An hypergraph based formulation for an Automatic Storage Design problem

Luis Marques, François Clautiaux, Aurélien Froger

Univ. Bordeaux, CNRS, Inria, Bordeaux INP, IMB, UMR 5251, F-33400 Talence, France

June 29, 2023

Table of Contents

Introduction

Arc flow formulations are increasingly popular in the field of integer programming, [de Lima et al., 2022].

Such formulations are built from graphs, often transitions graphs of dynamic programs.

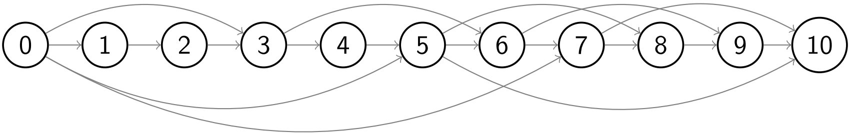


Figure: Example of arc flow formulation

Introduction

Arc flow formulations are increasingly popular in the field of integer programming, [de Lima et al., 2022].

Such formulations are built from graphs, often transitions graphs of dynamic programs.

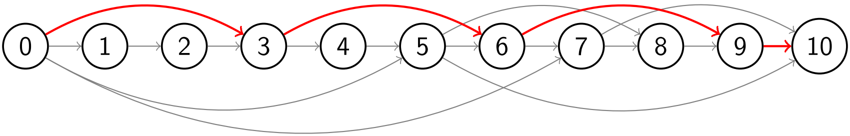


Figure: Example of arc flow formulation

Introduction

Arc flow formulations are increasingly popular in the field of integer programming, [de Lima et al., 2022].

Such formulations are built from graphs, often transitions graphs of dynamic programs.

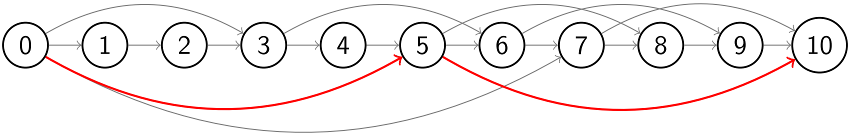


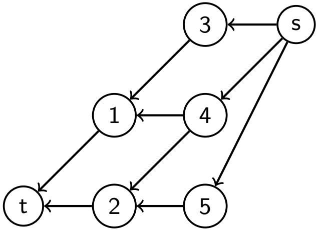
Figure: Example of arc flow formulation

Arc flow formulations and dynamic programming

Bacwkards recursive dynamic program:

- s: initial state
- t: terminal state
- f: function associating a state v to the minimum cost of going from state s to v
- $c_{u,v}$: cost of going from state u to state v
- $\Gamma^-(v)$: set of states preceding v

$$f(v) = \begin{cases} 0 & \text{if } v = s \\ \min_{u \in \Gamma^-(v)} \{c_{u,v} + f(u)\} & \text{otherwise} \end{cases}$$



Arc flow formulations and dynamic programming

- $G = (V, A)$: transition graph
- R: set of resources
- For every resource $r \in R$:
  - q_r: resource capacity
- For every arc $a \in A$:
  - c_a: arc cost
  - $b_{a,r}$: resource consumption of resource r

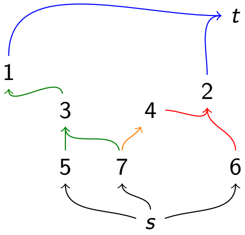
$$
\begin{aligned}
\text{minimize} \quad & \sum_{a \in A} c_a x_a && && (1) \\
\text{subject to} \quad & \sum_{a \in A^-(v)} x_a - \sum_{a \in A^+(v)} x_a = 0 && v \in V \setminus \{s, t\} && (2) \\
& \sum_{a \in A^-(t)} x_a = 1 && && (3) \\
& \sum_{a \in A(r)} b_{a,r} x_a \leq q_r && r \in R && (4) \\
& x_a \in \mathbb{N} && a \in A && (5)
\end{aligned}
$$

Extending arc flow formulation to hypergraphs

Bacwkards recursive dynamic program:

