Using decision hypergraphs to design a storage cabinet of limited size

Luis Marques[1], François Clautiaux[1], Aurélien Froger[1]

Univ. Bordeaux, CNRS, Inria, IMB, UMR 5251, F-33400 Talence, France
luis.lopes-marques@u-bordeaux.fr

Mots-clés : *Cutting and Packing, Integer linear programming, Temporal knapsack, Arc flow models, Decision hypergraphs*

We study the problem of designing a cabinet consisting of a set of shelves containing compartments whose contents slide forward on opening. Given a set of items candidate to be stored in the cabinet over a given time horizon, the problem is to design a set of shelves, a set of compartments in each shelf and to select the items to be placed in the compartments. The objective is to maximize the profit of the items selected in the device. Our problem is inspired by a real-world application involving inner-city pharmacies that use an automated under-the-counter storage and retrieval cabinet for the most common medications. We address a simplified version of such problems as a first step toward optimizing such capacity-constrained systems.

From a combinatorial optimization point of view, we study a two-phase and three-dimensional variant of the temporal knapsack problem, which we refer to as the *Storage Cabinet Physical Design* (SCPD) problem. In the first phase, called the *design phase*, a three-dimensional *storage cabinet* or *cabinet* for the rest of this paper, is divided horizontally into *shelves*, which are in turn divided vertically into *compartments*. The second phase, called the *assignment phase*, corresponds to the selection of the *items*, and in this case, their assignment to compartments. When an item is assigned to a compartment, it is by definition present during a given interval, which defines our temporal constraints.

We give a dynamic program for a relaxation of the SCPD problem where each item can be selected in more than one compartment. The states and transitions of this dynamic program define a decision hypergraph, a generalization of decision graphs. In Martin et al. (1990), a hyperarc is a triple $(\mathcal{F}, v, p)$, where $\mathcal{F}$ is a multiset of vertices called the *tail* of the hyperarc, v is a vertex called the *head* of the hyperarc, and p is a profit. In a hypergraph representation of a dynamic program, the vertices are the states of the dynamic program and the hyperarcs symbolize the decisions. Specifically, each hyperarc $(\mathcal{F}, v, p)$ represents the transition from the states at its tail to the state at its head. From the decision hypergraph of this relaxation of the SCPD problem, we derive an arc flow formulation equivalent to reformulating the SCPD problem as a maximum cost flow problem with additional linear constraints. Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be the decision hypergraph associated with the relaxation of the SCPD problem, with v^+ (resp. v^-) denoting its source (resp. sink). For each vertex $v \in \mathcal{V}$, let $\mathcal{A}^+$ (resp. $\mathcal{A}^-$) be the set of hyperarcs of which v is the head (resp. of which v belongs to the tail). For each hyperarc $a \in \mathcal{A}$, let p_a be its profit and let $x_a \in \{0, 1\}$ be its associated decision variable. The arc flow formulation is defined as follows, where $\mathcal{I}$ is the set of items and $\mathcal{A}(i)$ is the set of hyperarcs

carrying the decision of assigning item i.

$$\text{maximize} \quad \sum_{a \in \mathcal{A}} p_a x_a \tag{1a}$$

$$\text{subject to} \quad \sum_{a \in \mathcal{A}^-(v)} x_a - \sum_{a \in \mathcal{A}^+(v)} x_a = 0 \quad v \in \mathcal{V} \setminus \{v^+, v^-\} \tag{1b}$$

$$\sum_{a \in \mathcal{A}^-(v^-)} x_a = 1 \tag{1c}$$

$$\sum_{a \in \mathcal{A}(i)} x_a \leq 1 \qquad i \in \mathcal{I} \tag{1d}$$

$$x_a \in \mathbb{N} \qquad a \in \mathcal{A} \tag{1e}$$

Constraints (1a)-(1c) and (1e) define our maximum cost flow problem, and constraints (1d) define the additional linear constraints we add to ensure the feasible of the solution for the SCPD problem.

We show that the size of this arc flow formulation is a bottleneck when solving it directly with a MILP solver. Several techniques have been used in the literature to address this issue. Brandão and Pedroso (2016) proposed so-called *graph compression* techniques, which exploit the structure of the problem to reduce the number of arcs and vertices in the network. By studying the structure of the SCPD problem, we propose dominance rules to reduce the size of the hypergraph by (i) partially enforcing an order on the guillotine cuts and bounding the number of times a decision appears in a solution, (ii) merging states that are equivalent, and (iii) detecting and exploiting states that represent easily solvable subproblems. We also introduce a simple family of valid inequalities to improve the linear relaxation of the formulation.

To evaluate the performance of our formulations, we first randomly generate 180 instances by adapting a method from (Caprara et al., 2016). We first evaluate the impact of our dominance rules and valid inequalities on the performance of the arc flow formulation and show that adding them all gives the best reduction in the size of the decision hypergraph and the best increase in the number of instances solved. This configuration gives a reduction of about 61% and the solver solves 35 more instances than the arc flow formulation without the dominance rules and the valid inequalities. We then focus on the comparison between a simple and compact MILP formulation and the arc flow formulation with all dominance rules and valid inequalities. We confirm that the size of the arc flow formulation far exceeds the size of the compact MILP formulation, and we confirm the correlation between this increase in size and the increase in the quality of the linear relaxation. By counting the number of times the solver returned the best known primal bounds, the number of best known dual bounds, the number of instances solved and the optimality gap at the time limit, we see that the arc flow formulation outperforms the compact MILP formulation. In terms of best known primal bounds, the two formulations do not differ significantly, with the solver of the compact MILP formulation returning 151 times the best known primal bound against 137 for the solver of the arc flow formulation. However, the solver of the compact MILP formulation returned the best known dual bound only 81 times against 171 for the solver of the arc flow formulation. In addition, the solver of the arc flow formulation solved 25 more instances than the solver of the compact MILP formulation, with a total of 84 instances solved out of 180. We also observe a reduction in the optimality gap at the time limit, with 6.32% for the solver of the compact formulation and 4.38% for the solver of the arc flow formulation.

Références

Brandão, F. and Pedroso, J. P. (2016). Bin packing and related problems : General arc-flow formulation with graph compression. *Computers & Operations Research*, 69 :56–67.

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.

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