Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A geometric branch-and-bound algorithm for the service bundle design problem

Yifu Li[a,c,*], Xiangtong Qi[b]

[a] International Institute of Finance, School of Management, University of Science and Technology of China, Hefei, Anhui, 230026, PR China
[b] Department of Industrial Engineering and Decision Analytics, The Hong Kong University of Science and Technology, Kowloon, Hong Kong SAR
[c] International Business School of Suzhou, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, 215123, PR China

ARTICLE INFO

Article history:
Received 6 November 2020
Accepted 23 March 2022
Available online 26 March 2022

Keywords:
Scheduling
Service design
Geometric branch-and-bound algorithm
0–1 Sum-of-ratios problem

ABSTRACT

In the service industry, a service provider may sell a collection of service activities as a package, also known as a service bundle. Empirical studies indicate that the customer's ex-post perception of a service bundle depends on not only the utility of each activity, but also the sequence of the activities being delivered. The latter can be measured by certain sequence effects, such as the utility of the peak activity, the utility of the end activity, and the trend of utility change over the activities. This phenomenon gives a service provider an opportunity to optimize a service bundle by manipulating the activities and their sequence. Such a service bundle design problem can be formulated as a 0–1 sum-of-ratios problem. To solve the problem, we design a novel geometric branch-and-bound algorithm. The algorithm divides the objective function into several dimensions, and repeatedly strengthens the bounds of each dimension. This enables us to convert the 0–1 sum-of-ratios problem into a series of 0–1 quadratic optimization problems. Computational experiments show that the algorithm can solve the service bundle design problem efficiently.

1. Introduction

The service industry is a vital part of the world economy. This quickly developing industry creates various service commodities, from basic services, such as a meal in a cafeteria, to ultra-complex services, such as television shows engaging professionals from many different domains. Different from in the past, service providers are switching their attention from providing purely functional service encounters to designing services with an improved customer experience. As stated by Popat (2018, June 7), “Creating a great customer experience is worth the investment” and “the full customer journey has to be considered if we are seeking truly powerful results.” Thus, improving the customer experience has become a crucial factor in service design. We therefore seek to address this issue in designing bundled services.

A service bundle is a collection of different service encounters sold together as a value-added service package (Dixon & Verma, 2013). Among the vast category of service products, bundling services is now one of the most common modes of service in daily life. For example, art theatres offer bundled tickets that include different shows, sports venues sell seasonal subscriptions with tickets and activities, and travel agents provide trip plans that include several attractions within a destination. Thus, the service bundle design problem (SBDP) is a practical issue faced by service providers.

The SBDP is an optimization problem which has to take multiple types of customers into account. Faced with heterogeneous customers in practice, the service designer has to design several different service bundles at the same time to satisfy heterogeneous preferences. The service designer thus cares about the overall satisfaction of multiple bundles rather than a single bundle. While there are many studies on the single-service-bundle design problem (e.g., Chase & Dasu, 2001; Kahneman, Fredrickson, Schreiber, & Redelmeier, 1993), it is only recently that researchers have investigated the design of multiple service bundles.

Based on Dixon & Verma (2013), Dixon & Thompson (2016) formulate the SBDP with a linear-sum model of the sequence effects (i.e., peak effect, end effect, spread effect, and trend effect). The SBDP is a 0–1 sum-of-quadratic-ratios problem (0–1 SQRP). According to Prokopyev, Meneses, Oliveira, & Pardalos (2005) and Schaible & Shi (2003), the problem is NP-Hard. Dixon & Thompson (2016) solve the SBDP with a Simulated Annealing (SA) algorithm. In our study, we follow their linear model on the remembered utility, yet aim to compute the optimal assignment with an exact algorithm. The difficulty with solving the sum-of-ratios problem is in computing the bounds of the objective function that is strongly

* Corresponding author.
E-mail addresses: ylick@connect.ust.hk (Y. Li), ieemqi@ust.hk (X. Qi).

https://doi.org/10.1016/j.ejor.2022.03.044

nonlinear. Since there is no proper bounding method so far for the sum-of-ratio objective (Ursulenko, Butenko, & Prokopyev, 2013), it is thus virtually impossible to derive an efficient classical branch-and-bound (B&B) algorithm for the SBDP.

