Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.

Volume 210, Issue 3, 1 May 2011
ISSN 0377-2217
ELSEVIER
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
Available online at www.sciencedirect.com
ScienceDirect
LAST ISSUE OF THIS VOLUME
www.elsevier.com/locate/ejor

This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/copyright

Author's personal copy

European Journal of Operational Research 210 (2011) 670–683

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

Optimization models for targeted offers in direct marketing: Exact and heuristic algorithms

Fabrice Talla Nobibon [*], Roel Leus, Frits C.R. Spieksma

Operations Research Group, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium

ARTICLE INFO

Article history:
Received 29 September 2009
Accepted 9 October 2010
Available online 20 October 2010

Keywords:
Direct marketing campaign
Integer programming
Branch-and-price algorithm
Non-approximability
Heuristics
Tabu search

ABSTRACT

This paper presents an optimization model for the selection of sets of clients that will receive an offer for one or more products during a promotion campaign. We show that the problem is strongly NP-hard and that it is unlikely that a constant-factor approximation algorithm can be proposed for solving this problem. We propose an alternative set-covering formulation and develop a branch-and-price algorithm to solve it. We also describe eight heuristics to approximate an optimal solution, including a depth-first branch-and-price heuristic and a tabu-search algorithm. We perform extensive computational experiments both with the exact as well as with the heuristic algorithms. Based on our experiments, we suggest the use of optimal algorithms for small and medium-size instances, while heuristics (especially tabu search and branch-and-price-based routines) are preferable for large instances and when time is an important factor.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

In this paper, we examine the development of optimization models that decide on the details of a direct-marketing campaign that will make targeted offers to customers. Such campaigns are fundamental marketing tools for improving the economic profit of a firm, either by acquiring new customers or by generating additional revenue from existing customers [20]. The former action is called "acquisition" while the latter is "retention" [33]. In this paper, we are concerned with the latter case: campaigns that generate additional revenue by offering new products to existing customers. This study is justified by at least two practical facts. Reinartz et al. [33] point out that "When firms trade off between expenditures for acquisition and those for retention, a suboptimal allocation of retention expenditures will have a greater impact on long-term customer profitability than will suboptimal acquisition expenditures". Moreover, models and methods used for data analysis are more suited for retention [19] since more information is available. Retention boosts the customer lifetime value, which is defined by Kumar et al. [21] to be "the sum of cumulated cash flows – discounted using the weighted average cost of capital – of a customer over his or her entire lifetime with the firm". Customer lifetime value usually serves as a metric for a ranking or segmentation of the firm's customers [34]. During the last decades, the advances in data analysis coupled with the availability of customer data have pushed firms to develop a more customer-oriented strategy. Nowadays, such a strategy is globally accepted, but its practical implementation is far from being accomplished. This implementation delay is observed both in business-to-business and business-to-consumer settings, and is particularly pronounced in financial institutions such as large banks and insurance companies, which often have a large number of customers with full data available but may lack sophisticated tools that efficiently take into account these advantages in decision making [10]. Although retail banks have access to the deepest veins of customer data of any industry in the world [12], Bernstel [3] reveals that many banks are not using the full power of their customer databases.

Optimal product targeting models examine "Which products should be targeted to which customers to maximize profits, under the constraints that only a limited number can be targeted to each customer, and each product has a minimum sales target" [8,19]. Such problems are essentially characterized by two steps, which are "data analysis" and "problem formulation and solution". The first step, which is mainly statistical, has received increasing attention with the advances in data analysis, and is often referred to by the term "marketing response models", which are models intended to help scholars and managers understand how consumers

* Corresponding author. Tel.: +32 16326960.
E-mail addresses: Fabrice.TallaNobibon@econ.kuleuven.be (F. Talla Nobibon), Roel.Leus@econ.kuleuven.be (R. Leus), Frits.Spieksma@econ.kuleuven.be (F.C.R. Spieksma).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.10.019

Author's personal copy

respond to marketing activities [16]. In academia, these models were introduced in the early eighties (see, for instance, Guadagni and Little [14]), and in practice became popular in the nineties thanks to the scanner-data revolution. De Reyck and Degraeve [9] have developed a model in which a proxy is used for direct response rates, effectively reducing the effort spent on response modeling. Some improvements to this model were proposed by Tripathi and Nair [38]. Recently, Bhaskar et al. [4] have proposed a fuzzy mathematical-programming approach where fuzzy numbers are used to represent the output of the first step, allowing to take into account uncertainty. Their model, however, incorporates only a budget constraint and volume target constraints. An elaborate discussion on the inputs for direct-marketing models can be found in Bose and Chen [5].

This paper investigates the development of optimization models for targeted offers in a promotion campaign, based on integer programming. Motivation for studying this problem comes from a case occurring at FORTIS [17], until recently one of the leading banks in Belgium. More generally, Stone and Jacobs [36] point at direct marketing's recent growth in business categories such as banks, investment and insurance companies. We aim to maximize the profit subject to business constraints such as the campaign's return-on-investment hurdle rate that must be met (the hurdle rate is the minimum acceptable rate of return that management will accept for the campaign), a limitation on the funding available for each product, a restriction on the maximum number of possible offers to a client, and a minimum-quantity commitment (MQC) on the number of units of a product to be offered in order for that product to be part of the campaign.

The MQC constraint has been briefly mentioned by Cohen [8] as a technical issue for an application in a bank. However, he did not explicitly incorporate it into his model. Our model does take into account this constraint, making it an extension of the model used by Cohen. We also study the case where the budget of a product is managed by a business unit. These two constraints make our model substantially different from that proposed e.g., by Cohen. Contrary to Cohen, our model also considers a more general setting where a client can receive more than one offer. With respect to solution procedures, Cohen restricts himself to LP-based solutions for aggregated data, while our most important contributions are computational, in that we propose several exact and heuristic algorithms. In our formulation, we also impose a more general version of the MQC constraint, allowing the fixed minimum quantity to depend on the product, which distinguishes the constraint from comparable MQC restrictions studied in the analysis of transportation problems [25], bottleneck problems [26], and assignment problems [24]. Contrary to these references, our model also includes both the budget constraints and the hurdle-rate constraint, making it more difficult to solve.

In this paper, we present a basic integer-programming formulation for the problem at hand. We show the non-approximability of the problem, which makes the existence of an algorithm that will always provide a feasible solution and guarantee a specified proportion of optimal profit in polynomial time, highly unlikely. We next present a set-covering formulation and develop a branch-and-price algorithm for solving it. A dynamic-programming algorithm and a 2-approximation algorithm are presented for solving the pricing problem, which is closely related to the k-item knapsack problem. The size of instances that can be solved optimally using this algorithm allows its efficient use for business-to-business promotion campaigns (which have moderate size and high variable and fixed costs) and for sampling approaches in financial institutions [8]. We then present eight heuristics to approximate an optimal solution, which can be used for large instances and hence for business-to-consumer promotion campaigns. These heuristics are either variants of the algorithms used in practice for application in a bank (see [17,19]) or developed based on the structure of the problem.

The contributions of this paper are fourfold:

(1) The formulation of the product targeting problem as a mixed-integer programming (MIP) problem including more business constraints.
(2) An observation concerning the non-approximability of the problem.
(3) The reformulation of the problem as a set-covering model, which is solved via branch-and-price; a special cardinality-constrained knapsack problem is obtained as a pricing problem, which we solve by adapting well-known results concerning the knapsack problem.
(4) The development of efficient heuristics, including a depth-first branch-and-price heuristic and a tabu-search algorithm, that outperform methods used in practice.

This paper is organized as follows. Section 2 describes the basic integer-programming formulation for selecting clients that will receive targeted offers. The complexity of the problem makes it very difficult to produce optimal solutions by straightforward use of a MIP solver. We propose an alternative formulation, called the set-covering formulation, in Section 3 and develop a branch-and-price algorithm to solve it. In Section 4, we describe eight heuristics to approximate an optimal solution. Experimental results are presented in Section 5 both for the exact as well as for the heuristic algorithms. Finally, a number of possible extensions of the studied model are presented in Section 6, followed by a summary and conclusions in Section 7.

2. Basic formulation

The objective of a direct-marketing promotion campaign is to find a way to achieve a maximum profit by offering n different products to m customers while taking into account various business constraints. We incorporate the following restrictions: the return-on-investment hurdle rate must be met for the campaign, the budget allocated to each product is limited, an upper bound is imposed on the number of products that can be offered to each client and there is also an MQC constraint for each product. We define the parameter r_{ij} as the probability that client i reacts positively to an offer of product j (or the probability that product j is the next product bought by client i [19]) and DFV_{ij} as the return to the firm when client i responds positively to the offer of product j. The latter is termed the Delta Financial Value by FORTIS [17]. These two parameters are the basis for the computation of customer lifetime value [34]. Practically, these parameters are estimated using response models based on historical data [8,19,32] and are assumed to be available within the firm. We denote by p_{ij} the expected return to the firm (revenue) from the offer of product j to client i, so $p_{ij} = r_{ij}DFV_{ij}$. In the remainder of this paper, we will mostly use p_{ij}. Further, there is a variable cost c_{ij} associated with the offer of product j to client i, the upper bound M_i of offers that can be made to a client i (this quantity is related to the status of the client), the minimum-quantity commitment bound O_j associated with product j, the budget B_j allocated to the product j, a fixed cost f_j needed if product j is used for the campaign and finally the corporate hurdle rate R. The value of R is dependent on the firm and the riskiness of the investment. In practice, most firms use their weighted average cost of capital (WACC) as an estimation of R [6]. We define the decision variables $y_j \in \{0,1\}$, equal to 1 if product j is used during the campaign, 0 otherwise, and $x_{ij} \in \{0,1\}$, which is equal to 1 if product j is offered to client i and 0 otherwise. A basic formulation for the product targeting problem can be expressed as:

Author's personal copy

$$\text{(M1)}\quad \text{maximize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}(p_{ij}-c_{ij})x_{ij}-\sum_{j=1}^{n}f_jy_j \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}p_{ij}x_{ij} \geqslant (1+R)\left[\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij}+\sum_{j=1}^{n}f_jy_j\right], \tag{2}$$

$$\sum_{i=1}^{m}c_{ij}x_{ij} \leqslant B_j, \quad j=1,\ldots,n, \tag{3}$$

$$\sum_{j=1}^{n}x_{ij} \leqslant M_i, \quad i=1,\ldots,m, \tag{4}$$

$$\sum_{i=1}^{m}x_{ij} \leqslant my_j, \quad j=1,\ldots,n, \tag{5}$$

$$\sum_{i=1}^{m}x_{ij} \geqslant O_jy_j, \quad j=1,\ldots,n, \tag{6}$$

$$y_j,\ x_{ij} \in \{0,1\}, \quad i=1,\ldots,m,\ j=1,\ldots,n. \tag{7}$$

The objective function (1) is the maximization of the total net benefit received from the offer of products to clients minus the fixed cost of using the products for the campaign. The first constraint (2) is the corporate hurdle-rate constraint, which makes sure that the campaign's return on investment is at least R, and which was first suggested by Cohen [8] for an application in a bank. The set of constraints (3) enforces that we should not exceed the budget B_j allocated to the product j. Here, the product dependency of the budget reflects the situation in large firms where an individual business unit is responsible for the production and the sale of a product. Hence, each business unit has its own budget. The set of constraints (4) states that we cannot propose more than a certain number M_i of products to client i; the sets of constraints (5) and (6) constitute the MQC constraint, which specifies that when a product is not part of the campaign, no clients will receive an offer, while if product j takes part in the campaign then at least $O_j > 0$ clients receive an offer, and finally the last set of constraints (7) is the integrality constraint.

