Contents lists available at ScienceDirect

European Journal of Operational Research

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

Decision Support

A matheuristic for a customer assignment problem in direct marketing

T. Bigler[*], M. Kammermann, P. Baumann

Department of Business Administration, University of Bern, Engehaldenstrasse 4, Bern 3012, Switzerland

ARTICLE INFO

Article history:
Received 19 August 2020
Accepted 8 April 2022
Available online xxx

Keywords:
OR in marketing
Conflict constraints
Matheuristic
Mixed-binary linear programming

ABSTRACT

In direct marketing, companies use sales campaigns to target their customers with personalized product offers. The effectiveness of direct marketing greatly depends on the assignment of customers to campaigns. In this paper, we consider a real-world planning problem of a major telecommunications company that assigns its customers to individual activities of its direct marketing campaigns. Various side constraints, such as budgets and sales targets, must be met. Conflict constraints ensure that individual customers are not assigned too frequently to similar activities. Related problems have been addressed in the literature; however, none of the existing approaches cover all the side constraints considered here. To close this gap, we develop a matheuristic that employs a new decomposition strategy to cope with the large number of conflict constraints in typical problem instances. In a computational experiment, we compare the performance of the proposed matheuristic to the performance of two mixed-binary linear programs on a test set that includes large-scale real-world instances. The matheuristic derives near-optimal solutions in short running times for small- to medium-sized instances and scales to instances of practical size comprising millions of customers and hundreds of activities. The deployment of the matheuristic at the company has considerably increased the overall effectiveness of its direct marketing campaigns.

1. Introduction

Companies in competitive sectors such as banking and telecommunications strongly rely on direct marketing to promote their products (Miguéis, Camanho, & Borges, 2017). In direct marketing campaigns, companies contact their customers individually via *call*, *direct mail*, *email*, or *text message* to make a personalized offer. The success of such campaigns greatly depends on the assignment of customers to campaigns. Increasing the overall response rate by targeting the right customers can result in a substantial profit increase. However, contacting individual customers too frequently and offering products they are not interested in, negatively impacts the effectiveness of direct marketing.

We introduce a real-world planning problem of a major telecommunications provider. The problem input consists of a set of activities and a set of customers. Each activity is scheduled for a specific day, and the eligible customers, i.e., the customers who can be assigned, are known for each activity. An expected profit, a response probability, and a cost are given for every possible assignment of a customer to an activity. Various business and customer-specific constraints must be considered. The business constraints include assignment constraints that control the number of assignments to specific activities, budget constraints that ensure that the total cost associated with assignments to specific activities does not exceed a prescribed budget, as well as sales constraints that control the expected number of sales resulting from assignments to specific activities. The customer-specific constraints include lower and upper bounds on the number of contacts per customer within a prespecified time window (e.g., one month) and conflict constraints that ensure that each customer who is eligible for two conflicting activities is assigned to at most one of the two activities. Two activities are in conflict when they take place in close succession and are associated with the same channel or promote the same product. The objective is to assign the customers to the activities such that the total expected profit is maximized. The company solves this planning problem on a daily basis, each time with updated data. Before using our approach, a dedicated team of the company used to manually assign its customers to the activities. To preserve the existing workflow, a key requirement for the solution approach was that it is capable of producing solutions for practical instances within 30 minutes.

To the best of our knowledge, none of the existing approaches from the literature can be directly applied to the above-described

[*] Corresponding author.
E-mail addresses: tamara.bigler@unibe.ch (T. Bigler), manuel.kammermann@unibe.ch (M. Kammermann), philipp.baumann@unibe.ch (P. Baumann).

https://doi.org/10.1016/j.ejor.2022.04.009
0377-2217/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

Table 1
Activities of the illustrative example.

Activity	Day	Channel	Target products	Cost
A1	1	*text message*	*internet*	1
A2	5	*text message*	*internet*	1
A3	2	*call center*	*mobile*	10
A4	7	*call center*	*mobile*	10
A5	4	*direct mail*	*internet*	4
A6	6	*email*	*mobile*	1
A7	1	*call center*	*mobile*	10
A8	7	*text message*	*mobile*	1

planning problem. Approaches that have been proposed for optimizing direct marketing campaigns only consider subsets of the constraints of our problem setting (cf. Table 5 in Section 3.1 for an overview). In particular, none of the approaches consider customer-specific conflict constraints. Conflict constraints in a more general sense have received considerable attention in the literature on related planning problems such as the assignment problem and the bin packing problem. However, approaches for these problems also cannot be applied directly to our problem because they do not allow the assignment of the same customer to multiple activities or the assignment of multiple customers to the same activity. The planning problem here also differs from related planning problems in terms of the size of typical instances. Practical instances involve millions of customers and hundreds of activities, which may lead to hundreds of millions of conflict constraints. This large number of conflict constraints hinders the use of exact approaches that employ a mixed-binary linear programming formulation of the entire planning problem. With hundreds of millions of conflict constraints, the time required to construct the model alone exceeds the running time limit prescribed by the company. The large number also makes it difficult to adapt existing heuristics that were not specifically designed to address this challenge. Hence, the development of a new solution approach is required.

We propose a matheuristic for the above-described problem. The matheuristic follows the idea of solving a mathematical model for groups of customers rather than individual customers. The main methodological feature of the matheuristic is the problem decomposition strategy, which allows grouping of customers while still enforcing conflict constraints for individual customers. The decomposition works in four steps. In the first step, customers who are eligible for the same activities are grouped together such that the customers in one group are subject to the same conflicts among activities. In the second step, a clustering algorithm further divides each group into subgroups such that the customers in the same subgroup have similar expected profits. In the third step, a linear program decides how many customers of a subgroup are assigned to an activity. To consider conflicts among activities, the linear program employs new types of constraints which are defined for sets of activities of maximal size in which every two distinct activities are conflicting. To derive these sets efficiently without introducing redundancies, we developed a preprocessing technique. In the fourth step, an iterative algorithm assigns individual customers to the activities in a carefully selected sequence based on the solution derived by the linear program. The trade-off between solution quality and running time can be controlled by changing the number of subgroups created in the second step. The proposed decomposition scheme is the first to show how conflict constraints can be grouped and effectively incorporated into an aggregated optimization model. We believe that these ideas can be useful for the development of heuristics for related combinatorial optimization problems with conflict constraints.

In an experimental analysis, we use 27 generated and 13 real-world instances to compare the performance of the matheuristic to the performance of two mixed-binary linear programming formulations. The matheuristic finds near-optimal solutions in short running times for instances that could be solved to optimality by at least one of the two mixed-binary linear programming formulations. For all other instances, the matheuristic clearly outperforms both mixed-binary linear programming formulations in terms of solution quality and running time. We find that the preprocessing technique and the new types of constraints in the linear program contribute substantially to the effectiveness and the speed of the matheuristic. The matheuristic has been successfully deployed at the company and is now used on a daily basis. According to the company, introducing the matheuristic led to a substantial increase in the overall profitability of the campaigns. Moreover, the problem decomposition strategy of the matheuristic allows the company to approximate the impact of strategic decisions (e.g., an increase of budgets) in near real time by solving the linear program of the third step several times with different values for the parameters.

The rest of the paper is structured as follows. In Section 2, we describe the planning problem in more detail. In Section 3, we review the related literature. In Section 4, we formulate the planning problem as a mixed-binary linear program. In Section 5, we describe the four steps of the matheuristic. In Section 6, we explain the preprocessing technique used in the matheuristic in more detail and introduce an alternative mixed-binary linear programming formulation that makes use of the preprocessing technique. In Section 7, we report the results. Finally, in Section 8, we draw conclusions and give directions for future research.

2. Planning problem

In Section 2.1, we provide the business context of the planning problem. In Section 2.2, we specify the planning problem. In Section 2.3, we illustrate the planning problem with an example.

2.1. Business context

The telecommunications company that reported the planning problem simultaneously runs multiple direct marketing campaigns to promote its products and services related to different market segments to existing customers of the company. Each campaign is created by a marketing manager who selects a set of target products, designs the offer, identifies eligible customers, and determines the activities of the campaign, i.e., the days on which customers can be contacted via specific channels. The marketing managers also define some of the business constraints such as assignment constraints for activities of their campaigns. Other business constraints such as budgets are defined in a centralized manner by higher management. The assignment of the company's customers to its activities is performed by a central unit instead of the campaign managers to prevent customers who are eligible for activities of multiple campaigns from being contacted too frequently. The central unit considers all business constraints when assigning the customers to the activities. To replace the current practice of manually assigning the customers to the activities, the company asked us to develop a heuristic to assign its customers automatically. In the next section, we describe the planning problem from the perspective of the central unit.

2.2. Problem description

The problem input consists of a set of activities and a set of customers. Each customer can be assigned to one or multiple activities. An expected profit is given for each possible assignment. The goal is to assign the customers to the activities such that the total expected profit is maximized subject to various constraints. We distinguish between business and customer-specific constraints. The business constraints consist of:

- **Minimum assignment constraints**, which impose lower bounds on the number of assignments to sets of selected activities, and **maximum assignment constraints**, which impose upper bounds on the number of assignments to sets of selected activities. These constraints balance the number of assignments over activities and can be used, for example, to control the utilization of specific channels.
- **Budget constraints**, which impose upper bounds on the total costs that result from assignments to sets of selected activities. The cost per assignment depends on the channel over which a customer is contacted, and hence may differ among activities. These constraints ensure that funds allocated to individual channels or groups of channels are not exceeded.
- **Minimum sales constraints**, which impose lower bounds on the number of expected sales that result from assignments to sets of selected activities, and **maximum sales constraints**, which impose upper bounds on the number of expected sales that result from assignments to sets of selected activities. Each possible assignment is associated with a response probability, which states the likelihood of a positive customer reaction (i.e., a sale of the target product). The expected sales that result from assignments to a set of selected activities are computed as the sum of the corresponding response probabilities. These constraints enable the company to control the intensity of promoting new products or services. Maximum sales constraints are useful to prevent an overload of sales channels. For example, they can be used to distribute the customer volume in shops over time.

Each activity is associated with different characteristics, such as the day on which the assigned customers are contacted, the channel that is used to contact the customers, and the target products that are promoted. These characteristics allow definition of business constraints for specific time periods, channels, or target products by selecting the corresponding activities. More complicated selections of activities based on combinations of these characteristics are also possible. For example, minimum and maximum assignment constraints are typically defined for specific combinations of channels and time periods. The customer-specific constraints consist of:

- **Minimum contact constraints**, which impose a lower bound on the number of times that a customer is assigned to a set of selected activities, and **maximum contact constraints**, which impose an upper bound on the number of times that a customer is assigned to a set of selected activities. These constraints ensure that a customer is not contacted too frequently. The contact constraints are derived from contact rules, which may state, for example, that a customer cannot be contacted more than twice in January.
- **Conflict constraints**, which ensure that each customer is assigned to at most one out of two conflicting activities. A conflict arises when two activities take place within a certain number of consecutive days and use certain combinations of channels and target products. These constraints ensure that a customer is not assigned too frequently to similar activities. The conflict constraints are derived from conflict rules, which may state, for example, that a customer cannot be contacted twice via a *call* within one week. A separate conflict constraint would be imposed for each customer and each pair of activities which use the channel *call center* and take place within seven consecutive days.

This planning problem is strongly *NP*-hard, as it can be reduced to a generalized assignment problem that is known to be *NP*-hard (cf. Garey & Johnson, 2002). The minimum assignment constraints, the minimum and maximum sales constraints, and the minimum contact constraints reflect strategic decisions rather than operational requirements. Also, some of these constraints are defined independently by different marketing managers. This process may lead to contradictory constraints, for example, it is possible that two marketing managers each define a minimum assignment constraint that individually could be satisfied but together cannot be satisfied because of budget or conflict constraints. These dependencies get more complex if all types of minimum constraints are considered. Thus, the company wants to treat the constraints of the aforementioned four constraint types (minimum assignment, minimum and maximum sales, and minimum contact constraints) in almost all instances as soft constraints that can be violated subject to a penalty. The rest of the constraints must be treated as hard constraints because they represent operational requirements. The main purpose of using the soft constraints is to be able to generate a solution even if some constraints cannot be satisfied. Another advantage of the soft constraints is that users at the company are able to observe which slack variables take a positive value and thus, which constraints cannot be satisfied in the current setting. The penalty is computed for every soft constraint by multiplying the difference between the achieved assignments (or sales) and the prescribed bound by a constant. The constant can be set individually for each constraint type. The objective is to maximize the total expected profit minus the total penalty.

2.3. Illustrative example

The illustrative example comprises 20 customers and eight activities that belong to six different campaigns. Figure 1 shows for each activity the day on which it takes place, the channel over which the customers are contacted, and the target products that are promoted. Table 1 additionally lists the cost per assignment for each activity. Table 2 specifies the expected profit and the response probability for each possible assignment. The business constraints comprise a maximum assignment constraint, which ensures that at most 11 *calls* are used to contact the customers, a budget constraint, which ensures that all assignments associated with the channels *direct mail*, *email*, and *text message* do not incur costs of more than 40 Euros, and a maximum sales constraint, which ensures that the number of expected sales for the target product *mobile* does not exceed five. Moreover, minimum contact constraints are based on a contact rule that states that each customer must be assigned at least once. Maximum contact constraints are based on a contact rule that states that each customer must be assigned at most twice. Table 3 specifies for the business constraints and the contact rules the type, the start day, the end day, the channels, the target products, and the bound associated with the respective constraint or contact rule. The illustrative example has four conflict rules. First, each customer cannot be contacted more than once within two days. Second, each customer cannot be contacted more than once within five days via a *call*. Third, each customer cannot be contacted more than once within four days via *direct mail*. Fourth, each customer cannot be contacted more than once within four days via a *text message*. Table 4 indicates the combinations of channels and target products that lead to a conflict and the time lag related to a conflict rule. In Tables 3 and 4, the entry “ALL” indicates that all channels or target products are affected by this constraint or rule.

All constants used to compute the total penalty for the soft constraints are set to 112 Euros (maximum absolute expected profit). The optimal assignment is marked in bold in Table 2 and produces an expected profit of 2,973 Euros. The maximum sales constraint is

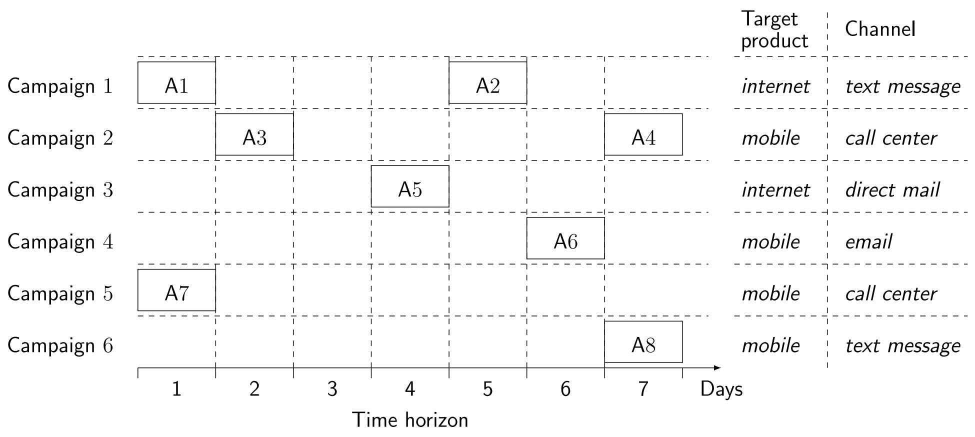

Fig. 1. Activities (A1 to A8) of the illustrative example.

