Automatic Storage Design

Luis Marques, François Clautiaux, Aurélien Froger

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

May 4, 2023

Table of Contents

Introduction

Automated storage and retrieval systems have been studied for years in the context of warehouse optimization. See [Roodbergen and Vis, 2009] for a survey.

Different objectives can be optimised, such as response time, wasted space or overall profit.

From a cutting and packing point of view, we see the problem as a variant of 3D knapsack problem. We also consider a time aspect.

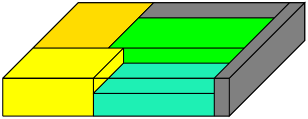


Figure: Example of 3D knapsack solution

Table of Contents

Automatic Storage Design

The problem is defined as a 3D temporal knapsack problem with additional constraints.

- $\mathcal{T}$: Set of consecutive time periods.
- 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.

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

Automatic Storage Design

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

- Partition the box horizontally to form shelves.
- Partition the shelves vertically to form compartments.
- Assign items to compartments to maximise the profit.

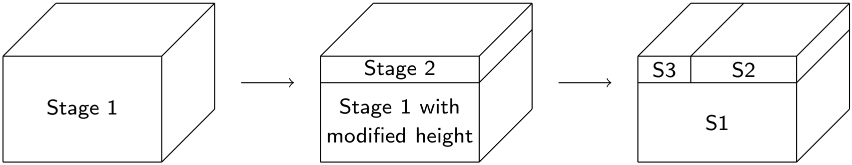


Figure: Example of automatic storage design

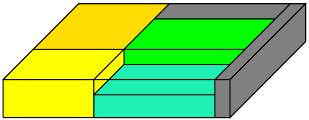


Figure: Example of design and assignment with $t = 0$

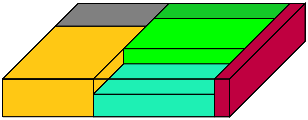


Figure: Example of design and assignment with $t = 1$

Relation with the literature

Our problem is related to the three-dimensional three-stage guillotine packing problem (see [de Queiroz et al., 2012] and [Martin et al., 2021] for works on this problem).

It generalizes the two-dimensional two-stage packing knapsack problem (see [Iori et al., 2021] for a survey on 2D packing).

It also generalizes the temporal knapsack and temporal bin packing problems (see [Caprara et al., 2013], [Caprara et al., 2016], [Clautiaux et al., 2021], [Dell'amico et al., 2020], [**?**]).

Automatic Storage Design

Decision variables:

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

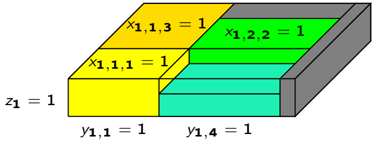


Figure: Example of design and assignment

Automatic Storage Design

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

Table of Contents

Dynamic programming formulation

Guillotine 2D-packing can be solved by MIP models based on flows in hypergraphs (see e.g. [Clautiaux et al., 2018]).

Temporal knapsack and bin packing problems can be solved by methods based on DP and arc-flow models [Clautiaux et al., 2021, **?**].

We propose a dynamic program to create a transition graph and use it to build an arc-flow formulation (on an hypergraph).

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

Dynamic programming formulation

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

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

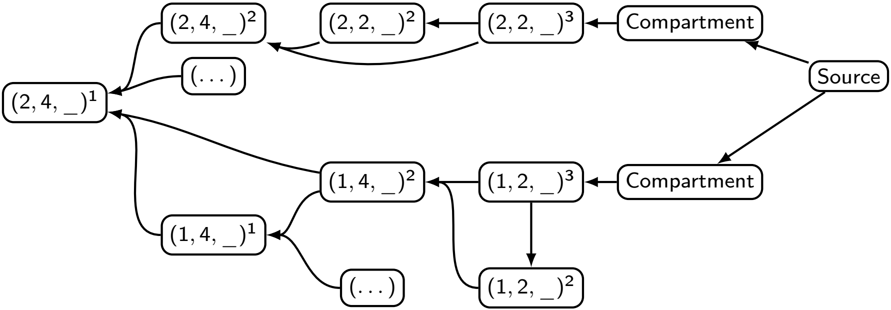


Figure: Example of hypergraph

Compartments, which are TKPs, are modeled by events as in [Clautiaux et al., 2021].

Dynamic programming formulation

Dynamic programming to MIP formulation

Let $G = (V, A)$ be the transition hypergraph associated with the dynamic program.

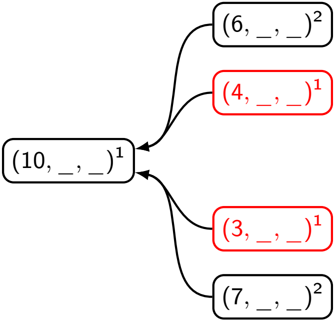
$$
\begin{aligned}
\text{maximize} \quad & \sum_{a \in A} p_a x_a \\
\text{subject to} \quad & \sum_{a \in A^-(v)} x_a - \sum_{a \in A^+(v)} x_a = 0 && v \in V \\
& \sum_{a \in A^-(v_0)} x_a = 1 \\
& \sum_{a \in A(i)} x_a \leq 1 && i \in \mathcal{I} \\
& x_a \in \{0, 1\} && a \in A
\end{aligned}
$$

Table of Contents

Reduce the size of the hypergraph

Aggregate equivalent states

Although the dimensions may be different, states that have the same candidate items are equivalent problems.