Definition 1. A *non-trivial feasible solution* for the basic formulation (M1) is a feasible solution that achieves a non-zero objective value.

The following result shows that there is little hope for finding a polynomial-time algorithm for solving (M1).

Proposition 1. *The product targeting problem defined by the formulation (M1) is strongly NP-hard, even for $O_j = 1$ for all j.*

Proof. See Appendix A. □

Moreover, the basic formulation (M1) is difficult to solve even approximately. We prove this non-approximability result by showing that it is NP-hard to find a non-trivial feasible solution to (M1).

Proposition 2. *Finding a non-trivial feasible solution to the basic formulation (M1) is NP-hard.*

Proof. See Appendix B. □

The results of Propositions 1 and 2 justify the intensive use of heuristics in practice [8,9,19].

The basic formulation (M1) can be strengthened by using the following disaggregate version of constraints (5):

$$x_{ij} \leqslant y_j, \quad i=1,\ldots,m,\ j=1,\ldots,n. \tag{8}$$

The following result allows the relaxation of the integrality constraint $y_j \in \{0,1\}$ to $0 \leqslant y_j \leqslant 1$ for all j.

Proposition 3. *The convex hull of the feasible solutions to the integer program (M1) is identical to the convex hull of the solutions that satisfy the constraints (2)–(4), (6) and (8), and $0 \leqslant y_j \leqslant 1$, $x_{ij} \in \{0,1\}$ for all i, j.*

Proof. See Appendix C. □

In the rest of this paper, the formulation obtained from (M1) by replacing (5) by (8) and by substituting $0 \leqslant y_j \leqslant 1$ for $y_j \in \{0,1\}$ for all j, is denoted by (M2).

The model described in the foregoing paragraphs starts from a number of premises, one of which is the independence between different products' promotions and sales. In practice, however, products may be dependent on each other in a number of ways. The promotion of similar products of the same firm (substitutes) can lead to *cannibalisation* effects: one product's sales growth can cause another product's sales to decline. For example, promoting a bank's savings account might reduce the probability that a customer opts for the bank's investment fund, bonds or shares. On the other hand, the effect of promoting one product can also boost the sales of other products through branding, leading to complementarities such as the *halo-effect*: positive spill-overs can occur of supporting one product or brand on other products of the same company. For example, a very competitive investment offer can positively stimulate the bank's overall image. Such interdependencies between different products' promotions and sales would necessitate making the probabilities r_{ij} a function of the x_{ik}-variables, which would lead to non-linearities. In our model, we also neglect the influence of other firms, and we also assume that there will be no averse timing aspects such as *forward buying*, where a campaign might motivate customers to buy a product now instead of later (as planned). In this way, a campaign might accelerate the purchase but not really add volume in the long run. Although interesting, it is fairly hard to incorporate the listed aspects into our model and they will not be further studied in this text.

3. Branch-and-price

This section is devoted to the application of a branch-and-price (B&P) algorithm for solving the product targeting problem. In the first subsection, we present an appropriate formulation, called a set-covering formulation. The next subsection studies the pricing problem and the last subsection presents a branching strategy.

3.1. A set-covering formulation

For every product j, let k_j be the number of distinct subsets of clients of cardinality at least O_j that can receive the offer of product j within the budget limit B_j. Explicitly, we define S_{pj} ($p = 1,\ldots,k_j$) as a set of at least O_j clients satisfying $\sum_{i\in S_{pj}} c_{ij} \leqslant B_j$. We use the binary variable z_{pj} to indicate whether the product j is offered to the set of clients S_{pj} ($z_{pj} = 1$) or not ($z_{pj} = 0$). A *set-covering formulation* of the product targeting problem is given by:

$$\text{(M3)}\quad \text{maximize} \quad \sum_{j=1}^{n}\left[\sum_{p=1}^{k_j}\left(\sum_{i\in S_{pj}}(p_{ij}-c_{ij})-f_j\right)z_{pj}\right] \tag{9}$$

$$\text{subject to} \quad \sum_{j=1}^{n}\left[\sum_{p=1}^{k_j}\left(\sum_{i\in S_{pj}}(p_{ij}-(1+R)c_{ij})-(1+R)f_j\right)z_{pj}\right] \geqslant 0, \tag{10}$$

$$\sum_{j=1}^{n}\sum_{p:i\in S_{pj}}z_{pj} \leqslant M_i, \quad i=1,\ldots,m, \tag{11}$$

$$\sum_{p=1}^{k_j}z_{pj} \leqslant 1, \quad j=1,\ldots,n, \tag{12}$$

$$z_{pj} \in \{0,1\}, \quad j=1,\ldots,n,\ p\in\{1,\ldots,k_j\}. \tag{13}$$

Author's personal copy

The first constraint (10) enforces that the campaign's return on investment must be at least R, the set of constraints (11) ensures that at most M_i products are offered to client i and the set of constraints (12) states that at most one non-empty set of clients is selected for each product. The formulation (M3) is called a set-covering formulation to reflect the fact that its solution can be viewed as a cover for the set of clients. More specifically, considering the subset X_0 of clients who do not receive any offer and the subsets X_j of clients who receive an offer of product j for $j = 1,\ldots,n$, the collection $\{X_j : j = 0,1,\ldots,n\}$ forms a cover of the total client set. We prove in [37] that the set-covering formulation is obtained by applying Dantzig-Wolfe decomposition to the LP relaxation of the basic formulation (M2). As a consequence, the value of the bound provided by the LP relaxation of (M3) is equal to the value of the Lagrangian dual obtained by dualizing the constraints (2) and (4) [30].

Let z be any feasible solution to the LP relaxation of (M3) and let $x^*_{ij} = \sum_{p:i\in S_{pj}} z_{pj}$, $y^*_j = \sum_{p=1}^{k_j} z_{pj}$, then (x^*,y^*) is a feasible solution to the LP relaxation of (M2) that achieves the same objective value as z. Furthermore, we have the following result, which is useful for the branching strategies.

Proposition 4. *Given a feasible solution z to the LP relaxation of (M3), if, for a given product j, z_{pj} is fractional, then there must be an i such that $x^*_{ij} = \sum_{p:i\in S_{pj}} z_{pj}$ is fractional.*

Proof. See Appendix D. □

Proposition 4 implies that the bound provided by the LP relaxation of the set-covering formulation is at least as strong as that obtained by the LP relaxation of the basic formulation; in fact, in Section 5 we show that for most instances the former bound is stronger than the latter.

3.2. The pricing problem

We consider the LP relaxation (LPM3) of (M3) obtained by replacing constraints (13) by

$$z_{pj} \geqslant 0, \quad j = 1,\ldots,n,\ p \in \{1,\ldots,k_j\}. \tag{14}$$

The formulation (LPM3) has an exponential number of variables, which makes it difficult to solve even instances of average size. Instead of solving the master problem (LPM3), we consider a restricted problem that includes only a subset of variables (columns) and can be solved directly. Additional columns for the restricted problem can be generated by looking at the dual of (LPM3) given by:

$$\begin{aligned}
\text{(DLPM3)}\quad & \text{minimize} && \sum_{i=1}^{m} M_i u_i + \sum_{j=1}^{n} v_j \\
& \text{subject to} && \sum_{i\in S_{pj}} [(p_{ij} - (1+R)c_{ij})d + u_i] - (1+R)f_j d + v_j \\
& && \geqslant \sum_{i\in S_{pj}} (p_{ij} - c_{ij}) - f_j, \quad \forall p,j, \qquad (15)\\
& && u_i,\ v_j \geqslant 0,\ d \leqslant 0, \quad i = 1,\ldots,m,\ j = 1,\ldots,n, \qquad (16)
\end{aligned}$$

where we use the dual variables d corresponding to (10), u_i corresponding to the set of constraints (11) and v_j corresponding to (12). The optimal solution found for the restricted problem is not optimal for the master problem if the associated dual variables u, v and d violate one of the constraints (15). Note that the set of constraints (16) is automatically satisfied by the dual variables. Since it is not computationally viable to compute and check the inequality (15) for all couples (p,j) not included in the restricted problem, we propose to proceed as follows. We drop the index j and assume that the product is given. We solve the following question called the *pricing problem*:

$$\exists S_p \text{ such that } \sum_{i\in S_p} [p_i(d-1) + c_i(1-(1+R)d) + u_i] + v + f(1-(1+R)d) < 0? \tag{17}$$

Remark that for a given product j, the left-hand side of the inequality is exactly the reduced cost of the variable z_{pj}, so that the pricing problem (17) checks the primal optimality condition.

3.2.1. Solving the pricing problem

A solution to the pricing problem (17) for a fixed product j can be obtained by solving the following variant of the *k-item knapsack problem*. The k-item knapsack problem, also known as cardinality-constrained knapsack problem [18], is a knapsack problem with an extra constraint enforcing a lower or upper bound on the number of items that a feasible solution must or may contain. For ease of exposition, we define $w_i = p_i(d-1) + c_i(1-(1+R)d) + u_i$ for all i

$$\begin{aligned}
\text{(Price)}\quad & \text{minimize} && \sum_{i=1}^{m} w_i x_i && (18)\\
& \text{subject to} && \sum_{i=1}^{m} c_i x_i \leqslant B, && (19)\\
& && \sum_{i=1}^{m} x_i \geqslant O, && (20)\\
& && x_i \in \{0,1\}, \quad i = 1,\ldots,m. && (21)
\end{aligned}$$

We refer to the problem corresponding to (18)–(21) as the problem kKP. We take the w_i's as the weights and the c_i's as the costs. Notice that in general the problem kKP is weakly NP-hard. Furthermore in our case, the w_i need not always be positive nor are they always integers. These last observations make the use of dynamic programming by weight [18] for solving kKP inefficient. Kellerer et al. [18] develop a dynamic-programming algorithm for the case where the cardinality constraint (20) is replaced by $\sum_{i=1}^{m} x_i \leqslant O$. We show here that this algorithm is easily modified to deal with (18)–(21).

3.2.2. Exact algorithm

We propose a dynamic-programming algorithm based on the algorithm described in [18] for solving kKP in pseudopolynomial time. Let k be the maximum number of items that can be used by a feasible solution to kKP. Clearly, since $c_i \geqslant 0$, k can be computed by taking the items with smallest cost c_i until the budget constraint (19) is violated. If $k < O$ then the problem kKP is infeasible. Assuming that $k \geqslant O$, we define the two-dimensional dynamic-programming function $Y_i(c,l)$ for $i = 0,1,\ldots,m$; $l = 0,1,\ldots,k$; $c = 0,1,\ldots,B$, as the optimal solution value of the following problem:

$$Y_i(c,l) = \min\left\{ \sum_{j=1}^{i} w_j x_j \,\middle|\, \sum_{j=1}^{i} c_j x_j = c,\ \sum_{j=1}^{i} x_j = l,\ x_j \in \{0,1\} \right\}.$$

An entry $Y_i(c,l) = q$ means that among the clients $1,2,\ldots,i$, the minimum-weight subset of clients with total cost c and cardinality l has weight q.

