An exact column-generation approach for the lot-type design problem

Miriam Kießling, Sascha Kurz, and Jörg Rambau

Lehrstuhl für Wirtschaftsmathematik, Universität Bayreuth, Germany
{miriam.kiessling,sascha.kurz,joerg.rambau}@uni-bayreuth.de

Abstract. We consider a fashion discounter distributing its many branches with integral multiples from a set of available lot-types. For the problem of approximating the branch and size dependent demand using those lots we propose a tailored exact column generation approach assisted by fast algorithms for intrinsic subproblems, which turns out to be very efficient on our real-world instances.

Keywords: p-median, facility location, lot-type design, real world data, column generation

1 Introduction

Due to small profit margins of most fashion discounters, applying OR methods is mandatory for them. In order to reduce the handling costs and the error proneness in the central warehouse, our business partner orders all products in multiples of so-called *lot-types* from the suppliers and distributes them without any replenishing to its branches. A lot-type specifies a number of pieces of a product for each available size, e.g., lot-type $(1, 2, 2, 1)$ means two pieces of size M and L, one piece of size S and XL, if the sizes are $(\mathrm{S}, \mathrm{M}, \mathrm{L}, \mathrm{XL})$.

We want to solve the following approximation problem: which (integral) multiples of which (integral) lot-types should be supplied to a set of branches in order to meet a (fractional) expected demand as closely as possible? We called this specific demand approximation problem the *lot-type design problem (LDP)* in [5]. In that paper, also a basic model for the LDP was introduced, accompanied by an integer linear programming formulation and a tailored heuristic, which turned out to perform very well for the real-world data of our partner.

For many practical instances the set of applicable lot-types is so large that the ILP formulation from [5] cannot be solved directly. In this paper, we therefore propose an exact column generation approach for the LDP, enhanced by properly chosen algorithms for important subproblems. We apply this algorithm to a stochastic version SLDP of the LDP, where the expectation over more than one demand-scenario is optimized. The SLDP is of the same form as the LDP, and thus all optimization techniques viable for the LDP immediately apply to the SLDP. Since using the SLDP instead of the LDP can make decisions more robust against forecasting errors, we based our investigations in this paper on the SLDP.

Branch-and-price algorithms are common for large-scale integer programming problems [7]. Unifying general remarks can be found in [2, 9]. Branch-price-and-cut algorithms are surveyed in [8]. The LDP is related to the p-median and the facility location problem: for recent computational results on large instances of the p-median problem we refer the reader to [1, 4, 6].

A formal problem statement is given in Section 2, followed by an ILP model in Section 3. Our algorithm is presented in Section 4. We show computational results on real-world data in Section 5, before we conclude with Section 6.

2 Formal problem statement

We consider the distribution of supply for a single product.

Data. Let $\mathcal{B}$ be the set of branches of our fashion discounter. Let $\mathcal{L}$ be a set of applicable *lot-types*: For a set of utilized sizes $\mathcal{S}$, a *lot-type* is a vector $(l_s)_{s\in\mathcal{S}} \in \mathbb{N}^{|\mathcal{S}|}$; it specifies the number of pieces of each size in any pre-packed lot of this lot-type. There is an upper bound $\overline{I}$ and a lower bound $\underline{I}$ given on the total supply of the product over all branches and sizes. Moreover, there is an upper bound $k \in \mathbb{N}$ on the number of lot-types used.

Let $\mathcal{A}$ be a set of *scenarios* (for the success of the product). For each scenario $a \in \mathcal{A}$ we denote by p^a its probability and with $d^a_{b,s} \in \mathbb{Q}_{\geq 0}$ the demand at Branch b in Size s in Scenario a for all $b \in \mathcal{B}$ and $s \in \mathcal{S}$.

Decisions. Consider an assignment of a unique lot-type $l(b) \in \mathcal{L}$ and an assignment of a unique multiplicity $m(b)$ to each branch $b \in \mathcal{B}$. These data specify that $m(b)$ lots of lot-type $l(b)$ are to be delivered to Branch b.

Decision-dependent entities. The *inventory of Branch b in Size s* given assignments $l(b)$ and $m(b)$ is given by $I_{b,s}(l, m) = m(b)l(b)_s$. Moreover, the *total supply* resulting from $l(b)$ and $m(b)$ is given by $I(l, m) = \sum_{b\in\mathcal{B}} \sum_{s\in\mathcal{S}} I_{b,s}(l, m)$.

Objective. The *deviation between inventory and demand in Branch b and Size s in Scenario a* given $l(b)$ and $m(b)$ is defined by $\delta^a_{b,s}(l, m) := |d^a_{b,s} - I^a_{b,s}(l, m)|$. The *expected total deviation between supply and demand* given assignments $l(b)$ and $m(b)$ is $\Delta(l, m) := \sum_{a\in\mathcal{A}} p^a \sum_{b\in\mathcal{B}} \sum_{s\in\mathcal{S}} \delta^a_{b,s}(l, m)$.

The goal is to find a subset $L \subseteq \mathcal{L}$ of at most k lot-types and assignments $l(b) \in \mathcal{L}$ and $m(b) \in \mathcal{M}$ such that the total supply is within the bounds $\left[\underline{I}, \overline{I}\right]$, and the expected total deviation $\Delta(l, m)$ is minimized.

We call this single-stage stochastic optimization problem the *Stochastic Lot-Type Design Problem (SLDP)*. The SLDP is equivalent to an ordinary LDP (see [5] for details concerning the LDP) with a modified objective function, since the expected total deviation $\Delta(l, m)$ can be written as $\sum_{b\in\mathcal{B}} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} p^a \delta^a_{b,s}(l, m)$. In other words, the certainty equivalence principle (see e.g. [3, p. 28]) holds if the input data are the expected deviations for all branches and sizes. Certainty equivalence does not hold if the input data are the expected demands, though.

3 Modelling

With binary assignment variables $x_{b,l,m}$ indicating whether $l(b) = l$ and $m(b) = m$ and binary selection variables y_l indicating whether $l \in L$, we can model the SLDP as the following integer linear program. As an abbreviation we use $|l| := \sum_{s \in \mathcal{S}} l_s$.

$$\min \quad \sum_{b \in \mathcal{B}} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} c_{b,l,m} \cdot x_{b,l,m} \tag{1}$$

$$s.t. \quad \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} x_{b,l,m} = 1 \qquad \forall b \in \mathcal{B} \tag{2}$$

$$\sum_{l \in \mathcal{L}} y_l \leq k \tag{3}$$

$$\sum_{m \in \mathcal{M}} x_{b,l,m} \leq y_l \qquad \forall b \in \mathcal{B}, l \in \mathcal{L} \tag{4}$$

$$\underline{I} \leq \sum_{b \in \mathcal{B}} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} m \cdot |l| \cdot x_{b,l,m} \leq \overline{I} \tag{5}$$

$$x_{b,l,m} \in \{0,1\} \qquad \forall b \in \mathcal{B}, l \in \mathcal{L}, m \in \mathcal{M} \tag{6}$$

$$y_l \in \{0,1\} \qquad \forall l \in \mathcal{L}, \tag{7}$$

where $c_{b,l,m} = \sum_{a \in \mathcal{A}} p^a \cdot \sum_{s \in \mathcal{S}} \left| d^a_{b,s} - m \cdot l_s \right| \geq 0$.

