Automatic Storage Design

Luis Marques, François Clautiaux, Aurélien Froger

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

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

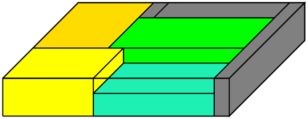


Figure: Example of 3D knapsack solution

Introduction

Three-dimensional guillotine packing has been less studied.

- [de Queiroz et al., 2012]
- [Martin et al., 2021]

Two-dimensional packing has been considered in many papers.

- see [Iori et al., 2021] for a survey

Temporal knapsack and temporal bin packing have also been studied.

- [Caprara et al., 2013]
- [Caprara et al., 2016]
- [Clautiaux et al., 2021]
- [Dell'amico et al., 2020]

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.

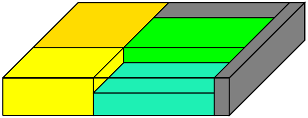


Figure: Example of design and assignment

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.

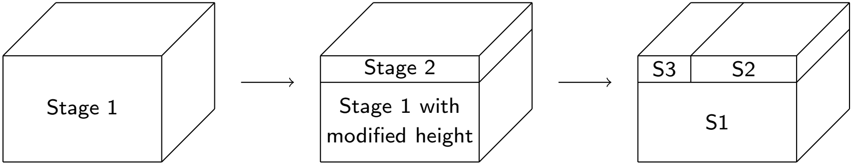


Figure: Example of automatic storage design

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.

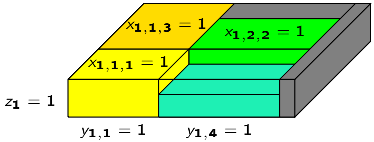


Figure: Example of design and assignment

Automatic Storage Design

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

Table of Contents

Dynamic programming formulation

We propose a dynamic program to create a transition graph and solve it through a MIP.

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

Notations:

- $\mathcal{H}$: Set of different heights, i.e. $\{h : \exists i \in \mathcal{I}, h_i = h\}$, sorted by non-increasing value.
- $\mathcal{W}$: Set of different widths, i.e. $\{w : \exists i \in \mathcal{I}, w_i = w\}$, sorted by non-increasing value.
- $\mathcal{W}_h$: Set of different candidate widths for a shelf of height h, i.e. $\{w : \exists i \in \mathcal{I}, h_i \leq h, w_i = w\}$, sorted by non-increasing value.

Dynamic programming formulation

Notations:

- $\alpha^s(h, w, l)$: Maximum profit of an automatic storage design problem in stage s with height h, width w and length l by using items from $\mathcal{I}$.

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

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

The optimal value of the dynamic program is given by $\alpha^1(H, W, L)$.

Dynamic programming formulation

Compartments are modeled by events as in [Clautiaux et al., 2021], each item leading two events.

Notations:

- $\mathcal{E}^{\text{in}}$: Set of in events.
- $\mathcal{E}^{\text{out}}$: Set of out events.
- $i(e)$: Item related to event e.
- $t(e)$: Time at which e occurs.

Events related to items are ordered from 1 to $2n$ as follows:

$$e < e' \text{ if } t(e) < t(e') \text{ or } \big(t(e) = t(e') \wedge e \in \mathcal{E}^{\text{out}} \wedge e' \in \mathcal{E}^{\text{in}}\big)$$

Dynamic programming formulation

$$\alpha^3(h, w, l) = \hat{\alpha}^3(h, w, l, 2n, 0)$$

If $e \in \mathcal{E}^{\text{in}} \wedge l + l_i \leq L$:

- $\hat{\alpha}^3(h, w, l, e, \mathrm{d}) = \max\left\{\frac{1}{2}p_{i(e)} + \hat{\alpha}^3(h, w, l + l_i, e - 1, \boldsymbol{\varepsilon_{i(e)}}), \hat{\alpha}^3(h, w, l, e - 1, \mathrm{d})\right\}$

If $e \in \mathcal{E}^{\text{in}} \wedge l + l_i > L$:

- $\hat{\alpha}^3(h, w, l, e, \mathrm{d}) = \hat{\alpha}^3(h, w, l, e - 1, \mathrm{d})$

If $e \in \mathcal{E}^{\text{out}} \wedge \mathrm{d}_i = 1$:

- $\hat{\alpha}^3(h, w, l, e, \mathrm{d}) = \frac{1}{2}p_{i(e)} + \hat{\alpha}^3(h, w, l - l_i, e - 1, \mathrm{d} - \boldsymbol{\varepsilon_{i(e)}})$

If $e \in \mathcal{E}^{\text{out}} \wedge \mathrm{d}_i = 0$:

- $\hat{\alpha}^3(h, w, l, e, \mathrm{d}) = \hat{\alpha}^3(h, w, l, e - 1, \mathrm{d})$

For every h, w, l and d, $\hat{\alpha}^3(h, w, l, 0, \mathrm{d}) = 0$.

Dynamic programming formulation

Dynamic programming to MIP formulation

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

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

Table of Contents

Improving the linear relaxation

Consistency constraints

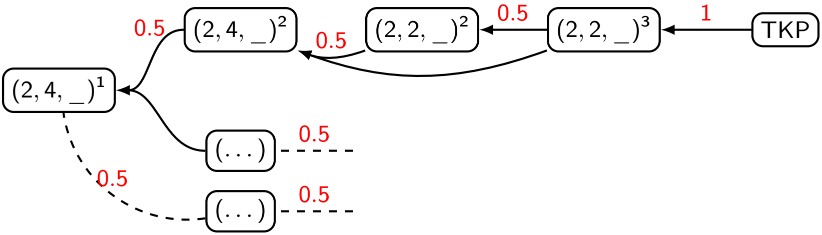


Figure: Example of fractional solution

We can assign items while using only "half" of the shelf.

Improving the linear relaxation

Consistency constraints

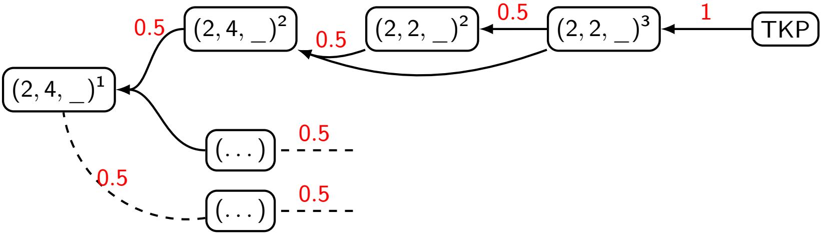


Figure: Example of fractional solution

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

Reduce the size of the hypergraph

Shelves height order

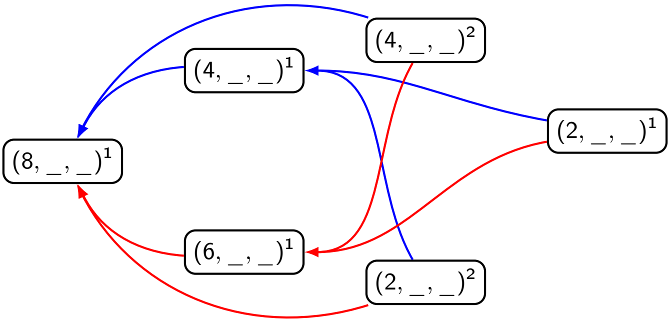


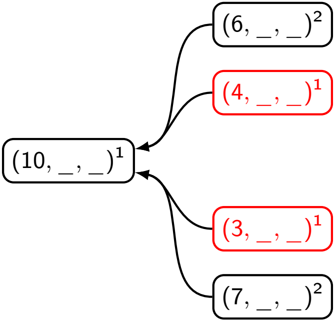
Figure: Example of symmetries

Solution: Consider shelf heights in a non-decreasing order similarly to [Becker et al., 2022] and [Rodrigues et al., 2023].

