Computers & Industrial Engineering 121 (2018) 150–160

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Distribution product packaging to maximize net revenue

Xiubin Bruce Wang[a,*], Yihua Li[b], Luca Quadrifoglio[a], Kai Yin[c]

[a] Zachry Department of Civil Engineering, Texas A&M University, College Station, TX 77843-3136, United States
[b] United Airlines, Willis Tower, 233 South. Wacker Drive, Chicago, IL 60606, United States
[c] HomeAway, Austin, TX 78727, United States

ARTICLE INFO

Keywords:
Packaging
Product distribution
Column generation

ABSTRACT

This paper studies a problem in which a distribution center of a chain store packages a small subset of a large number of available products (e.g. DVDs) to distribute to its local stores. Only a limited number of different packages are allowed. We determine what products are in each package and what package to distribute to each store for revenue maximization. A column generation method is developed in which the sub-problem generates candidate packages and corresponding stores to serve while the master problem determines the final packages on the production lines. Two heuristic methods are proposed to generate the candidate packages for the sub-problem. Bounds are derived for both the optimal number of packages and the total revenue in order to expedite the solution. The algorithm shows great promise with operational data from a chain store that serves several thousand retail locations.

1. Introduction

In the retail of large chain stores, similar products are typically packaged together at distribution centers (DC) to distribute to local stores. However, the limited space at retail stores and the large number of substitutable products available at a DC require only the most profitable products to be packaged and distributed. In addition, the number of package production lines, each for a different package, is subject to a limit due to space availability and the economies of scale in packaging, which often leads to several stores served by an identical package. Obviously, the packaging and distribution decision is complex in order to maximize the chain store's profitability.

This paper considers a chain store distribution center (DC) that has a set of DVD titles available from entertainment companies at the beginning of the sales season. DVDs of different titles are packed before being distributed to its local stores for sale. According to the industry convention, a title combination is referred to as a Plan-O-Gram later (or POG). Due to the shelf display capacity, a POG may not have more than N different titles in it. Each local store has a market for DVD titles. At a store, different titles may bring in different revenues. And the same title may have different revenues at different stores. In this problem, exactly one POG package is needed at each local retail store according to the identical fixture (e.g. shelf) for title display. Each POG has a constant setup cost c for the packaging line accounting for labor, capital and lot size at the DC. In addition, the number of POGs p must be within a range $[p_{\min}, p_{\max}]$ for practical considerations. At the end of the sales season, unsold DVDs are collected back to the DC at no additional cost. For simplicity, we assume that this problem does not replenish inventory during the sales season or equivalently, the replenish cost is assumed to be zero. The objective here is to decide the number of POGs, the titles in each of them, and the stores each POG serves in order to maximize the overall sales revenue less packaging cost.

In this problem, the demand at a particular retail store i is specified by a quantity for each title k at a preset retail price, $D_i = (d_{i1}, d_{i2}, \ldots, d_{ik})$. The demand can also be equivalently represented by its revenue $R_i = (r_{i1}, r_{i2}\ldots, r_{ik})$, where $r_{ik} = f(d_{ik})$, for all$i$, k. This paper adopts the latter. Furthermore, we assume that demand for one title is independent of another title's availability. This assumption appears restrictive, but is necessary in subsequently developing efficient algorithms. Because of the large number of available titles, the ones in a POG are most likely in different DVD categories. Therefore, the final solution turns out generally reasonable and acceptable. This independence assumption particularly applies to the situation in which products are from different categories and are not substitutable. In addition, only the packaging setup cost is considered. In comparison, shipping cost is not considered in this problem for two reasons. First, a package always needs to be shipped to each retail store regardless of its content. Secondly, the shipment is typically conducted with large volumes of other products, and has a negligible marginal cost. Naturally, there are a large number of potential combinations for POG. Ideally, each individual retail store would have a POG package of its most desired titles to maximize the revenue if other stores were not present.

* Corresponding author at: ROOM 303C, DLEB Building, 3136 TAMU, College Station, TX 77843, United States.
E-mail addresses: bwang@tamu.edu (X.B. Wang), yihua.li@united.com (Y. Li).

https://doi.org/10.1016/j.cie.2018.05.015
Received 27 December 2017; Received in revised form 16 April 2018; Accepted 10 May 2018
Available online 22 May 2018
0360-8352/ Published by Elsevier Ltd.

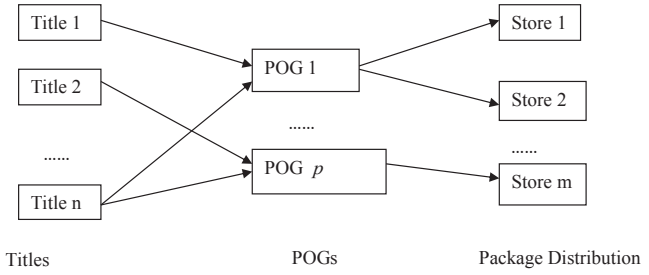

Fig. 1. An example DVD packaging and distribution process.

But, due to the packaging setup cost, only a (much) smaller number of packages will be chosen, a number much smaller than the number of titles. Fig. 1 illustrates the packaging and distribution process. As in Fig. 1, a combination of titles corresponds to one POG. And one POG serves one or more stores. Each store receives a particular POG package.

By background, a fixture at a local store is a special shelf for DVD title display. In one of our studies, a fixture has a capacity of displaying about 180 different titles. In this case, a serving POG fills the fixture perfectly up to this capacity when revenue is maximized. Depending on the shelf depth of a fixture, each title in a POG package can have a number of DVD duplications. As the shelf depth is exogenous to this problem, it suffices to only decide the titles in each POG.

Product packaging and distribution is a popular practice. Large chain stores such as War-Mart, Target and Best Buy decide their POGs and make pricing decisions about every 3 months according to demand updates. Although this problem is significant, we have not seen any other study directly related to this particular packaging and distribution problem.

Research on product packaging is traditionally about determination of packaging materials, package size, weight and other characteristics to balance cost with safety and handling efficiency (see Chan, Chan, & Choy, 2006, for example). One of the traditional functions of product packaging therein is product promotion. In another study, Prendergast and Pitt (1996) consider packaging for physical movement and environmental impact. In contrast, our paper determines bundling of substitutable products for distribution, a critical link being missing in the traditional retail product distribution. In literature, the product storage location assignment problem that determines the location for each product in a warehouse to minimize the total order picking distance appears inherently similar to our problem (see Hausman, Schwarz, & Graves, 1976; Muppani & Adil, 2008; Rosenblatt & Eynan, 1989, for examples). Both problems essentially involve two inter-related decisions: product grouping and location/space allocation, in which the location/space decision is about deciding the size of location space. However, the two are not analytically equivalent.

This problem challenges both modeling and solution development. The solution can be considered as one that pools multiple titles/products to serve a set of stores, or alternatively, that pools stores to sell identical POG packages. The number of store groups, each served by an identical POG, and the number of POGs, needs to balance with the POG setup cost and potential market revenues. A computationally efficient algorithm is necessary becausethe large number of potential combinations of titles and stores leads to a prohibitive complexity.