4 A custom-made branch-and-price algorithm

Since the set of applicable lot-types and, thus, the set of binary variables in the stated ILP formulation may become quite large, a natural approach is to consider applicable lot-types dynamically in a branch-and-price algorithm.

In this section we show how special structure can be used to obtain a fast branch-and-price algorithm for practically relevant instances: We typically have $300 \leq |\mathcal{B}| \leq 1600$ and $3 \leq |\mathcal{M}| \leq 7$ while $|\mathcal{L}|$ can be around 10^9 in the case where the set of applicable lot-types is parameterizable (see Appendix A).

The idea of our specialized exact branch-and-price algorithm is based on the following practical observations on real-world data:

- The integrality gap of our SLDP model is small.
- Solutions generated by heuristics perform very well (see [5]).
- There seems to be a "small" set of good and a "large" set of bad solutions.
- No mathematical structure of the set of good solutions is known a-priori.
- A proof of optimality is wanted.

This led us to the following idea for our branch-and-price algorithm:

1. In a node, set-up a subproblem with so few independent decision variables that it can be enumerated completely or that it can be solved fast by the static ILP model.
2. Generate two branches: One branch containing the completely enumerated decision options (this branch can be solved to optimality immediately, e.g., by complete enumeration), one branch with the remaining decision options only (this branch receives a single cover cut excluding the decision options considered in the other branch).

Although this kind of branching seems weird at first glance – splitting off at most a usually constant number of feasible solutions asymptotically yields a linear depth of the full tree –, there is a rationale behind this: Whenever we can find all the few good solutions by solving promising subproblems, we can prune the remaining subtree as soon as all subproblems containing a good solution have been generated. Because of the small integrality gap, this can be detected by the LP relaxation value. And a pricing algorithm can prove LP optimality even if not all variables have been generated.

Our branch-and-price algorithm must generate a promising subproblem in such a way that excluding that subproblem can be done efficiently in the restricted master problem. Moreover, we need fast implementations of the main workhorse methods in order to process large enough subproblems. Therefore, we searched for purely combinatorial algorithms for the crucial subproblems.

The master problem (MP) of a branch-and-price node is defined to consist of the static SLDP model plus some cover-cuts (14) (see Subsection 4.5) of the node, which exclude the decision options of subproblems. (MP) can be restricted to a manageable sized restricted master problem (RMP) by the following: We only consider a (small) subset $\mathcal{L}' \subseteq \mathcal{L}$ of the lot-types. For each branch $b \in \mathcal{B}$ we consider a subset $\zeta_{\mathcal{L}'}(b) = \zeta(b) \subseteq \mathcal{L}'$ of these lot-types and for each $l \in \zeta_{\mathcal{L}'}(b)$ we consider only a subset $\eta_{\mathcal{M}}(b,l) = \eta(b,l) \subseteq \mathcal{M}$ of the multiplicities.

The restricted master problem (RMP) then reads as follows:

$$\min \quad \sum_{b\in\mathcal{B}} \sum_{l\in\zeta(b)} \sum_{m\in\eta(b,l)} c_{b,l,m} \cdot x_{b,l,m} \tag{8}$$

$$s.t. \quad \sum_{l\in\zeta(b)} \sum_{m\in\eta(b,l)} x_{b,l,m} = 1 \qquad \forall b \in \mathcal{B} \tag{9}$$

$$\sum_{l\in\mathcal{L}'} -y_l \geq -k \tag{10}$$

$$\sum_{b\in\mathcal{B}} \sum_{l\in\zeta(b)} \sum_{m\in\eta(b,l)} m \cdot |l| \cdot x_{b,l,m} \geq \underline{I} \tag{11}$$

$$\sum_{b\in\mathcal{B}} \sum_{l\in\zeta(b)} \sum_{m\in\eta(b,l)} -m \cdot |l| \cdot x_{b,l,m} \geq -\overline{I} \tag{12}$$

$$\sum_{m\in\eta(b,l)} -x_{b,l,m} + y_l \geq 0 \qquad \forall b \in \mathcal{B}, l \in \zeta(b) \tag{13}$$

$$\sum_{l \in \mathcal{C}_i} -y_l \geq -\gamma_i \quad \forall i \in \mathcal{I} \tag{14}$$

$$x_{b,l,m} \geq 0 \qquad \forall b \in \mathcal{B}, l \in \zeta(b), m \in \eta(b,l) \tag{15}$$

$$y_l \geq 0 \qquad \forall l \in \mathcal{L}'. \tag{16}$$

The dual restricted master problem (DRMP) is then given by:

$$\max \quad \sum_{b \in \mathcal{B}} \alpha_b \; - k\pi + \underline{I}u - \overline{I}v - \sum_{i \in \mathcal{I}} \gamma_i \mu_i \tag{17}$$

$$s.t. \qquad \alpha_b - \beta_{b,l} + m|l|u - m|l|v \leq c_{b,l,m} \quad \forall b \in \mathcal{B}, l \in \zeta(b), m \in \eta(b,l) \tag{18}$$

$$-\pi + \sum_{b \in \mathcal{B} \,:\, l \in \zeta(b)} \beta_{b,l} - \sum_{i \in \mathcal{I} \,:\, l \in \mathcal{C}_i} \mu_i \leq 0 \qquad \forall l \in \mathcal{L}' \tag{19}$$

$$\alpha_b \in \mathbb{R} \qquad \forall b \in \mathcal{B} \tag{20}$$

$$\pi, u, v \geq 0 \tag{21}$$

$$\beta_{b,l} \geq 0 \qquad \forall b \in \mathcal{B}, l \in \zeta(b) \tag{22}$$

$$\mu_i \geq 0 \qquad \forall i \in \mathcal{I}. \tag{23}$$

The pricing problem is defined by finding those constraints in the dual (DMP) of the unrestricted master problem (MP) that are most violated by the current solution of (DRMP).

4.1 Workhorse 1: Finding best fitting lot-types for single branches

The following optimization problem is extensively used in the pricing step, see Lemma 1, and in primal heuristics in Sections 4.7 and 4.8 (setting $\Omega = 0$, $\mathcal{L}' = \emptyset$).