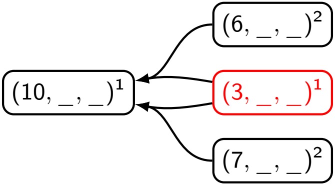
$h_i \in \{3, 6, 7\}$

Figure: Example of equivalent states

Reduce the size of the hypergraph

Aggregate equivalent states

Although the dimensions may be different, states that have the same candidate items are equivalent problems.



$h_i \in \{3, 6, 7\}$

Figure: Example of aggregated equivalent states

Reduce the size of the hypergraph

Trivial subproblems

A subproblem of dimensions (h, w, l) is trivial if all candidate items can be assigned to it.

If a subproblem is trivial, we can relax the dimensions related to the capacity of the compartment.

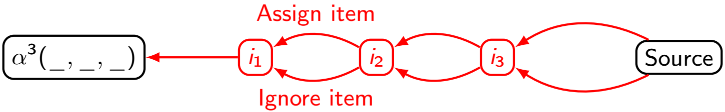


Figure: Example of vertices of trivial compartment

Reduce the size of the hypergraph

Other improvements

Other improvements:

- Break symmetries by imposing an order for heights and widths in the design.
- Detect trivial shelves, trivial and partially trivial compartments.
- Bound the number of times a height or a width is present.
- Aggregate states of similar compartments.
- Partially enumerate easy subproblem solutions.
- Reverse the events list.

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	$\|\mathcal{I}\|$
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	NbSolved
C10	17759	53230	1 / 10
C15	49068	151939	0 / 10
C30	363145	1120488	0 / 10
U50	5420	69781	9 / 10
U70	10438	181547	4 / 10
U100	21204	504690	1 / 10
U200	86361	3791192	1 / 10
Total	–	–	16 / 70

Table: Compact MIP formulation

Group	NbVertices	NbArcs	NbSolved
C10	146307	193277	2 / 10
C15	570665	698449	0 / 10
C30	5150654	5933438	0 / 10
U50	17183	32867	9 / 10
U70	30617	59912	3 / 10
U100	48738	102766	1 / 10
U200	150916	369086	4 / 10
Total	–	–	19 / 70

Table: Hypergraph MIP formulation

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

Numerical results

Impact of improvements on hypergraph formulation

Group	NbVertices	NbArcs	NbSolved
C10	146307	193277	2 / 10
C15	570665	698449	0 / 10
C30	5150654	5933438	0 / 10
U50	17183	32867	9 / 10
U70	30617	59912	3 / 10
U100	48738	102766	1 / 10
U200	150916	369086	4 / 10
Total	–	–	19 / 70

Table: Hypergraph MIP formulation

Group	NbVertices	NbArcs	NbSolved
C10	38398,4	55888,7	2 / 10
C15	156530,2	199988,2	0 / 10
C30	1860081,7	2157920,8	0 / 10
U50	2533,9	10250,2	10 / 10
U70	4467,7	17300,1	7 / 10
U100	5374,7	23790,7	7 / 10
U200	15976	96790,2	7 / 10
Total	–	–	33 / 70

Table: Improved hypergraph MIP formulation

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

Table of Contents

Conclusion

We have seen:

- A newly defined problem and a compact formulation for it.
- A dynamic programming reformulation and a MIP to solve the problem related to the DP's hypergraph.
- Several improvements useful to reduce the hypergraph's size and the solution space.

Perspectives:

- Improve trivial subproblems detection.
- Improve partial solution enumeration.
- Propose valid inequalities to improve the linear relaxation of the arc-flow model.
- Thorough computational study.

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.

Caprara, A., Furini, F., Malaguti, E., and Traversi, E. (2016).
Solving the temporal knapsack problem via recursive dantzig-wolfe reformulation.
Information Processing Letters, 116(5):379 – 386.

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.

Clautiaux, F., Sadykov, R., Vanderbeck, F., and Viaud, Q. (2018).
Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem.
Discrete Optimization, 29:18–44.

References II

de Queiroz, T. A., Miyazawa, F. K., Wakabayashi, Y., and Xavier, E. C. (2012).
Algorithms for 3d guillotine cutting problems: Unbounded knapsack, cutting stock and strip packing.
Computers and Operations Research, 39(2):200–212.

Dell'amico, M., Furini, F., and Iori, M. (2020).
A branch-and-price algorithm for the temporal bin packing problem.
Computers and Operations Research, 114:104825.

Iori, M., De Lima, V. L., Martello, S., Miyazawa, F. K., and Monaci, M. (2021).
Exact solution techniques for two-dimensional cutting and packing.
European Journal of Operational Research, 289(2):399–415.

Martin, M., Oliveira, J. F., Silva, E., Morabito, R., and Munari, P. (2021).
Three-dimensional guillotine cutting problems with constrained patterns: Milp formulations and a bottom-up algorithm.
Expert Systems with Applications, 168:114257.

References III

Roodbergen, K. J. and Vis, I. F. (2009).
A survey of literature on automated storage and retrieval systems.
European Journal of Operational Research, 194(2):343–362.