We propose a formulation and subsequently a column generation method for this problem. The column generation method dated back to Dantzig and Wolfe (1960). Lubbecke and Desrosiers (2005) provide a complete overview of literature on mixed integer column generation. Vanderbeck (2005) specifically provides a tutorial. In this paper, we propose two heuristics for generating columns (POGs), namely Bottom-Up (BU) and Top-Down (TD). Our numerical test shows clear advantage of the TD over the BU method. Therefore the TD method is adopted to generate POGs. In the iteration process, we develop bounds for the optimal number of POGs using results from both the TD and BU methods. The bounds tighten quickly with iterations. We further propose a simple search method based on a concavity property to find the optimal number of POGs. With operational data from a large chain store, the solution algorithm is tested to be satisfactory.

Table 1
Revenue matrix.

Store No	Title							
	1	2	3	4	5	6	7	8
1	$2000	$1000	$1000	$1000	$1000	$1000	$2000	$1000
2	$1000	$2000	$1000	$1000	$1000	$2000	$1000	$1000
3	$1000	$1000	$2000	$1000	$2000	$1000	$1000	$1000
4	$1000	$1000	$1000	$2000	$1000	$1000	$1000	$2000

The organization of this paper is as follows. Section 2 provides an illustrative example to facilitate understanding of the problem. In Section 3, a problem formulation is presented and is decomposed using the column generation method. Two algorithms are presented to solve the non-linear sub-problem. In Section 4, a flow chart of the solution algorithm is introduced. Some structural properties are studied in Section 5 followed by numerical tests based on large scale production data. Section 6 concludes this paper with a summary of major findings and remaining challenges.

2. An illustrative example

We present an example to illustrate the problem and solution. Assume an identical fixture at 4 stores. The fixture allows 4 titles in a POG, e.g., $N = 4$, selected from 8 DVD titles. The expected revenues of each title at the stores are shown in Table 1 below with a packaging cost c = $500 for each POG.

If we require the number of POGs $p = 1$, an optimal solution is to pack Title 1 through 4 in the POG to serve all the four stores, as in Table 2. The total revenue is $20,000.

At $p = 2$, the two POGs in an optimal solution *(T1, T4, T7, T8) and (T2, T3, T5, T6)* as in Table 3 serve stores (*S1, S4)* and (*S2, S3)* respectively with a total revenue of $24,000. The solution can be represented as below.

Triplet 1: ($12,000, (S1, S4), (T1, T4, T7, T8))
Triplet 2: ($12,000, (S2, S3), (T2, T3, T5, T6))

We define a triplet as a vector of a revenue, a set of stores served, and a set of titles in a POG. The revenue results from distributing the set of titles to the stores. More introduction to triplet is available in Section 3.

At $p = 3$, the solution is three triplets as in Table 4 with a total revenue of $24,000.

Triplet 1: ($12,000, (S1, S4), (T1, T4, T7, T8))
Triplet 2: ($6000, (S2), (T2, T3, T5, T6))
Triplet 3: ($6000, (S3),(T2, T3, T5, T6))

At $p = 4$, an optimal solution as in Table 5 has four triplets with a revenue of $24,000.

Triplet 1: ($6000, (S1), (T1, T4, T7, T8))
Tripelt 2: ($6000, *(S2), (T2, T3, T5, T6)*)
Triplet 3: ($6000, *(S3), (T2, T3, T5, T6)*)
Triplet 4: ($6000, *(S4), (T1, T4, T7, T8)*)

If each POG has a setup cost $500, the above example has an optimal solution at $p = 2$ with a profit being "$"24, 000−2 × 500 = "$"23, 000.

Table 2
Optimal Solution at $p = 1$.

POG	Store	Title								
		1	2	3	4	5	6	7	8	Total
1	1	***$2000***	*$1000*	*$1000*	*$1000*	$1000	$1000	$2000	$1000	$5000
	2	*$1000*	***$2000***	*$1000*	*$1000*	$1000	$2000	$1000	$1000	$5000
	3	*$1000*	*$1000*	***$2000***	*$1000*	$2000	$1000	$1000	$1000	$5000
	4	*$1000*	*$1000*	*$1000*	***$2000***	$1000	$1000	$1000	$2000	$5000

3. Problem formulation and solution

First, we define the following notation.

- x_{ijk} indicator variable which equals to 1 if POG k containing title j serves store i, equals to 0, otherwise.
- z_k indicator variable which equals to 1 if POG k serves at least one store and contains one title; equals to 0, otherwise. It indicates whether a production line for package k is needed.
- S set of retail stores.
- P set of all possible POGs.
- p number of POGs. p^* is the optimal number.
- T set of titles.
- $p_{\min}$ minimum number of POGs, which is zero if not specified otherwise.
- $p_{\max}$ maximum number of POGs, which equals to $|S|$ if not specified otherwise. The min/max number p is pre-determined by the decision maker.
- N the number of different titles included in each POG.
- c_k packaging cost of POG k, which is assumed a constant c later if not specified otherwise.
- r_{ij} revenue of title j at store i.
- q_k total revenue in a triplet from distributing POG k.
- s_{ik} indicator variable which equals to 1 if store i is served by POG k; and 0, otherwise.
- t_{ik} indicator variable which equals to 1 if title i is assigned to POG k; and 0, otherwise.
- Ω_k the set of triplets (or POGs with associated stores) by the k^{th} iteration in the column generation (CG) procedure.
- θ_{ij} indicator variable. It is 1.0 if store i is in POG j, 0 otherwise.

Of these notations, the decision variables include x_{ijk}, z_k, s_{ik}. The rest are given parameters in the formulation.

3.1. Mathematical formulation

Assuming an exhaustive set of POGs available, the problem can be formulated as follows.

$$\text{Max} \sum_{i\in S}\sum_{j\in T}\sum_{k\in P} r_{ij}x_{ijk} - \sum_{k\in P} c_k z_k \tag{1.1}$$

Subject to:

$$\sum_{k\in P} s_{ik} = 1 \quad \forall\, i \in S \tag{1.2}$$

$$s_{ik} \leqslant \sum_{j\in T} x_{ijk} \quad \forall\, i \in S,\ \forall\, k \in P \tag{1.3}$$

$$s_{ik} \geqslant x_{ijk} \quad \forall\, i \in S,\ \forall\, j \in T,\ \forall\, k \in P \tag{1.4}$$

$$t_{jk} \leqslant \sum_{i\in S} x_{ijk} \quad \forall\, j \in T,\ \forall\, k \in P \tag{1.5}$$

$$t_{jk} \geqslant x_{ijk} \quad \forall\, i \in S,\ \forall\, j \in T,\ \forall\, k \in P \tag{1.6}$$

$$\sum_{j\in T} t_{jk} \leqslant N \quad \forall\, k \in P \tag{1.7}$$

$$z_k \leqslant \sum_{i\in S}\sum_{j\in T} x_{ijk} \quad \forall\, k \in P \tag{1.8}$$

$$z_k \geqslant x_{ijk} \quad \forall\, i \in S,\ j \in T,\ k \in P \tag{1.9}$$

$$\sum_{k\in P} z_k \geqslant p_{\min} \tag{1.10}$$

$$\sum_{k\in P} z_k \leqslant p_{\max} \tag{1.11}$$

$$x_{ijk} = 0,\ 1 \quad \forall\, i \in S,\ \forall\, j \in T,\ \forall\, k \in P \tag{1.12}$$