For a given branch b we show how to solve the optimization problem

$$\min_{l \in \mathcal{L} \setminus \mathcal{L}', m \in \mathcal{M}} c_{b,l,m} - \Omega \cdot m \cdot |l|, \tag{24}$$

where $\Omega = u - v \in \mathbb{R}$ is given.

So let us assume that the cost coefficients $c_{b,l,m}$ are given by the expression in Section 2, the set of lot-types $\mathcal{L}$ is parameterized using the integers $\min_c$, $\min_t$, $\max_c$, $\max_t$, $\mathcal{L}'$ is given by an explicit list, and $\mathcal{M} = \left[\underline{m}, \overline{m}\right]$ for two integers $\underline{m}$, $\overline{m}$ (see Appendix A for details about this parametrization). In a preprocessing step we compute for each branch $b \in \mathcal{B}$, each size $s \in \mathcal{S}$, and each multiplicity $m \in \mathcal{M}$ the values $\gamma(b,s,m,l_s) = \sum_{a \in \mathcal{A}} p_a \cdot \left| d^a_{b,s} - m \cdot l_s \right|$, where $\min_c \leq l_s \leq \max_c$, and $\psi(b,s,m) = \min\left\{\gamma(b,s,m,l_s) \,:\, \min_c \leq l_s \leq \max_c\right\}$.

Let Λ be the value $c_{b,\hat{l},\tilde{m}} - \Omega \cdot \tilde{m} \cdot |\hat{l}|$ of our current champion, where we initialize $\Lambda = +\infty$. Next we fix the possible multiplicities $\underline{m} \leq m \leq \overline{m}$ and in each case start a branch&bound tree. To describe the nodes of the branch&bound tree we use a set $\mathcal{F} \subseteq \mathcal{S}$, where we set $\mathcal{F} = \emptyset$ for the root node. In each fixing step we choose a size $s \in \mathcal{S} \setminus \mathcal{F}$ and fix $\min_c \leq l_s \leq \max_c$, i.e., the number of items in size s of the emerging lot-type. If we either have $\sum_{s \in \mathcal{F}} l_s + \sum_{s \in \mathcal{S} \setminus \mathcal{F}} \min_c > \max_t$ or

$\sum_{s\in\mathcal{F}} l_s + \sum_{s\in\mathcal{S}\setminus\mathcal{F}} \max_c < \min_t$, we can prune the search tree, since the fixed values l_s can not be continued to an admissible lot-type.

Now let us estimate the minimal possible cost for the partially fixed lot-type corresponding to $\mathcal{F}$. The maximum number of items that can occur in the remaining sizes in $\mathcal{S}\setminus\mathcal{F}$ is given by $\overline{r} = \min\big(|\mathcal{S}\setminus\mathcal{F}| \cdot \max_c, \max_t - \sum_{s\in\mathcal{F}} l_s\big)$. Similarly the minimum number of items of the remaining sizes is given by $\underline{r} = \max\big(|\mathcal{S}\setminus\mathcal{F}| \cdot \min_c, \min_t - \sum_{s\in\mathcal{F}} l_s\big)$. If $\Omega > 0$ then we set $r = \overline{r}$, otherwise we set $r = \underline{r}$. With this every extension of the partially fixed lot-type corresponding to $\mathcal{F}$ results in costs of at least

$$\sum_{s\in\mathcal{F}} \gamma(b,s,m,l_s) + \sum_{s\in\mathcal{S}\setminus\mathcal{F}} \psi(b,s,m) - \Omega\cdot m\cdot\sum_{s\in\mathcal{F}} l_s - \Omega\cdot m\cdot r.$$

If these costs are at least as large as the costs Λ of our current champion, we can prune the search tree.

In the leafs, where the lot-type l is completely specified, we check whether $l \in \mathcal{L}'$. If this is not the case we have found a new champion.

4.2 Workhorse 2: Optimal multiplicities

A rather easy but relevant subproblem of the SLDP is the determination of an optimal multiplicity $\hat{m}$, i.e., for a given branch b and lot-type l we ask for $\hat{m} \in \mathcal{M}$ satisfying $c_{b,l,\hat{m}} \le c_{b,l,m}$ for all $m \in \mathcal{M}$. The most simple way would be to check all $|\mathcal{M}|$ possibilities for m and pick the best one.

If the deviations $c_{b,l,m}$ are convex in m and $\mathcal{M}$ consists of an interval of non-negative integers, like in our specific setting of Section 2, then we can determine $\hat{m}$ in at most $O(\log|\mathcal{M}|)$ evaluations using binary search.

4.3 Workhorse 3: Solving the SLDP-*k*

In this subsection we present an efficient algorithm for the SLDP-k, which is the SLDP with only k applicable lot-types. In other words we assume that the y_i of the ILP formulation from Section 2 are already fixed and it remains to determine the optimal $x_{b,l,m}$. To avoid some technical difficulties we assume that the set of possible multiplicities $\mathcal{M}$ is given by an interval of non-negative integers. If we additionally drop Inequality (5) on the overall supply the resulting optimization problem becomes quite easy. Since the number k of lot-types is a rather small number we may simply check them all for each branch $b \in \mathcal{B}$ and determine the corresponding optimal multiplicity using the methods from Subsection 4.2. This way, we can easily determine a best fitting lot-type $l(b)$ and an optimal multiplicity $m(b)$ for each branch separately.

If accidentally Inequality (5) is valid we have an optimum solution for the SLDP-k, or otherwise at least have derived a lower bound for its optimum target value. In the latter case we consider the SLDP-k and relax the integrality condition to $0 \le x_{b,l,m} \le 1$. Due to the convexity of the target function (1) this

problem can be efficiently solved by greedily adjusting the multiplicities $m(b)$ and the assignments $l(b)$ in order to fulfill Inequality (5).

For brevity we discuss only the case where the overall supply is strictly larger than $\overline{I}$. Here we have to iteratively take away items from some branches. To this end we introduce relative costs for each branch b and each alternative. If $m(b)-1$ is also an element of $\mathcal{M}$ then we can simply reduce $m(b)$ by one, which results in relative costs of $\frac{c_{b,l(b),m(b)-1}-c_{b,l(b),m(b)}}{|l(b)|} \geq 0$ per item. Another possibility is to change the used lot-type $l(b)$. Therefore we denote by $\varphi_b(l')$ the largest integer such that $\varphi_b(l') \cdot |l'| < m(b) \cdot |l(b)|$, i.e. $\varphi_b(l')$ is that multiplicity m for branch b and lot-type l resulting in a minimal coefficient $c_{b,l',m}$ while reducing the number of supplied items to branch b. If and only if $\varphi_b(l') \in \mathcal{L}$ we can modify the pair $(l(b), m(b))$ to $(l', \varphi_b(l'))$ resulting in relative costs of

$$\frac{c_{b,l,\varphi_b(l')} - c_{b,l(b),m(b)}}{m(b) \cdot |l(b)| - \varphi_b(l') \cdot |l'|} \geq 0$$

per item. So, after at most $O(1+(k-1)\log M)$ evaluations of coefficients $c_{b,l,m}$ we can determine the alternative with minimum relative costs Δ_b^- for each branch b, where we set $\Delta_b^- = \infty$ if there is no feasible alternative.