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.



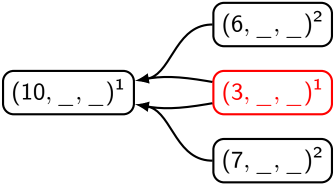
$h_i \in \{3, 6, 7\}$

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 shelves

A shelf of height h and width w is trivial if all candidate items can be assigned to it. If a shelf is trivial, the subproblem is replaced by a new stage.

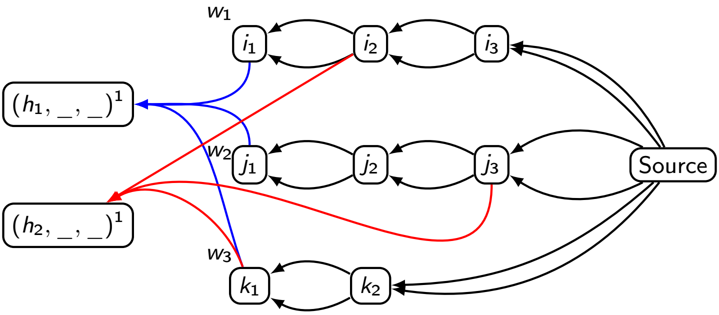


Figure: Example of trivial shelf stage

Reduce the size of the hypergraph

Other improvements we do

Other improvements:

- Reduce height of compartment design problem when a compartment is designed.
- Detect 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	$\lvert\mathcal{I}\rvert$
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 22.1.
Time limit: 30 minutes.

Numerical results

Comparison of formulations

Group	NbVariables	NbConstraints	NbSolved
C10	17978	53858	22 / 40
C15	52103	159082	8 / 40
C30	373325	1142640	0 / 40
U50	5344	69163	39 / 40
U70	10427	181536	34 / 40
U100	21111	509268	28 / 40
U200	85438	3763636	20 / 40
Total	–	–	151 / 280

Table: Compact MIP formulation

Group	NbVertices	NbArcs	NbSolved
C10	47562	61499	32 / 40
C15	180466	218640	28 / 40
C30	1533927	1761009	16 / 40
U50	5455	9800	39 / 40
U70	9186	17245	33 / 40
U100	14357	29223	31 / 40
U200	42620	101399	33 / 40
Total	–	–	212 / 280

Table: Hypergraph MIP formulation

The hypergraph formulation has around 4.8 times more variables but the compact formulation has around 3.1 times more constraints.

Numerical results

Impact of improvements on hypergraph formulation

Group	NbVertices	NbArcs	NbSolved
C10	47562	61499	32 / 40
C15	180466	218640	28 / 40
C30	1533927	1761009	16 / 40
U50	5455	9800	39 / 40
U70	9186	17245	33 / 40
U100	14357	29223	31 / 40
U200	42620	101399	33 / 40
Total	–	–	212 / 280

Table: Hypergraph MIP formulation

Group	NbVertices	NbArcs	NbSolved
C10	19793	26081	32 / 40
C15	75142	91703	30 / 40
C30	708735	813834	15 / 40
U50	1453	3650	40 / 40
U70	2199	5831	37 / 40
U100	2894	8264	37 / 40
U200	7640	30951	37 / 40
Total	–	–	228 / 280

Table: Improved hypergraph MIP formulation

The improved version has about 31% of the number of vertices and 37% 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:

- Generic valid cuts for hypergraphs.
- Improve trivial subproblems detection.
- Study how partial solution enumeration could help solve the problem.
- Study what aspects make the problem difficult to solve.

References I

Becker, H., Araújo, O., and Buriol, L. S. (2022).
Enhanced formulation for the Guillotine 2D Cutting Knapsack Problem.
Mathematical Programming Computation, 14(4):673–697.

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.

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

Rodrigues, C. D., Cherri, A. C., and de Araujo, S. A. (2023).
Strip based compact formulation for two-dimensional guillotine cutting problems.
Computers & Operations Research, 149:106044.

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.