$$s_{ik} = 0,\ 1 \quad \forall\, i \in S,\ \forall\, k \in P \tag{1.13}$$

$$t_{jk} = 0,\ 1 \quad \forall\, j \in T,\ \forall\, k \in P \tag{1.14}$$

$$z_k = 0,\ 1 \quad \forall\, k \in P \tag{1.15}$$

The first term of the objective function is for the total gross revenue, the second term the total packaging cost associated with the number of POGs. Here x is a binary decision variable assuming a known exhaustive set of combinations (i, j, k). The constraint (1.2) requires each store be covered exactly once. The constraints (1.3) and (1.4) specify an incidence relationship to indicate whether a store is served by a POG. (1.3) is binding only when store i is not served by POG k, in which case $\sum_{j\in T} x_{ijk} = 0$ that forces $s_{ik} = 0$ via constraint (1.3); (1.4) is binding only when store i is served by POGk, in which case some $x_{ijk} = 1$, $\exists\, j$ that forces $s_{ik} = 1$ via constraint (1.4). The constraints (1.5) and (1.6) define a binary variable to determine if a title is included in a POG. (1.5) and (1.6) are defined in a similar fashion to constraints (1.3) and (1.4), respectively. The constraint (1.7) limits the number of titles in a POG. Similarly, the constraints (1.8) and (1.9) specify an incidence relationship to define if POG k is used. Constraint (1.8) is binding when for some k, $x_{ijk} = 0$, $\forall\ (i, j)$, which forces $z_k = 0$. Constraint (1.9) is binding when for some $\exists\ (i, j)$, $x_{ijk} = 1$, which forces $z_k = 1$, meaning

Table 3
Optimal Solution at $p = 2$.

POG	Store	Title								
		1	2	3	4	5	6	7	8	Total
1	1	***$2000***	$1000	$1000	$1000	$1000	$1000	***$2000***	*$1000*	$6000
	4	*$1000*	$1000	$1000	***$2000***	$1000	$1000	*$1000*	***$2000***	$6000
2	2	$1000	***$2000***	*$1000*	$1000	*$1000*	***$2000***	$1000	$1000	$6000
	3	$1000	*$1000*	***$2000***	$1000	***$2000***	*$1000*	$1000	$1000	$6000

Table 4
Optimal Solution at $p = 3$.

POG	Store	Title								
		1	2	3	4	5	6	7	8	Total
1	1	***$2000***	$1000	$1000	$1000	$1000	$1000	***$2000***	*$1000*	$6000
	4	*$1000*	$1000	$1000	***$2000***	$1000	$1000	*$1000*	***$2000***	$6000
2	2	$1000	***$2000***	*$1000*	$1000	*$1000*	***$2000***	$1000	$1000	$6000
3	3	$1000	*$1000*	***$2000***	$1000	***$2000***	*$1000*	$1000	$1000	$6000

package k is being produced. The constraints (1.10) and (1.11) specify the lower and upper bounds to the number of POGs.

For profit maximization, the number of titles in each POG always reaches its maximum N provided the title's respective revenue is positive. Therefore, the constraint (1.7) is always binding in the solution. In addition, this proposed formulation implies enumeration of POGs and store assignments. Considering the large number of stores (up to 3500, for example) and DVD titles (up to 5000), an exhaustive enumeration is a prohibitive task.

To illustrate the solution structure, we define an entity called triplet (r, **s**, **t**) according to each POG, where **t** is a vector of titles in a particular POG, **s** is a vector of stores that the POG serves, $r \in \mathbb{R}^1$is a scalar for the total revenue from distributing this POG package to stores **s**. In our solution later, we always associate a triplet with each POG. The decision becomes to develop a set of triplets so that each retail store is covered once and only once by a triplet to maximize the overall revenue less packaging cost. Fig. 2 shows a set of triplets as a feasible solution to a packaging and distribution problem, in which the three POGs are planned covering the five stores.

The final solution shall be a set of triplets, specifying a one to one correspondence between each store and a package. Interestingly, if we know the set of stores sharing the same POG, the set of titles in this POG would be easily determined by maximizing the total revenue. The way to identify this set of titles is first to ranking order all titles, each have its total revenue to the stores served. Then we take the top titles according to revenue to pack into the package. The following proposition clarifies this relationship.

Proposition 1. *A set of stores in a triplet is sufficient to decide the optimal revenue from the titles in the according POG assigned to serve them.*

In light of Proposition 1, in what follows, we can find that the set of stores in a triplet also indicate a unique optimal revenue and a set of titles selected, therefore a set of stores in a triplet may represent an entire triplet in a solution. This is because the optimal set of titles to include in the package serving the same group of stores can be easily determined once the group of stores is known, as explained earlier. Therefore, first grouping the stores before deciding the titles to them represents an effective way to solve the problem. Specifically, we will often mention inclusion or assignment of a store to a POG. This implies a slight misuse of the term POG according to its earlier definition, which initially refers to the POG as a group of titles.

Obviously, it is very important to group the stores together with similar demand characteristics. Therefore, a solution process shall provide a means to guiding clustering of the stores into groups, each group corresponding to a triplet.

We propose to use the column generation method. In what follows, we introduce both the master problem and the sub-problem for the initial formulation.

3.2. A Column generation approach

The formulation (1.1)–(1.15) is decomposed into a master and a sub-problem. The master problem selects the optimal set of POGs to maximize revenue less packaging cost, which is essentially a set covering problem with a given set of triplets. The sub-problem uses the store specific dual values from the master problem to generate new POGs, or equivalently, triplets. Subsequently, the candidate triplets are fed back into the master problem. This process repeats until the optimal solution is reached from the master problem, or until a stopping criterion is met. In this section, a POG is referred to as a set of stores with an according optimal revenue, which is equivalent to a triplet as explained earlier in the paper.

The master problem is formulated as follows.

3.2.1. The master problem

In the following formulation, we present the master problem.

$$\text{Max} \sum_{k\in\Omega_m} q_k z_k - c \sum_{k\in\Omega_m} z_k \tag{2.1}$$

Subject to:

$$\sum_{k\in\Omega_m} \theta_{ik} z_k = 1 \quad \forall\, i \in S \tag{2.2}$$

$$\sum_{k\in\Omega_m} z_k \geqslant p_{\min} \tag{2.3}$$

$$\sum_{k\in\Omega_m} z_k \leqslant p_{\max} \tag{2.4}$$

$$z_k = 0, 1 \qquad \forall\, j \in \Omega_m \tag{2.5}$$

Note that θ_{ij} is an incidence indicator between store i and POG j. It is 1.0 when POG j serves store i; 0, otherwise. It is a known (or given) parameter in the master problem. The objective is to maximize the total gross revenue minus the total packaging cost of selected POGs, the net profit. The constraint (2.2) corresponds to (1.2), and means each store must be covered exactly once. The constraints (2.3) and (2.4) correspond to (1.10) and (1.11), ensuring a lower and an upper bound to the number of POGs. In the iterative process introduced later, the linear relaxation of the master problem is solved so that a dual value for

Table 5
Optimal Solution at $p = 4$.