If we have the relative costs Δ_b^- for all $b \in \mathcal{B}$ and the corresponding actions at hand, we can pick a $\hat{b} \in \mathcal{B}$ which minimizes Δ_b^-. Let $\delta > 0$ denote the number of items which are removed by the corresponding action and I denote the overall supply corresponding to the current pair of functions l, b. Due to the convexity of the target function (1) we can state the following:

(a) If $\Delta_{\hat{b}}^- = \infty$, then the SLDP-$k$ subproblem is infeasible.
(b) If $I - \delta \geq \overline{I}$, then after performing the greedily optimal action the new assignments $\tilde{l}(b)$ and $\tilde{m}(b)$ correspond to an optimal solution of SLDP-k, where Inequality (5) is replaced by $\sum_{b\in\mathcal{B}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} m \cdot |l| \cdot x_{b,l,m} \leq I - \delta$.
(c) If $I - \delta < \overline{I}$ we obtain the optimal solution of the SLDP-k with fractional variables $x_{b,l,m}$ by utilizing a suitable linear combination of the old assignment $(l(b), m(b))$ and the cheapest decreasing alternative $(\tilde{l}(b), \tilde{m}(b))$.

Thus, after a finite number of iterations, depending at most linearly on the difference between the initial overall supply and $\overline{I}$, we obtain the optimal solution of the SLDP-k with at most two fractional variables $x_{b,l,m}$. To also solve the integral SLDP-k we utilize a branch-bound approach.

In order to obtain an efficient algorithm we maintain the Δ_b^--values in a heap data structure, so that in each recursion step we only have to determine one new Δ_b^--value, while the update of the heap can be done in $O(\log |\mathcal{B}|)$.

4.4 The most promising subproblem SLDP$|_{\mathcal{C}}$

Given an optimal solution of the current (RMP), we consider the values of the attained y-variables. To ease the notation we assume that they are ordered downwards, i.e., $y_1 \geq y_2 \geq \cdots \geq y_{|\mathcal{L}'|}$. For a small constant $\varepsilon > 0$, e.g., $\varepsilon = 0.15$ we

consider an index q such that $y_q \geq \varepsilon$ and $y_{q+1} < \varepsilon$. We call the subproblem of the SLDP with a given $\mathcal{C} := \{1, \ldots, q\} \subseteq \mathcal{L}$ as its set of applicable lot-types the *most promising subproblem* $\text{SLDP}|_{\mathcal{C}}$ of SLDP.

4.5 Branching into the most promision subproblem and the rest

At a node of Depth i of the branch-and-price tree we split the current node into two: on branch contains the most promising subproblem $\text{SLDP}|_{\mathcal{C}_i}$, the other branch contains the remaining decision options. The node corresponding $\text{SLDP}|_{\mathcal{C}_i}$ is then solved exactly. If $\binom{\mathcal{C}_i}{k} \leq 100\,000$ then we completely enumerate all k-subsets of lot-types in $\mathcal{C}_i$ and subsequently solve the corresponding SLDP-k (see Subsection 4.3). Otherwise we solve the corresponding ILP formulation from Section 3 directly. The other node has to be worked on further by branch-and-price: Excluding the most promising subproblem in this branch can be achieved by adding a single cover cut $\sum_{l \in \mathcal{C}_i} y_l \leq \gamma_i := k - 1$.

4.6 The combinatorial pricing algorithm

We associate with the constraints of the master problem (MP) the dual variables α_b, π, u, v, $\beta_{b,l}$, and μ_i. With this for each lot-type l the reduced costs for a variable $x_{b,l,m}$ are given by

$$c_{b,l,m} - \alpha_b - m \cdot |l| \cdot (u - v) + \beta_{b,l} \tag{25}$$

and for a variable y_l the reduced costs are given by

$$0 + \pi - \sum_{b \in \mathcal{B}} \beta_{b,l} + \sum_{i \in \mathcal{I}: l \in \mathcal{C}_i} \mu_i. \tag{26}$$

Lemma 1. *If* $\cup_{i \in \mathcal{I}} \mathcal{C}_i \subseteq \mathcal{L}'$,

$$\min_{b \in \mathcal{B}, l \in \zeta(b), m \in \mathcal{M}} c_{b,l,m} - \alpha_b - m \cdot |l| \cdot (u - v) + \beta_{b,l} \geq 0, \tag{27}$$

$$\min_{b \in \mathcal{B}, l \in \mathcal{L}' \setminus \zeta(b), m \in \mathcal{M}} c_{b,l,m} - \alpha_b - m \cdot |l| \cdot (u - v) \geq 0, \text{ and} \tag{28}$$

$$\max_{\mathcal{L} \setminus \mathcal{L}'} \sum_{b \in \mathcal{B}} \max\Big(0, \max_{m \in \mathcal{M}} \alpha_b + m \cdot |l| \cdot (u - v) - c_{b,l,m}\Big) \leq \pi \tag{29}$$

then the current optimal solution of (RMP) is optimal for (MP).

Proof. If Inequality (27) is valid then there is no variable $x_{b,l,m}$ with $l \in \zeta(b)$ having negative reduced costs. For a given branch b and a given lot-type $l \in \mathcal{L} \setminus \zeta(b)$ (DRMP) does not contain the variable $\beta_{b,l}$ since (RMP) does not include the corresponding inequality. So we extend the dual solution by setting

$$\beta_{b,l} = \begin{cases} 0 : l \in \mathcal{L}' \setminus \zeta(b), \\ \max\left(0, \max_{m \in \mathcal{M}} \alpha_b + m \cdot |l| \cdot (u - v) - c_{b,l,m}\right) : l \in \mathcal{L} \setminus \mathcal{L}' \end{cases} \tag{30}$$

for all $b \in \mathcal{B}$. Due to Inequality (28) and $\beta_{b,l} = 0$ there is no variable $x_{b,l,m}$ with $l \in \mathcal{L}'\backslash\zeta(b)$ having negative reduced costs. The same is true for all lot-types $l \in \mathcal{L}\backslash L'$ and all branches $b \in \mathcal{B}$ due to the selection of $\beta_{b,l}$ in Equation (30).

Next we remark that Inequality (19) remains valid after appending some dual values $\beta_{b,l} = 0$ for lot-types $l \in \mathcal{L}'$. From Inequality (29) we conclude $\sum_{b\in\mathcal{B}} \beta_{b,l} \leq \pi$ for all $l \in \mathcal{L}\backslash\mathcal{L}'$. Since $\{i \in \mathcal{I} \mid l \in \mathcal{C}_i\} = \emptyset$ this is equivalent to Inequality (19).