Table 2
Expected profits and response probabilities of the illustrative example.

Customer	Activity	Expected profit	Response prob.
1	A3	105	0.29
1	**A4**	**105**	**0.29**
1	A5	88	0.16
2	A5	78	0.17
2	**A6**	**91**	**0.23**
3	A5	88	0.28
3	**A6**	**75**	**0.20**
4	A5	80	0.14
4	**A6**	**90**	**0.26**
5	A3	91	0.19
5	**A4**	**91**	**0.19**
5	**A5**	**107**	**0.26**
6	**A1**	**75**	**0.22**
6	A2	75	0.22
6	**A6**	**85**	**0.15**
7	A5	100	0.24
7	**A7**	**85**	**0.11**
7	**A8**	**71**	**0.22**
8	A3	109	0.25
8	**A4**	**109**	**0.25**
8	**A5**	**91**	**0.18**
9	A5	76	0.15
9	**A6**	**89**	**0.25**
10	A5	102	0.22
10	**A7**	**87**	**0.12**
10	**A8**	**68**	**0.23**
11	A3	90	0.18
11	**A4**	**90**	**0.18**
11	**A5**	**108**	**0.25**
12	A3	85	0.17
12	**A4**	**85**	**0.17**
12	**A5**	**106**	**0.25**
13	A3	88	0.18
13	**A4**	**88**	**0.18**
13	**A5**	**110**	**0.24**
14	A5	65	0.17
14	A7	75	0.23
14	**A8**	**80**	**0.24**
15	**A1**	**102**	**0.25**
15	A2	102	0.25
15	**A6**	**93**	**0.21**
16	A3	112	0.30
16	**A4**	**112**	**0.30**
16	**A5**	**90**	**0.20**
17	A5	89	0.29
17	**A6**	**77**	**0.21**
18	**A1**	**101**	**0.29**
18	A2	101	0.29
18	**A6**	**95**	**0.19**
19	A5	68	0.20
19	**A7**	**73**	**0.21**
19	**A8**	**85**	**0.26**
20	A5	66	0.19
20	**A7**	**76**	**0.22**
20	**A8**	**83**	**0.25**

Table 3
Business constraints and contact rules of the illustrative example.

Index	Type	Start day	End day	Channels	Target products	Bound
1	Maximum assignment	1	7	*call center*	ALL	11
2	Budget	1	7	*direct mail*, *email*, *text message*	ALL	40
3	Maximum sales	1	7	ALL	*mobile*	5
4	Minimum contact	1	7	ALL	ALL	1
5	Maximum contact	1	7	ALL	ALL	2

Table 4
Conflict rules of the illustrative example.

Index	Channel 1	Target product 1	Channel 2	Target product 2	Lag
1	ALL	ALL	ALL	ALL	2
2	*call center*	ALL	*call center*	ALL	5
3	*direct mail*	ALL	*direct mail*	ALL	4
4	*text message*	ALL	*text message*	ALL	4

Table 5
Approaches and problem features from related literature

		Approach		Time	Decision	Objective	Constraints							
Authors		Exact	Heuristic	Temporal dimension	Customer asgmt	Expected profit	Min asgmt	Max asgmt	Budget	Min sales	Max sales	Min contact	Max contact	Conflict
Direct marketing	Cohen (2004)	✓	✓		✓	✓	(✓)	✓	✓				(✓)	
	Bhaskar et al. (2009)	✓	✓		(✓)	✓			✓	(✓)				
	Nobibon et al. (2011)	✓	✓		✓	✓	(✓)	(✓)	✓				✓	
	Oliveira et al. (2015)		✓		✓	✓	(✓)	(✓)	✓				✓	
	Cetin & Alabas-Uslu (2017)		✓		✓	✓	(✓)	(✓)	✓				✓	
	Coelho et al. (2017)	✓	✓		✓	(✓)	(✓)	(✓)	✓				✓	
	Nair & Tarasewich (2003)	✓	✓	✓										(✓)
	Delanote et al. (2013)	✓		✓	(✓)	✓		(✓)	✓	(✓)			(✓)	
	Ma & Fildes (2017)	✓	✓	✓		✓	(✓)	(✓)					(✓)	
General	Darmann et al. (2011)	✓					(✓)	(✓)				(✓)	(✓)	(✓)
	Öncan et al. (2019)	✓					(✓)	(✓)				(✓)	(✓)	(✓)
	Elhedhli et al. (2011)	✓							(✓)			(✓)	(✓)	(✓)
	Sadykov & Vanderbeck (2013)	✓							(✓)			(✓)	(✓)	(✓)
	Bigler et al. (2019)	✓		✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)	✓	✓
	This paper	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

violated by 0.12, which leads to a penalty of $112(0.12) = 13.44$ Euros. The objective function value in the optimal solution is thus 2,959.56 Euros.

3. Literature

The literature review is organized as follows. In Section 3.1, we focus on related problems in direct marketing. In Section 3.2, we focus on more general combinatorial optimization problems that share specific constraints with our planning problem. Table 5 gives an overview of the discussed approaches and shows which problem features are exactly (✓) or partially ((✓)) covered.

3.1. Related problems in direct marketing

There are two large streams of literature in direct marketing. The first stream is concerned with the development of response models that predict the responses of customers to direct marketing efforts (e.g., Ma, Hou, Yao, & Lee, 2016 and Lessmann, Haupt, Coussement, & De Bock, 2021). The second stream pertains to the optimization of direct marketing operations given the output of response models. We focus on the second stream because it is more closely related to our planning problem.

Many of the planning problems considered in direct marketing involve the assignment of individual customers to product offers. The objective in these planning problems is to maximize the expected profit subject to various side constraints. Table 5 shows the side constraints that are covered by the different approaches. Cohen (2004) is the first to propose a binary linear program for assigning customers to direct marketing campaigns. For large-scale instances, he introduces a heuristic that is based on the idea of grouping similar customers and using a linear program to determine how many customers of a group are assigned to a campaign. Bhaskar, Sundararajan, & Krishnan (2009) formulate a similar problem as the one of Cohen (2004) as a binary linear program and propose a heuristic that builds on the idea of grouping customers. Sundararajan et al. (2011) describe the integration of the heuristic of Bhaskar et al. (2009) in a retail bank, which led to an estimated financial benefit of $20 million. Nobibon, Leus, & Spieksma (2011) consider a planning problem in which a subset of products must be selected for a campaign and the customers must be assigned to the selected products. They provide a binary linear formulation, a branch-and-price algorithm, and eight heuristics for this planning problem. The two heuristics that are based on column generation and on tabu search tend to perform best. For the same planning problem as Nobibon et al. (2011), Oliveira et al. (2015) develop a heuristic that is based on a greedy randomized adaptive search procedure combined with a variable neighborhood search, and Cetin & Alabas-Uslu (2017) propose two heuristics, which in a first step use a rule-based procedure to select the products that will be part of a campaign, and in a second step assign the customers to the selected products. These two-step heuristics are competitive with the best heuristics of Nobibon et al. (2011). Finally, Coelho et al. (2017) develop a metaheuristic for a variant of the planning problem considered by Nobibon et al. (2011) where the objective function includes a reward-to-variability indicator, which is inspired by the Sharpe ratio. One major difference between our planning problem (cf. Section 2) and the ones discussed above is the existence of a temporal dimension. In our planning problem, each activity is scheduled on a specific day of the time horizon. This timing information is relevant for various business constraints, and especially for the conflict constraints. In the direct marketing literature, only few planning problems include a temporal dimension. The planning problems which involve a temporal dimension, however, focus primarily on the design of campaigns rather than the assignment of individual customers.

Nair & Tarasewich (2003), for example, study a planning problem that consists of designing a series of promotions over time. Some of the side constraints are similar to the conflict constraints from our planning problem. These similar constraints ensure that selected pairs of promotions cannot take place within a certain number of consecutive days in the time horizon. Nair & Tarasewich (2003) formulate a non-linear integer program and develop a genetic algorithm for this planning problem. Delanote, Leus, & Nobibon (2013) formulate an integer linear model for the planning of multi-round direct marketing campaigns, which consists of determining how many customers of each customer segment are assigned to which product and which channel in each round, in order to maximize the total expected profit. Customers who react positively in one round cannot be assigned to the same product in later rounds. The number of customers who react positively is estimated by multiplying the response probability of a segment with the respective number of assigned customers. Ma & Fildes (2017) formulate a non-linear program and develop a genetic algorithm for the planning of multi-period promotions, which consists of determining for each period which products to advertise such that the total expected profit of the campaign is maximized. Bigler, Baumann, & Kammermann (2019) study a variant of the planning problem from Section 2, in which all constraints are hard constraints. The authors formulate a binary linear program for this slightly different planning problem and apply it to four small- to medium-sized instances and one large-sized instance. To the best of our knowledge, no other customer assignment approach solved instances of similar size. However, this binary linear program does not scale to very large real-world instances.

As we can see from Table 5, none of the discussed planning problems fully cover all the features of our planning problem. For example, most planning problems do not consider conflict constraints. Conflict constraints, however, have received considerable attention as an extension of more general combinatorial optimization problems such as the assignment problem and the bin packing problem. In the next section, we review these planning problems.

3.2. More general combinatorial optimization problems with conflict constraints

More general combinatorial optimization problems such as the assignment problem and the bin packing problem have been extended to consider conflict constraints. In the assignment problem, equal numbers of agents (here customers) and tasks (here activities) are given, and exactly one agent must be assigned to each task such that the total cost is minimized. In the assignment problem with conflict constraints (APC), an assignment of an agent to a task may conflict with another assignment of an agent to a task. This structure of the conflict constraints is visualized in Fig. 2. The dashed lines correspond to potential assignments of agents to tasks, the solid lines correspond to assignments of agents to tasks in a feasible solution, and the bold red line indicates a conflict. Because of the conflict, agent i_2 cannot be assigned to task j_2 when agent i_1 is assigned to task j_1, or vice versa. Darmann, Pferschy, Schauer, & Woeginger (2011) prove that the APC is an *NP*-hard optimization problem. Öncan, Şuvak, Akyüz, & Altınel (2019) propose a branch-and-bound and a Russian doll search algorithm for the APC. Figure 2 also illustrates the structure of the conflict constraints in this paper. The customers are eligible for some, but generally not all activities. Thus, a potential assignment in the context of our planning problem means that customer i is eligible for activity j. Other than in the APC, a customer can be assigned to multiple activities. The conflicts in this paper occur between activities and apply to all customers who are eligible for the conflicting activities. For example, a conflict exists between activities j_1 and j_2, and thus both customers i_1 and i_2 can at most be assigned to one

Table 6
Notation of MBLP.

Sets	
I	Customers
J	Activities
T	Pairs of conflicting activities
I_j	Eligible customers of activity j
J_i	Activities for which customer i is eligible
$J_l^{\underline{a}}$	Activities associated with minimum assignment constraint $l = 1, \ldots, n^{\underline{a}}$
$J_l^{\overline{a}}$	Activities associated with maximum assignment constraint $l = 1, \ldots, n^{\overline{a}}$
$J_l^{\overline{b}}$	Activities associated with budget constraint $l = 1, \ldots, n^{\overline{b}}$
$J_l^{\underline{m}}$	Activities associated with minimum contact rule $l = 1, \ldots, n^{\underline{m}}$
$J_l^{\overline{m}}$	Activities associated with maximum contact rule $l = 1, \ldots, n^{\overline{m}}$
$J_l^{\underline{s}}$	Activities associated with minimum sales constraint $l = 1, \ldots, n^{\underline{s}}$
$J_l^{\overline{s}}$	Activities associated with maximum sales constraint $l = 1, \ldots, n^{\overline{s}}$
Parameters	
$b_l^{\underline{a}}$	Lower bound of minimum assignment constraint $l = 1, \ldots, n^{\underline{a}}$
$b_l^{\overline{a}}$	Upper bound of maximum assignment constraint $l = 1, \ldots, n^{\overline{a}}$
$b_l^{\overline{b}}$	Upper bound of budget constraint $l = 1, \ldots, n^{\overline{b}}$
$b_l^{\underline{m}}$	Lower bound of minimum contact rule $l = 1, \ldots, n^{\underline{m}}$
$b_l^{\overline{m}}$	Upper bound of maximum contact rule $l = 1, \ldots, n^{\overline{m}}$
$b_l^{\underline{s}}$	Lower bound of minimum sales constraint $l = 1, \ldots, n^{\underline{s}}$
$b_l^{\overline{s}}$	Upper bound of maximum sales constraint $l = 1, \ldots, n^{\overline{s}}$
c_j	Cost per assignment to activity j
e_{ij}	Expected profit of customer i when assigned to activity j
$n^{\underline{a}}$	Number of minimum assignment constraints
$n^{\overline{a}}$	Number of maximum assignment constraints
$n^{\overline{b}}$	Number of budget constraints
$n^{\underline{m}}$	Number of minimum contact rules
$n^{\overline{m}}$	Number of maximum contact rules
$n^{\underline{s}}$	Number of minimum sales constraints
$n^{\overline{s}}$	Number of maximum sales constraints
q_{ij}	Response probability of customer i when assigned to activity j
α	Constant to penalize the extent to which bound in a minimum assignment constraint is violated
β	Constant to penalize the extent to which bound in a minimum sales constraint is violated
γ	Constant to penalize the extent to which bound in a maximum sales constraint is violated
δ	Constant to penalize the extent to which bound in a constraint resulting from a minimum contact rule is violated
Decision variables	
x_{ij}	$= 1$, if customer i is assigned to activity j; $= 0$, otherwise
$z_l^{\underline{a}} \in [0, b_l^{\underline{a}}]$,	Slack variable of minimum assignment constraint $l = 1, \ldots, n^{\underline{a}}$
$z_l^{\underline{s}} \in [0, b_l^{\underline{s}}]$,	Slack variable of minimum sales constraint $l = 1, \ldots, n^{\underline{s}}$
$z_l^{\overline{s}} \in [0, q]$,	Slack variable of maximum sales constraint $l = 1, \ldots, n^{\overline{s}}$ with $q = \sum_{j \in J} \sum_{i \in I_j} q_{ij}$
$z_{il}^{\underline{m}} \in [0, b_l^{\underline{m}}]$,	Slack variable of minimum contact rule $l = 1, \ldots, n^{\underline{m}}$ for customer $i \in I$

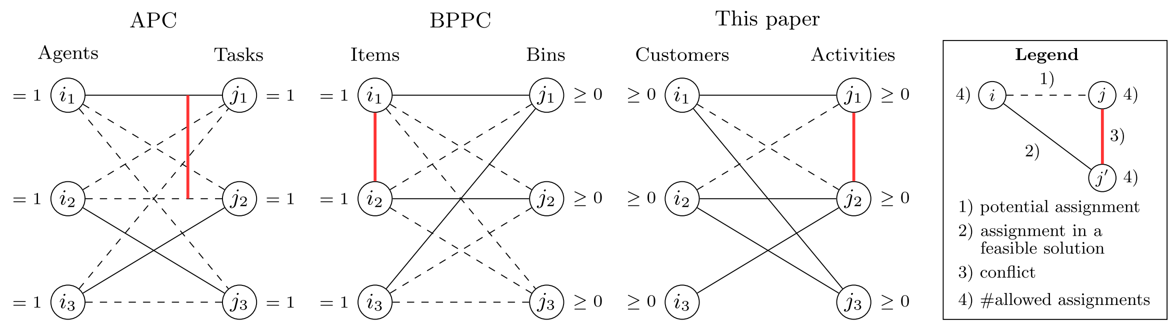

Fig. 2. Conflict constraints in more general combinatorial optimization problems.

of the two activities. In the feasible assignment shown in Fig. 2, customer i_1 is assigned only to activity j_1 and customer i_2 is assigned only to activity j_2. An infeasible assignment would be, for example, if customer i_1 was assigned to both activities j_1 and j_2.