POG	Store	Title								
		1	2	3	4	5	6	7	8	Total
1	1	***$2000***	$1000	$1000	$1000	$1000	$1000	***$2000***	*$1000*	$6000
2	2	$1000	***$2000***	*$1000*	$1000	*$1000*	***$2000***	$1000	$1000	$6000
3	3	$1000	*$1000*	***$2000***	$1000	***$2000***	*$1000*	$1000	$1000	$6000
4	4	*$1000*	$1000	$1000	***$2000***	$1000	$1000	*$1000*	***$2000***	$6000

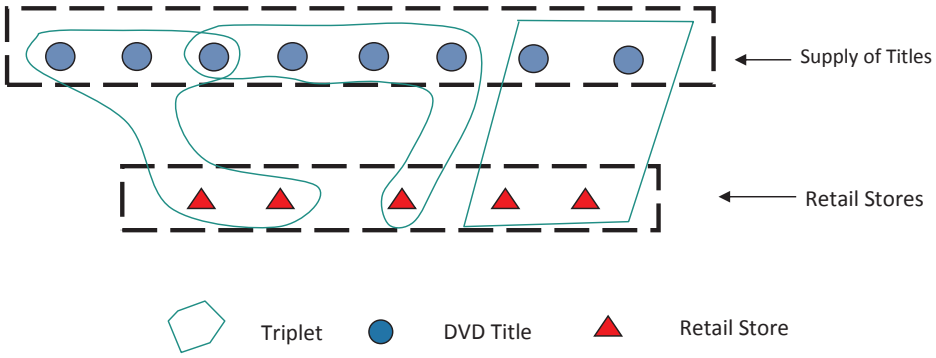

Fig. 2. Illustration of triplets and a feasible solution.

constraint (2.2) may be obtained.

3.2.2. Sub-problem (Column Generation)

We define $\widehat{r_{ij}}$ as the modified revenue of store i with title j, that is, $\widehat{r_{ij}} = r_{ij} - \pi_i$, where π_i is the dual value for store i from constraint (2.2) in the master problem. In some practical literature, the dual value of a constraint is also called shadow or bid price. In this case, one may interpret that the dual value associated to store i represents the locational potential for revenue. The sub-problem generates candidate POGs (sets of stores) with modified revenues for titles. Again, we take the POG as triplet equivalent.

$$\text{Max} \sum_{i \in S} \sum_{j \in T} \widehat{r_{ij}} x_{ijk} \tag{3.1}$$

Subject to:

$$s_{ik} \leqslant \sum_{j \in T} x_{ijk} \quad \forall\, i \in S \tag{3.2}$$

$$s_{ik} \geqslant x_{ijk} \quad \forall\, i \in S,\ \forall\, j \in T \tag{3.3}$$

$$t_{jk} \leqslant \sum_{i \in S} x_{ijk} \quad \forall\, j \in T \tag{3.4}$$

$$t_{jk} \geqslant x_{ijk} \quad \forall\, i \in S,\ \forall\, j \in T \tag{3.5}$$

$$\sum_{j \in T} t_{jk} = N \tag{3.6}$$

$$x_{ijk} = 0,\ 1 \quad \forall\, i \in S,\ \forall\, j \in T \tag{3.7}$$

$$s_{ik} = 0,\ 1 \quad \forall\, i \in S \tag{3.8}$$

$$t_{jk} = 0,\ 1 \quad \forall\, j \in T \tag{3.9}$$

The constraints are as defined in the original formulation (1.1)–(1.15). Here index k refers to a specific column being generated. Constraints (3.2) and (3.3) become redundant and can be eliminated.

The sub-problem is difficult to solve. It contains two major decisions: generating a set of titles and a set of stores served by the titles. Therefore, we must resort to heuristics.

3.2.3. Proposed Sub-problem algorithms

The sub-problem solves for a set of POGs as a solution to cover all stores, each POG containing exactly N titles. The logic is that in this way we can always guarantee a feasible solution in the master problem. However, this does not limit the master problem to choosing POGs generated at only one iteration from the sub-problem. The poor POGs from the sub-problem may be spelled out of the optimal solution in the master problem. In other words, the master problem can still bundle POGs generated by the sub-problem from multiple iterations. Interestingly, a poor POG at one iteration may become good at a later iteration due to the change of revised revenue. As indicated in our numerical tests, most of the optimal LP solutions to the master problem (after a large number of iterations between the master and sub-problems) turn out to be integers. This way of generating POGs is empirical and experimentally proved to be good.

Two heuristics are developed, namely Bottom-Up (BU) and Top-Down (TD). They are respectively explained in the following. Note that the sub-problem uses revised revenue with duals from the master problem.

3.2.3.1. BU algorithm. The BU algorithm starts with each store being served by an exclusive POG with the most desired N titles. In this case, each store selects its most profitable titles only subject to the limit N of the POG size. The set of titles selected for each store are the top N titles in the rank list of titles according to their revised revenue. Therefore, first rank order the titles for each store, which is a polynomial time process. Subsequently, the BU algorithm iterates by merging two POGs into one each time until a stopping criterion (to explained shortly) is met. Details of the merger will be introduced shortly. If we use p_k for the number of POGs at step $k \geqslant 1$, we have $p_1 = |S|$.

Among the $\frac{1}{2} p_k (p_k - 1)$ pairs of POGs potentially considered for merger at step k, the two POGs with the least reduced total revenue from merger are selected to merge, and the subsequent step has $p_{k+1} = p_k - 1$ POGs. When evaluating the revenue increase from a merger between two POGs, we pool together the stores served by both POGs. Then a new rank order of all the titles is developed accordingly based on their individual revenue to the pooled stores. The top N titles are selected. The revenue decrease due to merger is then assessed. This process does not stop until either of two conditions is met: (1) the total number of POGs is smaller than the maximum limit $p_{\max}$ AND the increased revenue from further merger is negative; (2) the total number of POG has reached the lower limit $p_{\min}$. Theoretically, it is possible that some mergers do not improve revenue but just serve to reach $p_{\max}$. Note that the cap on the number of POGs is exogenous and is often subjectively set by management due to factors not explained in the model.

This algorithm builds from each individual store having an exclusive POG, and is therefore called Bottom Up (or BU). It is briefly summarized below where k represents the index of iteration.

Step 1. Set $k = 1$. Each store $i \in S$ defines its own best set of N titles and is assigned to a unique POG *i. This* implies generation of a triplet for each store. This step ends with a number of POGs equal to the number of stores, $p_1 = |S|$. If we use P_1 for the set of POGs at this step, P_1 has $|S|$ POGs.

Step 2. Set $k = k + 1$. Merge POG i and POG j into a new POG if i and j have the least revenue reduction among all potential mergers. Set $p_k = p_{k-1} - 1$. Now P_k contains P_{k-1} plus the new POG less the two merged POGs.

Step 3. Repeat Step 2 till either of two conditions is met: (1) the total number of POGs is smaller than the maximum limit $p_{\max}$ AND the increased revenue from further merger is negative; (2) the total number of POGs has reached the lower limit $p_{\min}$.

The BU algorithm ends up with a set of POGs that constitute a feasible solution to the original problem. As indicated in Step 2, evaluating each merger and finding the optimal merger at each step is a polynomial time, but time-consuming process.

Associated with the BU algorithm is the following property.

Proposition 2. *To allow each store has its own POG generates an upper bound on the gross revenue.*

Proposition 2 is obvious. Be aware that the gross revenue in Proposition 2 does not account for the packaging cost.

3.2.3.2. TD algorithm. Here TD refers to a top down process. In contrast to the BU, this algorithm starts from having only one POG for all the stores. Set $p_1 = 1$, and accordingly P_1 contains the one POG. The best set of titles for this single POG can be selected easily in the following way. Rank order the titles based on their total revenues from all the stores, and select the top N titles in the triplet (or, POG). Revenue evaluation of titles for a set of stores is a common function in our algorithm, which has been explained in the BU method for POG mergers.