The initialization is given by $Y_0(c,l) = +\infty$ for $l = 0,1,\ldots,k$; $c = 0,1,\ldots,B$, with $Y_0(0,0) = 0$. Then, for $i = 1,\ldots,m$, the entries Y_i can be computed from those of Y_{i-1} by the following recursion

$$Y_i(c,l) = \begin{cases} Y_{i-1}(c,l), & \text{if } c_i > c,\\ \min\{Y_{i-1}(c,l), Y_{i-1}(c-c_i, l-1) + w_i\}, & \text{if } l > 0,\ c_i \leqslant c. \end{cases} \tag{22}$$

After computing Y_1 to Y_m, the optimal objective function value of kKP is given by

$$\min\{Y_m(c,l) \mid l \geqslant O,\ c = 0,1,\ldots,B\}.$$

Author's personal copy

We observe that only entries in $Y_{i-1}(c,l)$ need to be stored in order to derive $Y_i(c,l)$. Caprara et al. [7] propose an implementation based on pointers that achieves a space complexity of $O(k^2B)$ and a time complexity of $O(mkB)$.

3.2.3. Approximation algorithm

An approximation algorithm for kKP is based on the LP relaxation of the problem, which is obtained by replacing constraints (21) by

$$0 \leqslant x_i \leqslant 1, \quad i = 1,\ldots,m. \tag{23}$$

Denoting the optimal objective function value of the LP relaxation of kKP by z^{LP}, the following result proved in [7,18] holds.

Lemma 1. *An optimal basic solution x^* of the LP relaxation (18)–(20) and (23), has at most two fractional components. Let $J_1 := \{l|x_l^* = 1\}$. If the basic solution has two fractional components x_i^* and x_j^*, supposing without loss of generality $c_j \leqslant c_i$, then $w_i + \sum_{l\in J_1} w_l \leqslant z^{LP}$ and the solution defined by $J_1 \cup \{j\}$ is feasible for the kKP.*

Approximation algorithm for kKP

```
1: Let $x^{LP}$ be an optimal solution of the LP relaxation of kKP
2: $J_1 := \{l|x_l^{LP} = 1\}, \quad F := \{l|0 < x_l^{LP} < 1\}$   // fractional
   variables
3: if $F = \emptyset$ then
4:    $z^A := z^{LP}$
5: end if
6: if $F = \{i\}$ then
7:    $z^A := \min\left\{\sum_{l\in J_1} w_l, \sum_{l\in\{i\}\cup R_i} w_l\right\}$ where $R_i$ is the set of the
      $k-1$ items with the smallest weight in $\{1,\ldots,m\}\backslash\{i\}$
8: end if
9: if $F = \{i,j\}$ with $c_j \leqslant c_i$ then
10:   $z^A := \min\left\{\sum_{l\in J_1\cup\{j\}} w_l, \sum_{l\in\{i\}\cup R_i} w_l\right\}$ where $R_i$ is the set of
      the $k-1$ items with the smallest weight in $\{1,\ldots,m\}\backslash\{i\}$
11: end if
```

Proposition 5. *The approximation algorithm for kKP described by the pseudocode is a 2-approximation algorithm and runs in O(m) time.*

Proof. This follows from an easy modification of a proof in [18,7]. □

Remark that the pricing problem is solved for each product $j = 1,\ldots,n$. Therefore, up to n columns can be added to the master problem at each iteration of the column-generation procedure.

Using the column-generation procedure outlined above, we can solve the master problem (LPM3) in reasonable time. There is no guarantee, however, that the solution found will be integral; if this is not the case we will proceed with a branch-and-bound algorithm.

3.3. Branch-and-bound

Branching is needed when the optimal solution to (LPM3) turns out not to be integral. However, as Savelsbergh points out in [35], a naive branching strategy may sometimes lead to a conflict between the variable used for branching and the column generated. This occurs when the column generated by the pricing problem contains a client forbidden by previous branching decisions. Hence, it is worth looking for a branching strategy compatible with the pricing problem.

Proposition 4 allows the use of a hybrid branching policy [2,35], that is, to perform branching using the basic formulation while working with the set-covering formulation. We will then fix a single variable (variable dichotomy) [35,42]. This variable-dichotomy branching strategy is exactly the branching scheme proposed by Ryan and Foster for set-partitioning master problems [2].

In the basic formulation, fixing x_{ij} to zero forbids product j to be offered to client i, and fixing x_{ij} to one requires product j to be offered to client i. In the set-covering formulation, this is done by adding (not explicitly) one extra constraint. Fixing x_{ij} to zero leads to $\sum_{p:i\in S_{pj}} z_{pj} = 0$ and fixing to one leads to $\sum_{p:i\in S_{pj}} z_{pj} = 1$. Hence, at node u of the search tree, let $H(u) \subseteq \{1,\ldots,n\}$ be the subset of products j for which there exists a non-empty set of clients $R_j^u \subseteq \{1, \ldots, m\}$ who must receive an offer of product j. Similarly, let $L(u) \subseteq \{1,\ldots,n\}$ be the set of products j for which there exists a non-empty set of clients $N_j^u \subseteq \{1,\ldots,m\}$ who cannot receive an offer of product j. The LP problem (LP$_u$) to be solved at a node u is the combination of the LP relaxation (LPM3) of (M3) and the following two constraints

$$\sum_{p:R_j^u \subseteq S_{pj}} z_{pj} = 1, \quad j \in H(u), \tag{24}$$

$$\sum_{p:N_j^u \cap S_{pj} \neq \emptyset} z_{pj} = 0, \quad j \in L(u). \tag{25}$$

The branching scheme resulting from the variable-dichotomy branching strategy is compatible with the pricing problem for it does not render the pricing problem more difficult. To meet the set of constraints (25), we set $x_i = 0, \ \forall i \in N_j^u$ when solving the pricing problem associated with the product $j \in L(u)$. Similarly, for the set of constraints (24), we set $x_i = 1, \ \forall i \in R_j^u$ when solving the pricing problem corresponding to the product $j \in H(u)$. The constraints (19) and (20) are updated and the remaining problem is still a kKP, of reduced size. Moreover, the inequality (12) of the restricted master will be changed into equality for the product $j \in H(u)$ to make sure that the solution of (LP$_u$) effectively offers product j to clients in R_j^u.

An upper bound at node u is provided by the optimal solution to the master problem (LP$_u$) or estimated by dualizing some constraints; details on these computations are available in [37].

4. Heuristics

In this section, we present eight heuristics for the product targeting problem. These heuristics are either variants of the algorithms used in practice for application in a bank or specifically developed based on the structure of the problem. The first is a variant of the algorithm developed by FORTIS [17]. It assumes that the variable cost and the profitability of the different clients are identical and that the campaign involves all products. The second heuristic is also based on these simplifying assumptions, but allows for choosing the products to be used for the campaign. The third is a procedure that successively solves a number of *Exact k-item knapsack problems* (E-kKP) (which are kKP's with an equality for the cardinality constraint (20)). It uses an approximation algorithm for solving the (E-kKP) to identify the best product to be offered as well as the selected set of clients at each iteration. The fourth heuristic is also an iterative algorithm, inspired by the Next-Product-To-Buy model used by Knott et al. [19] for an application in a retail bank. The fifth heuristic is a depth-first B& P heuristic. The sixth heuristic is a truncated call to a MIP solver and the seventh one is an LP-rounding heuristic. The final heuristic is a tabu-search algorithm. Throughout the section, we will call a client i *active* if he/she has not yet received M_i offers.

Author's personal copy

4.1. Heuristic 1: Average costs and revenues

Heuristic 1 is a variant of the algorithm developed by FORTIS. It uses the average cost and the average revenue for each product. These quantities are defined by $C_j := \frac{1}{m}\sum_{i=1}^{m} c_{ij}$ and $P_j := \frac{1}{m}\sum_{i=1}^{m} p_{ij}$. This heuristic ignores the selection of products for the campaign and simply imposes the minimum quantity O_j on the number of offers of each product j. Using a new decision variable $u_j :=$ number of clients that receive an offer for the product j, the simplified formulation used by FORTIS is the following:

$$
\begin{aligned}
\text{(M4)}\quad & \text{maximize} && \sum_{j=1}^{n}[(P_j - C_j)u_j - f_j] \\
& \text{subject to} && \sum_{j=1}^{n}[P_j - (1+R)C_j]u_j \geqslant (1+R)\left(\sum_{j=1}^{n} f_j\right), \\
& && C_j u_j \leqslant B_j, \quad O_j \leqslant u_j \leqslant m, \qquad j = 1,\ldots,n, \\
& && u_j \in \mathbb{N}, \quad j = 1,\ldots,n.
\end{aligned}
$$

Heuristic 1 is formally described by the following pseudocode.

Heuristic 1

```
1: for each product $j$, compute the average revenue $P_j$ and the
   average cost $C_j$
2: solve the resulting integer-programming formulation (M4)
3: sort products such that for two products $j$ and $k$, $j < k$ if and only if
   $P_j u_j \geqslant P_k u_k$
4: for each product $j$, sort clients such that for two clients $i$ and $l$, $i < l$
   if and only if $p_{ij} \geqslant p_{lj}$
5: consider the products following the order obtained in line 3; offer
   each product $j$ to the first $u_j$ active clients in the sequence given by
   line 4
6: keep this solution if it is feasible and if its total profit is greater
   than 0; otherwise, output the trivial solution
```

We remark that the problem formulated here is still NP-hard as it includes an integer knapsack problem [18] as a special case. On the other hand, (M4) does not take into account the third constraint (4) of the basic formulation (M1), that is, it does not enforce an upper bound on the number of products that can be offered to a client.

4.2. Heuristic 2: Average costs and revenues + selection

Heuristic 2 is an improvement of Heuristic 1; here the average revenue and the average cost per product are still used, but the choice of products to be offered during the campaign is taken into account. Using an additional binary variable y_j that equals 1 if product j takes part in the campaign and 0 otherwise, the integer-programming problem to be solved is

$$
\begin{aligned}
\text{(M5)}\quad & \text{maximize} && \sum_{j=1}^{n}[(P_j - C_j)u_j - f_j y_j] \\
& \text{subject to} && \sum_{j=1}^{n}[(P_j - (1+R)C_j)u_j - (1+R)f_j y_j] \geqslant 0, \\
& && C_j u_j \leqslant B_j, \quad j = 1,\ldots,n, \\
& && O_j y_j \leqslant u_j \leqslant m y_j, \quad j = 1,\ldots,n, \\
& && y_j \in \{0,1\}, \quad u_j \in \mathbb{N}, \qquad j = 1,\ldots,n.
\end{aligned}
$$

Notice that (M5) generalizes (M4), since (M4) arises when we set $y_j = 1$ for all j. This formulation also does not take into account the upper bound on the number of products that can be offered to a client. Unlike formulation (M4), however, (M5) does incorporate the choice of products to be offered during the campaign. Heuristic 2 is formally described by the following pseudocode.