In the bin packing problem, items (here customers) of different size must be packed in a minimum number of identical bins (here activities) with limited capacity. Each item must be assigned to exactly one bin, and each bin may contain multiple items that do not exceed its bin capacity. In the bin packing problem with conflict constraints (BPPC), conflicts occur between items and apply to all bins. This structure is also illustrated in Fig. 2, where items i_1 and i_2 are in conflict and thus cannot be assigned to the same bin. Elhedhli, Li, Gzara, & Naoum-Sawaya (2011) and Sadykov & Vanderbeck (2013) develop different branch-and-price algorithms for the BPPC. Our planning problem differs from the BPPC in several ways. First, each item in the BPPC can be assigned to each bin if the capacity of the bin allows it. Second, while in our planning problem the conflicts occur between activities and apply to all customers who are eligible for these conflicting activities, the conflicts in the BPPC occur between items (here customers) and apply to all bins (here activities). Even if we consider items to be bins and bins to be items in the BPPC, there is no direct analogy to our planning problem because items must be assigned exactly once in the BPPC, while customers can be assigned multiple times and activities can

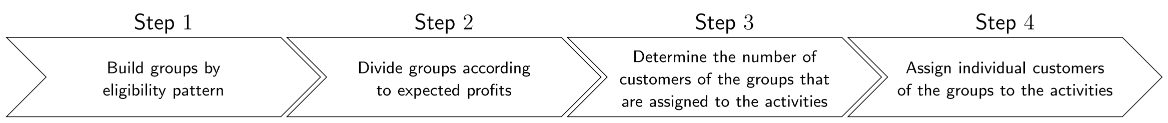
Step 1
Build groups by eligibility pattern
Step 2
Divide groups according to expected profits
Step 3
Determine the number of customers of the groups that are assigned to the activities
Step 4
Assign individual customers of the groups to the activities

Fig. 3. Steps of the matheuristic.

have multiple assigned customers in our planning problem. Thus, the conflict constraints have the same structure only if we adjust our planning problem such that all customers are eligible for all activities and each activity must have exactly one assigned customer.

Also other planning problems such as the knapsack problem with conflict constraints (cf. Bettinelli, Cacchiani, & Malaguti, 2017 and Coniglio, Furini, & San Segundo, 2021), the maximum flow problem with conflict constraints (cf. Şuvak, Altınel, & Aras, 2020), and the transportation problem with conflict constraints (cf. Sun, 2002) have attracted considerable attention in the literature. However, these planning problems differ considerably from our planning problem. In the knapsack problem, a conflict constraint ensures that only one of two items is included in the knapsack while in our planning problem a conflict constraint ensures that a customer is assigned to at most one of two conflicting activities. In the maximum flow problem, a conflict can involve any two edges of a graph while in our planning problem, a conflict always involves the assignments of one customer to two conflicting activities. In the transportation problem, a conflict constraint ensures that two conflicting goods cannot be shipped to the same warehouse. These conflict constraints are structurally similar to our conflict constraints if we consider warehouses to be customers and goods to be activities. However, in the transportation problem with conflict constraints, the supply of goods is given while in our planning problem the number of assignments to the activities is to be determined. Also, the warehouses in the transportation problem do not need to have a minimum number of assigned goods while the customers in our planning problem can be affected by minimum contact constraints. Finally, the assignment decisions in the transportation problem are integer (quantities of goods) while in our planning problem the assignment decisions are binary.

Table 5 also shows some side constraints besides the conflict constraints that the approaches for the APC and the BPPC cover. However, we can see that none of the existing approaches cover all problem features of our planning problem.

4. Mixed-binary linear programming formulation

In this section, we formulate the planning problem as a mixed-binary linear programming formulation (MBLP). Table 6 summarizes the sets, parameters and decision variables of the MBLP. The superscripts (i.e., $\underline{a}$, $\overline{a}$, $\overline{b}$, $\underline{m}$, $\overline{m}$, $\underline{s}$, $\overline{s}$) indicate for which type of constraint a set, parameter, or decision variable has been introduced. The information required to generate these sets and parameters for a specific instance can be derived from four tables. These four tables are exemplarily provided for the illustrative example in Section 2.3 (cf. Tables 1–4). From the equivalent of Table 1, we can derive information about all activities. From the equivalent of Table 2, we can derive information on all eligible customers for each activity. From the equivalent of Table 3, we can identify the activities that are associated with the business constraints and the contact rules. Based on the equivalents of Tables 1 and 4, we can identify all pairwise conflicts between activities. The MBLP uses binary decision variables x_{ij}, which take the value of one if customer i is assigned to activity j, and zero otherwise. Furthermore, continuous slack variables are introduced for all soft constraints. The expected profit e_{ij} is computed by multiplying the change in revenue that results if customer i accepts the offer times the corresponding response probability q_{ij} minus the cost per assignment c_j. The MBLP reads as follows:

$$
\text{(MBLP)}\left\{
\begin{array}{lll}
\text{Max.} & \sum_{j\in J}\sum_{i\in I_j} e_{ij}x_{ij} - \left(\alpha\sum_{l=1}^{n^{\underline{a}}} z_l^{\underline{a}} + \beta\sum_{l=1}^{n^{\underline{s}}} z_l^{\underline{s}} + \gamma\sum_{l=1}^{n^{\overline{s}}} z_l^{\overline{s}} + \delta\sum_{i\in I}\sum_{l=1}^{n^{\underline{m}}} z_{il}^{\underline{m}}\right) & \qquad (1)\\
\text{s.t.} & \sum_{j\in J_l^{\underline{a}}}\sum_{i\in I_j} x_{ij} + z_l^{\underline{a}} \geq b_l^{\underline{a}} & (l=1,\dots,n^{\underline{a}}) \qquad (2)\\
& \sum_{j\in J_l^{\overline{a}}}\sum_{i\in I_j} x_{ij} \leq b_l^{\overline{a}} & (l=1,\dots,n^{\overline{a}}) \qquad (3)\\
& \sum_{j\in J_l^{\overline{b}}}\sum_{i\in I_j} c_j x_{ij} \leq b_l^{\overline{b}} & (l=1,\dots,n^{\overline{b}}) \qquad (4)\\
& \sum_{j\in J_l^{\underline{s}}}\sum_{i\in I_j} q_{ij}x_{ij} + z_l^{\underline{s}} \geq b_l^{\underline{s}} & (l=1,\dots,n^{\underline{s}}) \qquad (5)\\
& \sum_{j\in J_l^{\overline{s}}}\sum_{i\in I_j} q_{ij}x_{ij} - z_l^{\overline{s}} \leq b_l^{\overline{s}} & (l=1,\dots,n^{\overline{s}}) \qquad (6)\\
& \sum_{j\in J_l^{\underline{m}}\cap J_i} x_{ij} + z_{il}^{\underline{m}} \geq b_l^{\underline{m}} & (i\in I;\ l=1,\dots,n^{\underline{m}}) \qquad (7)\\
& \sum_{j\in J_l^{\overline{m}}\cap J_i} x_{ij} \leq b_l^{\overline{m}} & (i\in I;\ l=1,\dots,n^{\overline{m}}:\ |J_l^{\overline{m}}\cap J_i| > b_l^{\overline{m}}) \qquad (8)\\
& x_{ij_1} + x_{ij_2} \leq 1 & ((j_1,j_2)\in T;\ i\in I_{j_1}\cap I_{j_2}) \qquad (9)\\
& x_{ij}\in\{0,1\} & (j\in J;\ i\in I_j) \qquad (10)\\
& z_l^{\underline{a}}\in[0,b_l^{\underline{a}}] & (l=1,\dots,n^{\underline{a}}) \qquad (11)\\
& z_l^{\underline{s}}\in[0,b_l^{\underline{s}}] & (l=1,\dots,n^{\underline{s}}) \qquad (12)\\
& z_l^{\overline{s}}\in[0,q] & (l=1,\dots,n^{\overline{s}}) \qquad (13)\\
& z_{il}^{\underline{m}}\in[0,b_l^{\underline{m}}] & (i\in I;\ l=1,\dots,n^{\underline{m}}) \qquad (14)
\end{array}\right.
$$

The objective function (1) is a linear combination of the total expected profit and the total penalty. The total expected profit corresponds to the sum over all expected profits e_{ij} that result from assigning a customer i to an activity j. The total penalty corresponds to the sum of the products of the slack variables and the corresponding penalty constants α, β, γ, or δ. Constraints (2) represent the minimum assignment constraints. For each minimum assignment constraint l, the number of customers assigned to the relevant activities $J_l^{\underline{a}}$ plus the corresponding slack variable $z_l^{\underline{a}}$ must satisfy the lower bound $b_l^{\underline{a}}$. Constraints (3) correspond to the maximum assignment constraints. For each maximum assignment constraint l, the number of customers assigned to the relevant activities $J_l^{\overline{a}}$ must not exceed the upper bound $b_l^{\overline{a}}$. Constraints (4) represent the budget constraints. Each assignment of a customer i to an activity j generates a cost c_j. For each budget constraint l, a

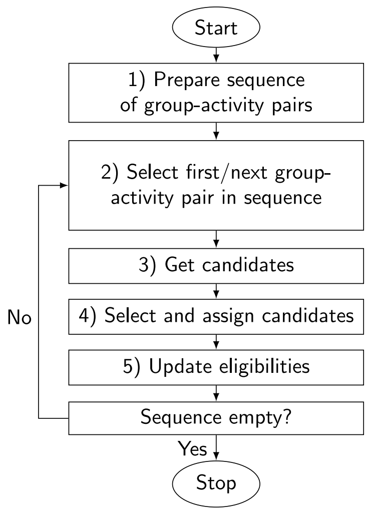
Start
1) Prepare sequence of group-activity pairs
2) Select first/next group-activity pair in sequence
3) Get candidates
4) Select and assign candidates
5) Update eligibilities
Sequence empty?
No
Yes
Stop

Fig. 4. Flowchart of the iterative algorithm.

budget $b_l^{\overline{b}}$ is imposed on the total cost that results from assigning customers to the relevant activities $J_l^{\overline{b}}$. Constraints (5) represent the minimum sales constraints. Each assignment of a customer i to an activity j leads to a positive customer response with a probability q_{ij}. For each minimum sales constraint l, the number of expected sales that results from assigning customers to the relevant activities $J_l^{\underline{s}}$, i.e., the sum of the corresponding response probabilities of the assigned customers, plus the corresponding slack variable $z_l^{\underline{s}}$ must satisfy the lower bound $b_l^{\underline{s}}$. Constraints (6) represent the maximum sales constraints. For each maximum sales constraint l, the number of expected sales that results from assigning customers to the relevant activities $J_l^{\overline{s}}$ minus the corresponding slack variable $z_l^{\overline{s}}$ must not exceed the upper bound $b_l^{\overline{s}}$. Constraints (7) represent the minimum contact constraints. For each contact rule $l = 1, \ldots, n^{\underline{m}}$, a separate constraint is imposed for each customer. This constraint ensures that the number of times customer i is assigned to the relevant activities $J_l^{\underline{m}} \cap J_i$ plus the corresponding slack variable $z_{il}^{\underline{m}}$ satisfies the lower bound $b_l^{\underline{m}}$. Constraints (8) correspond to the maximum contact constraints. For each contact rule $l = 1, \ldots, n^{\overline{m}}$, a separate constraint is imposed for each customer if this customer is eligible for more than $b_l^{\overline{m}}$ relevant activities $J_l^{\overline{m}} \cap J_i$. This constraint ensures that the number of times customer i is assigned to the relevant activities $J_l^{\overline{m}} \cap J_i$ does not exceed the upper bound $b_l^{\overline{m}}$. Constraints (9) represent the conflict constraints. The conflict constraints guarantee for each pair of conflicting activities (j_1, j_2) in set T and for each customer i who is eligible for both activities j_1 and j_2 that the customer can only be assigned to one of the two conflicting activities. Note that this formulation is very similar to the formulation of Bigler et al. (2019) that was developed for a slightly different planning problem. If one wants to consider all constraints as hard constraints, the upper bounds on the slack variables can be set to zero.

5. Matheuristic

The main idea of the matheuristic is to solve a mathematical model for groups of customers rather than individual customers. The key feature is that we incorporate customer-specific constraints in the group-level model, which allows transforming the solution from the group-level model into a customer-level solution with almost no loss in solution quality. Moreover, the matheuristic is designed in such a way that the user can control the trade-off between solution quality and running time with a single parameter. The matheuristic consists of four steps. Figure 3 provides an overview of these four steps. In Sections 5.1–5.4, we explain the four steps in detail. In Section 5.5, we apply the proposed matheuristic to the illustrative example from Section 2.3.

5.1. Build groups by eligibility pattern (Step 1)

In the first step of the matheuristic, the customers are grouped according to their eligibility patterns. An eligibility pattern p indicates the activities for which a customer is eligible. Two customers share the same eligibility pattern if they are eligible for the same activities. For example, for an instance with three activities and a customer who is eligible for activities 1 and 3, but not for activity 2, the customer's eligibility pattern corresponds to [1, 0, 1]. We generate an eligibility matrix with $|I|$ rows and $|J|$ columns. This matrix contains values of only zero and one, where a value of one indicates that a customer $i \in I$ is eligible for an activity $j \in J$. All customers with the same eligibility pattern are grouped together. This grouping can be efficiently produced by sorting the rows of the eligibility matrix. The grouping by eligibility patterns is essential for the decomposition strategy of the matheuristic because it ensures that the customers in the same group are affected by the same conflicts between activities. Only due to this feature of the customer groups we are able to enforce customer-specific constraints in the third step of the matheuristic. Note that if the number of eligibility patterns is not much smaller than the number of customers, one can reduce the number of different eligibility patterns by grouping similar but not identical patterns and replacing all patterns within each group with the intersection of the different patterns in the group. However, according to the telecommunications company, the number of different patterns is always substantially lower than the number of customers in real-world instances because the customers with the same subscriptions (or products) are mostly eligible for the same activities.