Thus by using Equation (30) we have constructed a feasible solution of the dual master problem without changing the objective value. Since the primal solution of the (RMP) can be extended to a feasible solution of (MP) having the same objective value, by setting the missing variables to zero, the current optimal solution of the (RMP) is also optimal for the (MP). □

In other words this means that we can restrict ourselves onto (25) to price out new variables. The first part of the proposed pricing algorithm loops over all branches $b \in \mathcal{B}$ and performs the following checks:

(1) For all $l \in \zeta(b)$ let $\mathcal{F}_l$ be the set of all multiplicities $m \in \mathcal{M}\backslash\eta(b,l)$ satisfying

$$\alpha_b - \beta_{b,l} + m \cdot |l| \cdot (u - v) > c_{b,l,m}. \tag{31}$$

If $\mathcal{F}_l$ is not empty let $\tilde{m}$ be an element with smallest distance to the elements of $\eta(b,l)$. We then add $\tilde{m}$ to $\eta(b,l)$, i.e., we generate the variable $x_{b,l,\tilde{m}}$.

(2) For all $l \in \mathcal{L}'\backslash\zeta(b)$ we perform the same test as in (1) while remarking that here we have $\eta(b,l) = \emptyset$. If there exists a pair (l,m) satisfying Inequality (31), then we choose $\hat{m} \in \mathcal{M}$ such that $c_{b,l,\hat{m}} \leq c_{b,l,m}$ for all $m \in \mathcal{M}$. We then add l to $\zeta(b)$ and $\hat{m}$ to $\eta(b,l)$, i.e., we generate the variable $x_{b,l,\hat{m}}$.

In step (3) we solve $\min_{\mathcal{L}\backslash\mathcal{L}'} \sum_{b\in\mathcal{B}} \min\left(0, \min_{m\in\mathcal{M}} c_{b,l,m} - \alpha_b - m \cdot |l| \cdot (u - v)\right)$ slightly adapting the algorithm given in Subsection 4.1, i.e. instead of considering costs of a single branch we sum over the costs of all branches. Let $\hat{l}$ be the optimal solution with objective value κ. If $\kappa > -\pi$, then we we add $\hat{l}$ to $\mathcal{L}'$, i.e. we generate the variable $y_{\hat{l}}$. For each $b \in \mathcal{B}$ we determine the optimal multiplicity $\hat{m}$, see Subsection 4.2, with respect to $\hat{l}$. If $c_{b',\hat{l},\hat{m}} - \alpha_{b'} - \hat{m} \cdot |\hat{l}| \cdot (u - v) < 0$ we add $\hat{l}$ to $\zeta(b')$ and $\hat{m}$ to $\eta(b',\hat{l})$.

Lemma 2. *If the pricing algorithm does not return a new column, then the current optimal solution of (RMP) is optimal for (MP).*

Proof. Let us assume that the pricing algorithm does not return a new variable. Due to step (1) we have $c_{b,l,m} - \alpha_b - m \cdot |l| \cdot (u - v) + \beta_{b,l} \geq 0$ for all $b \in \mathcal{B}$ and all $l \in \zeta(b)$. Similarly we have $c_{b,l,m} - \alpha_b - m \cdot |l| \cdot (u - v) \geq 0$ for all $l \in \mathcal{L}'\backslash\zeta(b)$ due to step (2). From step (3) we conclude Inequality (29). Due to the construction of the cover cuts we also have $\cup_{i\in\mathcal{I}} \mathcal{C}_i \subseteq \mathcal{L}'$ so that we can apply Lemma 1. □

4.7 Initial set of columns

As the behavior of a column generation algorithm depends on a suitable selection of the initial columns, we propose an initial variable selection in this subsection.

For each branch b we compute the three (locally) best fitting lot-types, see Subsection 4.1, and add them to ζ_b. Additionally we add all lot-types used in the best heuristically found solution of the algorithm sketched in Subsection 4.8. The initial set $\mathcal{L}'$ of lot-types is then given by $\cup_{b\in\mathcal{B}}\zeta(b)$. For each branch $b \in \mathcal{B}$ and each lot-type $l \in \zeta(b)$ we compute the corresponding optimal multiplicity $\hat{m}$, see Subsection 4.2, and set $\eta(b,l) = \{\hat{m}-1, \hat{m}, \hat{m}+1\} \cap \mathcal{M}$.

4.8 A starting heuristic

In [5] the authors have proposed the so-called Score-Fix-Adjust heuristic for the SLDP. Here we briefly describe the idea. For each branch we determine the three best fitting lot-types, see Subsection 4.1, and add a score of 100 to the best fitting lot-type, a score of 10 to the second best fitting lot-type and a score of 1 to the third best fitting lot-types. (Of course this can be generalized to the first t best fitting lot-types and different scoring schemes.) With this we have implicitly assigned a score to each lot-type $l \in \mathcal{L}$, where most of the lot-types obtain the score zero. We can extend this scoring to the k-subsets of $\mathcal{L}$ by summing up the individual scores so that we implicitly get an order of the $\binom{|\mathcal{L}|}{k}$ many feasible lot-type combinations. With this we traverse the k-subsets of $\mathcal{L}$ in descending order, where ties are broken arbitrarily. (This can be done without explicitly generating all such subsets beforehand.) In the fixing step we assume that the applicable lot-types are restricted to the current k-subset of $\mathcal{L}$. Now we are in the situation of Subsection 4.3, i.e., we have SLDP-k, and proceed by starting with a locally optimal assignment of the lot-types and multiplicities, which then is adjusted until it fits the global cardinality constraints (5). Remark: The adjustment step presented in Subsection 4.3 is an improved variant of the one in [5].

5 Computational results

In this section we compare our proposed branch-and-price algorithm, see Table 4[1], with the ILP model from Section 3 solved directly using ILOG CPLEX 12.1.0. The key parameters of some selected problem instances are summarized in Table 1. Although we consider only very few examples we have tried to include the whole range from rather small to very large problem instances. In Table 2 we demonstrate that the number of variables and constraints explodes as the problem size increases. To support our assertion that the formulation is really tight we have included the integrality gaps.

One can formulate an alternative compact formulation of the SLDP, see Appendix B for the details. Since its performance turns out to be really bad we do

[1] In the Instance 1 marked with a "⋆" we have not used the variables of the optimal solution found by the SFA heuristic.

