Alcom-FT Technical Report Series
ALCOMFT-TR-01-100

CP-based Lagrangian Relaxation for a Multimedia Application*

Meinolf Sellmann and Torsten Fahle

University of Paderborn
Department of Mathematics and Computer Science
Fürstenallee 11, D-33102 Paderborn
{sello,tef}@uni-paderborn.de

Abstract. Whereas CP methods are strong with respect to the detection of local infeasibilities, OR approaches have powerful optimization abilities that ground on tight global bounds on the objective. An integration of propagation ideas from CP and Lagrangian relaxation techniques from OR combines the merits of both approaches. We introduce a general way of how linear optimization constraints can strengthen their propagation abilities via Lagrangian relaxation. The algorithm is evaluated on a set of benchmark problems steming from a multimedia application. The experiments show the superiority of the combined method compared with pure CP and OR approaches and an algorithm based on two independent optimization constraints.

Keywords: Lagrangian relaxation, shortest path constraint, knapsack constraint, multimedia

1 Introduction

One important branch concerning the integration of CP and OR is the development of *optimization constraints* (see [10]). Assuming a maximization problem $P(x)$, the idea is to use upper bound information to detect non-profitable assignments for a variable: If an upper bound on $P(x_{|x_i=k})$ drops below the best known solution value, then k can be removed from the domain of x_i.

In many optimization problems, constraints can be grouped such that the overall problem can be modelled as a combination of optimization constraints. Assuming that cost-based filtering for these constraints can be performed efficiently, we still do not necessarily achieve an effective propagation. This is because tight bounds on the objective cannot be obtained by taking only a subset of the constraints into account.

For a multimedia application, we show exemplary how linear optimization constraints can be coupled via Lagrangian relaxation to overcome this weakness.

* This work was partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

1.1 A Multimedia Application

The technology of digital television offers new possibilities for individualized services that cannot be provided by nowadays analog broadcasts. Additional information like classification of content, or starting and ending times can be submitted within the digital broadcast stream. With those informations at hand, new services can be provided that make use of individual profiles and maximize customer satisfaction.

One service – which is available already today – is an "intelligent" digital video recorder that is aware of its users' preferences and records automatically (see [4]). The recorder tries to match a given user profile with the information submitted by the different TV channels. (E.g., a user may be interested in blockbusters, the more recent the better.) The digital video recorder is supposed to record movies such that the users' satisfaction is maximized. As the number of channels may be enourmous (more than 100 channels are possible), a service that automatically provides an individual selection is highly appreciated.

In this context, two restrictions have to be met. First, the storage capacity is limited (10h of MPEG-2 video need about 18 GB). And second, only one video can be recorded at a time.

More formally, we define the problem as follows:

Definition 1. *Let $n \in \mathbb{N}$, $V = \{0, \dots, n-1\}$ the set of movies, $s_i < e_i \; \forall\, i \in V$ the corresponding starting and ending times. $w = (w_i)_{0 \le i < n} \in \mathbb{R}_+^n$ the storage requirements, $K \in \mathbb{R}_+$ the storage capacity, and $p = (p_i)_{0 \le i < n} \in \mathbb{R}_+^n$ the profit vector.*

We say that the interval $I_i := [s_i, e_i]$ corresponds *to movie $i \in V$, and call two movies $i, j \in V$* overlapping *whose corresponding intervals overlap, i.e. $I_i \cap I_j \neq \emptyset$. For $X \subseteq V$ we call $p_X := \sum_{i \in X} p_i$ the* user satisfaction (with respect to X).

*The **Automatic Recording Problem (ARP)** then is to find a subset $X \subseteq V$ such that*

(a) X can be stored within the given disc size, i.e. $\sum_{i \in X} w_i \le K$.
(b) only one movie must be recorded at a time, i.e. $I_i \cap I_j = \emptyset \; \forall\, i \neq j \in X$
(c) X maximizes the user satisfaction, i.e. $p_X \ge p_Y \; \forall\, Y \subseteq V$, Y respecting (a) and (b).

1.2 Previous Work

The integration of concepts from Operations Research (OR) and Constraint Programming (CP) has been subject to intensive research in the last years. A recent overview on attempts of combining CP and OR can be found e.g. in [19].