5.2. Divide groups according to expected profits (Step 2)

In the second step of the matheuristic, each group is further divided into up to k subgroups. We determine the subgroups by considering a clustering problem for each group. The goal is to partition the customers of the group into k clusters (subgroups) such that the customers in the same cluster have similar expected profits for the activities for which they are eligible. Note that all customers of the same group are eligible for the same activities. Hence, the input to each clustering problem is a matrix which has one row for each customer and one column for each activity for which the customers are eligible. The values in the matrix correspond to the expected profits of the customers for the respective activities. To determine the partition, we apply the mini batch k-means algorithm of Sculley (2010). The mini batch k-means algorithm is particularly suitable for large-scale applications due to its speed and memory efficiency. If there are fewer than k customers in a group, all customers are placed in separate subgroups. Increasing the parameter k leads to smaller but more homogeneous subgroups. For the sake of simplicity, the subgroups that result from this grouping step are subsequently referred to as groups of customers.

5.3. Determine the number of customers of the groups that are assigned to the activities (Step 3)

In the third step of the matheuristic, a linear model (LP) determines how many customers of each group are assigned to the activities. We introduce continuous decision variables x_{gj} that indicate how many customers of group g are assigned to activity j. For

Table 7
Additional notation of linear model.

Sets	
G	Groups determined in step 2 of the matheuristic
P	Eligibility patterns
G_p	Groups with eligibility pattern p
J_g	Activities for which customers in group g are eligible
$J^{\bar{c}}_{lp}$	Activities associated with constraint $l = 1, \dots, n^{\bar{c}}_p$, which ensures conflict rules for groups with eligibility pattern p
Parameters	
$\bar{e}_{gj}$	Average expected profit of customers of group g when assigned to activity j
$n^{\bar{c}}_p$	Number of constraints that are set up for each group g with eligibility pattern p to ensure conflict rules
o_g	Number of customers in group g
$\bar{q}_{gj}$	Average response probability of customers of group g when assigned to activity j
Decision variables	
$x_{gj} \in [0, o_g]$,	Number of customers of group g that are assigned to activity j
$z^{\underline{m}}_{gl} \in [0, b^{\underline{m}}_l o_g]$,	Slack variable of group g for minimum contact rule $l = 1, \dots, n^{\underline{m}}$

each decision variable, we compute the corresponding average expected profit $\bar{e}_{gj}$ and the average response probability $\bar{q}_{gj}$ based on the respective expected profits e_{ij} and response probabilities q_{ij} of the customers of group g. Table 7 shows the notation that is used, in addition to the notation already introduced in Table 6. The LP reads as follows:

$$
\text{(LP)}\left\{
\begin{array}{lll}
\text{Max.} & \sum_{g\in G}\sum_{j\in J_g}\bar{e}_{gj}x_{gj} - \left(\alpha\sum_{l=1}^{n^{\underline{a}}} z^{\underline{a}}_l + \beta\sum_{l=1}^{n^{\underline{s}}} z^{\underline{s}}_l + \gamma\sum_{l=1}^{n^{\bar{s}}} z^{\bar{s}}_l + \delta\sum_{g\in G}\sum_{l=1}^{n^{\underline{m}}} z^{\underline{m}}_{gl}\right) & (15)\\
\text{s.t.} & \sum_{g\in G}\sum_{j\in J_g\cap J^{\underline{a}}_l} x_{gj} + z^{\underline{a}}_l \geq b^{\underline{a}}_l \quad (l = 1,\dots,n^{\underline{a}}) & (16)\\
 & \sum_{g\in G}\sum_{j\in J_g\cap J^{\bar{a}}_l} x_{gj} \leq b^{\bar{a}}_l \quad (l = 1,\dots,n^{\bar{a}}) & (17)\\
 & \sum_{g\in G}\sum_{j\in J_g\cap J^{\bar{b}}_l} c_j x_{gj} \leq b^{\bar{b}}_l \quad (l = 1,\dots,n^{\bar{b}}) & (18)\\
 & \sum_{g\in G}\sum_{j\in J_g\cap J^{\underline{s}}_l} \bar{q}_{gj}x_{gj} + z^{\underline{s}}_l \geq b^{\underline{s}}_l \quad (l = 1,\dots,n^{\underline{s}}) & (19)\\
 & \sum_{g\in G}\sum_{j\in J_g\cap J^{\bar{s}}_l} \bar{q}_{gj}x_{gj} - z^{\bar{s}}_l \leq b^{\bar{s}}_l \quad (l = 1,\dots,n^{\bar{s}}) & (20)\\
 & \sum_{j\in J_g\cap J^{\underline{m}}_l} x_{gj} + z^{\underline{m}}_{gl} \geq o_g b^{\underline{m}}_l \quad (g\in G;\ l = 1,\dots,n^{\underline{m}}) & (21)\\
 & \sum_{j\in J_g\cap J^{\bar{m}}_l} x_{gj} \leq o_g b^{\bar{m}}_l \quad (g\in G;\ l = 1,\dots,n^{\bar{m}}) & (22)\\
 & \sum_{j\in J^{\bar{c}}_{lp}} x_{gj} \leq o_g \quad (p\in P;\ g\in G_p;\ l = 1,\dots,n^{\bar{c}}_p) & (23)\\
 & x_{gj} \in [0, o_g] \quad (g\in G;\ j\in J_g) & (24)\\
 & z^{\underline{m}}_{gl} \in [0, b^{\underline{m}}_l o_g] \quad (g\in G;\ l = 1,\dots,n^{\underline{m}}) & (25)\\
 & (11) - (13) &
\end{array}
\right.
$$

In the objective function (15), the total average expected profit minus the total penalty associated with the soft constraints is maximized. Constraints (16)–(22) are formulated analogously to the constraints from Section 4 for groups of customers instead of individual customers. In the group-level model, it is not sufficient to simply consider pairs of conflicting activities for incorporating the conflict rules. Considering only pairs of activities, as in constraints (9), will result in excessive assignments on the group level, as shown in the following example. Consider two customers i_1 and i_2 who both belong to group g_1. Assume that these customers are eligible for three activities j_1, j_2, and j_3, which all have conflicts among each other. Therefore, each customer can only be assigned to one of the three activities. The analogous pairwise constraints to the constraints (9) for this example would be formulated as follows: $x_{g_1,j_1} + x_{g_1,j_2} \leq o_{g_1}$, $x_{g_1,j_1} + x_{g_1,j_3} \leq o_{g_1}$, $x_{g_1,j_2} + x_{g_1,j_3} \leq o_{g_1}$ with $o_{g_1} = 2$. The solution $x_{g_1,j_1} = x_{g_1,j_2} = x_{g_1,j_3} = 1$ satisfies these constraints. However, because there are only two customers in group g_1 and each customer can only be assigned to one of the three activities j_1, j_2, and j_3, this solution cannot be translated into a customer-level solution without losing one assignment. To better represent the conflict rules already in the group-level model, we introduce an alternative modeling technique. For each eligibility pattern p, we generate one or multiple sets $J^{\bar{c}}_{lp}$ of conflicting activities of maximal size. Constraints (23) ensure that a maximum of o_g customers can be assigned to two or more conflicting activities $J^{\bar{c}}_{lp}$ for each eligibility pattern p and each group g with eligibility pattern p. In Section 6, we will explain in detail how these sets of conflicting activities $J^{\bar{c}}_{lp}$ are efficiently generated.

Note that there are still special cases in which the LP might assign too many customers on the group level. Consider five activities j_1–j_5 with $T = \{(j_1, j_2), (j_2, j_3), (j_3, j_4), (j_4, j_5), (j_5, j_1)\}$. Assume that group g_1 has two customers i_1 and i_2 who are eligible for all five activities. Then, the solution $x_{g_1,j_1} = x_{g_1,j_2} = x_{g_1,j_3} = x_{g_1,j_4} = x_{g_1,j_5} = 1$ is feasible for constraints (23). However, one of these five assignments will be lost in the customer-level assignment.

5.4. Assign individual customers of the groups to the activities (Step 4)

In this step, we apply an iterative algorithm to assign individual customers to the activities based on the group-level assignment from the previous step. Figure 4 provides a flowchart of the algorithm. The basic idea is to assign the most profitable customers of group g to activity j for each variable x_{gj} of the LP with a non-zero value. The iterative algorithm assigns the customers to the activities without violating hard constraints. Next, we explain the iterative algorithm step-by-step.

First, the algorithm determines a specific sequence for the group-activity pairs of the decision variables x_{gj} with a value greater than or equal to one. This sequence determines in which order the customers are assigned to the activities. A random sequence is likely to lead to a loss of group-level assignments, as illustrated by the following example. Consider a group g_1 that contains two customers i_1 and i_2. Assume that the customers of group g_1 are eligible for the three activities j_1, j_2, and j_3 and that activity j_1 conflicts with both activities j_2 and j_3. Further assume that the solution of the LP is $x_{g_1,j_1} = x_{g_1,j_2} = x_{g_1,j_3} = 1$ and that the following random sequence $[(g_1, j_2), (g_1, j_3), (g_1, j_1)]$ of the group-activity pairs is given. If customer i_1 is assigned to activity j_2 in the first iteration, and customer i_2 is assigned to activity j_3 in the second iteration, then no customer can be assigned to activity j_1 in the third iteration because both customers are already assigned to activities that conflict with activity j_1. Instead of using a random sequence, we prepare a sequence based on a conflict graph $\mathcal{G} = (V, E)$ which can be constructed from the conflict rules. The nodes V of the conflict graph $\mathcal{G}$ correspond to the activities of an instance (i.e., $V = J$), and the edges E between nodes represent conflicts among activities. We first sort all group-activity pairs according to the group index such that all activities of the same group are processed sequentially. To determine the sequence of activities for each group g, we construct a subgraph of the conflict graph $\mathcal{G}$ that only contains the activities as nodes to which customers from group g are assigned. The activity that corresponds to the node with the highest degree in this subgraph is selected first. Then, activities are added iteratively to the sequence in decreasing order of the number of edges that connect the respective nodes in the subgraph to nodes that represent already added activities. Note that this number of edges is recomputed every time an activity is added to the sequence. In case of ties, the activity with the lower index is first. For the example from above, this sorting mechanism results in the sequence $[(g_1, j_1), (g_1, j_2), (g_1, j_3)]$, based on which all three assignments can be conducted on the customer level.

Second, the iterative algorithm selects the first/next group-activity pair (g, j) in the sequence.

Third, the iterative algorithm identifies the customers of group g that can be assigned to activity j without violating any hard customer-specific constraints. Even though all customers of group g are initially eligible for activity j, it is possible that some customers of group g cannot be assigned to activity j because such an assignment would violate a conflict rule or a maximum contact rule due to assignments in earlier iterations. The customers of group g that can be assigned are referred to as candidates.

Fourth, the iterative algorithm selects the $\lfloor x_{gj} \rfloor$ candidates with the highest expected profits for activity j and assigns them. If the number of candidates is lower than $\lfloor x_{gj} \rfloor$, then all candidates are assigned. The number of candidates can be lower than $\lfloor x_{gj} \rfloor$ in special cases. Such a special case is the example from Section 5.3 in which the conflict graph $\mathcal{G}$ represents a circle of five activities j_1, j_2, j_3, j_4, and j_5, i.e., set $T = \{(j_1, j_2), (j_2, j_3), (j_3, j_4), (j_4, j_5), (j_5, j_1)\}$. The last group-activity pair in this example may have no candidate because all customers of the group have been assigned to conflicting activities in previous iterations, even though the solution of the LP intended to assign one or more customers.

Fifth, the iterative algorithm updates the eligibilities of the assigned candidates for other activities. All assigned candidates are no longer eligible for activities that are in conflict with activity j. Moreover, it is possible that with the assignment to activity j, some of the candidates will reach their maximum number of assignments for one or multiple maximum contact rules. These customers are no longer eligible for other activities that are affected by these maximum contact rules. Finally, the iterative algorithm determines whether the sequence of group-activity pairs is empty. If that is the case, the iterative algorithm stops and returns the customer-level assignment x_{ij}; otherwise, the next group-activity pair is selected from the sequence.

5.5. Illustrative example

We apply the matheuristic to the illustrative example from Section 2.3 step-by-step. In the first step, the customers who are eligible for the same activities are grouped together. This leads to the following four groups: $\{7, 10, 14, 19, 20\}$, $\{2, 3, 4, 9, 17\}$, $\{1, 5, 8, 11, 12, 13, 16\}$, and $\{6, 15, 18\}$. In the second step, these groups are further divided using the mini batch k-means algorithm with a value of $k = 2$. The first two columns of Table 8 show the resulting eight groups and the customers I_g who belong to these groups. In the third step, the LP is set up and solved for the eight groups and eight activities. Table 8 shows the resulting values of the decision variables x_{gj}. A dash (-) indicates that the customers of this group are not eligible for this activity, and thus no assignment can be conducted. In the fourth step, the customers are iteratively assigned to the activities. Figure 5 shows the conflict graph $\mathcal{G}$ based on which the sequence of the group-activity pairs is determined. The complete sequence of the group-activity pairs is: $[(1, \text{A7}), (1, \text{A8}), (2, \text{A7}), (2, \text{A8}), (3, \text{A6}), (4, \text{A6}), (5, \text{A4}), (5, \text{A5}), (6, \text{A3}), (6, \text{A4}), (6, \text{A5}), (7, \text{A1}), (7, \text{A6}), (8, \text{A1}), (8, \text{A6})]$. The algorithm iterates over all group-activity pairs in the derived order. In most of the iterations, all customers of the respective groups are assigned to the activities (as intended by the solution of the LP). Next, we will describe the iterations in which not all customers of a group are assigned to the considered activity. In the third iteration, customers of group 2 are assigned to activity A7. All three customers of group 2 are candidates, but $\lfloor x_{2,\text{A7}} \rfloor = 2$. Thus, only the two customers with the highest expected profits for activity A7 are assigned (here, customers 20 and 14). Also, in iteration 9, only one customer of group 6 must be assigned to activity A3. Thus, customer 16 is assigned. In iterations 10 and 11, only customers 8 and 1 of group 6 are candidates because customer 16 has been assigned to a conflicting activity in a previous iteration. Thus, these two customers are assigned to both activities A4 and A5. The total expected profit of the solution obtained by the matheuristic is 2,973 Euros (the same as in the optimal solution). The maximum sales constraint is violated by 0.14 (+0.02 as compared to the optimal solution), which leads to a penalty of $112(0.14) = 15.68$ Euros. The objective function value of the solution derived by the matheuristic is thus 2,957.32 Euros, which is 2.24 Euros below the objective function value of the optimal solution.