Pseudocode of Heuristic 2

```
1: for each product $j$, compute the average revenue $P_j$ and the
   average cost $C_j$
2: solve the resulting integer programming formulation (M5)
3: sort products such that for two products $j$ and $k$, $j < k$ if and only if
   $P_j u_j \geqslant P_k u_k$
4: for each product $j$, sort clients such that for two clients $i$ and $l$, $i < l$
   if and only if $p_{ij} \geqslant p_{lj}$
5: consider the products following the order obtained in line 3; offer
   each product $j$ to the first $u_j$ active clients in the sequence given by
   line 4
6: keep this solution if it is feasible and if its total profit is greater
   than 0; otherwise, output the trivial solution
```

Notice that if the last line (line 6) is not included in the description of Heuristic 2 then its outcome may be an infeasible solution; the same is true for Heuristic 1. As an example, consider an instance with one product and three clients, where $c_{11} = 5$, $c_{21} = 2$ and $c_{31} = 2$. The revenues are $p_{11} = 10$, $p_{21} = 8$ and $p_{31} = 3$. The hurdle rate $R = 50\%$, the budget $B_1 = 6$, the lower bound $O_1 = 2$ and the fixed cost $f_1 = 0$. The application of either Heuristic 1 or Heuristic 2 will offer the product to the first and second client. This solution is not feasible, however, as the total cost is 7, which is greater than the budget.

4.3. Heuristic 3: Successive E-kKP

Heuristic 3 iteratively solves a number of instances of the Exact k-item knapsack problem (E-kKP). For a given product j, we define the following E-kKP:

$$
\begin{aligned}
\text{(E-kKPj)}\quad & \text{maximize} && \sum_{i=1}^{m}[p_{ij} - (1+R)c_{ij}]x_{ij} - (1+R)f_j \\
& \text{subject to} && \sum_{i=1}^{m} c_{ij}x_{ij} \leqslant B_j, \\
& && \sum_{i=1}^{m} x_{ij} = O_j, \\
& && x_{ij} \in \{0,1\}, \quad i = 1,\ldots,m.
\end{aligned}
$$

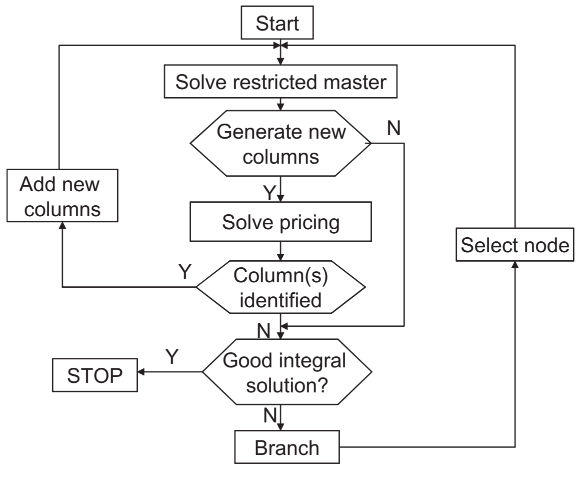

Fig. 1. Flow chart of the B&P heuristic (Heuristic 5).

Author's personal copy

We denote the associated k-item knapsack problem by kKPj. Remark that the LP relaxation of (E-kKPj) has a solution with either zero or two fractional components. The approximation algorithm for kKP is used as a 2-approximation algorithm for (E-kKPj).

Heuristic 3

```
1: $val := 0$, $Rate := 0$, $V := \{1,\ldots,m\}$    //$val$ is the objective value
2: for each $j = 1,\ldots,n$, solve (E-kKPj) and compute $val_j^E$
3: while there exists a product $j$ with $val_j^E > 0$ and $Rate + z_j^E \geqslant 0$ do
4:   select such product $j^*$ with the highest profit $val_{j^*}^E$
5:   $y_{j^*} := 1,\ val := val + val_{j^*}^E,\ Rate := Rate + z_{j^*}^E$
6:   for $i \in J_{j^*}^E$ do
7:     $x_{ij^*} := 1,\ M_i := M_i - 1$
8:     if $M_i = 0$ then
9:       $V := V \backslash \{i\}$
10:    end if
11:    forbid any further offer of the product $j^*$ to client $i$    //by setting $c_{ij^*}$ greater than $B_{j^*}$
12:  end for
13:  $O_{j^*} := 1,\ f_{j^*} := 0$
14:  update $B_{j^*}$
15:  for each $j = 1,\ldots,n$ satisfying $O_j \leqslant |V|$, solve (E-kKPj) and compute $val_j^E$
16: end while
17: $val := \max\left\{val; \max\left\{val_j^A : j = 1,\ldots,n,\ z_j^A \geqslant 0\right\}\right\}$
```

The pseudocode of Heuristic 3 describes an iterative procedure for finding a solution to the product targeting problem using a 2-approximate solution to (E-kKPj). We denote the objective value found by the approximation algorithm for the problem (E-kKPj) (respectively kKPj) by z_j^E (respectively z_j^A) and the set of solution components equal to 1 by J_j^E (respectively J_j^A). The quantity $val_j^E = \sum_{i \in J_j^E}[p_{ij} - c_{ij}] - f_j$ (respectively $val_j^A = \sum_{i \in J_j^A}[p_{ij} - c_{ij}] - f_j$) is the profit collected by offering product j to the clients selected by J_j^E (respectively J_j^A). Heuristic 3 works as follows: it first selects the product j^* with the highest positive profit $val_{j^*}^E$ such that the hurdle-rate constraint is not violated. Then, this product is offered to the set of clients $J_{j^*}^E$, the problem is updated and the procedure is repeated until no more product can be offered to clients; the quantity *val* represents the objective value of the solution obtained by the algorithm at each stage of its execution. Notice that after each iteration, the reconstructed problem is still a product targeting problem. The last line (line 17) chooses the best solution between the obtained solution and the solutions achievable by solving kKPj for each product j.

Heuristic 3 considers the clients for a single product. We focus mainly on the feasibility, which is why in the E-kKP$_j$ we consider the product targeting problem associated with product j, and instead of using the objective function defined by (1) we rather introduce the hurdle-rate constraint in the objective.

4.4. Heuristic 4: Next-Product-To-Buy

This heuristic is inspired by the Next-Product-To-Buy model proposed by Knott et al. [19] for an application in a retail bank. This heuristic is based on the probability r_{ij} that product j is the next product bought by client i.

Heuristic 4

```
1: for $j = 1,\ldots,n$ do
2:   sort and re-index clients such that $i < l$ if and only if $r_{ij} \geqslant r_{lj}$
3:   for $i = 1,\ldots,m$ do
4:     if $i$ is active and there is enough budget then
5:       $x_{ij} := 1$, $M_i := M_i - 1$
6:       if $M_i = 0$ then
7:         $i$ becomes inactive
8:       end if
9:     else
10:      $x_{ij} := 0$
11:    end if
12:  end for
13: end for
14: keep this solution if it is feasible and if its total profit is greater than 0; otherwise, output the trivial solution
```

4.5. Heuristic 5: B&P heuristic

This is a depth-first heuristic based on the B&P approach. The goal of this heuristic is to find a feasible and high-quality solution as quickly as possible. To achieve this, Heuristic 5 performs a partial traversal of the nodes corresponding to a particular branching decision within our B&P approach. In the heuristic, we branch on a variable with value closest to 1 and we have preference for following the branch where that variable is set to 1.

Fig. 1 describes the precise steps to follow. To start the algorithm, we need an initial restricted master problem with a feasible LP relaxation. Therefore, for each product, we solve the kKP problem (kKPj) with the 2-approximation algorithm. If a feasible solution is obtained, it is added to the master problem. We speed up our heuristic using ideas proposed by Vance et al. [41]. At the root node, the restricted master problem is solved only once (so without using column generation). Next, we branch. As the goal of this heuristic is to find a good feasible solution and not necessarily an optimal one, we do not have to branch in such a way that the solution space is divided evenly. Thus, using the hybrid branching strategy, we branch on the fractional variable x_{ij} closest to 1 and set $x_{ij} = 1$ in one branch and $x_{ij} = 0$ in the other branch. If the solution at the root node is integral we branch on a variable $x_{ij} = 1$. In case of a tie, we select the variable with the highest revenue p_{ij}. With this branching rule, we are more likely to find a good solution in the node with $x_{ij} = 1$, and we investigate that node first. At each node, an upper bound on the number of calls to the pricing problem is used to stop the column generation; our implementation uses a bound of $20 \times n$, which was chosen based on some preliminary experiments. Moreover, a target value equal to 110% of the value obtained at the root node is used at each node to stop the column-generation procedure. That is, if at a given node the value of the objective function becomes larger than the target value, the column generation is stopped. We stop the algorithm when we find a feasible solution with an objective value greater than or equal to the target value. In other words, the algorithm continues when it finds a feasible solution with a value less than the target value. Similar heuristics have been used successfully in a variety of applications, including raw-materials logistics planning [27] and the airline crew pairing problem [2,41].

Author's personal copy

4.6. Heuristic 6: Truncated MIP

This is a truncated call to the MIP solver CPLEX: the solver is used to solve (M2) and is interrupted when a time limit is reached. For our implementation, we use 1 hour as time limit. This limit is set according to the average time spent by the other heuristics on large instances. Consequently, after at most 1 hour, CPLEX stops and either has already found a feasible solution, in which case the best feasible solution obtained so far is output by the heuristic, or CPLEX has not yet found a feasible solution and the heuristic outputs the trivial solution.

4.7. Heuristic 7: LP rounding

This heuristic is based on the LP relaxation of (M2). Given a solution x_{ij}, $i = 1,\dots,m$, $j = 1,\dots,n$ to (M2), there are two possibilities: either that solution is integer, in which case it is the output of the heuristic, or it is fractional and we round it up to integer values as follows. The product j is used if $y_j \geqslant \frac{1}{2}$ and we set x_{ij} to 1 if it is greater than or equal to $\frac{1}{2}$. On the other hand, if $y_j < \frac{1}{2}$ then we set $y_j = 0$ and all corresponding x-variables also receive the value 0. If this solution is infeasible or has a non-positive total profit, then the heuristic outputs the trivial solution. A time limit of 1 hour is imposed for solving the LP relaxation. When the limit is reached, we use the best solution found as solution of the LP.

4.8. Heuristic 8: Tabu search

In this section, we present a tabu-search algorithm. Our rationale behind the choice for tabu search is the fact that it has been successfully applied to solve the generalized assignment problem [22], which is a special case of the problem studied in this paper. The global structure of the algorithm is described by the pseudocode below. In the description, we use the function $g(x,y)$ to represent the objective function (1) of our problem and $\mathcal{N}_k(x,y)$ to identify a set of solutions that constitutes the neighborhood of a given solution (x,y) at iteration k. At any time, (x^*,y^*) is the best feasible solution obtained so far.

Tabu search

```
1: choose an initial solution $(x,y)$ and set $x^* = x$, $y^* = y$ and $k = 0$
2: set $k = k + 1$ and consider the first solution $(x',y') \in \mathcal{N}_k(x,y)$ with $g(x^*,y^*) < g(x',y')$ that is not tabu
3: if $(x',y')$ exists, then set $x^* = x = x'$, $y^* = y = y'$
4: else choose $(x',y')$ to be the best non-tabu solution in $\mathcal{N}_k(x,y)$ and set $x = x'$, $y = y'$
5: update the tabu list
6: if a stopping condition is met, then stop; else goto 2
```

Our algorithm differs from a standard tabu-search implementation [1] in two main points. First, in line 2 we do not necessarily investigate the entire neighborhood $\mathcal{N}_k(x,y)$: when we find a solution with a better objective value than the currently best solution, we do not investigate the remaining solutions in $\mathcal{N}_k(x,y)$ but rather update the current solution (and the incumbent) immediately. The main reason for this is the fact that $\mathcal{N}_k(x,y)$ is very large. The second difference is the absence of aspiration conditions (conditions that allow a tabu solution to become the new current solution). Below, we describe in more details the most important steps of the tabu-search algorithm.