The work presented in this paper is based on some previous results on shortest path and knapsack constraints: In [8, 15], an efficient shortest path constraint was introduced. It is based on a variant of Djikstra algorithm for directed acyclic graphs (see [1]) and achieves bound consistency in time $O(n + e)$ on DAGs with n nodes and e edges. In [23], we showed that bound consistency for a shortest path on a node-weighted co-interval graph (given as list of negative profits and starting and ending times) can be computed in amortized time $\Theta(n)$.

Knapsack constraints have been studied in [9]. There, amortized linear time propagation routines based on continuous bounds were developed.

The multimedia application we use to introduce CP-based Lagrangian Relaxation is investigated in [16]. The goal there was to develop heuristics for an extension of the ARP.

Lagrangian relaxation was first presented in [7] for resource allocation problems. Held and Karp used it for the TSP [13, 14] and it has been applied in many different areas since then. We refer to [1] for an introduction.

In [10], Focacci et al. introduced a method to strengthen bound propagation using Lagrangian Relaxation. In that approach, Lagrangian multipliers were used to incorporate additional cuts to tighten the bound used for propagation. In our application, each group of relaxed constraints together with the objective forms a structured optimization constraint itself; structured meaning, that certain properties of the constraints can be exploited to achieve efficient propagation methods. To our knowlegde, the combination of propagation routines via Lagrangian relaxation as presented in this paper has not been considered before.

1.3 Optimization Constraints

2 CP and OR

To solve problems like the ARP, there exist two canonical approaches using CP or OR techniques, respectively. In the following, we present those approaches and discuss their respective advantages and disadvantages.

2.1 Pure CP Approach

From a Constraint Programming point of view, we search for a selection of movies $X \subseteq V$

(a) that can be stored within the given capacity of the hard disc, i.e. $\sum_{i \in X} w_i \leq K$,
(b) that does not contain movies overlapping in time, i.e. $\forall\, i \neq j \in X : I_i \cap I_j = \emptyset$,
(c) and that has a better objective function value than the currently best solution $\overline{X}$, i.e. $p_X > p_{\overline{X}}$.

Propagation and Tree Search The common way in CP to compute a solution is to apply propagation within a tree search. In every search node, the three constraints corresponding to (a), (b) and (c) are applied to shrink the domain of a set variable X that originally can take values in 2^V. In every search node, the domain of X can be specified by defining two sets. The first set is the *required set* $Req(X)$ containing all movies that already have been decided to be recorded in the current search node. The second set is the *possible set* $Pos(X)$ that contains all movies that still can possibly be recorded. Thus,

$$Req(X) \subseteq X \subseteq Pos(X).$$

All constraints try to add movies to $Req(X)$ and to delete elements from $Pos(X)$ by drawing logical implications. E.g., the non-overlapping constraint can delete all movies from $Pos(X)$ that overlap with one or more elements in $Req(X)$. This process is called *constraint propagation*. Of course, when one constraint has changed the domain of X, another constraint may force the domain to be shrunk even further. Thus, the constraints are being propagated until no further reduction is possible. If $Req(X) = Pos(X)$ – i.e., if X is bound – we have found a new and improving solution. To obtain a bounded variable X, typically constraint propagation alone is not sufficient, i.e. the propagation calculus is not complete. We therefore need to incorporate branching: for example, after all logical implications have been used to shrink the domain of X as much as possible, a branching variable $x \in Pos(X) \setminus Req(X)$ is chosen. In one branch of the search tree, we continue by setting

$$Req(X) \leftarrow Req(X) \cup \{x\}.$$

And in the other we proceed by setting

$$Pos(X) \leftarrow Pos(X) \setminus \{x\}.$$

Partitioning the search space by applying disjunctions yields a number of active search nodes that all must be investigated further.