Therefore, we develop a geometric-branch-and-bound (GB&B) algorithm for the SBDP. Rather than dealing with the objective function directly, we divide the objective into several "dimensions", and introduce the concept of a "box" to divide the SBDP into several subproblems. The upper bound of a box is the sum of the upper bound of each dimension in the box, and the upper bound of the objective function is the maximum upper bound of the list of boxes. In each iteration, the algorithm starts by selecting a box and a dimension to search. The GB&B algorithm computes the bounds of the selected dimension by solving the "subproblem". If a solution with higher satisfaction is found, the global lower bound is updated. Then the algorithm starts the branching process by dividing the current box into two smaller sub-boxes. Finally, boxes that cannot possibly contain the optimal solution will be deleted during the pruning procedure.

We offer the first algorithm to compute an exact solution for the SBDP. Our algorithm differs from past studies in the following aspects. First, the GB&B algorithm decomposes the objective into several dimensions, and it branches on the value of each dimension. This is different from the classical GB&B algorithm, which branches on the entire objective function. This method decomposes the sum-of-ratios problem so that we only need to deal with one ratio at a time. Second, for the bounding process, we introduce a parametric approach to convert the single ratio optimization problem into a 0–1 quadratic optimization problem. Together with the branching process, our algorithm decomposes the 0–1 sum-of-ratios problem into several less complicated 0–1 polynomial optimization subproblems, which reduces the computational burden. Finally, we discuss several methods of speeding up the algorithm. We analyze the performance of the GB&B algorithm with numerical studies, and our algorithm can also be modified into an adequate heuristic when the size of the problem does not allow us to solve it optimally.

The rest of the paper is organized as follows: In Section 2, we review the relevant literature and summarize our contributions. In Section 3, we describe the key concepts and formulation of the SBDP. We propose the GB&B algorithm in Section 4 and discuss the methods of improving the algorithm in Section 5. We report on the numerical studies of the algorithm in Section 6 and conclude our paper in Section 7.

2. Literature review

Our paper contributes to the literature in three different aspects: Service design, the fractional optimization problem, and the GB&B algorithm literature.

2.1. Service bundle design

Service design is a broad concept, with rich studies in this area. We focus on the SBDP, whereby a service designer selects and sequences the various activities to create multiple bundles with an enhanced customer experience. We refer readers to Karmarkar (2015) for a comprehensive review of the relevant literature.

One stream of the service design literature investigates the ideal schedule of a service bundle by studying the characteristics of the "experience profile". Kahneman et al. (1993) propose that customers have a preference for service encounters with high peak and end effects. Researchers also consider the trend effect (e.g., Loewenstein & Sicherman, 1991), and the spread effect (e.g., Thaler & Johnson, 1990). Dixon & Verma (2013) show with their empirical study that the retrospective perception of a service bundle is influenced by the four sequence effects.

Based on this empirical study, Dixon & Thompson (2016) formulate an optimization problem to maximize the customer experience over several service bundles, in which the customer's remembered utility is modeled as the weighted sum of the four sequence effects. They solve the problem using an SA algorithm and further discuss the effect of flexibility on the optimal bundle design. We adopt the basic concepts in Dixon & Thompson (2016) and the weighted-sum formed objective function. In contrast to their research, however, we focus on developing an exact algorithm for the SBDP, which may facilitate further investigations in this area.

Other relevant research includes Dixon & Thompson (2019) on the service package design problem with bundling flexibility, Page, Pérez, Telha, García-Echalar, & López-Ospina (2021) on the bundle composition design problem under competition, Li, Dai, & Qi (2021a) on service package design with non-homogeneous memory decay, Li, Ryan, & Sheng (2021b) on the game design problem, and Hanukov (2022) on service system design problem. Our work advances the service design literature by proposing an exact algorithm, which can be applied in solving practical service design problems.

2.2. Fractional optimization problem

The SBDP is a 0–1 SQRP, which is NP-hard (Prokopyev et al., 2005; Schaible & Shi, 2003). It belongs to the domain of 0–1 fractional optimization problems. We refer readers to Stancu-Minasian (2019) for a summary of the fractional programming literature, and Conforti, Cornuéjols, Zambelli et al. (2014) for a review of integer programming studies.