- f: function associating a state v to the minimum cost of going from state s to v
- $\Gamma^{-}(v)$: set of sets of states v can be decomposed into
- $c_{U,v}$: cost of decomposing state v into the set of states U

$$f(v) = \begin{cases} 0 & \text{if } v = s \\ \min_{U \in \Gamma^{-}(v)} \{c_{U,v} + \sum_{u \in U} f(u)\} & \text{otherwise} \end{cases}$$



Extending arc flow formulations to hypergraphs

Graph

- The flow conservation constraints form a totally unimodular matrix
- The domain of variables is $\{0, 1\}$

Hypergraph

- The flow conservation constraints form a totally dual integral matrix (TDI) [Martin et al., 1990]
- The domains of variables is unbounded and bounding those we can breaks the TDI property
- Flow may be multiplied without cost

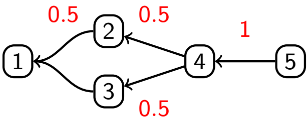


Figure: Example of half a unit of flow creating one unit of flow

Table of Contents

Automatic Storage Design
Introduction

We consider a two-phase and three-dimensional variant of the temporal knapsack problem called *Automatic Storage Design* problem:

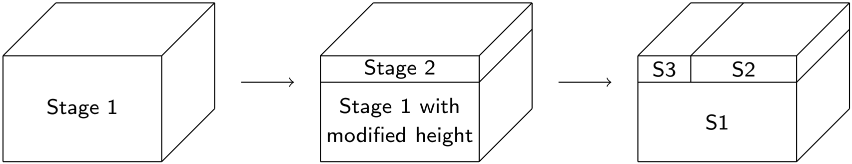
- M: Storage box with a *height* H, a *width* W and a *length* L.
- $\mathcal{I}$: Set of items. Each item i has a *height* h_i, a *width* w_i, a *length* l_i, a *profit* p_i, a *time period* s_i at which it enters the storage and a *duration* d_i during which the item is stored.
- $\mathcal{T}$: Set of consecutive time periods.

Automatic Storage Design
Introduction

The decisions represent a box design followed by an assignment of items.

- Partition the box to form shelves and partition each shelf to form compartments.
- Assign items to compartments.

Objective: Maximise the sum of profits of items assigned to the box.

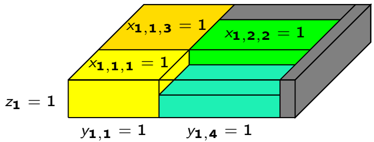


Automatic Storage Design

Compact formulation

Decision variables:

- z_i: 1 if a shelf of height h_i has been created, 0 otherwise.
- $y_{i,j}$: 1 if a compartment of width w_j has been created in the shelf of height h_i such that $i \leq j$, 0 otherwise.
- $x_{i,j,k}$: 1 if item k has been assigned to the compartment of height h_i and width w_j such that $i \leq j \leq k$, 0 otherwise.



Automatic Storage Design

Compact formulation

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i \in \mathcal{I}} \sum_{\substack{j \in \mathcal{I} \\ j \geq i}} \sum_{\substack{k \in \mathcal{I} \\ k \geq j}} p_i x_{i,j,k} \\
\text{subject to} \quad & \sum_{i \in \mathcal{I}} z_i h_i \leq H \\
& \sum_{\substack{j \in \mathcal{I} \\ j \geq i}} y_{i,j} w_j \leq W && i \in \mathcal{I} \\
& \sum_{\substack{k \in \mathcal{I} \\ k \geq j \\ s_k \leq t \leq s_k + d_k}} l_k x_{i,j,k} \leq L && i, j \in \mathcal{I}, j \geq i, t \in \mathcal{T} \\
& \sum_{\substack{i,j \in \mathcal{I} \\ i \leq j \leq k}} x_{i,j,k} \leq 1 && k \in \mathcal{I} \\
& x_{i,j,k} \leq y_{i,j} && i, j, k \in \mathcal{I}, k \geq j \geq i \\
& y_{i,j} \leq z_i && i, j \in \mathcal{I}, j \geq i \\
& z_i \in \{0, 1\} && i \in \mathcal{I} \\
& y_{i,j} \in \{0, 1\} && i, j \in \mathcal{I}, j \geq i \\
& x_{i,j,k} \in \{0, 1\} && i, j, k \in \mathcal{I}, k \geq j \geq i
\end{aligned}
$$