To obtain a good algorithm, the propagation routines must be both, efficient and effective. This means, that propagation needs to be done fast, and as many logical implications must be drawn as possible to shrink the domains of variables. On top of that, the above tree search outline allows choices concerning the selection of the next search node, and the branching variable. Both choices are highly important regarding the efficiency of the resulting algorithm. With respect to the selection of the next search node, tree traversal methods like depth first search, breadth first search, and more sophisticated strategies like e.g. limited or depth-bounded discreapancy search (see [24]) have been developed. The choice of the branching variable is usually being tuned towards the specific application. Strategies like choosing the variable with the smallest domain show good results for many applications. For our puposes however, the smallest domain strategy is arbitrary, as there is only one set variable X, and all movies are either in X or not.

CP Pros and Contras While the CP approach is usually strong solving feasibility problems, it is often very poor in its optimization abilities. Treating the objective function as a constraint while using trivial bounds for propagation is ineffective. This remains true even if the search for the optimal objective value is being done in a bisectioning manner. Moreover, as the constraints are being treated independently from each other, logical implications caused by the interaction of two or more constraints cannot be seen. In our application e.g., movies that cause the objective to drop below the value of the incumbant solution because they consume the entire remaining storage capacity could not be removed from the possible set, as the capacity constraint and the objective constraint are being propagated independently.

2.2 Pure OR Approach

Using mathematical programming, the multimedia problem can be stated as an integer program:

$$\begin{array}{lrl} \text{Maximize} & \sum_{0\le i<n} p_i x_i & \qquad\qquad\qquad\qquad (IP\,1) \\ \text{subject to} & x_i + x_j \le 1 & \forall\, 0 \le i < j < n,\ I_i \cap I_j \ne \emptyset \\ & \sum_{0\le i<n} w_i x_i \le K & \\ & x \in \{0,1\}^n & \end{array}$$

The objective function maximizes the user satisfaction. Constraints of the form $x_i + x_j \le 1$ ensure that for overlapping intervals I_i, I_j at most one movie can be selected. Storage restrictions are enforced by the last row.

The formulation can be tightened by replacing the overlapping constraints with maximal clique constraints.

Definition 2. *A set $C \subseteq V$ is called a* conflict clique*, iff $I_i \cap I_j \ne \emptyset\ \forall\, i,j \in C$. A conflict clique C is called* maximal*, iff $\forall\, D \subseteq V$, D conflict clique: $C \subseteq D \Rightarrow C = D$. Let $M := \{C_0, \ldots, C_{m-1}\} \subseteq 2^V$ the set of maximal conflict cliques in G.*

Then restrictions of the form

$$\sum_{i\in C_p} x_i \le 1 \qquad \forall\, 0 \le p < m$$

imply that $x_i + x_j \le 1$ for all nodes $i, j \in V$ whose corresponding intervals overlap. On the other hand, if $x_i + x_j \le 1$ for all overlapping intervals, it is also true that $\sum_{i\in C_p} x_i \le 1\ \forall\, 0 \le p < m$. Thus, $(IP\,1)$ is equivalent to

$$\begin{array}{lrl} \text{Maximize} & \sum_{0\le i<n} p_i x_i & \qquad\qquad\qquad (IP\,2) \\ \text{subject to} & \sum_{i\in C_p} x_i \le 1 & \forall\, 0 \le p < m \\ & \sum_{0\le i<n} w_i x_i \le K & \\ & x \in \{0,1\}^n & \end{array}$$

Branch-And-Bound Although there exist methods that do not split the search space – like e.g. cutting plane algorithms – to solve a mixed integer program branch-and-bound approaches have proofed to be efficient, widely applicable and thus are most commonly used. In every search node, a bound based on some (usually continuous) relaxation is being computed. If that bound is worse than the objective value B of the incumbant solution, backtracking occurs. Like for the CP approach an effective and efficient propagation is needed, a successful application of the branch-and-bound paradigm analogically relies heavily on tight bounds that can be computed quickly. Moreover, we have the same freedom regarding the choice of the branching variable and the selection of the next search node as in any tree search.

Continuous Relaxation Pros and Contras As the continuous relaxation bound incorporates all or at least a set of constraints and all variables, it reflects a global view on the problem. And if it is too weak, it can be strengthened by adding cuts, e.g. in a branch-and-cut approach. However, the OR approach lacks the ability to tighten the problem formulation by quickly drawing logical implications of the constraints. Thus, infeasibilities are often detected rather late, and computational effort is wasted in branches that do not contain feasible solutions.