After a single POG is generated for all the stores, we first split it into two. We continue splitting, one POG at a step. At step $k - 1$, we end with a set of POGs, P_{k-1}, to consider for further splitting. We examine the revenue increase potential from the split of each POG, and split the POG j with the most incremental revenue into two new POGs, *j(1)* and *j (2)*. Then set the total POG $p_k = p_{k-1} + 1$.

In this process, an assignment function *Assign* (n, s) is proposed, where n is the number of POGs resulting from splitting an original POG serving stores s. Specially in our proposed TD method, $n = 2$. The function *Assign* (n, s), to be explained in more details shortly, returns n best triplets from s and a total revenue increase from the split. The POG with the most revenue increase potential from split is finally selected to split. After the split, P_k at next step has two new additions less the one being split. This process does not terminate until either of two conditions is met: (1) The total number of POG at step k $p_k \geqslant p_{\min}$ and profit increase from a split is negative; OR, (2) the maximum number of POG $p_{\max}$ is reached, i.e., $p_k = p_{\max}$.

The function *Assign* (*2*, s) is applied to each POG in P_k to assess the revenue potential. We introduce more details about *Assign* (2, s) here. The split starts with setting two empty POGs $j(1)$ and $j(2)$ for an original POG j with stores s. It examines each store in s to decide which of j (1) and $j(2)$ to assign it to. The revenue potential from an addition of a store from s, say store m, to a new POG $j(i)$, $i \in \{1, 2\}$, is evaluated by the difference between the revenue from the stores already in POG $j(i)$ without store m and the revenue from the stores in POG $j(i)$ with an addition of store m. Store m is assigned to the new POG $j(i)$with the bigger revenue increase. Then update the set of stores in this new POG with the addition of store m. Repeat the process until every store in s is assigned to either $j(1)$ or $j(2)$. Clearly, this is a myopic policy for split.

Again, the revenue from the stores in a POG is evaluated in the same way as in the BU method: rank order the titles based on revenues from this set of stores, and select the top N titles. In this way, triplets (r, **s**, **t**) are implicitly generated.

Note that in this splitting process, stores with similar demand and revenue potentials *tend* to remain together. However, as a special case, the first n stores considered in s by the function *Assign*(n, s) are always spread evenly into the n new POGs. The reason is that the incremental revenue from including a store to an empty POG is always not smaller than that from assigning the store to a non-empty POG.

The TD algorithm is summarized below.

Step 1. Initialize $k = 1$. Define the best set of titles to one POG serving all stores and set $p_1 = 1$ and P_1 has only one POG, e.g. $P_1 = \{S\}$. Set $k = k + 1$, and go to step 2.

Step 2. At step k, apply *Assign* (*2*, s) to every POG in P_{k-1}. Select the POG which potentially generates the most incremental revenue from the split, and split into *2* POGs. Update the set of POGs, P_k, with addition of the two new POGs to, and reduction of the one split from, P_{k-1}. Set $p_k = p_{k-1} + 1$ and $k = k + 1$.

Step 3. Repeat Step 2 until one of two conditions is met: (1) The total number of POG at step k, $p_k \geqslant p_{\min}$, and profit increase from a split is negative; (2) the maximum number of POG $p_{\max}$ is reached, i.e., $p_k = p_{\max}$.

Proposition 3. *One single POG serving all the stores gives rise to a lower bound on the total gross revenue from the optimal solution.*

3.3. Bounds for the optimal number of POGs and a search heuristic

As seen, if we know the optimal, or a specified, number of POGs, we will directly generate the exact number of POGs in either the BU or TD method at each iteration. Accordingly, in the master problem, we would set both the upper bound and the lower bound to be this specific number of POGs (if no other bounds are imposed). This method is adopted in our subsequent numerical test. Experimentally, knowing this optimal POG number expedites the solution process for the optimal POGs. We call this a point solution. In comparison, without knowing the exact optimal number of POGs, the master problem is extremely low efficient. Our solution follows the idea of this point solution. This is conducted by first narrowly bounding the optimal number of POGs with a limited number of point solutions.

In what follows, we develop methodologies to iteratively reduce the range for the optimal number of POGs, which builds on the following.

Observation 1 The objective value from the Integer Programming (IP) solutions to the formulation (1.1)–(1.15) increases approximately at a decreasing rate with the number of POGs (e.g. z values).

Specially, Observation 1 is correct when the POG increases from zero to two, which is easy to verify. For a number of stores beyond two, the proof becomes difficult. Unfortunately, we do not have a grasp to prove Observation 1, although we know the LP relaxation of (1.1)–(1.15) is concave in z.

Observation 1 is supported by numerical tests. In the subsequent 100 store example, the total revenue and profit functions both appear concave in the number of POGs.

The following result is partially derived from Observation 1.

Proposition 5. *With Observation 1, the optimal number of POGs* p^* *satisfies* $p^{DN} \leqslant p^* \leqslant p^{UP}$, *where* $p^{DN} = \max\left(1, \frac{R^{UP} - |S| \times c}{R^{DN} - c}\right)$ *and* $p^{UP} = \min\left(|S|, \frac{R^{UP} - R^{DN}}{c} + 1\right)$. *The optimal net profit, revenue less packaging cost, is greater than or equal to* $\max\{R^{UP} - |S| \times c, R^{DN} - c\}$, *where*$\max\{x, y\}$ *and* $\min\{x, y\}$ *return the larger and smaller values of* x *and* y, *respectively*.

Proof. Denote the revenue at the optimal POG number p^* by R^*. Note that $R^{UP} - |S| \times c$ is the profit when the number of POGs allowed is equal to the store number.

If we draw a line from $R = 0$ at z = 0 through $R = R^{DN}$ at z = 1 on the revenue curve, the rate of revenue increase is R^{DN}, an upper bound for the increase rate due to Observation 1. It takes a minimum of $\frac{R^{UP} - |S| \times c}{R^{DN} - c}$ packages to achieve a profit $R^{UP} - |S| \times c$. The number of packages necessary to reach the optimal profit is obviously higher than this. In addition, this number should not be less than one. This proves the first inequality.

Furthermore,

$R^{DN} - c \leqslant R^* - p^* c$ (Both sides represent revenue minus packaging costs), and

$R^{UP} > R^*$ (This is from Proposition 1).

The first inequality minus the second one gives$R^{DN} - R^{UP} - c \leqslant -p^* c$, which shows $p^* \leqslant p^{UP}$.

It follows that $R^* - p^* c \geqslant R^{DN} - p^* c \geqslant R^{DN} - (R^{UP} - R^{DN}) - c = 2R^{DN} - R^{UP} - c$, which provides a potential lower bound to the

optimal profit. This bound is not as tight as a bound from the TD method, $R^{DN}-c$. The BU method provides another lower bound in profit, $R^{UP}-|S| \times c$.

Clearly a lower bound results: $\max\{R^{UP}-|S| \times c, R^{DN}-c\}$. □

As a special case in Proposition 5, if the (marginal) packaging cost is zero, $p^{DN} \geqslant \frac{R^{UP}}{R^{DN}}$. Proposition 5 helps to decide an approximate range for the number of POGs. Computationally, it is very costly to solve for the optimal POGs even at a given number z of POGs, as experienced in the numerical test. Our test shows that this bound significantly reduces the range for the optimal number of POGs. A practical meaning in our algorithm of having this bound is that we do not need to generate solutions in the BU or TD method with the number of POGs exceeding this bound. By developing a tight bound on the optimal number of POGs, a significant savings in computational time is achieved. However, as will be seen in our test, the lower bound on $p^*$is usually very close to 1, very loose.