Initial solution procedure

```
1: $val := 0$, $exp := 0$, $V := \{1,\dots,m\}$, $S := \emptyset$
2: for each $j = 1,\dots,n$ and for each $i \in V$ compute $NPP_{ij} := \frac{p_{ij}-c_{ij}}{c_{ij}}$
3: for all products $j \notin S$ do
4:   sort the clients in $V$ in non-increasing order of their $NPP_{ij}$
5:   $C_j$:= the cost of offers to the first $O_j$ clients in the sorted list
6:   $P_j$ := the revenue of offers to the first $O_j$ clients in the sorted list
7:   $PR_j := P_j - C_j - f_j$
8: end for
9: select $j^* \notin S$ with the highest $PR_{j^*}$ and satisfying $C_{j^*} \leqslant B_{j^*}$ and $val + P_{j^*} \geqslant (1+R)(exp + C_{j^*} + f_{j^*})$
10: if $j^*$ exists and $PR_{j^*} > 0$ then
11:   $S := S \cup \{j^*\},\ y_{j^*} := 1,\ val := val + P_{j^*},\ exp := exp + C_{j^*} + f_{j^*}$
12:   for each client $i$ amongst the first $O_{j^*}$ clients in the sorted list do
13:     $x_{ij^*} := 1,\ M_i := M_i - 1$
14:     if $M_i = 0$ then
15:       $V := V \backslash \{i\}$
16:     end if
17:   end for
18:   goto 3
19: end if
20: for each active client $i$ do
21:   for each $j \in S$ do
22:     if $p_{ij} > c_{ij}$ and the offer of product $j$ to client $i$ leads to a feasible solution, make that offer and update the current solution
23:   end for
24: end for
```

4.8.1. Initial solution

The initial solution used by the tabu search is obtained using the procedure proposed in [40]. The pseudocode of this procedure is given as 'Initial solution procedure'. This algorithm runs in polynomial time, and so from Proposition 2, there is no guarantee that it will always produce a non-trivial feasible solution. For the benchmark instances used in Section 5, however, this procedure turns out to be quite effective, as it produces a non-trivial feasible solution to all the instances.

4.8.2. Neighborhood $\mathcal{N}_k(x,y)$ and selection of (x',y')

Let (x,y) be the current solution at iteration k. The neighborhood $\mathcal{N}_k(x,y) = \mathcal{N}_k^1(x,y) \cup \mathcal{N}_k^2(x,y) \cup \mathcal{N}_k^3(x,y)$ of (x,y) is the union of three sets. The set $\mathcal{N}_k^1(x,y)$ contains the feasible solutions (x',y') obtained from (x,y) by considering two clients i_1 and i_2, and a product j satisfying $y_j = 1,\ x_{i_1j} = 1$ and $x_{i_2j} = 0$; then we set $x'_{i_1j} = 0$ and $x'_{i_2j} = 1$. The second set $\mathcal{N}_k^2(x,y)$ contains the feasible solutions (x',y') obtained from (x,y) by considering two clients i_1 and i_2, and two products j and ℓ satisfying $y_j = y_\ell = 1,\ x_{i_1j} = 1,\ x_{i_2j} = 0,\ x_{i_1\ell} = 0$ and $x_{i_2\ell} = 1$; then we set $x'_{i_1j} = 0,\ x'_{i_2j} = 1,\ x'_{i_1\ell} = 1$ and $x'_{i_2\ell} = 0$. The last set, $\mathcal{N}_k^3(x,y)$, contains the feasible solutions (x',y') obtained from (x,y) by permuting two clients i_1 and i_2; that is, for all products j we set $x'_{i_1j} = x_{i_2j}$ and $x'_{i_2j} = x_{i_1j}$. Notice that a new solution (x',y') is in $\mathcal{N}_k(x,y)$ only if it is feasible.

The three sets $\mathcal{N}_k^1(x,y)$, $\mathcal{N}_k^2(x,y)$ and $\mathcal{N}_k^3(x,y)$ are examined successively in this order. Once a non-tabu solution (x',y') is found with better objective than the incumbent, the iteration is interrupted and the current solution is immediately updated. If such a non-tabu solution is not found then the iteration halts with the best non-tabu solution in $\mathcal{N}_k(x,y)$, which subsequently becomes the new solution. In the implementation, when this happens five

Author's personal copy

times consecutively (finding a non-improving solution), we apply a diversification procedure to the current solution (the number five was chosen based on preliminary experiments). This diversification procedure looks for a feasible solution (x',y') obtained from the current solution (x,y) by replacing a product j currently in the campaign ($y_j = 1$) by another product ℓ not used in the current solution ($y_\ell = 0$), so $y'_j = 0$ and $y'_\ell = 1$. Of course, from the set of clients $S = \{i|x_{ij} = 1$ or i is still active$\}$, we need to be able to find at least O_ℓ clients who can receive an offer of product ℓ.

4.8.3. Tabu list

From iteration k to the next iteration $k + 1$, we move from a solution (x,y) to a solution (x',y'). As mentioned above, this new solution can be obtained in different ways; it is either in $\mathcal{N}^i_k(x,y)$ $(i = 1,2,3)$ or obtained after a diversification procedure. In any case, the reverse modification leading back from (x',y') to (x,y) is considered as tabu for the next 20 iterations with the same neighborhood.

4.8.4. Stopping conditions

In our implementation, we combine two stopping conditions. We stop the tabu search when either a time limit of 1 hour is reached or when the diversification procedure is unable to find any other non-tabu feasible solution.

5. Computational experiments

All algorithms have been coded in C using Visual Studio C++ 2005; all the experiments were run on a Dell Optiplex *GX*620 personal computer with Pentium R processor with 2.8 GHz clock speed and 1.49 GB RAM, equipped with Windows XP. CPLEX 10.2 was used for solving the linear programs. A number of issues related to the implementation of the algorithms is explained in more detail in [37]. Below, we first provide some details on the generation of the datasets and subsequently, we discuss the computational results.

5.1. Generating test instances

The instances used for the experiments are randomly generated, with cost c_{ij} randomly generated from the set $\{1,2,3\}$ and the return to the firm $p_{ij} = r_{ij}DFV_{ij}$ is an integer randomly generated between 0 and 16. The choice of these figures is guided by examining the real-life data used by Cohen [8]. The corporate hurdle rate R belongs to the set $\{5\%,10\%,15\%\}$. There are six different values for the number of clients m: these are 100, 200 and 300 clients (small size; S1, S2 and S3), 1000 and 2000 clients (medium size; M1 and M2) and 10,000 clients for instances of large size (L). For each number of clients, we have three different numbers of products n; these are 5, 10 and 15 products. In total, we have 54 groups of instances, as described in Table 1.

For each group, we generate a minimum-quantity commitment bound O_j as a random integer selected between $\left\lceil \frac{\sum_i M_i}{n} \right\rceil$ and $\left\lceil \frac{2\sum_i M_i}{n} \right\rceil$. We consider three values for the budget B_j, namely a random integer chosen between $O_j \frac{\sum_i c_{ij}}{m}$ and $2\frac{\sum_i c_{ij}}{n}$ and the two extreme budgets $\left\lfloor O_j \frac{\sum_i c_{ij}}{m} \right\rfloor$ and $\left\lceil 2\frac{\sum_i M_i}{n} \frac{\sum_i c_{ij}}{m} \right\rceil$. The fixed cost f_j is a random integer between $\frac{O_j}{2m(1+R)}\sum_i[p_{ij} - (1+R)c_{ij}]$ and $\frac{O_j}{m(1+R)}\sum_i[p_{ij} - (1+R)c_{ij}]$. These bounds have been chosen in such a way that the instances become feasible and consistent. We generate one instance with a small upper bound M_i for each client, selected between 1 and $\frac{n}{5}$ (denoted by s) and another one with a large random upper bound M_i for each client, selected between $\left\lceil \frac{n}{3} \right\rceil$ and $\left\lceil \frac{2n}{3} \right\rceil$ (denoted by ℓ). In conclusion, we have $3 \times 2 \times 54 = 324$ test instances in total.

Since Heuristic 4 requires as input r_{ij} (and not p_{ij}), we have decided to generate for each of the 324 test instances, three different sets of values for r_{ij}. The first set is generated randomly from $]0,1[$, while the second is also randomly generated in $]0,1[$ but proportional to c_{ij}, and the last is inversely proportional to p_{ij}. Note that by choosing r_{ij} in $]0,1[$, we discard certainty. These $3 \times 324 = 972$ test instances are solved using Heuristic 4. All instances can be found at http://www.econ.kuleuven.be/public/NDBAC96/promotion_campaigns.htm.

Table 1
Size of the generated inputs.

Group	Rate R (%)	Number of clients (m)	Number of products (n)
S1	5, 10 or 15	100	5, 10 or 15
S2	5, 10 or 15	200	5, 10 or 15
S3	5, 10 or 15	300	5, 10 or 15
M1	5, 10 or 15	1000	5, 10 or 15
M2	5, 10 or 15	2000	5, 10 or 15
L	5, 10 or 15	10,000	5, 10 or 15

5.2. Computational results

In this section, we report on different implementations of the column-generation procedure and the B&P algorithm (Section 5.2.1), we provide a comparison of the basic model and the set-covering formulation (Section 5.2.2) and we examine the performance of the heuristics (Section 5.2.3). Throughout this section, the computation time (Time) is expressed in seconds.

5.2.1. Different implementations of the column-generation procedure and the B&P algorithm

Table 2 compares two implementations of the column-generation procedure for solving the LP relaxation of the set-covering formulation. In this table, each cell is the average of nine values. The first implementation (Single column) adds a single column per product to the master problem. For each product, the approximation algorithm is first used to solve the pricing problem; if a column is identified, it is added to the master problem. If no column is found for any product, on the other hand, we successively use the exact algorithm for solving the pricing problem for each product. Once a column is identified, we stop and do not use the exact algorithm for the remaining products. The LP problem obtained is then solved using CPLEX. The second implementation (Multiple columns) can add up to five different columns per product, but only in the case where the first column is identified using the approximation algorithm. For each product, the approximation algorithm is first used to solve the pricing problem; if a column is identified, we generate at most four additional columns, as follows. We randomly select a single client i_1 with $x_{i_1} = 1$, we fix $x_{i_1} = 0$ and the pricing problem is re-solved using the approximation algorithm. If a feasible solution with negative reduced cost is obtained, the corresponding column is added, a new client i_2 with $x_{i_2} = 1$ in the above column is randomly chosen and we set $x_{i_1} = x_{i_2} = 0$. The pricing problem is re-solved and this procedure is repeated until either five columns are added or the pricing problem (with these additional constraints) is either infeasible or the solution has a non-negative reduced cost.