3 Two Different Ways of Possible Integration

3.1 First Order Integration

A first and obvious way of integrating CP and OR is to apply a tree search combining variable fixing and bound calculations using constraint propagation and linear programming, respectively. This idea can be refined by using the OR techniques – like shortest path algorithms – not only for pruning, but also to perform variable fixing, e.g. by applying reduced cost propagation.

Regarding our application, propagation routines for knapsack problems and shortest path problems are of interest: Knapsack problems evolve when dropping the constraints (b), i.e. when combining the constraints (a) and (c). And shortest path problems on weighted co-interval graphs occur when relaxing the initial setting by ignoring the capacity constraint (a), i.e. when tieing constraints (b) and (c) together. (An approach to combine constraints (a) and (b) is obviously ineffective.)

We presented ways of how variable fixing can be done for knapsack problems in [9], and for shortest path problems in [8]. Moreover, we developed a specialized algorithm for shortest paths on weighted co-interval graphs in [23]. To keep the paper self contained, we briefly review the relevant findings regarding knapsack and shortest path propagation. For a detailed description, we refer to the original papers.

Propagation for Knapsack Problems Let the problem be given as

$$\begin{array}{ll} \text{Maximize} & \sum_{0\le i<n} p_i x_i \\ \text{subject to} & \sum_{0\le i<n} w_i x_i \le K \\ & \qquad\qquad x \in \{0,1\}^n \end{array}$$

It has been known for a long time that bounds based on the linear continuous relaxation can be found in time $\Theta(n)$ (see [5]).

Bound U_1 Without loss of generality, we may assume that the items are ordered according to decreasing efficiency, i.e. $\frac{p_1}{w_1} \ge \frac{p_2}{w_2} \ge \cdots \ge \frac{p_n}{w_n}$. We define the *critical item* s of a knapsack problem as the first item that overloads the knapsack, that is $s = \min\{j \mid \sum_{i=0}^{j} w_i > K \text{ or } p_j \le 0\}$ (we omit the trivial case here where no such s exists). In [6] Dantzig showed that the linear relaxation of the 0-1 knapsack has the optimal value $\sum_{j=0}^{s-1} p_j$ if $p_s \leq 0$ and $\sum_{j=0}^{s-1} p_j + \overline{k}\frac{p_s}{w_s}$ otherwise, where $\overline{k}$ is defined as the remaining capacity of the knapsack after filling in the first $s-1$ items: $\overline{k} := K - \sum_{j=0}^{s-1} w_j$. Let $q \in \{0,1\}$ with $q = 1 \Leftrightarrow p_s > 0$. The first upper bound on KP is defined as

$$U_1 = \sum_{j=0}^{s-1} p_j + q \cdot \left(\overline{k}\frac{p_s}{w_s}\right)$$

Bound U_2 A second bound U_2 was introduced in [17]. Assume that $p_s > 0$. Then, we impose the integrality of the critical item s. Either item s belongs to the optimal solution (yielding an upper bound U^1) or not (leading to an upper bound U^0):

$$U^0 = \sum_{j=0}^{s-1} p_j + q \cdot \left(\overline{k}\frac{p_{s+1}}{w_{s+1}}\right)$$

$$U^1 = \sum_{j=0}^{s-1} p_j + q \cdot \left(p_s - (w_s - \overline{k})\frac{p_{s-1}}{w_{s-1}}\right)$$

$$U_2 = \max(U^0, U^1)$$

Defining U_2 as the maximum of U^0 and U^1 results in a bound dominating U_1, i.e. $U_2 \leq U_1$. We developed a variable fixing routine for knapsack problems that is based on bound U_2 and that runs in time $\Theta(n \log n)$. In a tree search with branching decisions of the form $x_i := 1$ or $x_i := 0$ for some item $0 \leq i < n$ and a minimum of $\Omega(\log n)$ search nodes an amortized linear time propagation algorithm exists (see [9]).