6. Preprocessing technique

In this section, we present the preprocessing technique that is used in the third step of the matheuristic in more detail. To implement constraints (23), we need to determine for each eligibility pattern $p \in P$ one or several sets of conflicting activities J^{c}_{lp}. Determining these sets without introducing redundancies is nontrivial and may become a computational bottleneck for large-scale instances if not implemented in an efficient manner. We propose a preprocessing technique that combines concepts from graph theory with array-computing to efficiently generate these sets. By using arrays (matrices) as the fundamental data structure, we can benefit from highly optimized array-computing libraries. The preprocessing technique consists of six steps. In Sections 6.1–6.6, we explain each step by means of an example that involves five activities j_1, j_2, j_3, j_4, and j_5. Figure 6 shows an overview of the different steps of the preprocessing technique using this example. Table 9 provides the notation of the matrices used for the preprocessing technique. In Section 6.7, we introduce an alternative mixed-binary linear programming formulation that uses the sets generated by the preprocessing technique.

Table 8
Groups in the illustrative example.

Group g	Customers I_g	x_{gj}							
		$j = $ A1	$j = $ A2	$j = $ A3	$j = $ A4	$j = $ A5	$j = $ A6	$j = $ A7	$j = $ A8
1	{7, 10}	-	-	-	-	0	-	2	2
2	{14, 19, 20}	-	-	-	-	0	-	2	3
3	{3, 17}	-	-	-	-	0	2	-	-
4	{2, 4, 9}	-	-	-	-	0	3	-	-
5	{5, 11, 12, 13}	-	-	0	4	4	-	-	-
6	{1, 8, 16}	-	-	1	2	2	-	-	-
7	{15, 18}	2	0	-	-	-	2	-	-
8	{6}	1	0	-	-	-	1	-	-

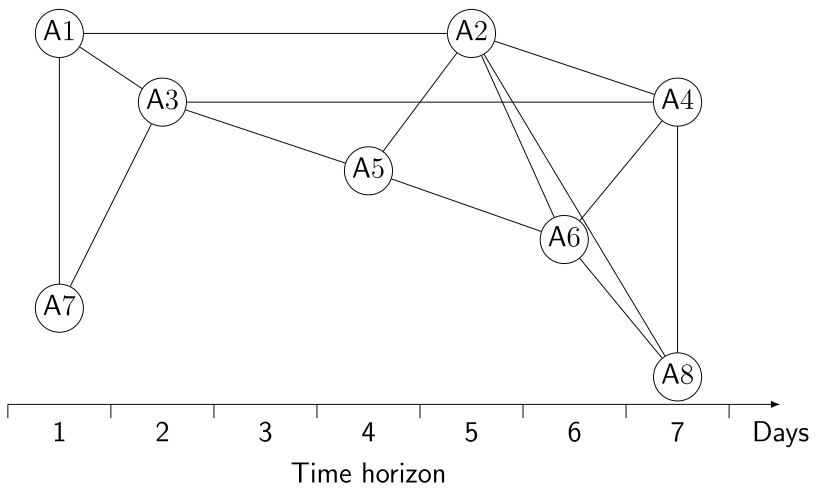

Fig. 5. Conflict graph $\mathcal{G}$ for the illustrative example.

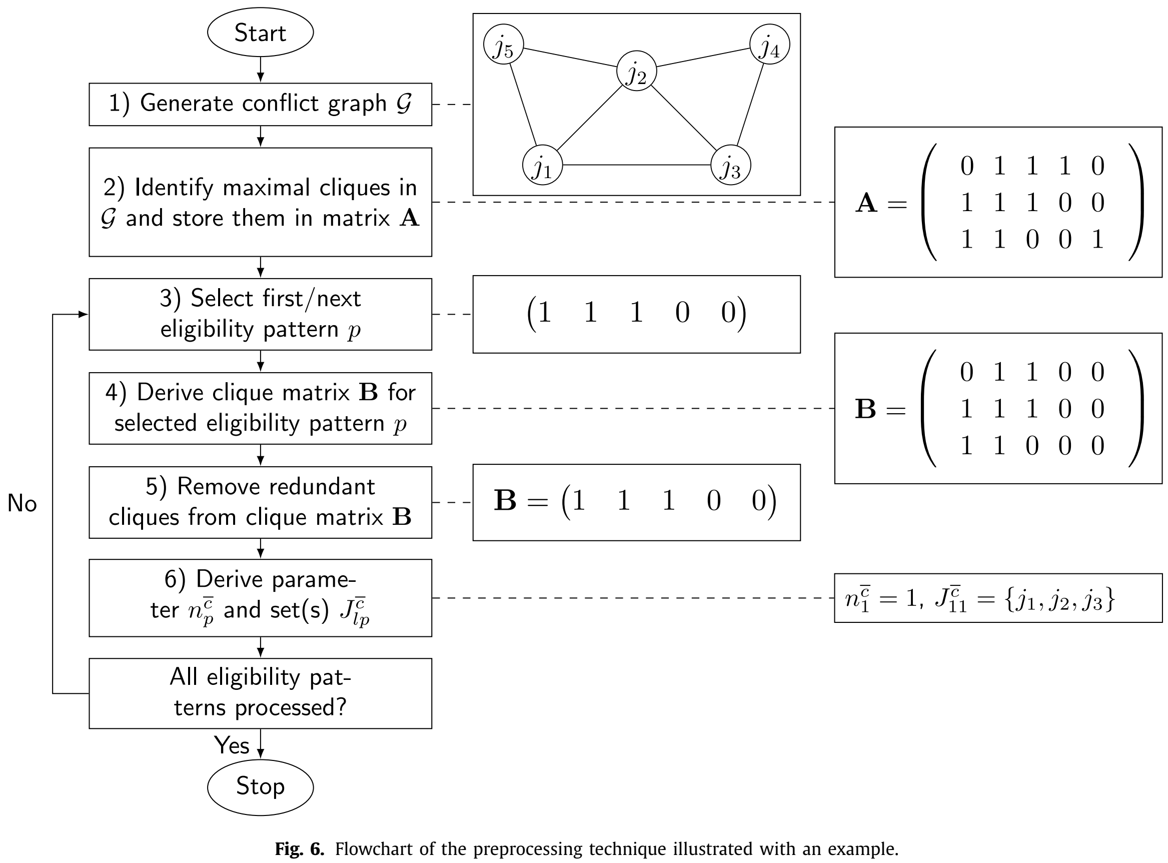
Start
1) Generate conflict graph $\mathcal{G}$
2) Identify maximal cliques in $\mathcal{G}$ and store them in matrix $\mathbf{A}$
3) Select first/next eligibility pattern p
4) Derive clique matrix $\mathbf{B}$ for selected eligibility pattern p
5) Remove redundant cliques from clique matrix $\mathbf{B}$
6) Derive parameter $n^{\bar{c}}_p$ and set(s) $J^{\bar{c}}_{lp}$
All eligibility patterns processed?
No
Yes
Stop
j_1
j_2
j_3
j_4
j_5
$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \end{pmatrix}$
$\mathbf{B} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$
$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \end{pmatrix}$
$n^{\bar{c}}_1 = 1,\ J^{\bar{c}}_{11} = \{j_1, j_2, j_3\}$

Fig. 6. Flowchart of the preprocessing technique illustrated with an example.

Table 9
Matrices used in the preprocessing technique.

Matrix	Initial dimensions	Domain	Description
A	$(c \times \|J\|)$	Binary	Contains the c maximal cliques of conflict graph $\mathcal{G}$ in rows
B	$(c' \times \|J\|)$	Binary	Contains the c' rows with two or more non-zero entries that result from an element-wise multiplication of each row of matrix **A** with eligibility pattern p
C	$(c' \times c')$	Integer	Obtained by multiplying the matrices **B** and $(\mathbf{B})^T$
D	$(c' \times c')$	Integer	Contains diagonal elements of matrix **C** in each row
E	$(c' \times c')$	Integer	Obtained by subtracting matrix **D** from matrix **C**

Table 10
Generated and real-world problem instances.

ID	Customers	Activities	Eligibility fraction [%]	Eligibility patterns
GS1	10,000	50	small (5)	few (50)
GS1'	10,000	50	small (5)	few (50)
GS2	10,000	50	large (15)	few (50)
GS3	10,000	50	small (5)	many (100)
GS4	10,000	50	large (15)	many (100)
GS5	20,000	75	small (5)	few (50)
GS6	20,000	75	large (15)	few (50)
GS7	20,000	75	small (5)	many (100)
GS8	20,000	75	large (15)	many (100)
GM1	100,000	100	small (5)	few (300)
GM1'	100,000	100	small (5)	few (300)
GM2	100,000	100	large (15)	few (300)
GM3	100,000	100	small (5)	many (800)
GM4	100,000	100	large (15)	many (800)
GM5	200,000	125	small (5)	few (300)
GM6	200,000	125	large (15)	few (300)
GM7	200,000	125	small (5)	many (800)
GM8	200,000	125	large (15)	many (800)
GL1	500,000	150	small (5)	few (300)
GL1'	500,000	150	small (5)	few (300)
GL2	500,000	150	large (15)	few (300)
GL3	500,000	150	small (5)	many (1,000)
GL4	500,000	150	large (15)	many (1,000)
GL5	1,000,000	175	small (5)	few (300)
GL6	1,000,000	175	large (15)	few (300)
GL7	1,000,000	175	small (5)	many (1,000)
GL8	1,000,000	175	large (15)	many (1,000)
RL1	987,486	133	small (1.0)	many (1,830)
RL1'	987,486	133	small (1.0)	many (1,830)
RL2	1,101,432	215	small (0.8)	many (3,196)
RL3	1,401,582	308	small (0.7)	many (5,833)
RL4	1,401,582	385	small (0.6)	many (5,833)
RVL1	2,171,792	50	large (17.3)	few (61)
RVL1'	2,171,792	50	large (17.3)	few (61)
RVL2	2,171,792	75	large (17.3)	few (61)
RVL3	2,171,792	100	large (16.6)	few (61)
RVL4	2,171,792	150	large (16.9)	few (61)
RVL5	2,171,792	200	large (16.9)	few (61)
RVL6	2,180,831	250	large (16.9)	few (108)
RVL7	2,180,831	295	large (16.3)	few (108)

6.1. Step one of the preprocessing technique

In step 1), we generate the conflict graph $\mathcal{G}$ from the predefined conflict rules. The nodes represent the activities and the edges represent the conflicts between activities. The conflict graph of the example used in this section is shown at the top of Fig. 6.

6.2. Step two of the preprocessing technique

In step 2), we identify all maximal cliques in the conflict graph $\mathcal{G}$ using the NetworkX implementation (cf. Hagberg, Schult & Swart, 2008) of the algorithm of Bron & Kerbosch (1973). Even though the problem of finding all maximal cliques in a graph is *NP*-hard, real-world graphs often exhibit properties that enable solving clique problems in a few seconds, even when the graph has millions of nodes. Walteros & Buchanan (2020) found that the maximum clique problem is easy to solve on graphs with a small clique-core gap which they define to be the difference between the graph's clique number and its degeneracy-based upper bound on the clique number. Like many other real-world graphs, also the conflict graphs considered in this paper have a clique-core gap of zero. We store these maximal cliques in a binary matrix **A**. Each row of matrix **A** corresponds to a maximal clique, and each column corresponds to an activity. A value of one indicates that the activity in the corresponding column is part of the maximal clique in the corresponding row. The conflict graph $\mathcal{G}$ of the example has three maximal cliques, and thus matrix **A** consists of $c = 3$ rows and $|J| = 5$ columns (cf. Fig. 6).

6.3. Step three of the preprocessing technique

In step 3), we select the first/next eligibility pattern p from set P. For illustrative purposes, assume that we select an eligibility pattern $[1, 1, 1, 0, 0]$ with index $p = 1$. All customers with this eligibility pattern are eligible for the three activities j_1, j_2, and j_3 but not for the activities j_4 and j_5.

6.4. Step four of the preprocessing technique

In step 4), we derive the clique matrix **B** for the selected eligibility pattern p. The clique matrix **B** is computed by an element-wise multiplication of each row of matrix **A** with the eligibility pattern p. Only rows that contain two or more non-zero entries are relevant for setting up constraints to ensure conflict rules, because a conflict must always occur between at least two activities. Thus, we remove all rows that contain less than two non-zero entries. In the example, the resulting matrix **B** has $c' = 3$ rows (cf. Fig. 6).

6.5. Step five of the preprocessing technique

In step 5), we remove cliques (rows) from the clique matrix **B** that are a subset of another clique (row) of matrix **B**. Here, a clique is a subset of another clique if it is either identical to the other clique or if it is contained in the other clique. A clique is contained in another clique if it has only values of one in columns in which the other clique also has values of one. In the example, the first and third cliques of matrix **B** are a subset of the second clique (cf. Fig. 6). We detect all cliques which are a subset of another clique using array-computing steps. These steps are illustrated for the matrix **B** in Fig. 7. We start by multiplying matrix **B** with the transposed matrix $(\mathbf{B})^T$. The resulting integer matrix **C** indicates, in the diagonal, how many values of one the corresponding clique in the matrix **B** contains. The off-diagonal elements indicate, for each pair of cliques of the matrix **B**, how many values of one they have in common (how many values of one occur in the same columns). For the example, matrix **C** is displayed in Fig. 7 (cf. ①). Moreover, we generate a matrix **D** that contains the diagonal elements of matrix **C** in the rows. In the example, the first and the third rows of matrix **D** correspond to twos, and the second row of matrix **D** corresponds to threes. We then subtract matrix **D** from matrix **C** and

$$\mathbf{B} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\textcircled{1}} \mathbf{B}(\mathbf{B})^T = \mathbf{C} = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 2 \end{pmatrix}, \mathbf{D} = \begin{pmatrix} 2 & 2 & 2 \\ 3 & 3 & 3 \\ 2 & 2 & 2 \end{pmatrix} \xrightarrow{\textcircled{2}} \mathbf{C} - \mathbf{D} = \mathbf{E} =$$

$$\begin{pmatrix} 0 & 0 & -1 \\ -1 & 0 & -1 \\ -1 & 0 & 0 \end{pmatrix} \xrightarrow{\textcircled{3}} \mathbf{E} = \begin{pmatrix} -1 & 0 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{pmatrix} \xrightarrow{\textcircled{4}} \mathbf{B} = \begin{pmatrix} \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{0} \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{C} = \begin{pmatrix} \cancel{2} & \cancel{2} & \cancel{1} \\ \cancel{2} & 3 & 2 \\ \cancel{1} & 2 & 2 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 2 & 2 \end{pmatrix}, \mathbf{D} = \begin{pmatrix} \cancel{2} & \cancel{2} & \cancel{2} \\ \cancel{3} & 3 & 3 \\ \cancel{2} & 2 & 2 \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ 2 & 2 \end{pmatrix} \xrightarrow{\textcircled{5}} \mathbf{C} - \mathbf{D} = \mathbf{E} = \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix}$$

$$\textcircled{6}\; \mathbf{E} = \begin{pmatrix} -1 & -1 \\ 0 & -1 \end{pmatrix} \xrightarrow{\textcircled{7}} \mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ \cancel{1} & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{0} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Fig. 7. Example for removing cliques from the binary matrix **B** that are a subset of another clique.