Automatic Storage Design

Dynamic program

Each state is noted by $(h, w, l)^s$, h being the height, w the width, l the length and s being the stage.

$$\alpha^1(h, w, l) = \max_{h' \in \mathcal{H}, h' \leq h} \{\alpha^2(h', w, l) + \alpha^1(h - h', w, l)\}$$

$$\alpha^2(h, w, l) = \max_{w' \in \mathcal{W}_h, w' \leq w} \{\alpha^3(h, w', l) + \alpha^2(h, w - w', l)\}$$

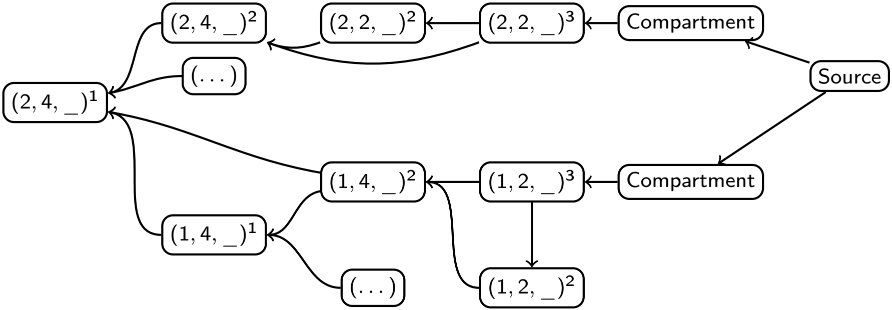
States $(h, w, l)^3$, which are temporal knapsack problems, are modeled by events as in [Clautiaux et al., 2021].

Automatic Storage Design

Dynamic program

$$\alpha^1(h,w,l) = \max_{h' \in \mathcal{H}, h' \leq h} \{\alpha^2(h',w,l) + \alpha^1(h-h',w,l)\}$$

$$\alpha^2(h,w,l) = \max_{w' \in \mathcal{W}_h, w' \leq w} \{\alpha^3(h,w',l) + \alpha^2(h,w-w',l)\}$$



Automatic Storage Design

Arc flow formulation

Let $G = (V, A)$ be the transition hypergraph associated with the dynamic program. Let $A(i)$ be the set of arcs corresponding to assigning item i in a compartment.

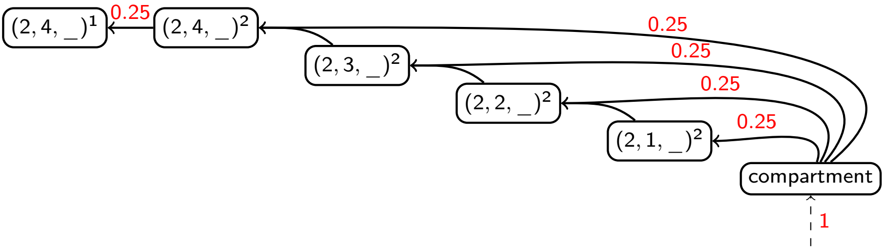
$$
\begin{aligned}
\text{maximize} \quad & \sum_{a \in A} p_a x_a & \\
\text{subject to} \quad & \sum_{a \in A^-(v)} x_a - \sum_{a \in A^+(v)} x_a = 0 & v \in V \setminus \{s, t\} \\
& \sum_{a \in A^-(t)} x_a = 1 & \\
& \sum_{a \in A(i)} x_a \leq 1 & i \in \mathcal{I} \\
& x_a \in \mathbb{N} & a \in A
\end{aligned}
$$

Table of Contents

Improve the linear relaxation

Valid cuts and algorithm changes

Suppose a box with height 2 and width 4 and two items with heights 1 and 2 and widths 4 and 1.