In Table 2, we report for the approximation algorithm (Approx.), the exact algorithm (Exact) and CPLEX (CPLEX) the number of calls (nr.) and the total time spent (Time). We observe that the average time spent is 0.00081 second (respectively 0.01539 second) per

Author's personal copy

Table 2
Comparison of column-generation procedures for solving the LP relaxation of the set-covering formulation.

Group	n	M	Single column						Multiple columns					
			Approx.		Exact		CPLEX		Approx.		Exact		CPLEX	
			Nr.	Time	Nr.	Time	Nr.	Time	Nr.	Time	Nr.	Time	Nr.	Time
S1	5	s	2643	2.05	7	0.03	597	7.42	11,788	3.17	7	0.04	545	10.98
		ℓ	881	0.76	22	0.74	233	1.20	5056	1.41	35	1.22	266	3.69
	10	s	2902	2.37	37	0.18	412	6.47	16,020	4.28	140	0.81	455	16.14
		ℓ	1325	1.08	24	0.70	179	2.17	6880	1.72	19	0.61	185	7.02
	15	s	2896	2.30	126	0.42	319	5.83	14,601	3.96	226	0.87	337	12.02
		ℓ	1054	0.96	72	2.36	108	1.24	5449	1.61	68	2.36	167	3.86

call of the approximation algorithm (respectively the exact algorithm) for the "Single column" implementation, which is greater than the corresponding average time for the "Multiple columns" implementation, 0.00027 second and 0.0114 second, respectively. However, the increasing number of columns and the increasing average time of using CPLEX for solving intermediary LP problems (0.02748 second for the second implementation compared to 0.01317 second for the first implementation) make the overall time of the "Multiple columns" implementation larger than the overall time of the first implementation.

We have implemented a local-search algorithm to generate multiple columns per product (a similar technique has been used by Van Den Akker et al. [39] for application in parallel-machine scheduling), but the results were not better than those reported above. In the sequel, the LP relaxation of the set covering is solved using the first implementation.

Table 3 compares three different strategies of traversing the branching tree in the B&P algorithm: breadth first, depth first, and best first. The main difference between these three strategies is observed after the branching, when selecting the child to investigate first [35].

The comparison is made for 36 instances from group S1. We clearly see from Table 3 that the breadth-first B&P algorithm is dominated either by the depth-first B&P algorithm or by the best-first branch-and-price B&P. Note that the number of nodes might be important here, as more nodes is likely to imply more columns generated. The generation of these columns is time-consuming, and too many columns may sometimes lead to a memory problem for our B&P algorithm. Observe that for $n = 10$ and $M = s$, the best-first strategy explores more nodes than the depth-first strategy; this is due to a single instance for which the best-first strategy has to investigate a very high number of nodes in order to find a guaranteed optimal solution.

5.2.2. A comparison of the basic model and the set-covering formulation

Table 4 shows a comparison between the LP relaxation of the basic formulation and of the set-covering formulation. Here also, each cell is the average of nine values. The results of the group S3 are not reported in this table because there are instances in that group for which we do not know the optimal value, while for group S1 and group S2 we have obtained an optimal solution using either the MIP solver CPLEX for solving (M2) or the B&P algorithms (see Table 5). In Table 4, LP-Gap is a percentage $\frac{z_{LP}-z_{OP}}{z_{OP}} \times 100\%$, where z_{OP} is the optimal objective value and z_{LP} is the objective value of the LP relaxation.

Table 3
Comparison of different tree-traversal strategies for the B&P algorithm.

	n	M	Breadth first		Depth first		Best first	
			Time	Nr. nodes	Time	Nr. nodes	Time	Nr. nodes
S1	5	s	10131.57	178	7365.78	74	2031.13	26
		ℓ	64.68	8	38.75	6	72.85	2
	10	s	5057.54	936	1141.08	210	3035.86	410
		ℓ	9967.78	754	20172.64	1395	8587.53	208

Table 4
LP relaxation of the basic formulation and the set-covering formulation.

Group	n	M	Basic formulation				Set-covering formulation			
			LP-Gap	Time			LP-Gap	Time		
				Min	Mean	Max		Min	Mean	Max
S1	5	s	8.33	0.06	0.16	0.31	5.05	0.44	9.56	24.47
		ℓ	6.51	0.08	0.30	0.55	0.03	1.05	2.20	3.86
	10	s	3.11	0.05	0.13	0.23	0.62	6.05	9.02	14.83
		ℓ	3.26	0.08	0.26	0.50	0.18	1.81	3.65	7.76
	15	s	2.96	0.05	0.14	0.27	0.45	5.51	8.59	10.66
		ℓ	1.28	0.06	0.25	0.53	0.07	1.69	4.68	11.36
S2	5	s	2.51	0.11	0.31	0.67	0.78	152.18	378.81	852.84
		ℓ	12.80	0.19	0.93	2.31	5.47	0.47	36.39	108.04
	10	s	3.37	0.11	0.46	0.91	0.21	105.60	220.75	375.74
		ℓ	10.51	0.20	1.06	2.28	0.80	0.45	92.01	176.33
	15	s	2.47	0.11	0.43	0.83	0.15	80.75	190.46	271.88
		ℓ	5.00	0.16	1.12	2.23	0.00	4.73	49.48	119.54

Author's personal copy

Table 5
Basic formulation and B&P algorithms.

Group	n	M	Basic formulation				Depth first				Best first			
			Solved	Time			Solved	Time			Solved	Time		
				Min	Mean	Max		Min	Mean	Max		Min	Mean	Max
S1	5	s	9	1.72	57.39	190.13	9	103.69	7365.78	21135.56	9	279.97	2031.13	4999.85
		ℓ	9	0.11	3.91	12.28	9	11.95	38.75	75.78	9	11.20	72.85	168.86
	10	s	9	1.33	27.90	114.84	9	24.98	1141.08	11052.67	9	578.39	3035.86	5493.34
		ℓ	9	0.17	6.05	35.95	9	61.72	20172.64	35798.08	9	74.16	8587.53	16683.14
	15	s	9	1.50	22.57	74.73	8	17259.00	39845.53	62432.07	8	3038.83	19521.5	36004.17
		ℓ	9	0.11	3.22	14.03	9	23.81	31117.1	62210.36	9	23.56	18014.18	36004.79
S2	5	s	9	0.23	253.47	1036.68	9	2862.80	36725.30	62143.55	7	2862.80	36382.61	51143.56
		ℓ	8	0.44	138.18	1094.44	9	358.72	12125.45	36017.63	9	251.53	15699.19	36101.53
	10	s	8	3.58	2246.54	16249.26	9	518.71	36138.39	57844.56	8	422.7	36696.35	57842.13
		ℓ	9	0.33	20.40	124.16	9	336.17	12010.04	36021.12	9	631.33	18334.06	36036.80
	15	s	7	3.59	1811.47	14259.19	9	1575.14	12545.51	36061.38	7	444.75	36209.22	64422.08
		ℓ	9	0.30	9.47	53.83	9	458.99	5742.13	11052.67	9	179.95	18183.22	36186.49
S3	5	s	8	0.48	241.79	1311.32	–	×	×	×	–	×	×	×
		ℓ	9	2.63	103.94	596.19	–	×	×	×	–	×	×	×
	10	s	4	12.81	823.01	3213.22	–	×	×	×	–	×	×	×
		ℓ	9	0.75	809.26	5646.58	–	×	×	×	–	×	×	×
	15	s	5	1.53	127.03	370.22	–	×	×	×	–	×	×	×
		ℓ	9	0.66	172.78	1490.54	–	×	×	×	–	×	×	×

Table 4 confirms the theoretical result obtained in Section 3.1 that the set-covering formulation is at least as strong as the basic formulation: that is, it gives a solution very close to the optimal solution compared to the solution obtained by the basic formulation. This difference is reflected by LP-Gap for the set-covering formulation, which is less than 1% for many instances. However, this quality of the LP relaxation of the set-covering formulation comes with a relatively high computation time. Notice that this computation time increases with the size of the problem (number of clients). Table 4 explicitly displays the trade-off between the solution quality and the computation time. Notice also that the computation time reported in Table 4 for the set-covering formulation is not necessarily the computation time spent by the B&P algorithm at the root node as we stop solving the LP at each node of the branching tree (and therefore at the root node) when we go through 30 iterations without an improvement of the objective value.

Table 5 displays the results of the exact algorithms for the product targeting problem. The basic formulation (M2) is solved using the MIP solver of CPLEX 10.2. In Table 5, the minimum, the mean (average) and the maximum computation time as well as the number of instances solved (Solved) are recorded. Notice that each cell is based on nine instances. For group S1, CPLEX was able to solve every instance. However, the computation time increases when we move to group S2. For that group, CPLEX could not solve four instances, due to memory problems. Amongst these four instances, three have a small upper bound per client M_i and only one was an instance with large upper bound. For group S3, up to ten instances were not solved by CPLEX. In that group, all unsolved instances have a small upper bound M_i. The above observations coupled with an analysis of the computation time given in Table 5 for the basic formulation lead to the following conclusions. Computation time seems to increase exponentially with the size of the instance (number of clients). Moreover, instances with small upper bound M_i seem to be harder to solve than instances with large upper bound.

Table 5 also shows the computation times obtained by the B&P algorithms. We concentrate only on depth-first and best-first versions, as Table 3 has shown that these two dominate the breadth-first. Using the depth-first B&P, we are able to solve all the instances in the group S1 and the group S2 except for one instance in S1 while the best-first B&P algorithm fails to find an optimal solution to six instances, one in the group S1 and five in S2. It is clear that using the MIP solver of CPLEX 10.2 for solving the basic formulation (M2) (enhanced with the default cutting planes) is much faster than the B&P approaches. Although the pricing problem is solved fast and LP-Gap is smaller at the root node as shown in Tables 2 and 4, the number of nodes in the branching tree can be excessive, as witnessed in Table 5. We do not report results for the B&P algorithms applied to the instances in S3 because of excessive run times.

5.2.3. The heuristics

In Table 6, the outcomes of the heuristics are exhibited for the set of instances S3, M1, M2 and L. The groups S1 and S2 are not considered because we have obtained optimal solutions to all instances in these groups. Here, Gap is a percentage $\frac{z_{UB}-z_{AP}}{z_{UB}} \times 100\%$, where z_{UB} is either the optimal objective value, found by solving (M2) with the MIP solver CPLEX, or an upper bound of that value obtained by solving the LP relaxation of the problem and z_{AP} is the objective value obtained by the heuristic. For some instances, the LP relaxation was not solved to optimality, and we impose a time limit of 1 hour and consider the value obtained at that time as z_{UB}, which is an upper bound since we use CPLEX's dual simplex. The Gap is computed only for instances for which a non-trivial feasible solution is found, so when the heuristic finds a feasible solution with $z_{AP} > 0$. Feas is the percentage of feasible solutions with positive objective value. Unlike the previous tables, here each cell is the average of 18 values, corresponding with three values for R, three values for the budget B and the two values of M. The experimental results of Heuristics 1 and 4 are not displayed in Table 6 because they could not produce any non-trivial feasible solution. A time limit of 1 hour is imposed for each heuristic.