store the values in an integer matrix **E** (cf. ② in Fig. 7). The elements of matrix **E** show for each pair of cliques of the matrix **B** if the clique is contained in the other clique. For example, the element in the first row and second column of matrix **E** is zero, which means that the first clique of the matrix **B** is contained in the second clique of the matrix **B**. Naturally, the diagonal elements of the matrix **E** are zero, because each clique of the matrix **B** is identical to itself. We then want to remove the cliques that are a subset of another clique, i.e., the rows that contain a value of zero in an off-diagonal element of the matrix **E**. Here it is important to notice that if two cliques of the matrix **B** were exactly identical, then both of the corresponding rows in matrix **E** would contain a value of zero as an off-diagonal element, but we only want to remove one of these rows and keep the other one (otherwise we would miss a clique). To avoid removing too many cliques, we first check whether each clique in matrix **B** is a subset of a clique *below* it. Therefore, we fill the lower half and the diagonal of matrix **E** with values of negative one (cf. ③ in Fig. 7). Note that we could fill in any arbitrary non-zero value. Next, we check which rows in matrix **E** contain a value of zero. The rows that contain a value of zero in matrix **E** indicate that the corresponding clique in matrix **B** is a subset of another clique and can be removed. In the example, we remove the first clique from matrix **B** (cf. ④ in Fig. 7). We also update the matrices **C** and **D** by removing the first row and first column. Next, we check whether each clique of matrix **B** is a subset of a clique *above* it. We continue with the updated integer matrix **C** and again subtract the updated matrix **D** to obtain the updated matrix **E** (cf. ⑤ in Fig. 7). Then, we again fill the diagonal of matrix **E** with values of negative one (the upper half of matrix **E** no longer contains any zeros because these rows have already been removed) and determine whether any values of zero occur in the lower half of matrix **E** (cf. ⑥ in Fig. 7). The rows that contain a value of zero in matrix **E** again indicate that the corresponding clique in matrix **B** is a subset of another clique and can be removed. In the example, we remove the second clique from matrix **B** (cf. ⑦ in Fig. 7).

6.6. Step six of the preprocessing technique

In step 6), we derive the parameter $n_p^{\bar{c}}$ and the set(s) $J_{lp}^{\bar{c}}$ from matrix **B**. Each row of matrix **B** results in a constraint for the customers with eligibility pattern p. In the example, matrix **B** ultimately contains one row, and thus $n_1^{\bar{c}} = 1$. Set $J_{lp}^{\bar{c}}$ corresponds to the activities associated with constraint $l = 1, \dots, n_p^{\bar{c}}$. In the example, $J_{11}^{\bar{c}} = \{j_1, j_2, j_3\}$ because the first row of matrix **B** includes activities j_1, j_2, and j_3. Finally, we verify whether all eligibility patterns have been processed. If that is the case, we stop; otherwise, we select the next eligibility pattern p. For large-scale instances, steps 3) to 6) of Fig. 6 can be parallelized for different eligibility patterns to further reduce running times.

As an alternative to applying our preprocessing technique, the sets $J_{lp}^{\bar{c}}$ could be determined by computing an individual conflict graph for each eligibility pattern and by deriving the sets $J_{lp}^{\bar{c}}$ from these conflict graphs directly using the algorithm of Bron & Kerbosch (1973) on each of these conflict graphs. However, constructing a separate conflict graph for each eligibility pattern and computing the maximal cliques in each of these conflict graphs is much slower than the array-computing, especially for instances comprising a large number of eligibility patterns. Here, it is important to note that our preprocessing technique is not an alternative algorithm to compute maximal cliques but rather a procedure that generates the sets of conflicting activities $J_{lp}^{\bar{c}}$ for each eligibility pattern p without introducing redundancies; i.e., cliques that are a subset of another clique for a specific eligibility pattern p are identified efficiently, which prevents introducing a large number of redundant constraints (see Tables 12 and 14).

6.7. An alternative mixed-binary linear programming formulation

The parameters $n_p^{\bar{c}}$ and sets $J_{lp}^{\bar{c}}$ can also be used in a mixed-binary linear programming formulation to incorporate the conflict rules. Constraints (26) ensure that each customer i with eligibility pattern p is assigned to at most one of two or more conflicting activities $J_{lp}^{\bar{c}}$. Set I_p includes all customers with the eligibility pattern p.

$$\sum_{j \in J_{lp}^{\bar{c}}} x_{ij} \leq 1 \qquad (p \in P, i \in I_p, l = 1, \dots, n_p^{\bar{c}}) \tag{26}$$

We formulate an alternative mixed-binary linear program that uses constraints (26) instead of constraints (9) and reads as follows:

$$(\text{MBLP'}) \begin{cases} \text{Max.} & (1) \\ \text{s.t.} & (2) - (8), (10) - (14), (26) \end{cases}$$

With constraints (26), fewer constraints are required to ensure the conflict rules without loss of generality. One advantage that we noticed is that for our problem instances the linear programming relaxation of model MBLP′ was tighter than the linear programming relaxation of model MBLP.

7. Results

In this section, we compare the performance of the matheuristic to the performance of the MBLP and MBLP′. In Section 7.1, we describe the generated and real-world instances. In Section 7.2, we present the experimental design. In Section 7.3, we compare the performance of the MBLP to the performance of the MBLP′. In Section 7.4, we assess the overall performance of our matheuristic and investigate the effect of two key components of the matheuristic on running time and solution quality.

7.1. Problem instances

Our test set comprises 13 real-world instances and 27 instances that we manually generated based on real-world data (cf. Table 10). First, we describe the generated instances in more detail. These instances include small (GS), medium (GM), and large (GL) instances. The small instances comprise up to 20,000 customers and 75 activities, the medium instances comprise up to 200,000 customers and 125 activities, and the large instances comprise up to 1,000,000 customers and 175 activities. We generated different instances by varying the eligibility fraction (small/large) and the number of eligibility patterns (few/many). The eligibility fraction specifies the percentage of activities a customer is eligible for on average. For the generated instances, the eligibility fraction in Table 10 might slightly differ from the actual eligibility fraction of the instance as a consequence of the randomized generation process. The response probabilities, the costs per assignment, and the expected profits were derived from real-world data. The costs per assignment and the expected profits were scaled with a factor to preserve confidentiality. The constraints were defined for each instance in consultation with the company. All generated instances are publicly available (cf. https://github.com/phil85/customer-assignment-instances). The real-world instances consist of five large instances (RL) comprising up to 1.4 million customers and 385 activities, and eight very large instances (RVL) comprising over 2 million customers and up to 295 activities. While the RL instances have small eligibility fractions but many eligibility patterns, the RVL instances have high eligibility fractions but few eligibility patterns. The real-world instances are confidential and thus not publicly available. The instances GS1’, GM1’, GL1’, RL1’, RVL1’ differ from the respective instances GS1, GM1, GL1, RL1, RVL1 only in terms of constraints. The instances GS1’, GM1’, GL1’, RL1’, RVL1’ are infeasible if all soft constraints are considered as hard constraints.

7.2. Experimental design

The matheuristic, the MBLP, and the MBLP′ are implemented in Python 3.7 and the Gurobi 8.1 solver is used. All computations are performed on an HP workstation with one Intel Xeon CPU with 2.20 GHz clock speed and 128 GB RAM. Even though the running time budget of the company is 30 minutes, we applied the exact approaches with a time limit of 10,000 seconds to obtain better reference values for evaluating the solutions of the matheuristic. For the matheuristic, we set parameter $k = 20$ for all instances. The matheuristic is further applied to selected instances with different values of k. In consultation with the company, the constants α, β, γ, and δ are each set to the maximum absolute expected profit of the corresponding instance. Setting α and δ to the maximum absolute expected profit ensures that the objective function value cannot be improved by having fewer assignments or contacts than prescribed by the bounds. This reflects the preference of the company to reach the prescribed bounds if possible. The company accepts a shortfall or an exceedance of the lower and upper bounds on the number of sales if the corresponding assignments have a small or a large expected profit, respectively. Setting β and γ to the maximum absolute expected profit is an adequate penalty for sales constraints from the perspective of the company. The reported running times of all approaches include the time to compute relevant sets and parameters, the time to set up and solve the optimization models with Gurobi, and the time used for the iterative algorithm of the matheuristic. The time used for importing and exporting data is excluded because it is equivalent for all three approaches.

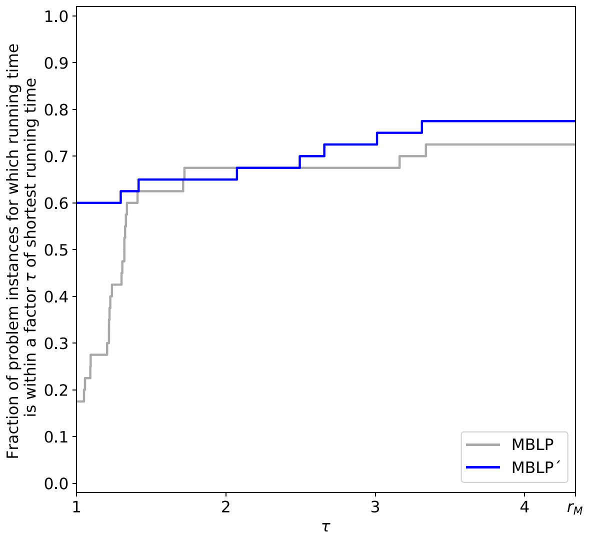

Fig. 8. Performance profiles for the MBLP and the MBLP′ (cf. Dolan & Moré, 2002).

7.3. Comparison of MBLP and MBLP′

First, we compare the performances of the MBLP and the MBLP′. Table 11 reports, for each instance and each formulation, the objective function value (OFV), the total penalty and in brackets the number of slack variables that take a positive value, the MipGap, the total number of constraints, and the total running time. The entry “lim” means that the time limit was reached. A dash (-) indicates that setting up the respective model resulted in an out-of-memory error. From Table 11, we can conclude that the MBLP′ has much fewer constraints than the MBLP for all instances. As a consequence, a larger number of feasible and also optimal solutions can be derived, and the running times are generally shorter. Figure 8 compares the running times of the MBLP and the MBLP′ using performance profiles (Dolan & Moré, 2002). Each curve corresponds to an approach and indicates for what fraction of problem instances the running time of the respective approach was within a factor τ of the shortest running time. The parameter r_M is set to the highest factor that occurs plus one; and if an approach cannot solve an instance to optimality within the time limit, the factor for the corresponding instance is set to r_M. From Fig. 8 (at $\tau = 1$), we can see that the MBLP′ is the faster approach for 60% of the instances while the MBLP is faster for 17.5% of the instances. Note that the MBLP is only faster when solving small instances. The MBLP′ maintains a considerably larger fraction up to $\tau = 1.3$. Also for large values of τ (when the performance profiles become flat), we can see that the fraction of instances that can be solved to optimality within the time limit is higher for the MBLP′ than for the MBLP. To further investigate the substantial difference in the number of constraints of the MBLP and the MBLP′, we applied the MBLP′ with and without step 5) of the preprocessing technique. As explained in Section 6, step 5) removes the cliques from clique matrices that are a subset of another clique. Table 12 reports the total number of constraints to ensure conflict rules ($\sum$) for groups of instances for the MBLP, the MBLP′, and the MBLP′ without conduct-

Table 11
Results of the MBLP, the MBLP′ and the matheuristic for generated and real-world instances.