4. Flow chart of the iterative algorithm

The POGs generated from the sub-problem heuristics are first fed to the master problem for LP solution. If the solution to the master problem is not optimal to the original problem, the dual values from the master problem will be used to revise the revenue from each title. The revised revenues are used in the subproblem heuristics, which will in turn generate new candidate columns for the master problem. The standard iteration between a master problem and a sub-problem is described as in Fig. 4.

Applying the B&B method in the process of column generation is explained well in Vanderbeck (2005) and Lubbecke and Desrosiers (2005).

5. Tightening bounds and numerical test

The master problem always contains a feasible solution due to the TD or BU method. The quality feasible solutions play significant roles in the master problem because of two advantages. The first is that they provide a good lower bound to the master problem when the branch and bound algorithm (B&B) is called. A good lower bound is critical to application of the B&B algorithm. The second advantage is that the feasible solution allows us to further tighten the bound on the optimal POG number p^* for the master problem, as described in Proposition 5. In addition, a tight bound for the POG number is useful to both the BU and TD methods. As will explained later, our solution algorithm depends on having a fixed POG number.

The LP solution to the master problem provides dual values to guide generation of POGs in the subproblem algorithm implementation. In order for an efficient feasible solution, if the computational time exceeds 20 h, or if there is no improvement to the master problem objective function within 500 iterations between the master and

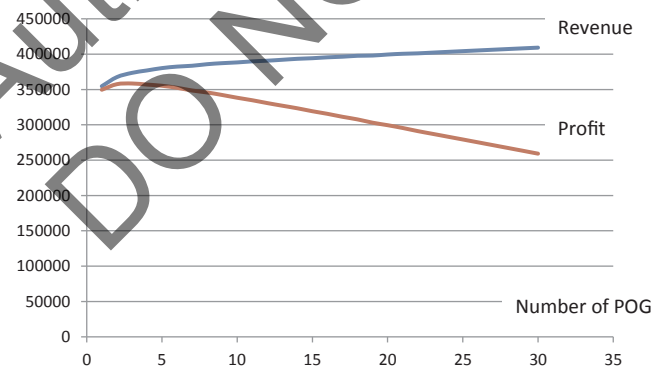

Fig. 3. Example Revenue and profit function with the number of POGs.

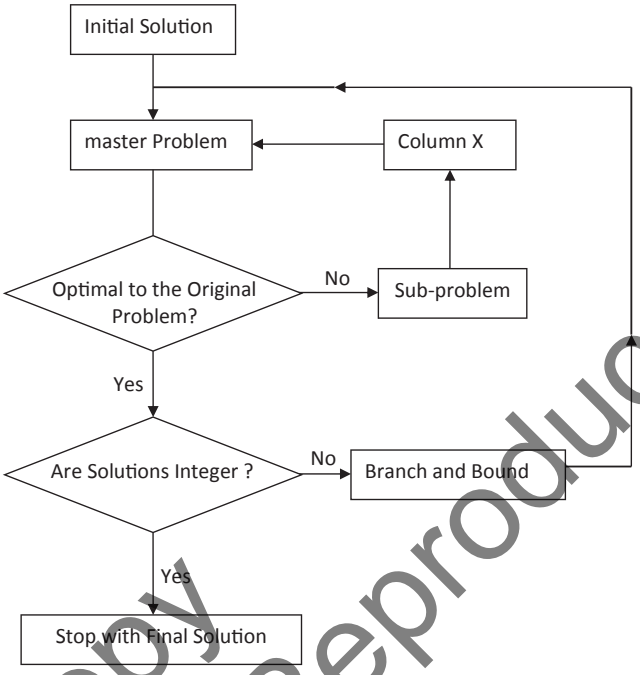

Fig. 4. The standard column generation flow chart.

subproblems, the solution process terminates. The best solution that has been generated from the subproblem is selected as the optimal solution to this problem. Our heuristic feasible solutions allow us to stop the master problem at anytime of our choice and always with a solution.

We next explain how a quality feasible solution helps tighten bound on the optimal number of POGs. Suppose at some iteration, we have a feasible solution from the sub-problem with revenue R_I and n POGs that we know $p^* \leqslant n$. It is clear that this feasible solution implies a lower bound for the objective value in the master problem, R_I-nc. In addition, if we denote the optimal revenue with two POGs byR_2, we have,

Proposition 6. Suppose a feasible solution has n POGs with a total revenueR_I. In the case $n \geqslant p^* \geqslant 2$, there holds $\max\{2, \frac{R_I-R_2-(n-2)c}{R_2-R^{DN}-c}\} \leqslant p^* \leqslant \frac{R_I-R^{DN}}{c}+1$ in light of Observation 1.

Observation 1 implies that profit increase from $z = 1$ to $z = 2$ gives an upper bound on the profit increase rate with z for $z > 2$. This helps prove the lower bound for p^*. Proof for the second inequality is obvious because R_I is a new upper bound of revenue.

5.1. Heuristic search for the optimal number of POGs

As indicated in the subsequent numerical tests, our bound on the optimal number of POGs tightens fast, but is hardly further improved after a number of iterations. As in the larger numerical test shortly later, the range for the optimal POG number is improved from [1, 2500] to be within [1, 20] after two iterations, after which, Proposition 6 does not further tighten the bound. In this case, based on Observation 1, we propose the following algorithm to further narrow the scope for the optimal POG number which is most lately assumed bounded to within a range$[p^{DN}, p^{UP}]$.

Step 1 Calculate the optimal revenues for two POG numbers p and $p + 1$ in the middle of the range $[p^{DN}, p^{UP}]$ with their revenues being R_p and R_{p+1} respectively.
Step 2 If $R_p < R_{p+1}-c$, update with $p^{DN} = p + 1$, and $p^{UP} = p^{UP}$. Repeat Step 1. If $R_p > R_{p+1}-c$, update with $p^{DN} = p^{DN}$ and $p^{UP} = p$, and repeat Step 1. If $R_p = R_{p+1}-c$, stop the algorithm, and the optimal POG number is p or $p + 1$.

This heuristic search is based on Observation 1. One can easily find that the maximum number of steps in the proposed heuristic search is bounded by $\log_2^{|S|}$. For a store with 2500 locations, this bound is 11.3. Note that in this process, calculation of revenue at a given POG number with the master problem requires replacing constraints (2.3) and (2.4) with $\sum_{j\in\Omega_k} z_j = 1.0$.

5.2. Performance of the BU and TD methods

Both the BU and TD methods generate revenues that can be used for calculation of bounds to the optimal POG number as indicated in Proposition 5 and 6. We would like to choose between BU and TD for the sub-problem based on their numerical performances.

A data set with 100 stores and 500 titles from the Target chain stores is available. The fixture capacity allows display of 172 titles. The number of POGs ranges from 1 to 100. Both the BU and TD methods are applied to this data. The performance comparison is shown in Table 6 in the Appendix A.