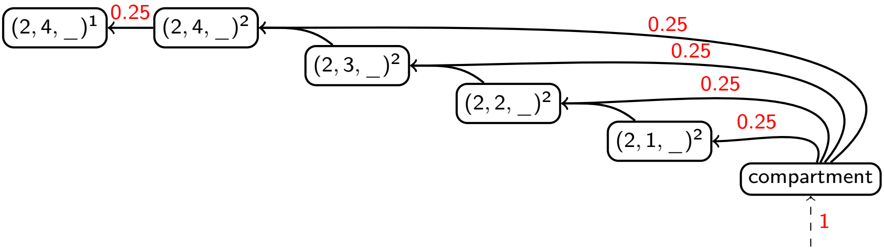


Flow recombines to form a compartment.

Improve the linear relaxation

Valid cuts and algorithm changes

Suppose a box with height 2 and width 4 and two items with heights 1 and 2 and widths 4 and 1.



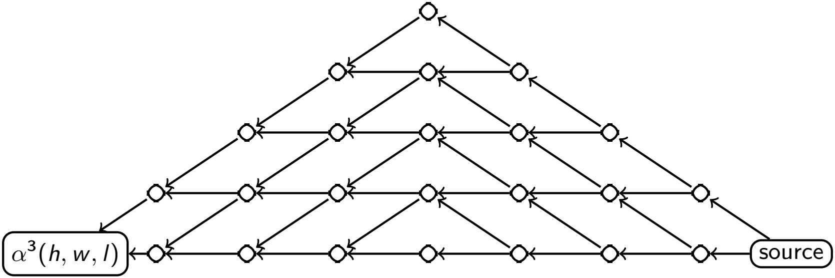
Flow recombines to form a compartment.

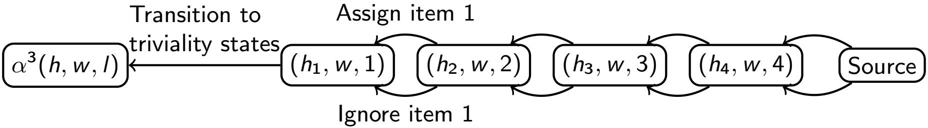
Valid cut:

$$\sum_{a \in A(i)} x_a - \sum_{h' \geq h_i} \sum_{a \in A(h)} x_a \leq 0 \quad i \in \mathcal{I}$$

Detect trivial subproblems

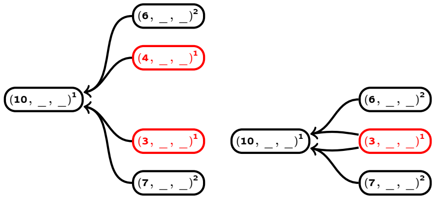
A subproblem is trivial if there exists a solution such that every item that fits is assigned.

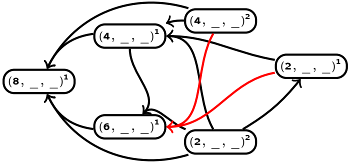


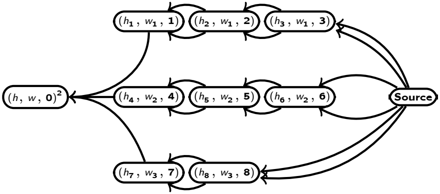


Reduce the size of the hypergraph

Other improvements







and many others...

Table of Contents

Numerical results

Experiments configuration

Two classes of instances are generated:

- $H = W = L = 500$.
- $h_i,\ w_i,\ l_i,\ p_i \in \mathcal{U}(80, 170)$.
- s_i and d_i are generated by cliques for the first class, $s_i,\ d_i \in \mathcal{U}(0, 1000)$ for the second class, similarly to [Caprara et al., 2013]

Group	$\lvert\mathcal{I}\rvert$
C10	~ 75
C15	~ 110
C30	~ 215
U50	50
U70	70
U100	100
U200	200

Table: Average number of items per group of instances

Machine setup: Haswell Intel Xeon E5-2680 v3 CPU at 2.5 GHz with 128 Go RAM.
Solver: CPLEX 20.1.
Time limit: 30 minutes.