	MBLP					MBLP′					Matheuristic					
ID	OFV [100k]	Penalty [100k] (#pos. slack.)	MipGap [%]	Constr. [1k]	CPU [sec]	OFV [100k]	Penalty [100k] (#pos. slack.)	MipGap [%]	Constr. [1k]	CPU [sec]	OFV [100k]	Penalty [100k] (#pos. slack.)	Sum slack var.	Constr. [1k]	CPU [sec]	Gap [%]
GS1	**1.5**	0.00 (0)	0.0	34	**3.4**	**1.5**	0.00 (0)	0.0	14	8.4	1.5	0.00 (0)	0.00	2	5.7	0.3
GS1’	**1.2**	0.27 (1)	0.0	34	**2.7**	**1.2**	0.27 (1)	0.0	14	8.2	1.2	0.27 (1)	47.00	2	4.3	0.4
GS2	**1.4**	0.00 (0)	0.0	172	8.1	**1.4**	0.00 (0)	0.0	36	11.5	1.4	0.00 (2)	0.03	4	**5.4**	1.7
GS3	**1.0**	0.00 (0)	0.0	41	**2.7**	**1.0**	0.00 (0)	0.0	17	9.0	1.0	0.00 (0)	0.00	5	7.9	0.3
GS4	**5.1**	0.00 (0)	0.0	180	11.4	**5.1**	0.00 (0)	0.0	36	14.7	5.0	0.00 (0)	0.00	7	**10.3**	1.6
GS5	**1.3**	0.00 (0)	0.0	36	**3.3**	**1.3**	0.00 (1)	0.0	20	8.9	1.3	0.00 (1)	0.05	2	4.8	0.2
GS6	**13.9**	0.00 (0)	0.0	815	37.6	**13.9**	0.00 (0)	0.0	83	26.7	13.5	0.01 (2)	1.00	4	**7.3**	2.5
GS7	**4.2**	0.00 (0)	0.0	72	**5.0**	**4.2**	0.00 (0)	0.0	29	10.3	4.2	0.01 (1)	1.00	4	8.9	0.5
GS8	**9.2**	0.00 (0)	0.0	736	39.9	**9.2**	0.00 (0)	0.0	83	29.8	8.9	0.00 (0)	0.00	8	**12.4**	2.9
GM1	**36.0**	0.00 (0)	0.0	1,154	95.0	**36.0**	0.00 (0)	0.0	275	78.0	35.5	0.00 (1)	0.05	18	**32.8**	1.4
GM1’	**32.8**	1.16 (1)	0.0	1,154	109.2	**32.8**	1.16 (1)	0.0	275	82.7	32.3	1.16 (2)	135.05	18	**32.9**	1.3
GM2	**88.4**	0.00 (0)	0.0	5,993	317.1	**88.4**	0.00 (0)	0.0	529	184.1	84.8	0.00 (1)	0.10	32	**49.8**	4.0
GM3	**25.1**	0.00 (0)	0.0	1,244	68.3	**25.1**	0.00 (0)	0.0	285	**52.3**	24.9	0.00 (1)	0.03	48	84.3	0.8
GM4	**30.1**	0.00 (0)	0.0	5,811	869.0	**30.1**	0.00 (0)	0.0	524	653.0	29.3	0.00 (0)	0.00	84	**114.7**	2.7
GM5	**47.6**	0.00 (0)	0.0	3,162	450.9	**47.6**	0.00 (0)	0.0	623	341.5	46.8	0.01 (2)	1.13	19	**39.0**	1.8
GM6	**147.4**	0.00 (0)	0.0	18,358	1,449.5	**147.4**	0.00 (0)	0.0	1,210	845.8	141.3	0.00 (1)	0.19	36	**65.6**	4.2
GM7	**71.4**	0.00 (0)	0.0	3,292	303.4	**71.4**	0.00 (0)	0.0	617	233.2	70.5	0.00 (2)	0.15	51	**97.8**	1.4
GM8	**130.0**	0.00 (0)	0.0	18,309	7,282.6	**130.0**	0.00 (0)	0.0	1,200	5,982.5	123.6	0.00 (0)	0.00	96	**150.8**	4.9
GL1	**248.8**	0.00 (0)	0.0	10,033	3,708.4	**248.8**	0.00 (0)	0.0	1,673	3,037.1	243.0	0.01 (2)	1.01	21	**58.3**	2.4
GL1’	**-389.7**	616.69 (2)	0.0	10,033	3,335.9	**-389.7**	616.69 (2)	0.0	1,673	2,696.2	-394.9	616.69 (2)	53,273.00	21	**57.2**	1.3
GL2	-172.2	295.48 (3)	402.4	69,790	lim	-172.2	295.48 (3)	402.1	3,533	lim	**256.1**	0.00 (0)	0.00	42	**115.1**	-248.7
GL3	**197.3**	0.00 (0)	0.0	10,137	1,371.5	**197.3**	0.00 (0)	0.0	1,641	1,034.9	194.0	0.00 (0)	0.00	67	**139.3**	1.7
GL4	58.2	31.69 (1)	630.9	67,492	lim	58.2	31.69 (1)	631.2	3,491	lim	**211.8**	0.01 (2)	1.30	139	**251.2**	-263.9
GL5	-229.9	323.26 (2)	216.3	25,676	lim	-229.9	323.26 (2)	216.3	3,496	lim	**202.0**	0.00 (1)	0.09	22	**89.7**	-187.8
GL6	-	- (-)	-	-	-	166.7	4.89 (1)	443.6	7,163	lim	**452.1**	0.00 (1)	0.53	43	**209.3**	-171.3
GL7	**475.3**	0.00 (0)	0.0	26,239	5,087.4	**475.3**	0.00 (0)	0.0	3,528	4,148.5	465.9	0.00 (1)	0.03	72	**185.4**	2.0
GL8	-	- (-)	-	-	-	-2,084.1	2,378.82 (3)	201.8	7,136	lim	**1,022.2**	0.00 (1)	0.01	143	**354.6**	-149.0
RL1	**109.8**	0.00 (0)	0.0	388	579.6	**109.8**	0.00 (0)	0.0	244	548.5	109.2	0.00 (0)	0.00	21	**46.5**	0.6
RL1’	**110.4**	1.83 (1)	0.0	388	623.4	**110.4**	1.83 (1)	0.0	244	593.6	109.7	1.83 (1)	415.00	21	**46.5**	0.6
RL2	**149.3**	0.00 (0)	0.0	755	2,570.6	**149.3**	0.00 (0)	0.0	538	2,352.9	148.7	0.00 (0)	0.00	39	**85.2**	0.4
RL3	**224.6**	0.00 (0)	0.0	2,079	4,937.4	**224.6**	0.00 (0)	0.0	1,012	4,097.6	223.0	0.00 (0)	0.00	85	**173.9**	0.7
RL4	**237.6**	0.00 (0)	0.0	3,200	5,523.6	**237.6**	0.00 (0)	0.0	1,197	5,044.9	236.0	0.00 (0)	0.00	106	**205.8**	0.7
RVL1	**292.6**	0.00 (0)	0.0	64,911	1,724.3	**292.6**	0.00 (0)	0.0	3,052	545.2	291.7	0.00 (0)	0.00	6	**100.2**	0.3
RVL1’	**247.8**	44.87 (1)	0.0	64,911	1,735.8	**247.8**	44.87 (1)	0.0	3,052	519.8	246.8	44.87 (1)	2,000.00	6	**99.8**	0.4
RVL2	-	- (-)	-	-	-	**378.6**	0.00 (0)	0.0	3,999	1,917.1	377.0	0.00 (0)	0.00	9	**144.9**	0.4
RVL3	-	- (-)	-	-	-	**537.1**	0.00 (0)	0.0	9,431	3,868.9	534.7	0.00 (0)	0.00	15	**182.3**	0.4
RVL4	-	- (-)	-	-	-	-	- (-)	-	-	-	**751.0**	0.00 (0)	0.00	28	**273.5**	-
RVL5	-	- (-)	-	-	-	-	- (-)	-	-	-	**975.9**	0.00 (0)	0.00	40	**365.2**	-
RVL6	-	- (-)	-	-	-	-	- (-)	-	-	-	**1,273.3**	0.00 (0)	0.00	84	**476.4**	-
RVL7	-	- (-)	-	-	-	-	- (-)	-	-	-	**1,501.8**	0.00 (0)	0.00	102	**554.0**	-

(-) Not available due to out-of-memory error

Table 12
Number of constraints to ensure the conflict rules.

ID	MBLP	MBLP′	MBLP′ without step 5)
∑ GS	2,039,866	250,808 (-88%)	1,528,941 (-25%)
∑ GM	57,273,804	4,334,985 (-92%)	47,112,825 (-18%)
∑ GL	569,555,040	27,113,838 (-95%)	422,135,617 (-26%)
∑ (RL+RVL)	71,089,558	5,653,887 (-92%)	23,701,370 (-67%)

ing step 5). The results clearly show that the preprocessing technique is effective and that step 5) is essential.

We also examined two alternatives to the use of soft constraints. First, we tested a variant of the model MBLP′ in which all soft constraints are replaced by hard constraints. Of course, this variant was not able to devise solutions for the instances GS1', GM1', GL1', RL1', and RVL1'. For the other instances, the variant with hard constraints obtained identical or very similar results in terms of solution quality and running time as the variant with soft constraints.

Second, we tested a Lagrangian relaxation scheme (LRS). To obtain the LRS, we first formulated all soft constraints as hard constraints and deleted the penalty terms in the objective function. Next, we dualized the former soft constraints to obtain the Lagrangian subproblem. We iteratively solved the Lagrangian subproblem using the subgradient algorithm as described in Fisher (1981) and Fisher (1985). It turns out that for most instances, the LRS is inferior to the MBLP′ in terms of running time and solution quality. Only for some of the large instances, the LRS was able to devise better solutions than the MBLP′. However, these solutions are with one exception (GL5) still much worse than the solutions obtained by the matheuristic. For problem instance GL5, the LRS obtained a slightly better solution than the matheuristic (gap of 2.3%). The LRS also runs out of memory for the largest real-world problem instances. In our view, a main factor that negatively affects the performance of the LRS is that the Lagrangian subproblem in each iteration cannot be solved much faster than the original problem with hard or soft constraints because it still contains the large number of conflict constraints.

7.4. Performance of matheuristic

Next, we compare the performance of the matheuristic to the performance of the MBLP′. The right part of Table 11 reports the results of the matheuristic. The columns for the matheuristic are the same as for the MBLP′ except for the column sum slack variables, which states the sum of the slack variables, and the last column, which reports the gap between the OFV of the solution derived by the matheuristic and the OFV of the solution derived by the MBLP′. For each instance, we highlight the shortest running time and the highest OFV of all three approaches in bold. Note that some small positive penalties are rounded down to zero in Table 11 (cf. e.g., instance GS2). First, we compare the matheuristic and the MBLP′ in terms of solution quality. Most of the solutions of the matheuristic obtained with $k = 20$ are near-optimal. For problem instance GM8, we investigate how changing the value of parameter k affects the gap to the optimal solution and the running time of the matheuristic. Figure 9 visualizes the gap to the optimal solution and the running time of the matheuristic for various values of k. We can see that the gap can be further reduced by increasing parameter k. Interestingly, the gap decreases faster than linearly, whereas the running time appears to increase linearly. Parameter k can therefore be used to control the trade-off between solution quality and speed. Overall, there are only a few slack variables that take positive values, and the resulting penalties for the matheuristic are minor. Exceptions are the instances GS1', GM1', GL1', RL1', RVL1' which are infeasible if all soft constraints are considered as hard constraints and thus, a (large) positive penalty value cannot be avoided.

Table 13
Average gaps of the matheuristic for generated instances by eligibility fraction and eligibility patterns. Only instances that are solved to optimality by the MBLP′ are considered.

		Eligibility patterns		
		few	many	aggregated
Eligibility fraction	small	1.1	1.1	1.1
	large	3.1	3.0	3.1
	aggregated	1.8	1.9	1.8

Next, we compare the two approaches in terms of running time. The matheuristic is substantially faster than the MBLP′, especially for medium instances with a high eligibility fraction and for large and very large instances. Furthermore, the matheuristic is scalable to very large real-world instances. Figure 10 shows the running time for different steps of the matheuristic. From the bars, we can see that setting up the LP only plays a significant role if the instance has many eligibility patterns (cf. e.g., GL3–GL4, GL7–GL8, and RL1–RL4). In the first step of the matheuristic, customers who are eligible for the same activities are grouped together. Thus, instances with many eligibility patterns result in more groups and exhibit larger models, which explains the higher time consumption for setting up the LP for these instances. With increasing size of the RVL instances, primarily the running time for generating all sets and parameters increases (e.g., for deriving subsets of the activities that are associated with a specific constraint), whereas the rest of the steps require only a slightly longer running time. The preprocessing technique runs fast for all instances. Even for the largest instances, the matheuristic (with $k = 20$) always terminates in less than 10 minutes and is thus well within the running time budget prescribed by the company.

Next, we analyze the impact of the complexity parameters on the quality of the solutions obtained by the matheuristic. Therefore, we only consider the generated instances for which we systematically varied the complexity parameters. Generally, the average gaps increase with increasing eligibility fraction, as we can see from Table 13. While the eligibility fraction has an effect on the average gaps, the number of eligibility patterns has almost no effect. We can see from Table 11 that the gaps remain small even with increasing numbers of customers and activities in the instances. Overall, the matheuristic provides high-quality solutions in a shorter running time than the MBLP and the MBLP′.

Finally, we performed two experiments to assess the effect of two key components of the matheuristic on running time and solution quality. In the first experiment, we analyze the impact of step 5) of the preprocessing technique and in the second experiment, we analyze the impact of the new modeling technique used to consider the conflict constraints in the LP.

Table 14 summarizes the results of the first experiment. It shows the total number of constraints to ensure conflict rules (# conflict const. in LP) for groups of instances (∑) for the matheuristic and the matheuristic without conducting step 5) of the preprocessing technique. For the matheuristic without step 5), the increase in percent of the number of constraints to ensure conflict rules in the LP is stated in brackets (increase). Moreover, Table 14 states the total running time of both approaches (CPU). We can see that applying step 5) removes a substantial number of conflict constraints in the LP, and that setting up a smaller model leads to a lower total running time for all instance groups. Note that the solution quality is not affected by step 5) of the preprocessing technique.

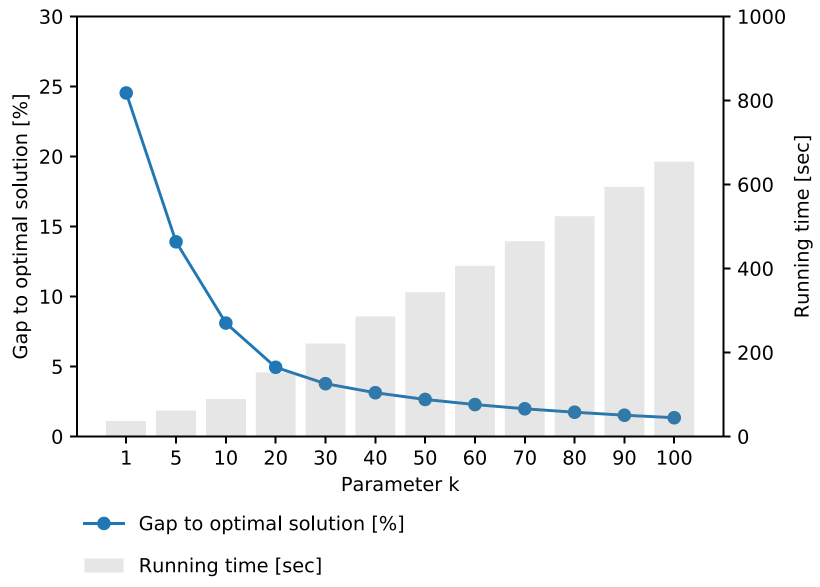

Fig. 9. Instance GM8: gap to optimal solution vs. running time of the matheuristic for different values of k.

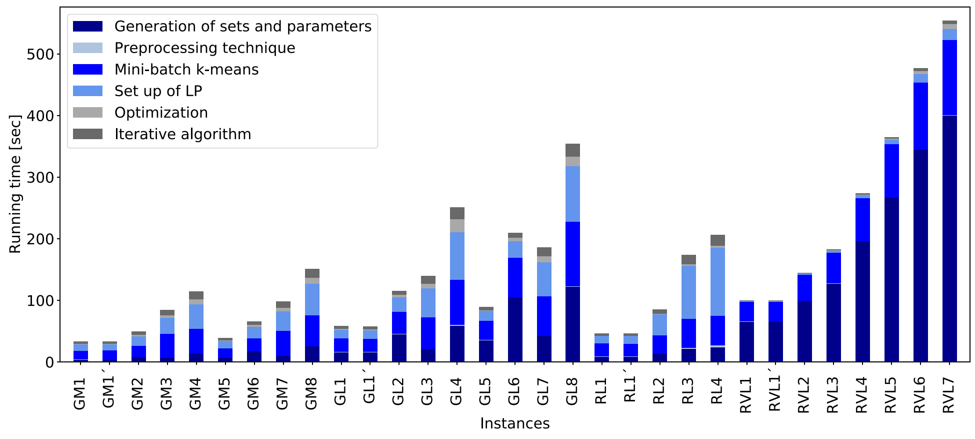

Fig. 10. Running times for the different matheuristic steps.

Table 14
Number of constraints to ensure the conflict rules in matheuristic.