In Table 6, the comparison shows an overwhelming advantage of the TD over BU method in terms of the total revenue when the POG number is below 60. For the POG numbers above 60, the BU method is slightly better in revenue. The computational time is a total for all the 100 cases, which shows that the TD method consumes about 50% more time than the BU method. As computer time does not appear to be a major concern for the application of the heuristics, we only adopt the TD method in the following test problems. Although we could adopt both and choose the better solution each time, we believe the marginal gain in the improved solution is not worth the additional computational time.

The following test uses the TD method.

Test 1

A total of 372 titles is available for 100 Target chain stores. The fixture has a capacity of displaying 96 DVD titles. A cost of $5000 is set per POG. In this relatively small test problem, we are able to see in detail the performance of our proposed algorithm, part of which is illustrated in Fig. 3 earlier. In this revised solution, we replace the constraints (2.3) and (2.4) with $\sum_{j\in\Omega_k} z_j = p_{given}$, where p_{given} is a chosen number of POGs.

Here, $|S| = 100$, $c = \$5000$, the revenue $R^{UP}(p = 100) = \$456315$, $R^{DN}(p = 1) = \$354626$, where the parameter in parentheses is for the number of corresponding POGs.

Iteration 1.

$$p^{DN} = \max\left(1,\ \frac{R^{UP} - |S| \times c}{R^{DN} - c}\right) = \max\left(1,\ \frac{456315 - 100 \times 5000}{354626 - 5000}\right) = 1(\text{POG})$$

$$p^{UP} = \min\left(100,\ \frac{R^{UP} - R^{DN}}{c} + 1\right) = \min\left(100,\ \frac{456315 - 354626}{5000} + 1\right) \approx 21(\text{POGs})$$

Thus, $1 \le p^* \le 21$.

Iteration 2. Update R^{up} with $R(p^{UP} = 21) = \$400527$:

$$p^{UP} = \min\left(100,\ \frac{R^{UP} - R^{DN}}{c} + 1\right) = \min\left(100,\ \frac{400527 - 354626}{5000} + 1\right) \approx 10,$$

Thus, $1 \le p^* \le 10$.

Iteration 3. Update R^{up} with $R(p^{UP} = 10) = \$388302$:

$$p^{UP} = \min\left(100,\ \frac{R^{UP} - R^{DN}}{c} + 1\right) = \min\left(100,\ \frac{388302 - 354626}{5000} + 1\right) \approx 7,$$

Thus, $1 \le p^* \le 7$.

Iteration 4. Update R^{up} with $R(p^{UP} = 7) = \$383569$:

$$p^{UP} = \min\left(100,\ \frac{R^{UP} - R^{DN}}{c} + 1\right) = \min\left(100,\ \frac{383569 - 354626}{5000} + 1\right) \approx 6,$$

Thus, $1 \le p^* \le 6$.

Since 6 is small enough, we are able to find the optimal solution for each POG number as follows.

Maximum Revenue(6) = $382380, profit = $352,380
Maximum Revenue(5) = $380249, profit = $355,249
Maximum Revenue(4) = $376920, profit = $356,920
Maximum Revenue(3) = $373487, ∗profit = $373,487 − 3∗5000 = **$358,487** (Optimal)
Maximum Revenue(2) = $367476, profit = $357,476
Maximum Revenue(1) = $354626, profit = $349,626

The optimal solution from CG includes $p^* = 3$, $R^* = \$373183$, and a profit = $358,183.

Test 2

Test 2 is a full-sized problem from real production data. At each of the 2500 locations of a chain store, there is a fixture for displaying 180 DVD titles selected from a total of 500 titles. The packaging cost is subjectively determined to be $250,000 per POG by the management. By background, the current annual revenue from DVD sales is in multi-billion dollars. Thus, a small percentage improvement of revenue is significant.

A sample forecasted revenue matrix is illustrated in Table 7.

With the TD method, it is easy to get $R^{DN} = R(p = 1) = \$69,666,310$, $R^{UP} = R(p = 2500) = \$91,958,456$. Here, $|S| = 2500$, $c = \$250,000$. Then we start to tighten the bound for the optimal number of POG p∗ as follows.

Iteration 1.

$$p^{DN} = \max\left(1,\ \frac{R^{UP} - |S| \times c}{R^{DN} - c}\right) = \max\left(1,\ \frac{91958456 - 2500 \times 250000}{69666310 - 250000}\right) = 1(\text{POG})$$

$$p^{UP} = \min\left(2500,\ \frac{R^{UP} - R^{DN}}{c} + 1\right) = \min\left(2500,\ \frac{91958456 - 69666310}{250000} + 1\right) \approx 90(\text{POGs})$$

Thus, $1 \le p^* \le 90$. Here the optimal revenue at $p = 90$is $R =$ "$"74, 437, 632. Update $R^{UP}$with R, which will be used in the next iteration to strengthen the bound on p^*.

Iteration 2.

$$p^{UP} = \min\left(2500,\ \frac{R^{UP} - R^{DN}}{c} + 1\right) = \min\left(2500,\ \frac{74437632 - 69666310}{250000} + 1\right) \approx 20(\text{POGs})$$

Table 7
Revenue Forecast Matrix (Stores/Titles).

StoreID\TitleID	Title001	Title002	Title003	…	Title498	Title499	Title500
S0001	$741	$55	$659	…	$14	$151	$741
S0002	$196	$82	$96	…	$247	$181	$316
…	…	…	…	…	…	…	…
S2500	$226	$92	$206	…	$41	$247	$412

Table 8
Numerical Results.

POG #	Revenue	Profit	ComputingTime (sec)
2500	**91,958,456**	**−533041544**	**135**
…	…	…	…
90	**76,072,616**	**53,572,616m**	**71,120**
…	…	…	…
22	74,472,455	68,972,455	53,114
21	74,455,444m	69,205,444	45,760
20	**74,437,632**	**69,437,632**	**38,393**
19	74,423,094	69,673,094	31,391
18	74,390,289	69,890,289	25,412
17	74,362,531	70,112,531	20,566
16	74,283,151	70,283,151	16,115
15	74,245,915	70,495,915	13,579
14	74,222,158	70,722,158	11,200
13	74,181,375	70,931,375	9018
12	74,146,173	71,146,173	7233
11	74,063,730m	71,313,730	5868
10	73,974,746	71,474,746	4709
9	73,765,906	71,515,906	3853
8	73,772,721	71,772,721	3172
7	73,570,780	71,820,780	2504
6	**73,421,947**	**71,921,947**	2028
5*	**73,278,391**	**72,028,391**	1645
4	**72,993,037**	**71,993,037**	1300
3	**72,289,653**	**71,539,653**	1146
2	71,900,411	71,650,411	907
1	**69,666,310**	**69,416,310**	**103**

Within 2509 iterations between the master and Sub-problems when the number of POGs is 20, CG generated 49,681 POGs to get the final solution. The solution takes 10.5 h on an HP workstation. And a total revenue of $74,437,632 is achieved.

After iteration 2, we cannot use Proposition 5 to further tighten the bound on the optimal number of POGs. It is clear now that the optimal number of POGs must be within the range [1, 20]. We subsequently adopt the proposed search algorithm, and calculate the revenue at POG numbers 10 and 11 respectively. The revenues turn out to be $73,974,746 and $74,063,730 accordingly. The revenue increase from POG = 10 to POG = 11 is outweighed by the packaging cost $25,000. Therefore we next calculate the revenues at POG = 5, and POG = 6 respectively. The revenues are $73,278,391 and $73,421,947 respectively. At the third step, we calculate the revenues at POG = 3 and POG = 4 respectively. The according revenues are $72,289,653 and $72,993,037, whose differential revenue still outnumbers the packaging cost. By now, it is clear that the optimal number of POGs is 5. Table 8 summarizes the numerical results in this test, with revenues at several additional POG numbers for the purpose of verification. The bold lines indicate cases actually calculated in our algorithm.