Instance	1	2	3	4	5
k	3	5	5	4	5
$\lvert\mathcal{B}\rvert$	10	10	1303	1328	682
$\lvert\mathcal{S}\rvert$	4	4	4	7	12
$\lvert\mathcal{M}\rvert$	3	3	3	3	3
$\min_c$	0	0	0	1	0
$\max_c$	2	5	5	3	5
$\min_t$	4	3	3	7	12
$\max_t$	8	15	15	14	30
$\lvert\mathcal{L}\rvert$	50	1211	1211	1290	1 159 533 584
$\overline{I}$	66	66	12 100	9898	16 200
$\underline{I}$	54	54	11 900	9702	15 500

Table 1. Key parameters for some selected problem instances.

Instance	1	2	3	4	5
# variables	1550	37 541	3 634 211	5 140 650	2 373 565 246 448
# constraints	513	12 123	1 212 003	1 714 451	790 801 904 973
CPU time [min]	0	0	27	36	N/A
objective	11.479	11.367	1482.952	6096.536	N/A
LP relaxation	11.479	11.367	1482.952	6095.525	N/A
integrality gap	0%	0%	0%	0.017%	N/A

Table 2. Performance of the ILP model (Section 3). Instance 5 could not be solved.

Instance	1	2	3	4	5
CPU time [s]	1	1	1	1	2
successful node	10	12	648	180	463

Table 3. Performance of the SFA heuristic from Subsection 4.8.

Instance	$1^\star$	2	3	4	5
CPU time [s]	1	1	2	4	937
# variables (initial LP)	74	76	20 952	7975	13 129
# constraints (initial LP)	43	43	7944	5315	6642
# variables (final LP)	76	76	20 952	9937	64 877
# constraints (final LP)	44	43	7944	6248	32 183
# cover cuts	0	0	0	2	4
# pricing steps	2	1	1	4	141

Table 4. Performance of the column generation algorithm from Section 4.

not give the results except for Instance 1, the smallest one: This instance has 502 variables and 1204 constraints. (In general we have roughly $k \cdot |\mathcal{B}| \cdot |\mathcal{M}| \cdot |\mathcal{S}|$ instead of $|\mathcal{B}| \cdot |\mathcal{L}| \cdot |\mathcal{M}|$ variables.) The root relaxation of the compact model yields a trivial lower bound of zero; it takes 3 335 836 nodes and 97 minutes to solve this tiny example to optimality.

Table 3 underpins our assertion that the SLDP can be solved heuristically quite well, and the SLDP-k can be solved exactly fast: It takes less than 15 seconds to solve 100 000 SLDP-k instances, even for Instance 5.

6 Conclusion and future work

We have considered the stochastic lot-type design problem SLDP, which is an industrially relevant problem. We provided a tight ILP model of it. This model, however, has in many cases too many variables for solvers of the shelf. Thus, we presented a custom-made branch-and-price algorithm that is able to solve the SLDP exactly for real-world instances in preliminary computational results.

Currently we perform field experiments at our business partner to compare the deviation measure $\Delta(l, m)$ with explicit recourse costs for markdowns during the sales process. This results in a large-scale two-stage stochastic integer program with recourse, and the algorithm in this paper for the SLDP is needed for its solution as a subroutine, which is required to provide an optimality certificate.

References

1. P. Avella, A. Sassano, and I. Vasilyev. Computational study of large-scale p-median problems. *Math. Program.*, 109(1):89–114, 2007.
2. C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.*, 46(3):316–329, 1998.
3. Dimitri P. Bertsekas. *Dynamic Programming & Optimal Control, Vol. I.* Athena Scientific, 3rd edition, 2005.
4. M. Boccia, A. Sforza, C. Sterle, and I. Vasilyev. A cut and branch approach for the capacitated p-median problem based on Fenchel cutting planes. *J. Math. Model. Algorithms*, 7(1):43–58, 2008.
5. C. Gaul, S. Kurz, and J. Rambau. On the lot-type design problem. *Optim. Methods Softw.*, 25(2):217–227, 2010.
6. A. Klose and S. Görtz. A branch-and-price algorithm for the capacitated facility location problem. *Eur. J. Oper. Res.*, 179(3):1109–1125, 2007.
7. M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Oper. Res.*, 53(6):1007–1023, 2005.
8. M. E. Lübbecke and J. Desrosiers. *Branch-Price-and-Cut Algorithms.* John Wiley & Sons, Inc., 2011.
9. F. Vanderbeck. Branching in branch-and-price: a generic scheme. *Math. Program., Ser. A*, 130(2):249–294, 2011.

A Parameterizable sets of applicable lot-types

We call the set $\mathcal{L}$ of applicable lot-types *parameterizable* if it can be described as the set of all vectors $(l_s)_{s\in\mathcal{S}} \in \mathbb{N}^{|\mathcal{S}|}$ satisfying $\min_c \le l_s \le \max_c$ for all $s \in \mathcal{S}$ and $\min_t \le \sum_{s\in\mathcal{S}} l_s \le \max_t$. Here, $\min_c$, $\min_t$, $\max_c$, $\max_t$ are integers with $\min_c \le \max_c$, $\min_t \ge |\mathcal{S}| \cdot \min_c$, $\max_t \le |\mathcal{S}| \cdot \max_c$, and $\min_t \le \max_t$.

A parameterizable set of lot-types is a practically relevant case: By setting $\min_c = 1$ we can enforce that each branch is supplied in each size with at least one item, a requirement which legally arises for advertised products. Since the main advantage of using lot-types lies in the reduction of the number of picks in the central warehouse, we should guarantee, that this effect does not dwindle away by selecting lot-types with too few items, which can be controlled by a suitable value for $\min_t$. There are practical reasons for the parameter $\max_t$, too: combine too many winter coats in a lot would cause serious handling problems.

For a parameterizable set of applicable lot-types, we can of course describe a very large set $\mathcal{L}$ of applicable lot-types by few numbers. The formula for the number $\tau\left(\min_c, \max_c, \min_t, \max_t, |\mathcal{S}|\right)$ of resulting lot-types given the mentioned parameters is given in the following. First, we observe that we have

$$\tau\left(\min_c, \max_c, \min_t, \max_t, |\mathcal{S}|\right) = \tau\left(0, \max_c - \min_c, \min_t - \Lambda, \max_t - \Lambda, |\mathcal{S}|\right), \quad (32)$$

where $\Lambda = \min_c \cdot |\mathcal{S}|$. Since the number of non-negative integer solutions of $\sum_{i=1}^{r} x_i = n$ equals $\binom{n+r-1}{r-1}$, we can use the inclusion-exclusion principle to obtain

$$\tau\left(0, a, b, c, d\right) = \sum_{n=b}^{c} \sum_{i=0}^{\min\left(\left\lfloor \frac{n}{a+1} \right\rfloor, d\right)} (-1)^i \binom{d}{i} \cdot \binom{n - i(a+1) + d - 1}{d-1}, \quad (33)$$

where the index i counts the number of sizes s with $l_s \ge a + 1$. As examples we state $\tau(1, 5, 5, 15, 5) = 1753$ and $\tau(0, 5, 12, 30, 12) = 1\,159\,533\,584$[2].