There are two common methods of converting the fractional problem into a polynomial problem. The first method is the linearization technique, which converts the fractional problem into a mixed integer linear problem (MILP) by introducing additional variables and constraints. Research in this area includes, for example, Wu (1997), Chang (2001), and Borrero, Gillen, & Prokopyev (2016) on the 0–1 fractional optimization problem, and Xia & Yuan (2006) on the 0–1 quadratic programming problem.

The other method is the parametric approach (e.g., Ibaraki, 1983). This method converts the fractional problem into a polynomial optimization problem by introducing a parameter. Recent advances include the work of Baldacci, Lim, Traversi, & Wolfler Calvo (2020) on the fractional vehicle routing problem, and Park & Lim (2021) on the road maintenance optimization problem. Both studies apply the parametric approach to solving the single ratio optimization problem. The parametric approach is an efficient method to solve the fractional problem; however, Falk & Palocsay (1992) use a counter example to show that the parametric approach cannot be directly used to solve the sum-of-ratios problem, which is proposed by Almogy & Levin (1971). Hence, in our study, we only apply the parametric approach to resolving the 0–1 single ratio subproblems.

For the problem that cannot be resolved by the above methods, researchers have developed various algorithms to resolve it. For example, Jiao & Liu (2015) use the B&B algorithm with linear reformulation to solve the sum of linear ratios problem; and Shen, Zhu, & Chen (2019) introduce a contraction and iteration algorithm to solve the polynomial ratio optimization problem. Recently, Dias, Brito, & Amaral (2021) has extended the application of fractional programming to data science in addressing the classification of the data.

2.3. The GB&B algorithm

In our paper, we develop a GB&B algorithm to solve the SBDP. The GB&B algorithm is a modification of the classical B&B algorithm with similar idea–the algorithm recursively divides the feasible set and computes the upper and lower bounds of the objective value. If the upper and lower bounds meet, then the optimal solution is found. The feasible set has to be further divided if the optimal solution has not so far been found. The GB&B algorithm is developed as an implementation of the B&B idea for continuous optimization problems (Pfeuffer, Stiglmayr, & Klamroth, 2012).

Scholz (2011) provides a thorough review of the GB&B algorithm, and readers are referred to Schöbel & Scholz (2014) for the latest advances. Whilst the GB&B algorithm has been developed for decades, it has only recently been applied in solving combinatorial problems (Schöbel & Scholz, 2014). The GB&B algorithm is found to be especially efficient in solving problems with special structures, such as a mixed-integer problem with box constraints (e.g., Schöbel & Scholz, 2014), multi-objective problems in location theory (e.g., Fernández & Tóth, 2009), and the sum-of-ratios minimum spanning tree problem (e.g., Skiscim & Palocsay, 2001; Ursulenko et al., 2013).

We contribute to the GB&B literature by extending it to the 0–1 sum-of-ratios problem. For an SBDP with K bundles, rather than computing the bounds of the objective, we reformulate the SBDP into a problem with $K+1$ dimensions and bound each dimension by solving the subproblems. We apply both a linearization technique and a parametric approach to solving the subproblems. In addition, we discuss how to speed up the algorithm, which may shed light on future work in this area.

3. Model and solution approach

In this section, we discuss both the model and the solution approach of the SBDP. We describe the problem in Section 3.1, and discuss the solution approach in Section 3.2.

3.1. Problem description

We consider an SBDP in which a set of activities $\mathcal{N}$ has been scheduled into different dates, and the service designer needs to assign the activities to K bundles. Let $\mathcal{N}=\{1,\dots,N\}$, and $\mathcal{K}=\{1,\dots,K\}$ be the sets of activities and bundles respectively. Each activity $i\in\mathcal{N}$ has a given unidimensional utility u_i and a scheduled date d_i. The service provider needs to design a set of bundles, in which each bundle has multiple activities.

As pointed out by Dixon & Verma (2013) the bundles are usually of different themes. The service provider has to take the theme into consideration when assigning the activities to the bundles. We assume each bundle k has a unique theme, and it contains m_k activities. Because of the content of the themes, some activities may be simultaneously assigned to several bundles at the same time, whereas others can just be assigned to a specific bundle. We assume an activity can be assigned to at most b bundles ($b<K$), and there is a feasibility constraint on the assignment between event i and bundle k as given by the parameter $c_{i,k}$. If it is feasible, we have $c_{i,k}=1$, otherwise we have $c_{i,k}=0$.