Propagation for Shortest Paths on Weighted Co-Interval Graphs We model the problem like in $(IP\,2)$ and achieve a totally unimodular LP formulation:

$$\begin{array}{lll} \text{Minimize} & \sum_{0 \leq i < n} -p_i x_i & \\ \text{subject to} & \sum_{i \in C_p} x_i \leq 1 & \forall\, 0 \leq p < m \\ & x \geq 0 & \end{array} \qquad (LP\,3)$$

For $(LP\,3)$, we were able to develop a greedy pivot selection strategy that solves the problem after at most n simplex iterations. Moreover, by using an efficient implementation, the problem can be solved in time $\Theta(n \log n)$. The algorithm works in two phases. In phase I, a data structure independent of the objective function is being set up in time $\Theta(n \log n)$. In phase II, we solve the shortest path problem in linear time.

An extension of the algorithm scetched allows the fixing of variables with respect to

- the deletion of nodes which are blocked with some required node,

– the deletion of nodes that would cause the shortest path value to exceed a given value B when being required,
– and the requiring of nodes that must be on any path with a corresponding value below a given value B.

All this can also be done in time $\Theta(n \log n)$. For at least $\Omega(\log n)$ calls of the propagation algorithm, it even runs in amortized linear time. The subsequent calls may vary with respect to the deletion or requiring of nodes, and also regarding a change of the objective function. Thus, the method is well suited to be applied within a Column Generation or Lagrangian Relaxation framework.

Our first approach to solve the ARP by integrating methods from CP and OR is based on a shortest path constraint and a knapsack constraint:

We apply a branch-and-bound algorithm based on the linear relaxation bound of the ARP[1]. If a search node cannot be pruned, it is being thightened by using the propagation routines developed for knapsack and shortest path constraints that we have just outlined.

3.2 Second Order Integration

Obviously, the shortest path and the knapsack relaxation are both rather weak. The first evolves from dropping constraint (a), the latter from ignoring constraints (b). But neither assuming an unlimited capacity of our storing device nor ignoring the fact that only one movie can be recorded at a time naturally results in a tight upper bound on the objective. But the accuracy of the upper bound is essential for the effectiveness of pruning and propagation. We therefore need to improve here.

In the pure OR approach, we use the linear relaxation on all constraints and variables to compute the upper bound. That linear relaxation bound can easily be obtained by applying a standard LP solver or by using specialized methods tailored for this specific application (see section 4.1). Then however, except for the conventional reduced cost propagation, we do not know how to draw logical implications to shrink the search space.

On the other hand, we know good propagation methods already for the shortest path and the knapsack relaxation. Those methods are able to draw logical implications from special properties of the constraints. The propagation algorithms, however, rely heavily on a tight bound on the objective. But only a global bound can be accurate, i.e. it cannot be achieved by looking at either one constraint only.

Lagrangian relaxation allows us to bring together the advantages of a tight continuous global bound and the existing propagation algorithms that exploit the special structure of their respective constraints. As the shortest path propagation method allows us to incorporate changing objectives at a low computational cost, we decide to relax the capacity constraint. We introduce a non-negative Lagrange multiplier $\lambda \geq 0$ and define the Lagrangian subproblem

[1] See section 4.1 for a description on how we compute that bound.

$$
\begin{array}{lll}
\text{Maximize} & z(\lambda) := z & L(\lambda) \\
\text{subject to} & z = \sum_{0 \le i < n} (p_i - \lambda w_i) x_i + \lambda K & \\
& \sum_{i \in C_p} x_i \le 1 & \forall\, 0 \le p < m \\
& x \in \{0,1\}^n &
\end{array}
$$

The Lagrange multiplier problem then is to solve

$$
\begin{array}{ll}
\text{Minimize} & z(\lambda) \\
\text{subject to} & \lambda \ge 0
\end{array}
$$

For every $\lambda \ge 0$, $z(\lambda)$ is a valid upper bound on the objective. Therefore, we can apply the propagation method for shortest path problems on weighted co-interval graphs each time we solve the Lagrangian subproblem. After we have found an optimal Lagrange multiplier λ^*, i.e. $z(\lambda^*) \le z(\lambda)\ \forall\ \lambda \ge 0$, we can use the (optimal) dual information $\pi \in \mathbb{R}^m$ from the corresponding shortest path subproblem to propagate the knapsack constraint now. By Lagrange relaxing the maximal clique constraints with multipliers $\pi \ge 0$, we obtain a knapsack problem:

Let $\mu_i := \sum_{j \mid i \in C_j} \pi_j\ \forall\ 0 \le i < n$ and $\overline{\pi} := \sum_{0 \le j < m} \pi_j$. The problem then is to

$$
\begin{array}{ll}
\text{Maximize} & \sum_{0 \le i < n} (p_i + \mu_i) x_i - \overline{\pi} \\
\text{subject to} & \sum_{0 \le i < n} w_i x_i \le K \\
& x \in \{0,1\}^n
\end{array}
$$

Relaxations of this problem again yield an upper bound on the objective, and we can apply the algorithms for knapsack propagation using e.g. bound U_2.

We conclude, that two linear optimization constraints for which efficient propagation methods are known can be combined effectively by computing Lagrangian multipliers for the first, using the second for propagation on each Lagrangian subproblem $L(\lambda)$, and then handing back dual information of the optimal $L(\lambda^*)$ to propagate the first constraint with the corresponding (optimal) reduced cost objective.

4 Numerical Results

4.1 Implementation Details

As we focus on the general principle of how existing propagation routines for optimization constraints can be exploited more effectively by tieing them together using Lagrangian relaxation, so far we have left out implementation details concerning the choice of the branching variable and the computation of the Lagrange multiplier λ^*. In this section, we will give an insight in the implementation the tests were performed with. However, we would like to stress that we were not aiming at developing a most efficient ARP solver. In fact, we only used that application to outline the idea behind the – as we believe: widely applicable – paradigm we call CP-based Lagrangian Relaxation.

Continuous Bound Computation For the pure OR, the first and the second order integration algorithms, the computation of a linear bound on the objective is needed for pruning. The second order integration algorithm obviously uses the objective value corresponding to $L(\lambda^*)$ for this purpose. As the computation via Lagrangian relaxation with shortest path subproblems turned out to be very efficient, we used that algorithm also for the other two approaches.

Computation of $\boldsymbol{\lambda^*}$ To determine λ^*, we used a method to maximize one-dimensional concave functions based on the golden section. We obtain a sequence of λ^k, $k \in \mathbb{N}$. Let $e_{max} := \max\{p_i/w_i \mid 0 \leq i < n\}$. Then, for all $\varepsilon > 0$ there exists a constant $c > 0$ such that

$$|\lambda^k - \lambda^*| < \varepsilon \qquad \forall\, k \geq c \cdot \log e_{max}$$

Thus, after at most $O(\log e_{max})$ iterations we can numerically approximate the optimal Lagrange multiplier λ^*. Each iteration costs amortized linear time for at least $\Omega(\log n)$ search nodes. Finally, in every search node we add $O(n \log n)$ for the succeeding knapsack propagation. Therefore, the propagation method for the tight global continuous relaxation bound runs in time $O(n \log e_{max} + n \log n)$.

Branching Variable Selection All algorithms except the pure CP approach choose the first node on the shortest path[2] with maximal efficiency p_i/w_i as branching variable. In the pure CP approach, the first node with maximal efficiency is being chosen.

4.2 Experiments

We consider the experiments as preliminary, as we were only able to run a small number of tests before submitting the paper. E.g., at the moment we have only tested 10 instances per benchmark class. For a detailed analysis a bigger sample would be desirable, and we will include more experiments in case of acceptance.

All experiments were performed on a PC with an AMD-Athlon 600 processor and 256 MB ram running Linux 2.2. The implementation was done in C++ and compiled by gcc 2.95 with maximal optimization (O3). The algorithms were built on top of ILOG SOLVER 5.0 [12].

The test instances were generated by specifying the time horizon and the number of channels. The generator sequentially fills the channels by starting each new movie one minute after the last. First, a class for the next movie is being chosen randomly. That class then determines the intervals from which the length and the profit are chosen randomly (for now, we have only been using 3 different classes of movies). The disc space necessary to store each movie equals its length, and the storage capacity is randomly chosen as 40%–60% of the entire time horizon.