Numerical results

Comparison of formulations

Group	NbVariables	NbConstraints	IP/LP Gap[1,2]	NbSolved
C10	17759	53230	5.97 %	1 / 10
C15	49068	151939	11.73 %	0 / 10
C30	363145	1120488	46.13 %	0 / 10
U50	5420	69781	0.14 %	9 / 10
U70	10438	181547	1.30 %	4 / 10
U100	21204	504690	11.41 %	1 / 10
U200	86361	3791192	70.08 %	1 / 10
Total	–	–	–	16 / 70

Table: Compact MIP formulation

Group	NbVertices	NbArcs	IP/LP Gap[1,2]	NbSolved
C10	146307	193277	4.48 %	2 / 10
C15	570665	698449	6.50 %	0 / 10
C30	5150654	5933438	31.33 %	0 / 10
U50	17183	32867	0.14 %	9 / 10
U70	30617	59912	1.30 %	3 / 10
U100	48738	102766	5.74 %	1 / 10
U200	150916	369086	3.27 %	4 / 10
Total	–	–	–	19 / 70

Table: Arc flow formulation

Both formulations have around the same number of constraints but the hypergraph formulation has around 13.3 times more variables.

[1] IP/LP Gap formula : $\frac{|\mathbf{LP} - \mathbf{BestInteger}|}{|\mathbf{BestInteger}|}$

[2] Only instances where the three formulations LPs could be solved are included

Numerical results

Impact of improvements on hypergraph formulation

Group	NbVertices	NbArcs	IP/LP Gap[1,2]	NbSolved
C10	146307	193277	4.48 %	2 / 10
C15	570665	698449	6.50 %	0 / 10
C30	5150654	5933438	31.33 %	0 / 10
U50	17183	32867	0.14 %	9 / 10
U70	30617	59912	1.30 %	3 / 10
U100	48738	102766	5.74 %	1 / 10
U200	150916	369086	3.27 %	4 / 10
Total	–	–	–	19 / 70

Table: Arc flow formulation

Group	NbVertices	NbArcs	IP/LP Gap[1,2]	NbSolved
C10	38398	55888	4.42 %	2 / 10
C15	156530	199988	6.39 %	0 / 10
C30	1860081	2157920	31.32 %	0 / 10
U50	2533	10250	0.14 %	10 / 10
U70	4467	17300	1.30 %	7 / 10
U100	5374	23790	5.27 %	7 / 10
U200	15976	96790	2.77 %	7 / 10
Total	–	–	–	33 / 70

Table: Improved arc flow formulation

The improved version has about 20% of the number of vertices and 29% of the number of arcs.

[1] IP/LP Gap formula : $\frac{|\mathbf{LP} - \mathbf{BestInteger}|}{|\mathbf{BestInteger}|}$

[2] Only instances where the three formulations LPs could be solved are included

Table of Contents

Conclusion

We have seen:

- Arc flow formulations on graphs and hypergraphs and the differences between both tools.
- A newly defined problem and a compact and an arc flow formulation for it.
- Several improvements useful to improve the linear relaxation and reduce the size of the arc flow formulation.

Perspectives:

- Propose further valid inequalities to improve the linear relaxation of the arc flow formulation.
- Thorough computational study on what reductions improve best the linear relaxation and the size of the formulation.

References I

Caprara, A., Furini, F., and Malaguti, E. (2013).
Uncommon dantzig-wolfe reformulation for the temporal knapsack problem.
INFORMS Journal on Computing, 25(3):560–571.

Clautiaux, F., Detienne, B., and Guillot, G. (2021).
An iterative dynamic programming approach for the temporal knapsack problem.
European Journal of Operational Research, 293(2):442–456.

de Lima, V. L., Alves, C., Clautiaux, F., Iori, M., and de Carvalho, J. M. V. (2022).
Arc Flow Formulations Based on Dynamic Programming: Theoretical Foundations and Applications.
European Journal of Operational Research, 296(1):3–21.

Martin, R. K., Rardin, R. L., and Campbell, B. A. (1990).
Polyhedral Characterization of Discrete Dynamic Programming.
Operations Research, 38(1):127–138.