	Matheuristic		Matheuristic without step 5)	
ID	#conflict const. in LP	CPU [s]	#conflict const. in LP (increase)	CPU [s]
$\sum$ GS	24,910	**67.1**	138,030 (+454.1%)	73.7
$\sum$ GM	307,346	**667.6**	3,285,126 (+968.9%)	837.9
$\sum$ GL	458,967	**1,460.0**	7,404,226 (+1,513.2%)	1,912.5
$\sum$ (RL+RVL)	389,873	**2,754.1**	1,570,423 (+302.8%)	2,905.1

In the second experiment, we compare the proposed matheuristic to a benchmark version of the matheuristic that does not use the new modeling technique to incorporate the conflict constraints in the linear program. Instead of using the new modeling technique, the benchmark version incorporates the conflict constraints in the linear program by formulating constraints (9) of the MBLP for groups of customers. The resulting linear program $\overline{\text{LP}}$ reads as follows:

$$(\overline{\text{LP}}) \begin{cases} \text{Max.} & (15) \\ \text{s.t.} & (11)-(13),(16)-(22),(24),(25) \\ & x_{gj_1}+x_{gj_2} \leq o_g \qquad (g \in G;\ (j_1,j_2) \in T:\ j_1,j_2 \in J_g) \end{cases} \qquad (27)$$

Table 15 summarizes the results of the second experiment. It reports the objective function value (OFV), the penalty, the number of assignments (#agmts) in the LP solution (or in the $\overline{\text{LP}}$ solution), the number of assignments that are conducted in the iterative algorithm (#agmts (it. alg.)), and the total running time (CPU). The results in Table 15 demonstrate the advantages of using the new modeling technique. The new modeling technique considers the customer-specific constraints very effectively already in the group-level model such that almost all assignments in the LP solution can be conducted by the iterative algorithm. In contrast, the $\overline{\text{LP}}$ solution has many assignments that cannot be conducted by the iterative algorithm because they would violate customer-specific con-

Table 15
Effectiveness of new modeling technique to incorporate conflict constraints in group-level model.

ID	Matheuristic					Matheuristic without new modeling technique				
	OFV [100k]	Penalty [100k]	#agmts (LP)	#agmts (it. alg.)	CPU [s]	OFV [100k]	Penalty [100k]	#agmts ($\overline{\text{LP}}$)	#agmts (it. alg.)	CPU [s]
GS1	**1.5**	0.0	8,331	8,331	5.7	1.0	0.4	8,600	7,946	**5.4**
GS1’	**1.2**	0.3	8,331	8,331	**4.3**	0.7	0.7	8,600	7,946	6.3
GS2	**1.4**	0.0	17,079	17,079	**5.4**	-0.1	1.3	20,020	15,722	7.7
GS3	**1.0**	0.0	9,826	9,826	**7.9**	0.9	0.1	9,943	9,263	9.3
GS4	**5.0**	0.0	17,594	17,594	**10.3**	3.2	1.0	19,193	15,786	13.2
GS5	**1.3**	0.0	16,698	16,698	**4.8**	1.3	0.0	18,295	16,014	5.5
GS6	**13.5**	0.0	40,975	40,975	**7.3**	1.2	10.2	51,067	34,559	10.6
GS7	**4.2**	0.0	20,897	20,897	**8.9**	3.1	0.7	23,304	19,530	10.6
GS8	**8.9**	0.0	38,903	38,902	**12.4**	-3.7	10.8	48,212	33,501	18.9
GM1	**35.5**	0.0	149,790	149,790	**32.8**	3.3	27.5	184,503	135,298	45.7
GM1’	**32.3**	1.2	143,902	143,902	**32.9**	-46.2	76.4	174,792	133,806	46.7
GM2	**84.8**	0.0	220,773	220,773	**49.8**	31.1	35.3	279,628	175,282	81.2
GM3	**24.9**	0.0	151,642	151,639	**84.3**	-18.0	38.1	177,748	130,791	118.8
GM4	**29.3**	0.0	215,672	215,672	**114.7**	-75.5	98.0	257,281	174,272	216.4
GM5	**46.8**	0.0	310,369	310,350	**39.0**	14.5	25.6	390,896	274,002	59.4
GM6	**141.3**	0.0	448,297	448,297	**65.6**	71.2	41.4	573,383	365,234	120.7
GM7	**70.5**	0.0	320,507	320,507	**97.8**	-22.9	81.0	401,804	279,353	155.2
GM8	**123.6**	0.0	461,223	461,223	**150.8**	49.8	42.0	564,119	345,302	308.6
GL1	**243.0**	0.0	834,008	834,008	**58.3**	28.3	186.5	1,030,637	760,945	83.1
GL1’	**-394.9**	616.7	838,189	838,189	**57.2**	-492.7	694.0	988,698	787,555	81.6
GL2	**256.1**	0.0	1,149,637	1,149,637	**115.1**	190.7	15.4	1,468,184	931,097	179.2
GL3	**194.0**	0.0	787,395	787,395	**139.3**	146.3	24.0	951,477	712,955	244.0
GL4	**211.8**	0.0	1,193,477	1,193,477	**251.2**	-326.5	499.1	1,446,367	936,972	499.2
GL5	**202.0**	0.0	1,637,333	1,637,333	**89.7**	124.7	49.5	2,086,706	1,461,946	125.0
GL6	**452.1**	0.0	2,418,241	2,418,241	**209.3**	312.0	67.9	2,901,633	1,956,749	306.9
GL7	**465.9**	0.0	1,611,220	1,611,220	**185.4**	-404.3	801.3	2,003,448	1,427,594	314.2
GL8	**1,022.2**	0.0	2,378,114	2,378,114	**354.6**	74.4	762.1	2,942,362	1,861,592	830.3
RL1	**109.2**	0.0	231,224	231,224	**46.5**	109.0	0.0	231,047	230,778	95.8
RL1’	**109.7**	1.8	240,704	240,704	**46.5**	109.0	2.3	240,494	240,008	95.3
RL2	**148.7**	0.0	369,377	369,377	**85.2**	147.0	0.0	368,244	365,422	232.9
RL3	**223.0**	0.0	534,595	534,595	**173.9**	213.5	1.6	545,604	518,135	614.8
RL4	**236.0**	0.0	596,515	596,515	**205.8**	217.3	4.1	594,679	565,202	814.5
RVL1	**291.7**	0.0	186,499	186,499	**100.2**	236.4	0.0	186,125	159,361	101.8
RVL1’	**246.8**	44.9	187,086	187,086	**99.8**	191.5	44.9	186,712	159,948	102.7
RVL2	**377.0**	0.0	257,332	257,309	**144.9**	288.6	0.0	279,060	195,974	148.6
RVL3	**534.7**	0.0	350,901	350,751	**182.3**	403.7	0.0	372,490	274,505	187.9
RVL4	**751.0**	0.0	510,249	509,998	**273.5**	554.1	0.0	554,051	380,332	282.8
RVL5	**975.9**	0.0	658,216	657,843	**365.2**	742.1	0.0	723,998	497,519	377.2
RVL6	**1,273.3**	0.0	823,361	822,628	**476.4**	959.8	6.9	886,967	612,202	506.0
RVL7	**1,501.8**	0.0	941,824	941,284	**554.0**	1,119.8	0.0	1,004,740	676,107	592.6

flict constraints. When a large fraction of assignments cannot be conducted by the iterative algorithm, some bounds of minimum assignment and sales constraints that were satisfied in the group-level model are violated after applying the iterative algorithm. This explains the large penalties and hence lower objective function values of the solutions of the matheuristic which uses model $\overline{\text{LP}}$. The new modeling technique not only improves the solution quality but also reduces the running time because the group-level model has fewer constraints and a smaller number of assignments needs to be processed by the iterative algorithm.

8. Conclusion

In this study, we introduced a real-world planning problem of a telecommunications company. The planning problem consists of assigning existing customers to direct marketing activities subject to various business constraints, such as budget and sales constraints, and various customer-specific constraints. The customer-specific constraints ensure, for example, that individual customers are generally not assigned to the activities too often and that the customers are not assigned to activities that are subject to a conflict. Such a conflict may exist, for example, between two activities that are scheduled within the same week. Existing approaches that deal with customer assignment in direct marketing do not consider such customer-specific constraints and are thus not applicable to the planning problem at hand. We developed a matheuristic that first solves an optimization problem for groups of customers and then iteratively assigns individual customers to the activities based on the solution to the group-level problem. New modeling techniques and a preprocessing technique are introduced to consider customer-specific constraints already in the group-level model. In a computational analysis, we demonstrated the effectiveness of the preprocessing technique and the problem decomposition strategy of the matheuristic based on a test set that includes generated and real-world instances. The proposed preprocessing technique is able to reduce the number of constraints in the models by up to 95%. Even when the number of groups is relatively small, the average gap of the solutions derived by the matheuristic to the optimal solutions of the generated instances is only 1.8%. Increasing the number of groups further reduces the gap while prolonging the running time only slightly. The matheuristic is currently in use at the company and has lead to an overall improvement of its key performance indicators. The company estimates based on a proof of benefit conducted on a selected campaign that the use of the matheuristic increased the number of sales by 90%, which improved the profitability of this campaign by around 300%.

In future research, we will extend the planning problem to include multi-stage campaigns. In such campaigns, customers can only be assigned to activities of a non-initial stage when they have been assigned to an activity of each previous stage. A promising direction for future research is the development of strategies for grouping customers with different eligibility patterns, as pointed out in Section 5.1. Moreover, it would be interesting to adapt the preprocessing technique and the decomposition strategy to related planning problems such as the bin packing problem with conflict constraints. Finally, another direction for future research is to incorporate the conflict constraints with branching rules as done in the exact solution approaches of Şuvak et al. (2020) for the maximum flow problem with conflict constraints and of Şuvak, Altınel, & Aras (2021) for the minimum cost flow problem with conflict constraints.

Acknowledgement

We would like to thank the company for their support and for the excellent collaboration.

References

Bettinelli, A., Cacchiani, V., & Malaguti, E. (2017). A branch-and-bound algorithm for the knapsack problem with conflict graph. *INFORMS Journal on Computing, 29*, 457–473.

Bhaskar, T., Sundararajan, R., & Krishnan, P. G. (2009). A fuzzy mathematical programming approach for cross-sell optimization in retail banking. *Journal of the Operational Research Society, 60*, 717–727.

Bigler, T., Baumann, P., & Kammermann, M. (2019). Optimizing customer assignments to direct marketing activities: A binary linear programming formulation. In M. Wang, J. Li, F. Tsung, & A. Yeung (Eds.), *Proceedings of the 2019 IEEE international conference on industrial engineering and engineering management: Macau* (pp. 601–605).

Bron, C., & Kerbosch, J. (1973). Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM, 16*, 575–577.

Cetin, F., & Alabas-Uslu, C. (2017). Heuristic solution to the product targeting problem based on mathematical programming. *International Journal of Production Research, 55*, 3–17.

Coelho, V. N., Oliveira, T. A., Coelho, I. M., Coelho, B. N., Fleming, P. J., Guimarães, F. G., … Lust, T. (2017). Generic Pareto local search metaheuristic for optimization of targeted offers in a bi-objective direct marketing campaign. *Computers & Operations Research, 78*, 578–587.

Cohen, M.-D. (2004). Exploiting response models — optimizing cross-sell and up-sell opportunities in banking. *Information Systems, 29*, 327–341.

Coniglio, S., Furini, F., & San Segundo, P. (2021). A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts. *European Journal of Operational Research, 289*, 435–455.

Darmann, A., Pferschy, U., Schauer, J., & Woeginger, G. J. (2011). Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics, 159*, 1726–1735.

Delanote, S., Leus, R., & Nobibon, F. T. (2013). Optimization of the annual planning of targeted offers in direct marketing. *Journal of the Operational Research Society, 64*, 1770–1779.

Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming, 91*, 201–213.

Elhedhli, S., Li, L., Gzara, M., & Naoum-Sawaya, J. (2011). A branch-and-price algorithm for the bin packing problem with conflicts. *INFORMS Journal on Computing, 23*, 404–415.

Fisher, M. L. (1981). The lagrangian relaxation method for solving integer programming problems. *Management Science, 27*, 1–18.

Fisher, M. L. (1985). An applications oriented guide to lagrangian relaxation. *Interfaces, 15*, 10–21.

Garey, M. R., & Johnson, D. S. (2002). *Computers and intractability*: vol. 29. New York: WH Freeman.

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th python in science conference (SciPy2008)* (pp. 11–15). Pasadena.

Lessmann, S., Haupt, J., Coussement, K., & De Bock, K. W. (2021). Targeting customers for profit: An ensemble learning framework to support marketing decision-making. *Information Sciences, 557*, 286–301.

Ma, S., & Fildes, R. (2017). A retail store SKU promotions optimization model for category multi-period profit maximization. *European Journal of Operational Research, 260*, 680–692.

Ma, S., Hou, L., Yao, W., & Lee, B. (2016). A nonhomogeneous hidden Markov model of response dynamics and mailing optimization in direct marketing. *European Journal of Operational Research, 253*, 514–523.

Miguéis, V. L., Camanho, A. S., & Borges, J. (2017). Predicting direct marketing response in banking: Comparison of class imbalance methods. *Service Business, 11*, 831–849.

Nair, S. K., & Tarasewich, P. (2003). A model and solution method for multi-period sales promotion design. *European Journal of Operational Research, 150*, 672–687.

Nobibon, F. T., Leus, R., & Spieksma, F. C. (2011). Optimization models for targeted offers in direct marketing: Exact and heuristic algorithms. *European Journal of Operational Research, 210*, 670–683.

Oliveira, T. A., Coelho, V. N., Souza, M. J., Boava, D. L. T., Boava, F., Coelho, I. M., & Coelho, B. N. (2015). A hybrid variable neighborhood search algorithm for targeted offers in direct marketing. *Electronic Notes in Discrete Mathematics, 47*, 205–212.

Öncan, T., Şuvak, Z., Akyüz, M. H., & Altınel, İ. K. (2019). Assignment problem with conflicts. *Computers & Operations Research, 111*, 214–229.

Sadykov, R., & Vanderbeck, F. (2013). Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing, 25*, 244–255.

Sculley, D. (2010). Web-scale k-means clustering. In M. Rappa, P. Jones, J. Freire, & S. Chakrabarti (Eds.), *Proceedings of the 19th international conference on world wide web: Raleigh* (pp. 1177–1178).

Sun, M. (2002). The transportation problem with exclusionary side constraints and two branch-and-bound algorithms. *European Journal of Operational Research, 140*, 629–647.

Sundararajan, R., Bhaskar, T., Sarkar, A., Dasaratha, S., Bal, D., Marasanapalle, J. K., … Bak, K. (2011). Marketing optimization in retail banking. *Interfaces, 41*, 485–505.

Şuvak, Z., Altınel, İ. K., & Aras, N. (2020). Exact solution algorithms for the maximum flow problem with additional conflict constraints. *European Journal of Operational Research, 287*, 410–437.

Şuvak, Z., Altınel, İ. K., & Aras, N. (2021). Minimum cost flow problem with conflicts. *Networks, 78*, 421–442.

Walteros, J. L., & Buchanan, A. (2020). Why is maximum clique often easy in practice? *Operations Research, 68*, 1866–1895.