This shows that we can face very large sets of applicable lot-types in the SLDP. A straight-forward attempt to by-pass the problem is presented in the next section.

B A compact model for parameterizable sets of lot-types

In the special case of a parameterizable set of lot-types we can model the SLDP with fewer variables:

$$\min \qquad \sum_{b\in\mathcal{B}} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \delta_{b,s}^{a}$$

[2] For t-shirts or sweaters the typically number of offered sizes varies between 4 and 6, but for, e.g., lingerie or children's clothing 12 or even more different sizes may occur.

$$
\begin{aligned}
s.t. \quad \sum_{i\in\mathcal{K}}\sum_{m\in\mathcal{M}} x_{b,i,m} &= 1 && \forall b\in\mathcal{B}\\
v_{b,s,i,m} - max_c\cdot x_{b,i,m} &\le 0 && \forall (b,s,m,i)\in\mathcal{U}\\
v_{b,s,i,m} - l_{i,s} &\le 0 && \forall (b,s,m,i)\in\mathcal{U}\\
v_{b,s,i,m} - l_{i,s} - max_c\cdot x_{b,i,m} &\ge -max_c && \forall (b,s,m,i)\in\mathcal{U}\\
\underline{I} \le \sum_{b\in\mathcal{B}}\sum_{s\in\mathcal{S}}\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{K}} m\cdot v_{b,s,i,m} &\le \overline{I}\\
l_{i,s} &\le max_c && \forall i\in\mathcal{K}, s\in\mathcal{S}\\
l_{i,s} &\ge min_c && \forall i\in\mathcal{K}, s\in\mathcal{S}\\
\sum_{s\in\mathcal{S}} l_{i,s} &\le max_t && i\in\mathcal{K}\\
\sum_{s\in\mathcal{S}} l_{i,s} &\ge min_t && \forall i\in\mathcal{K}\\
\delta^a_{b,s} + \sum_{i\in\mathcal{K}}\sum_{m\in\mathcal{M}} m\cdot v_{b,s,i,m} &\ge d^a_{b,s} && \forall b\in\mathcal{B}, s\in\mathcal{S}, a\in\mathcal{A}\\
\delta^a_{b,s} - \sum_{i\in\mathcal{K}}\sum_{m\in\mathcal{M}} m\cdot v_{b,s,i,m} &\ge -d^a_{b,s} && \forall b\in\mathcal{B}, s\in\mathcal{S}, a\in\mathcal{A}\\
x_{b,i,m} &\in \{0,1\} && \forall b\in\mathcal{B}, i\in\mathcal{K}, m\in\mathcal{M}\\
v_{b,s,i,m} &\ge 0 && \forall (b,s,m,i)\in\mathcal{U}\\
l_{i,s} &\in \mathbb{Z} && \forall i\in\mathcal{K}, s\in\mathcal{S}\\
\delta^a_{b,s} &\ge 0 && \forall b\in\mathcal{B}, s\in\mathcal{S}, a\in\mathcal{A},
\end{aligned}
$$

where we use the abbreviations $\mathcal{K} := \{1,\dots,k\}$ and $\mathcal{U} := \mathcal{B}\times\mathcal{S}\times\mathcal{M}\times\mathcal{K}$. We utilize the binary variable $x_{b,i,m}$ to model the assignment of the lot-type and multiplicity to a certain branch b, i.e., we have $x_{b,i,m} = 1$ if and only if branch b is supplied with lot-type i in multiplicity m. Of course, for each branch b only one $x_{b,i,m}$ is one. In order to incorporate the bounds on the total number of supplied items we utilize the auxiliary variables $v_{b,s,i,m}$. For a given branch b and size s we set $v_{b,s,i,m} = l_{i,s}\cdot x_{b,i,m}$, i.e. $v_{b,s,i,m} = l_{i,s}$ if branch b is supplied with lot-type i in multiplicity m and $v_{b,s,i,m} = 0$ otherwise. The linearization of this non-linear equation is quite standard using suitable big-M constants. The deviations $\delta^a_{b,s}$ then are given by

$$\delta^a_{b,s} = \left| d^a_{b,s} - \sum_{i=1}^{k}\sum_{m\in\mathcal{M}} m\cdot v_{b,s,i,m} \right|.$$

Again the linearization of this non-linear equation is standard.[3]

[3] We would like to remark that one can express the deviations $\delta^a_{b,s}$ also using the binary assignment variables $x_{b,i,m}$ instead of the item counts $v_{b,s,i,m}$. In this case we have to replace the two inequalities containing $\delta^a_{b,s}$ by $\delta^a_{b,s} + m\cdot l_{i,s} - d^a_{b,s}\cdot x_{b,i,m} \ge 0$ and $\delta^a_{b,s} - m\cdot l_{i,s} + m\cdot max_c\cdot(1-x_{b,i,m}) \ge -d^a_{b,s}$ for all $(b,s,m,i)\in\mathcal{U}, a\in\mathcal{A}$. Or

Since the symmetric group on k elements acts on the k different lot-types one should additionally destroy the symmetry in the stated ILP formulation. This can be achieved by assuming that the lot-types $(l_{i,s})_{s\in\mathcal{S}}$ are lexicographically ordered. For the ease of notation we assume that the set of sizes $\mathcal{S}$ is given by $\{1,\ldots,s\}$. As an abbreviation we set $t := \max_c - \min_c + 1$. With this the additional inequalities

$$\sum_{j=1}^{s}\left(l_{i,j}-\min_c\right)\cdot t^{s-j} \geq 1+\sum_{j=1}^{s}\left(l_{i+1,j}-\min_c\right)\cdot t^{s-j} \quad \forall 1\leq i\leq k-1,$$

which are equivalent to $\sum_{j=1}^{s}(l_{i,j}-l_{i+1,j})\cdot t^{s-j}\geq 1$ for all $1\leq i\leq k-1$, will do the job. We remark that using some additional auxiliary variables a numerical stable variant can be stated easily. (This becomes indispensable if t^s gets too large.)

We remark that the LP relaxation of the compact model yields large integrality gaps [4]. The typical approach would now be to solve a Dantzig-Wolfe decomposition of the compact model by dynamic column generation, leading to a master problem very similar to the SLDP model we presented in the first place. And column generation on that SLDP model is what we proposed in the paper.

C A worked example

In order to demonstrate the proposed algorithm we consider a small example consisting of ten branches, four sizes, and one scenario. As elements of these sets we use the first non-negative integers, i.e., we set $\mathcal{B}=\{1,\ldots,10\}$, $\mathcal{S}=\{1,\ldots,4\}$, and $\mathcal{A}=\{1\}$. In Table 5 we give the demands.