The results of Table 6 confirm the existence of a trade-off between computation time and solution quality. For the instances in S3, Heuristic 2 (H2) did not find any non-trivial feasible solution while Heuristic 3 (H3) gives a feasible solution with strictly positive objective value for all the test instances. Moreover, the Gap reported for each instance is less than 10% and the computation time is at most 10 seconds. The solutions provided by Heuristic 5 (H5) are of good quality, with Gap less than 9%. However, unlike Heuristic 3, Heuristic 5 requires much more time. Heuristic 6 (H6) provides feasible solutions with strictly positive objective value for each instance in S3. Although it takes more time than Heuristic 3

Author's personal copy

Table 6
Comparison of the different heuristics.

n		S3			M1			M2			L		
		5	10	15	5	10	15	5	10	15	5	10	15
H2	Gap	–	–	–	–	81.61	–	97.77	62.21	–	–	–	62.81
	Time	–	–	–	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.04	0.04
	Feas	–	–	–	0.00	2.00	0.00	2.25	2.25	0.00	0.00	0.04	1.50
H3	Gap	8.46	8.50	9.17	11.20	13.23	14.46	13.10	13.15	12.31	34.04	33.52	37.80
	Time	2.80	9.63	1.66	146.60	394.54	304.32	2125.62	2231.14	1195.45	3573.92	3143.41	3353.81
	Feas	100	100	100	100	100	100	100	100	100	100	100	100
H5	Gap	8.15	8.17	8.30	8.72	13.04	14.43	12.40	12.84	12.01	13.60	24.54	34.26
	Time	1498.87	2455.66	2301.48	2441.74	3318.04	3274.59	1978.80	3565.28	3384.92	3730.05	3711.15	3634.33
	Feas	100	100	100	100	100	100	100	100	100	100	100	100
H6	Gap	6.91	4.11	4.59	4.44	2.97	23.11	2.77	17.74	23.20	14.19	13.11	–
	Time	320.54	1452.93	1653.53	1845.20	2414.50	3018.14	2048.79	3132.60	3523.35	3667.33	3763.12	3803.61
	Feas	100	100	100	100	100	77.78	100	83.33	77.78	61.11	5.56	0.00
H7	Gap	0.54	0.00	0.00	0.00	0.00	0.00	0.00	0.39	0.27	0.00	2.75	51.10
	Time	0.47	1.20	3.13	8.63	28.30	64.26	17.63	124.36	298.37	558.66	2974.50	3558.28
	Feas	11.11	5.56	22.22	11.11	11.11	16.67	11.11	22.22	16.67	22.22	5.56	11.11
Init.	Gap	19.20	17.85	18.34	17.90	16.20	18.36	18.92	19.23	17.84	26.22	24.86	25.65
	Time	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.03	0.28	0.32	0.36
	Feas	100	100	100	100	100	100	100	100	100	100	100	100
H8	Gap	6.86	6.52	7.76	7.22	8.54	7.60	9.75	9.58	9.11	10.86	11.04	10.23
	Time	1.17	0.60	0.79	16.10	11.62	15.36	56.58	50.33	67.24	1268.30	1347.23	1149.28
	Feas	100	100	100	100	100	100	100	100	100	100	100	100

and the tabu search, the Gap is the smallest among the heuristics able to solve all the instances. Heuristic 7 (H7) has a competitive computation time (less than 4 seconds) but provides very few feasible solutions with strictly positive objective value (less than 23%). The tabu-search algorithm (H8) produces non-trivial feasible solutions to all the test instances. Moreover, the Gap reported for each instance is less than 7.77% and the computation time is at most 1.17 seconds. Notice that for each instance, the initial solution is feasible and is obtained almost instantaneously (computation time is 0 second). The average smallest gap is 17.85% for instances with $n = 10$ while the final output of the tabu search has a gap of 6.52%, indicating an improvement of more than 11% over the initial solution.

For medium and large size instances in M1, M2 and L, we have a confirmation of the difficulty experienced by Heuristic 2 (average costs and revenues + selection) in finding a feasible solution with strictly positive objective value. Moreover, when it finds a feasible solution, the Gap is very large (more than 60%). The only good news comes from the computation time, which increases very slowly. H3 (successive E-kKP), by contrast, behaves very well for large instances: all the instances are solved with Gap less than 15% for group M1 and group M2, and less that 38% for the instances with 10,000 clients. This good quality of the solutions is coupled with an increased running time. H5 (B&P heuristic) finds non-trivial feasible solutions to all instances, with a smaller Gap than H3 but with higher running time. The solutions proposed by H6 (truncated MIP) are usually close to the optimal solution (judging from the Gap), but for large instances, this heuristic is quite limited in terms of the number of non-trivial feasible solutions it can find. We see in Table 6 that for instances in the set L, H6 provides at most 62% of feasible solutions when $n = 5$ and at most 6% for $n = 10$. When $n = 15$, this heuristic fails to find any non-trivial feasible solution. LP rounding (H7) provides good solutions from time to time, but is very limited by the number of feasible solutions it is able to find (at most 23%). As for the tabu search (H8), it has an excellent behavior: all the instances are solved with Gap less than 10% for group M1 and group M2 in about one minute, and with Gap less than 11.5% for the instances with 10,000 clients with average time less than 23 minutes. As observed in Table 6, the initial solution procedure is very fast and always provides a feasible solution. Further, there is a difference of at least 7% between the gap of the initial solution and of the tabu search.

To summarize the comparison between the heuristics reported in Table 6, we formulate the following advice. If the solution quality is the only criterion to be taken into account, the use of the successive E-kKP, the B&P heuristic, the truncated MIP or the tabu search is advised if an efficient exact algorithm is not available. When dealing with large instances in the setting where both the computation time and the solution quality are relevant, we strongly recommend the use of the successive E-kKP, the B&P heuristic or the tabu-search algorithm, with a clear preference for the latter.

In practice, there may be instances with millions of clients [8]. We believe that the use of the tabu search, possibly with more running time, may provide a good feasible solution. In practice nowadays when dealing with such instances, one often-used technique is to cluster the clients into groups (these techniques are also called *aggregate techniques* [8]). Next, clients within a group are treated as identical. We point out here that a variant of (M1) provides a model for this situation where x_{ij} is then defined as the number of clients of group i that receive the product j and the parameters c_{ij} and p_{ij} are redefined accordingly.

6. Extensions

In this section, we briefly touch upon some extensions regularly encountered in financial institutions. The first is related to purchase sequences: there may be a logical succession due to complementary goods – e.g., a VCR is bought after a television set is acquired (see [23,31]). An extension of our model towards sequentially ordered products would probably need to entail multiple time buckets and is an avenue for future work.

A second extension is related to the maximum number of offers that can be made for a given product during the campaign (see, e.g., Cohen [8]). This constraint is easily incorporated into the basic

Author's personal copy

formulation by changing Eq. (5) into $\sum_{i=1}^{m} x_{ij} \leqslant N_j y_j$ for each product j, where N_j represents the maximum number of offers allowed for the product j. The set-covering formulation (M3) remains valid under a slight modification in the definition of S_{pj}, such that it contains at least O_j clients and at most N_j clients. The major effect appears in the pricing problem: instead of solving a k-item knapsack problem kKP (18)–(21), we have to solve a $k_1|k_2$-item knapsack problem, which is a k-item knapsack problem in which the cardinality constraint (20) is replaced by

$$k_1 \leqslant \sum_{i=1}^{m} x_i \leqslant k_2.$$

The dynamic-programming recursion for the k-item knapsack problem is also valid for the $k_1|k_2$-item knapsack problem if we replace k by $\min\{k, k_2\}$, while the 2-approximation algorithm is valid without any modification. This is due to the fact that a solution to the $k_1|k_2$-item knapsack problem has at most two fractional components. This second constraint is also easily incorporated in all the heuristics defined above after some minor modifications.

A price bundle is a selling offer of several services at a special price, or with a free gift etc., [15,28]. The model presented can be extended to take into account such 'package deals': each bundle can also constitute an individual 'product', next to the separate products. A new type of interdependence between the products in the model now arises (namely, between the bundle on the one hand and the constituent products on the other hand). We leave this extension open for possible future work.

Another research direction that may be pursued in the future is an extension of this work to multichannel offers, where the products can be offered through many different channels and each channel has its own constraints (e.g., minimum and maximum number of offers through this channel [8]).

7. Summary and conclusions

In this text, we have modeled a product targeting problem, taking into account both business constraints and customer preferences and specificities. Although the inspiration for the work undertaken draws from a financial institution, application in other sectors are also possible. A key requirement, obviously, is the availability of the appropriate input data. Our work shows that the problem is strongly NP-hard and that it is unlikely that a constant-factor approximation algorithm can be proposed for solving this problem. We have also presented a set-covering formulation in which each product is associated with a subset of clients (which can be empty) in the optimal solution and developed a branch-and-price algorithm for solving it. We have shown that this last formulation is stronger than the basic formulation. These two formulations are used to produce optimal solutions. Experimental results confirm that these two formulations are limited by the size of instances that they can efficiently solve, which makes their application more suited for business-to-business applications in which variable (and fixed) costs are more important and the number of clients is not very large (around 1000 clients [13]).

To extend the application to a business-to-consumer environment with considerably more customers, we have presented eight heuristics, including a branch-and-price heuristic and a tabu search. Some of these heuristics are currently used in practice. Based on extensive experimental results, we provide a comparison of and comments on the efficiency and quality of the results obtained using the different formulations and the heuristics. We find that the main interest of the branch-and-price algorithm lies in its use as a heuristic, outputting high-quality results for large instances. More generally, we observe a trade-off between computation time and solution quality and suggest the use of optimal algorithms for small and medium-size instances, while heuristics are preferable for large instances (tabu search and branch-and-price heuristic) and when time is an important factor (tabu search and successive solution of exact k-item knapsack instances).

Appendix A. Proof Proposition 1

This result follows directly from the fact that the basic formulation (M1) has the generalized assignment problem (GAP) [29,35] as a special case for $R = 0$, $M_i = 1$, $\forall i$, and $O_j = 1$, $f_j = 0$, $\forall j$ and $p_{ij} \geqslant c_{ij}$ for all i and j. The latter problem is known to be strongly NP-hard [35].

Appendix B. Proof Proposition 2

The proof uses the following variant of Partition (see [11]; by adding $|A|$ dummy elements of size 0 to an instance of the usual Partition problem, one easily sees that this variant of Partition is as hard as the original problem):

INSTANCE:	A finite set $A = \{1,2,\ldots,2q\}$ (where q is an integer greater than 0) with size $s(i) \in \mathbb{Z}^+$ for each $i \in A$ and $K = \frac{1}{2}\sum_{i\in A} s(i)$.
QUESTION:	Does there exist a subset $A' \subset A$ with $\lvert A'\rvert = q$ and $\sum_{i\in A'} s(i) = K$?

For a given arbitrary instance of Partition, consider the following polynomial reduction to an instance of (M1). Each $i \in A$ indicates a client with upper bound equal to 1, so $m = 2q$ and $M_i = 1$ for all $i = 1,2,\ldots,2q$. Suppose there is one product, $n = 1$. Let $\delta = 2K + 1$. For each client $i = 1,2,\ldots,2q$, the cost $c_{i1} = \delta + s(i)$, the revenue $p_{i1} = \delta s(i)$. Suppose also that for our product, product 1, the lower bound $O_1 = q$, the budget $B_1 = q\delta + K$, and the fixed cost $f_1 = 0$. Finally, we set the hurdle rate $R = \frac{(K-q)\delta - K}{q\delta + K}$. Now we prove that a non-trivial feasible solution to (M1) exists if and only if there is a solution A' to Partition.