Optimality of the final feasible solution is hard to evaluate. If one believes the optimal revenue generated with 20 POGs in the test problem provides an upper bound to the revenue from 5 POGs, the solution is within 97% of the optimal net revenue.

6. Conclusion

The retail industry often distributes products in packages. This paper studies a special product selection and packaging for distribution in order to maximize the total profit. In this particular application, the decision concerns packaging a small number of DVDs out of hundreds of thousands of them available to serve several thousand retail stores. Only a limited number of different packages are allowed. We propose an optimization algorithm using column generation. We define a master problem and a sub-problem, and propose two heuristic algorithms for the sub-problem, namely the Bottom UP (BU) and Top Down (TD) methods respectively. The TD method appears superior to the BU based on the numerical tests. We further develop bounds on the optimal number of POGs and the optimal profit for the master problem. The bounds tighten fast and significantly reduce the computational time. Our full-sized problem test shows that our proposed algorithm is promising for application. DVD sale is a multi-billion dollar business annually at major chain stores. According to an internal estimate by the management of the chain store, the annual revenue improvement potential from using our developed methodologies, compared with using the current manual method, would be in dozens of millions of dollars.

This practical problem has more features than studied in this paper. First, each store could have different fixtures. For this particular example of DVD distribution, our simplification might represent a practically acceptable approximation to the more complex problems. Secondly, it would be interesting to examine if and how the relaxation of the assumption on demand independence between titles affects the distribution decisions.

Appendix A

See Table 6.

Table 6
Comparison between BU and TD Methods.

Number of POGS	Total Revenue		Revenue Difference (TD-BU)/BU
	BU	TD	
1	354,626	354,626	0.000000
2	358,129	367,474	0.026094
3	363,472	371,947	0.023317
4	366,495	372,944	0.017596
5	368,009	373,061	0.013728
6	368,662	373,760	0.013828
7	370,904	378,193	0.019652
8	371,635	379,282	0.020577
9	373,030	380,130	0.019033
10	373,542	380,447	0.018485
11	374,955	380,622	0.015114
12	375,827	381,868	0.016074
13	376,654	383,872	0.019163
14	377,322	384,004	0.017709
15	378,066	381,131	0.008107
16	380,125	386,571	0.016958

(*continued on next page*)

Table 6 (*continued*)

Number of POGS	Total Revenue		Revenue Difference (TD-BU)/BU
	BU	TD	
17	381,956	387,670	0.014960
18	385,233	389,782	0.011808
19	386,477	390,919	0.011494
20	387,393	391,403	0.010351
21	388,497	391,952	0.008893
22	389,165	393,771	0.011836
23	390,049m	393,981m	0.010081
24	393,839	395,541	0.004322
25	395,336	396,939	0.004055
26	395,074	397,831	0.006978
27	395,730	400,397	0.011793
28	397,262	401,754	0.011307
29	397,921	403,772	0.014704
30	398,814	403,772	0.012432
31	401,187	404,441	0.008111
32	402,492	405,116	0.006519
33	403,569	405,837	0.005620
34	403,769	406,736m	0.007348
35	404,334	407,367m	0.007501
36	405,673	408,901	0.007957
37	406,697	410,173	0.008547
38	408,256	411,624	0.008250
39	407,748	412,445	0.011519
40	408,810	413,619	0.011763
41	409,309	415,247	0.014507
42	410,546	416,489	0.014476
43	412,485	416,893	0.010686
44	413,192	418,454	0.012735
45	412,893	419,452	0.015885
46	415,607	420,620	0.012062
47	417,024	421,565	0.010889
48	417,508	422,292	0.011458
49	417,849	422,880	0.012040
50	418,649	424,006	0.012796
51	419,310	424,560	0.012521
52	420,141	424,869	0.011253
53	420,164	424,917	0.011312
54	421,147	424,982	0.009106
55	423,139	425,824	0.006345
56	423,226	426,702	0.008213
57	424,228m	426,829	0.006131
58	424,727	427,021	0.005401
59	425,195	427,396	0.005176
60	426,076	427,399	0.003105
61	427,060	427,818m	0.001775
62	428,531	428,458	−0.00017
63	429,609	428,740	−0.00202
64	430,381	428,953	−0.00332
65	431,004	430,474	−0.00123
66	432,063	431,923	−0.00032
67	432,527	432,087	−0.00102
68	433,354	432,213	−0.00263
69	434,548	432,618	−0.00444
70	435,761	432,915	−0.00653
71	435,763	433,809	−0.00448
72	436,114	435,112	−0.00230
73	437,010	436,299	−0.00163
74	437,361	436,568	−0.00181
75	438,788	437,324	−0.00334
76	438,795	437,835	−0.00219
77	439,947	438,943	−0.00228
78	440,671	439,607	−0.00241
79	440,721	440,199	−0.00118
80	442,020	440,423	−0.00361
81	443,202	441,448	−0.00396
82	443,895	441,837	−0.00464
83	444,696	442,721	−0.00444
84	445,399	443,428	−0.00443
85	446,193	444,309	−0.00422
86	447,227	445,394	−0.00410
87	448,114	446,761	−0.00302
88	448,787	447,340	−0.00322
89	449,108	448,559	−0.00122

(*continued on next page*)

Table 6 (*continued*)

Number of POGS	Total Revenue		Revenue Difference (TD-BU)/BU
	BU	TD	
90	450,185	449,196	−0.00220
91	451,599	450,001	−0.00354
92	451,625	450,385	−0.00275
93	452,264	451,026	−0.00274
94	452,822	451,967	−0.00189
95	453,632	452,704	−0.00205
96	454,211	453,791	−0.00092
97	454,903	454,508	−0.00087
98	455,223	455,236	0.0000286
99	455,913	455,913	0.000000
100	456,315	456,315	0.000000
Total Computing Time (sec)	7350	11,275	SUM = 0.6044

References

Chan, F. T. S., Chan, H. K., & Choy, K. L. (2006). A systematic approach to manufacturing packaging logistics. *International Journal of Advanced Manufacturing Technology, 29*, 1088–1101.

Dantzig, G. B., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research, 8*, 101–111.

Hausman, W. H., Schwarz, L. B., & Graves, S. C. (1976). Optimal storage assignment in automatic warehousing systems. *Management Science, 22*(6), 629–638.

Lubbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research, 53*(6), 1007–1023.

Muppani, V. R., & Adil, G. K. (2008). Class-based storage-location assignment to minimize pick travel distance. *International Journal of Logistics: Research and Application, 11*(4), 247–265.

Prendergast, G., & Pitt, L. (1996). Packaging, marketing, logistics and the environment: Are there trade-offs? *International Journal of Physical Distribution and Logistics Management, 26*(6), 69–72.

Rosenblatt, M. J., & Eynan, A. (1989). Deriving the optimal boundaries for class-based automatic storage/retrieval systems. *Management Science, 35*(12), 1519–1524.

Vanderbeck, F. (2005). In G. Desaulniers, J. Desrosiers, & M. M. Solomon (eds.), *Implementing mixed integer column generation* (pp. 331–358). Springer.