The set of applicable lot-types $\mathcal{L}$ is parameterized by $\min_c = 0$, $\max_c = 2$, $\min_t = 4$, $\max_t = 8$, see Table 6 for the exhaustive list. As possible multiplicities we allow $\mathcal{M}=\{1,2,3\}$. The number of simultaneously usable lot-types is restricted to $k=3$. The bounds on the total given supply are given by $\underline{I}=54$ and $\overline{I}=66$.

The scoring of the SFA heuristic outlined in Subsection 4.8 is presented in Table 7. In node number ten it found the lot-type selection $\{15,27,49\}$ achieving a target value of 11.479. The ILP formulation from Section 3 with its 1550 variables and 513 constraints can be utilized to prove the optimality of the corresponding solution.

The initial set of columns for the proposed algorithm, see Subsection 4.7 is given by the three best-fitting lot-types per branch, see Table 8.

---

we may use both sets of inequalities. It turns out that this alternative formulations is solved slightly faster on some inspected instances.

[4] Assuming some technical conditions on the demands, the four parameters for the lot-tyes, and the applicable multiplicities, which are not too restricting, one can construct a solution of the LP relaxation having an objective value of zero. E.g. we assume $\min_c \leq d^a_{b,s} \leq \max_c \cdot \max\{m\in\mathcal{M}\}$

branch b	$d^1_{b,1}$	$d^1_{b,2}$	$d^1_{b,3}$	$d^1_{b,4}$
1	1.57854	1.66130	1.63925	1.40429
2	1.55737	1.82154	2.13340	1.75363
3	1.25112	1.31302	1.48253	1.46916
4	1.79848	2.32990	1.80533	1.84418
5	1.49304	2.02603	1.49577	1.35903
6	0.82655	1.48188	1.54487	1.07092
7	0.85965	0.96385	1.26121	1.32116
8	2.01761	2.22984	2.17453	2.22337
9	0.90348	0.74222	0.60461	0.73267
10	1.14625	1.60841	1.52226	1.54775

Table 5. Demands per branch, size, and scenario.

no.	l_1	l_2	l_3	l_4	no.	l_1	l_2	l_3	l_4	no.	l_1	l_2	l_3	l_4	no.	l_1	l_2	l_3	l_4
1	2	2	0	0	14	2	0	1	1	27	2	2	2	1	40	1	2	1	2
2	2	1	1	0	15	1	1	1	1	28	2	0	0	2	41	2	2	1	2
3	1	2	1	0	16	2	1	1	1	29	1	1	0	2	42	0	0	2	2
4	2	2	1	0	17	0	2	1	1	30	2	1	0	2	43	1	0	2	2
5	2	0	2	0	18	1	2	1	1	31	0	2	0	2	44	2	0	2	2
6	1	1	2	0	19	2	2	1	1	32	1	2	0	2	45	0	1	2	2
7	2	1	2	0	20	1	0	2	1	33	2	2	0	2	46	1	1	2	2
8	0	2	2	0	21	2	0	2	1	34	1	0	1	2	47	2	1	2	2
9	1	2	2	0	22	0	1	2	1	35	2	0	1	2	48	0	2	2	2
10	2	2	2	0	23	1	1	2	1	36	0	1	1	2	49	1	2	2	2
11	2	1	0	1	24	2	1	2	1	37	1	1	1	2	50	2	2	2	2
12	1	2	0	1	25	0	2	2	1	38	2	1	1	2					
13	2	2	0	1	26	1	2	2	1	39	0	2	1	2					

Table 6. List of all applicable lot-types.

score	l_1	l_2	l_3	l_4	no.
331	1	1	1	1	15
301	2	2	2	2	50
112	1	1	2	1	23
102	1	2	2	2	49
101	2	2	2	1	27
100	1	2	1	1	18
31	1	2	2	1	26
11	1	1	1	2	37
10	2	1	1	1	16
10	1	2	1	2	40
1	2	2	1	1	19

Table 7. Lot-types with non-zero scores.

branch	lot-types
1	27,26,50
2	15,50,49
3	15,23,37
4	15,50,49
5	18,26,19
6	23,26,15
7	15,37,23
8	15,50,49
9	15,16,23
10	49,40,26

Table 8. Locally best fitting lot-types per branch.

Solving this linear program consisting of 74 variables and 43 constraints ends up in an optimal target value of 11.507. The non-zero dual multipliers are given by $\alpha_1 = 1.620900$, $\alpha_2 = 1.000600$, $\alpha_3 = 1.515700$, $\alpha_4 = 0.881200$, $\alpha_5 = 1.424500$, $\alpha_6 = 1.271000$, $\alpha_7 = 0.758820$, $\alpha_8 = 0.740600$, $\alpha_9 = 1.017030$, $\alpha_{10} = 1.563300$, $\pi = 0.095400$, $\beta_{1,27} = 0.095400$, $\beta_{3,23} = 0.005800$, $\beta_{5,18} = 0.050800$, $\beta_{5,26} = 0.042200$, $\beta_{5,19} = 0.036800$, $\beta_{6,23} = 0.089600$, $\beta_{6,26} = 0.053200$, $\beta_{8,15} = 0.095400$, $\beta_{8,50} = 0.095400$, $\beta_{10,49} = 0.095400$, and $\beta_{10,40} = 0.051000$.

The pricing algorithm from Subsection 4.6 then finds out that lot-type 27 should be added to branch 5. So the variables $x_{5,27,1}$ and $x_{5,27,2}$ are generated (y_{27} already exists). The optimal objective value of the resulting linear program consisting of 76 variables and 44 constraints is given by 11.479. Using the corresponding dual multipliers and the pricing algorithm proves the optimality of this solution (which is by chance integral).

Instead of pricing out new variables one might have also tried to cut off the optimal solution. Since the initial linear program has the following optimal non-zero y_l-variables: $y_{15} = 1$, $y_{26} = 1$, $y_{27} = 1$, we compute the optimal solution of the SLDP-k with $\mathcal{L} = \{15, 26, 27\}$, see Subsection 4.3, and add the cover cut $y_{15} + y_{26} + y_{27} \leq 2$ afterwards. The resulting linear program has an optimal solution with $y_{15} = 1$, $y_{18} = \frac{1}{2}$, $y_{26} = \frac{1}{2}$, $y_{27} = \frac{1}{2}$, $y_{49} = \frac{1}{2}$ and an objective value of 11.561. By solving the SLDP-k for all ten three-element subsets of $\{15, 18, 26, 27, 49\}$ we can add the cover cut $y_{15} + y_{18} + y_{26} + y_{27} + y_{49} \leq 2$ afterwards. The resulting linear program has an optimal objective value of 11.585. Adding the next six-element cover results in an objective value of 11.663. In this small example the primal-dual bounds are too weak to prevent the algorithm from generating at least one additional column.