[2] according to the optimal reduced costs objective $\sum_{0 \leq i < n} (p_i - \lambda^* w_i) x_i + \lambda^* K$

Test set			*runtime (in sec.)*			
minutes	*#channels*	*Ø movies*	*CP approach*	*OR approach*	*Int. 1*	*Int. 2*
360	3	16.8	0.3	0.0	0.0	0.0
360	5	27.6	3.2	0.0	0.0	0.0
360	20	114.7	—	0.2	0.1	0.1
360	50	284.4	—	0.5	0.4	0.4
360	100	564.8	—	1.5	1.0	0.8
720	5	54.2	—	0.1	0.1	0.1
720	20	206.8	—	0.7	0.5	0.4
720	50	521.8	—	3.1	2.1	1.3
720	100	1043.4	—	9.5	5.9	3.7
1440	5	101.0	—	0.4	0.3	0.2
1440	20	401.8	—	2.4	1.7	1.4
1440	50	1010.0	—	33.0	12.2	6.2
1440	100	2032.9	—	341.8	145.0	25.9
4320	5	292.3	—	2.6	2.0	1.7
4320	20	1168.0	—	131.4	69.2	26.9
4320	50	2936.7	—	4654.1	2718.1	279.1
7200	5	487.6	—	11.0	7.9	5.4
7200	20	1938.4	—	—	2656.5	526.6

Table 1. Numerical results for the ARP. The first three columns characterize each group of 10 instances by giving the planning horizon in minutes, the number of channels and the average number of movies. Columns 4 to 7 present running times for the pure CP, pure OR, Integration 1, and Integration 2 approaches.

The experiment consists of 10 random instances per test set. For each instance, the approaches presented in the previous sections were run to find and prove an optimal solution. The minutes for the time horizon equal 6 hours, 12 hours, one day, 3 days, and 5 days of digital television, respectively.

As Table 1 shows, the pure CP approach is not competitive at all. This is caused by the weak independent propagation of the constraints (a), (b) and (c). Therefore, we did not consider that approach to solve the bigger test instances.

The impact of a tight global bound used for pruning becomes obvious in the results for the other approaches. For up to 600 movies, all approaches perform well and prove an optimal solution within 3.1 seconds. For bigger instances, the pure OR approach performs worse compared to the combined approaches. For one test set, it is not able to prove the optimality of a solution within 3 hours.

Both integrative algorithms show the expected effect: A combination of CP and OR is superior and forms an efficient solution method. Integration 2 clearly outperforms integration 1. The main reason for this is that a tight bound is essential for an effective pruning.

It should be noted that all compeditive approaches find a first solution rather early in the search. Therefore, the main work lies in the prove of optimality rather than in the construction of the solution. We conclude that the branching variable selection we used efficiently supports finding near-optimal solutions in a non-exhaustive search.

A topic of further investigation is the comparison of test sets with an equal number of movies: E.g., (720min, 100ch), (1440min, 50ch), (4320min, 20ch) all contain about 1000 movies, and the solution time decreases with the number of channels increasing. In a test set with a high number of channels, many nodes are overlapping, hence the shortest path constraint can easily eliminate many variables from consideration. With a lower number of channels and a wider time horizon, the knapsack constraint gains more importance. We assume that a vice versa approach (i.e. Lagrange relaxing the non-overlapping constraints and solving a knapsack as subproblem) would be a suitable choice for such a situation.

5 Conclusions and Future Work

For a multimedia problem, we exemplary introduced the idea of CP-based Lagrangian Relaxation. Several extensions are possible for the application. A digital video recorder could have more than one recording unit which allows the recording of a limited number of channels simultaneously. In an IP context, this modification can be introduced easily. For the new approach presented, a fast and efficient propagation routine for this type of relaxed non-overlapping constraint is subject to further research.

CP-based Lagrangian Relaxation allows to combine existing propagation routines of linear optimization constraints to obtain effective and efficient cost-based filtering algorithms based on tight global bounds. We believe, that this idea is generic and independent of the specific application we presented here. In the future, we will formalize a framework for CP-based Lagrangian Relaxation.