On one hand, if Partition has a solution A', we offer the product to the clients in A'. This is a non-trivial feasible solution to the instance for (M1) constructed above since: (i) the lower bound $O_1 = q$ is met, (ii) the budget is met $\left(q\delta + \sum_{i\in A'} s(i) = q\delta + K = B_1\right)$ and (iii) the hurdle rate is achieved: $\sum_{i\in A'} p_{i1} = \delta \sum_{i\in A'} s(i) = \delta K = \frac{\delta K}{q\delta + K}(q\delta + K) = \left(1 + \frac{(K-q)\delta - K}{q\delta + K}\right)(q\delta + K) = (1 + R)(q\delta + K)$.

If we have a non-trivial feasible solution, on the other hand, then consider the set of clients A' receiving the product. We have $|A'| = q$ and $\sum_{i\in A'} c_{i1} = \sum_{i\in A'}(\delta + s_i) \leqslant B_1 = q\delta + K$. Suppose that $\sum_{i\in A'} c_{i1} < B_1$, then $\sum_{i\in A'} c_{i1} = q\delta + K_1$ with $K_1 < K$. Thus

$$\sum_{i\in A'} p_{i1} = K_1\delta = K\delta\frac{K_1}{K} < K\delta\frac{q\delta + K_1}{q\delta + K} = (q\delta + K_1)\frac{K\delta}{q\delta + K}$$
$$= \left(1 + \frac{(K-q)\delta - K}{q\delta + K}\right)(q\delta + K_1) = (1 + R)(q\delta + K_1),$$

contradicting the fact that we have a non-trivial feasible solution because the hurdle-rate constraint is not satisfied. Therefore $|A'| = q$ and $\sum_{i\in A'} c_{i1} = q\delta + K$, which implies that A' is a solution to the variant of Partition defined above.

Appendix C. Proof Proposition 3

Denote the convex hull of the feasible solutions to (M1) by $CH(M1)$ and the convex hull of the solutions that satisfy the constraints (2)–(4), (6) and (8), and $0 \leqslant y_j \leqslant 1$, $x_{ij} \in \{0,1\}$ for all i, j by CH^*. Because any feasible solution to (M1) satisfies (2)–(4), (6) and (8), and $0 \leqslant y_j \leqslant 1$, $x_{ij} \in \{0,1\}$, the set of feasible solutions to (M1) is included in CH^*, therefore, the convexity of CH^* implies that $CH(M1) \subseteq CH^*$.

Author's personal copy

On the other hand, let (x^0,y^0) be an extreme point of CH^*, so (x^0,y^0) satisfies (2)–(4) and (6), and $x_{ij} \in \{0,1\}$, and (x^0,y^0) satisfies (8) which implies that (x^0,y^0) satisfies (5). Furthermore $y_j^0 \in \{0, 1\}$ for all j because if there was a j_0 such that $0 < y_{j_0}^0 < 1$, then $x_{ij_0}^0 = 0$ for all i and constraints (6) would be violated. We then have $y_j^0 \in \{0,1\}$ for all j and therefore (x^0,y^0) is a feasible solution to (M1), so belongs to $CH(M1)$. Since (x^0,y^0) is an extreme point of CH^*, it is also an extreme point of $CH(M1)$ (because $CH(M1) \subseteq CH^*$), implying that $CH^* \subseteq CH(M1)$.

Appendix D. Proof Proposition 4

Suppose there is no client i such that x_{ij}^* is fractional. Let $F_j = \{p \in \{1,\ldots,k_j\} | 0 < z_{pj} < 1\}$ be the set of fractional variables associated with product j. We may assume that $|F_j| \geqslant 2$, otherwise the hypothesis (there is no client i such that x_{ij}^* is fractional) will be violated. We have $\sum_{p\in F_j} z_{pj} = y_j^* \leqslant 1$ for $j = 1,\ldots,n$ and $\sum_{p\in F_j : i\in S_{pj}} z_{pj} = x_{ij}^* \in \{0,1\},\ i = 1,\ldots,n$. In particular for $i \in \cup_{p\in F_j} S_{pj}$, we have $\sum_{p\in F_j : i\in S_{pj}} z_{pj} = 1$. But $\sum_{p\in F_j : i\in S_{pj}} z_{pj} = \sum_{p\in F_j} \mathbf{1}_{S_{pj}}(i) z_{pj} = 1$ and $\sum_{p\in F_j} z_{pj} = 1$; where $\mathbf{1}_{S_{pj}}$ is the indicator function of S_{pj}. Hence, $\mathbf{1}_{S_{pj}}(i) = 1,\ \forall p \in F_j$, meaning that for all $p_1,\ p_2 \in F_j$, we have $S_{p_1} = S_{p_2}$, contradicting $|F_j| \geqslant 2$.

References

[1] E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, 1997.
[2] C. Barnhart, E. Johnson, M. Savelsbergh, Branch-and-price: Column generation for solving huge integer programs, Operations Research 46 (1998) 316–329.
[3] J.B. Bernstel, Smart move: Creating 'intelligent' database, ABA Bank Marketing 34 (2002) 14–19.
[4] T. Bhaskar, R. Sundararajan, P.G. Krishnan, A fuzzy mathematical programming approach for cross-sell optimization in retail banking, Journal of the Operational Research Society 60 (2009) 717–727.
[5] I. Bose, X. Chen, Quantitative models for direct marketing: A review from a systems perspective, European Journal of Operational Research 195 (2009) 1–16.
[6] R.C. Bruner, K.M. Eades, R.S. Harris, R.C. Higgins, Best practices in estimating the cost of capital: Survey and synthesis, Financial Practice and Education 8 (1998) 13–28.
[7] A. Caprara, H. Kellerer, U. Pferschy, D. Pisinger, Approximation algorithms for knapsack problems with cardinality constraints, European Journal of Operational Research 123 (2000) 333–345.
[8] M. Cohen, Exploiting response models – Optimizing cross-sell and up-sell opportunities in banking, Information Systems 29 (2004) 327–341.
[9] B. De Reyck, Z. Degraeve, Broadcast scheduling for mobile advertising, Operations Research 51 (2003) 509–517.
[10] F.R. Dwyer, Customer lifetime valuation to support marketing decision making, Journal of Direct Marketing 11 (1997) 6–13.
[11] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Co., 1979.
[12] V. Gerson, Right on target, ABA Bank Marketing 30 (1998) 277–293.
[13] Air Transport Action Group, The Economic and Social Benefits of Air Transport, Technical Report, December 2007.
[14] P.M. Guadagni, J.D.C. Little, A logit model of brand choice calibrated on scanner data, Marketing Science 2 (1983) 203–238.
[15] J.P. Guiltinan, The price bundling of services: A normative framework, Journal of Marketing 51 (1987) 74–85.
[16] D.M. Hanssens, P.S.H. Leeflang, D.R. Wittink, Market response models and marketing practice, Applied Stochastic Models in Business and Industry 21 (2005) 423–434.
[17] E. Hellinckx, Customer Relationship Management: De optimalisatie van de Planning van campagnes, Master's Thesis, Department of Applied Economics, KULeuven, 2004 (in Dutch).
[18] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer, 2004.
[19] A. Knott, A. Hayes, S.A. Neslin, Next-product-to-buy models for cross-selling applications, Journal of Interactive Marketing 16 (2002) 59–75.
[20] P. Kotler, G. Armstrong, Principles of Marketing, Pearson Prentice Hall, 2006.
[21] V. Kumar, G. Ramani, T. Bohling, Customer lifetime value approaches and best practice applications, Journal of Interactive Marketing 18 (2004) 60–72.
[22] M. Laguna, J.P. Kelly, J.L. González-Velarde, F. Glover, Tabu search for the multilevel generalized assignment problem, European Journal of Operational Research 82 (1995) 176–189.
[23] S. Li, B. Sun, R.T. Wilcox, Cross-selling sequentially ordered products: An application to consumer banking services, Journal of Marketing Research XLII (2005) 233–239.
[24] A. Lim, F. Wang, Z. Xu, On the selection and assignment with minimum quantity commitments, Lecture Notes in Computer Science 3106 (2004) 102–111.
[25] A. Lim, F. Wang, Z. Xu, A transportation problem with minimum quantity commitment, Transportation Science 40 (2006) 117–129.
[26] A. Lim, Z. Xu, The bottleneck problem with minimum quantity commitment, Naval Research Logistics 53 (2006) 91–100.
[27] Z. Luo, L. Tang, W. Zhang, Using branch-and-price algorithm to solve raw materials logistics planning problem in iron and steel industry, in: 2007 International Conference on Management Science and Engineering, 2007, pp. 529–536.
[28] M. Mankila, Retaining students in retail banking through price bundling: Evidence from the swedish market, European Journal of Operational Research 155 (2004) 299–316.
[29] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementation, John Wiley and Sons, 1990.
[30] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley, 1988.
[31] A. Prinzie, D. Van Den Poel, Investigating purchasing-sequence patterns for financial services using Markov, MTD and MTDg models, European Journal of Operational Research 170 (2006) 710–734.
[32] A. Prinzie, D. Van Den Poel, Predicting home-appliance acquisition sequences: Markov/Markov for discrimination and survival analysis for modeling sequential information in NPTB models, Decision Support Systems 44 (2007) 28–45.
[33] W. Reinartz, J.S. Thomas, V. Kumar, Balancing acquisition and retention resources to maximize customer profitability, Journal of Marketing 69 (2005) 63–79.
[34] L. Ryals, Making customer relationship management work: The measurement and the profitable management of customer relationships, Journal of Marketing 69 (2005) 252–261.
[35] M. Savelsbergh, A branch-and-price algorithm for the generalized assignment problem, Operations Research 45 (1997) 831–841.
[36] B. Stone, R. Jacobs, Successful Direct Marketing Methods: Interactive, Database, and Customer-based Marketing for Digital Age, McGraw-Hill Professional, 2008.
[37] F. Talla Nobibon, R. Leus, F.C.R. Spieksma, Models for the Optimization of Promotion Campaigns: Exact and Heuristic Algorithms, Research Report KBI_0814, Katholieke Universiteit Leuven, 2008.
[38] A.K. Tripathi, S.K. Nair, Narrowcasting of wireless advertising in malls, European Journal of Operational Research 182 (2007) 1023–1038.
[39] J.M. Van Den Akker, J.A. Hoogeveen, J.W. Van Kempen, Parallel machine scheduling through column generation: Minimax objectives (extended abstract), Lecture Notes in Computer Science 4168 (2006) 648–659.
[40] N. Van Praag, Optimization of Promotion Campaigns Using Tabu Search, Master's Thesis, Faculty of Business and Economics, KULeuven, 2010.
[41] P.H. Vance, A. Atamturk, C. Barnhart, E. Gelman, E.L. Johnson, A. Krishna, G.L. Nemhauser, R. Rebello, A Heuristic Branch-and-Price Approach for the Airline Crew Pairing Problem, Technical Report LEC-97-06, Georgia Institute of Technology, 1997.
[42] L.A. Wolsey, Integer Programming, Wiley, 1998.