Moreover, variations of the method presented here are possible: E.g., for each Lagrangian subproblem, dual information of the shortest path constraint could be handed over to the knapsack constraint to establish a deeper integration and to enforce an even more effective cost-based filtering.

References

1. R.K. Ahuja, T.L. Magnati, J.B. Orlin. Network Flows. *Prentice Hall*, 1993.
2. M. Atallah and D. Chen and D. Lee. An optimal algorithm for shortest paths on weighted interval and circular-arc graphs, with applications. *Algorithmica*, 14(5): 429–441, 1995.
3. G. Ausiello, G.F. Italiano, A.M. Spaccamela, U. Nanni. Incremental Algorithms for Minimal Length Paths. *Journal of Algorithms*, 12(4): 615–638, 1990.
4. *mediaTV*, technical description, Axcent AG. `http://www.axcent.de`.
5. E. Balas and E. Zemel. An algorithm for large-scale zero-one knapsack problems. *Operations Research*, 28:119–148, 1980.
6. G.B. Dantzig. Discrete variable extremum problems. *Operations Research*, 5:266–277, 1957.
7. H. Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resource. *Operations Research* 11:399–417, 1963.
8. T. Fahle, U. Junker, S.E. Karisch, N. Kohl, M. Sellmann, B. Vaaben. Constraint programming based column generation for crew assignment. University of Paderborn, Technical Report tr-ri-99-212, 1999, accepted by Journal of Heuristics.
9. T. Fahle and M. Sellmann. Constraint Programming Based Column Generation with Knapsack Subproblems. *Proceedings of the CP-AI-OR'00*, Paderborn Center for Parallel Computing, Technical Report tr-001-2000:33–44, 2000.
10. F. Focacci, A. Lodi, M. Milano. Cutting Planes in Constraint Programming: An Hybrid Approach. *Proceedings of the CP-AI-OR'00*, Paderborn Center for Parallel Computing, Technical Report tr-001-2000:45–51, 2000.
11. M.C. Golumbic. Algorithmic Graph Theory and Perfect Graphs. *Academic Press, New York*, 1991.
12. ILOG. Ilog Solver. Reference manual and user manual. V5.0, ILOG, 2000.
13. M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees *Operations Research* 18:1138–1162, 1970.
14. M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees: Part II *Mathematical Programming* 1:6–25, 1971.
15. U. Junker, S.E. Karisch, N. Kohl, B. Vaaben, T. Fahle, M. Sellmann. A Framework for Constraint Programming based Column Generation. *Proceedings of the CP'99*, Springer LNCS 1713: 261–274, 1999.
16. M. Lehradt. Basisalgorithmen für ein TV Anytime System, *Diploma Thesis* (in german), University of Paderborn, 2000.
17. S. Martello and P. Toth. An upper bound for the zero-one knapsack problem and a branch and bound algorithm. *European Journal of Operational Research*, 1:169–175, 1977.
18. S. Martello and P. Toth. Knapsack Problems – Algorithms and Computer Implementations. *Wiley Interscience*, 1990.

19. M. Milano. *Integration of Mathematical Programming and Constraint Programming for Combinatorial Optimization Problems*, Tutorial at CP2000, Sept. 2000, Singapore.
20. G.L. Nemhauser, L.A. Wolsey. Integer and Combinatorial Optimization. *Wiley Interscience*, 1988.
21. G. Ramalingam, T. Reps. An Incremental Algorithm for a Generalization of the Shortest-Path Problem. *Journal of Algorithms*, 21(2): 267–305, 1992.
22. G. Ramalingam, T. Reps. On the computational complexity of dynamic graph problems. *Theoretical Computer Science*, 158(1–2): 233–277, 1995.
23. M. Sellmann. Shortest Paths on Weighted Co-Interval Graphs. University of Paderborn, Technical Report tr-ri-00-219, 2000.
24. T. Walsh. Depth-bounded discrepancy search. *Proceedings of the IJCAI'97*, 1997.
25. F.B. Zhan, C.E. Noon. Shortest Path Algorithms: An Evaluation using Real Road Networks. *Transportation Science*, 32(1), 1988.