We take the linear model introduced by Dixon & Thompson (2016) on the objective function of the SBDP. The objective of the service designer is to maximize the remembered utility, which is modeled as the sum of sequence effects:

$$S=\max_{\forall k}\sum_{k\in\mathcal{K}}(w_1\text{PeakEffect}_k+w_2\text{EndEffect}_k+w_3\text{SpreadEffect}_k+w_4\text{TrendEffect}_k),\tag{1}$$

where PeakEffect_k, EndEffect_k, SpreadEffect_k, and TrendEffect_k are the sequence effects of bundle k defined in Appendix A. Given that peak, end, and spread effects can be represented by linear functions of utilities and dates, we call these effects *linear effects* in the rest of the paper.

Let $x_{i,k}$ be a binary variable, denoting the assignment decision between activity i and bundle k. If $x_{i,k}=1$, then the activity i is assigned to bundle k; otherwise, activity i is not assigned to bundle k. We introduce the binary variable $p_{i,k}$ to denote the peak activity in bundle k. If $p_{i,k}=1$, then activity i is the peak activity in bundle k; otherwise, it is not the peak activity in the bundle k. Let the binary variable $q_{i,k}$ denote the end activity in bundle k. If $q_{i,k}=1$, then the ith date is the last date assigned to bundle k; otherwise, it is not the last date assigned.

By (1) and the definition of the sequence effects in Appendix A, the SBDP is given by:

$$\max\ S(\boldsymbol{x},\boldsymbol{p},\boldsymbol{q})=\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{N}}\left(w_1u_ip_{i,k}+w_2u_iq_{i,k}+w_3d_iq_{i,k}-w_3d_ip_{i,k}\right)+w_4\sum_{k\in\mathcal{K}}\frac{\sum_{i\in\mathcal{N}}d_ix_{i,k}\left(m_ku_i-\sum_{j\in\mathcal{N}}u_jx_{j,k}\right)}{\sum_{i\in\mathcal{N}}u_ix_{i,k}\left(m_ku_i-\sum_{j\in\mathcal{N}}u_jx_{j,k}\right)},\tag{2}$$

$$s.t.\ \sum_{j\in\mathcal{N}}u_jp_{j,k}\ge u_ix_{i,k},\quad\forall i\in\mathcal{N}\text{ and }\forall k\in\mathcal{K},\tag{3}$$

$$\sum_{j\in\mathcal{N}}d_jq_{j,k}\ge d_ix_{i,k},\quad\forall i\in\mathcal{N}\text{ and }\forall k\in\mathcal{K},\tag{4}$$

$$x_{i,k}\ge p_{i,k},\quad\forall i\in\mathcal{N}\text{ and }\forall k\in\mathcal{K},\tag{5}$$

$$x_{i,k}\ge q_{i,k},\quad\forall i\in\mathcal{N}\text{ and }\forall k\in\mathcal{K},\tag{6}$$

$$\sum_{i\in\mathcal{N}}p_{i,k}=1,\quad\forall k\in\mathcal{K},\tag{7}$$

$$\sum_{i\in\mathcal{N}}q_{i,k}=1,\quad\forall k\in\mathcal{K},\tag{8}$$

$$\sum_{i\in\mathcal{N}}x_{i,k}=m_k,\quad\forall k\in\mathcal{K},\tag{9}$$

$$\sum_{k\in\mathcal{K}}x_{i,k}\le b,\quad\forall i\in\mathcal{N},\tag{10}$$

$$x_{i,k}\le c_{i,k},\quad\forall i\in\mathcal{N}\text{ and }\forall k\in\mathcal{K},\tag{11}$$

$$\boldsymbol{x},\boldsymbol{p},\boldsymbol{q}\in\{0,1\},\tag{12}$$

where $S(\boldsymbol{x},\boldsymbol{p},\boldsymbol{q})$ is the overall remembered utility of the K service bundles.

To facilitate our analysis, we replace $\left(\boldsymbol{x}^{\mathrm{T}},\boldsymbol{p}^{\mathrm{T}},\boldsymbol{q}^{\mathrm{T}}\right)^{\mathrm{T}}$ with variable $\hat{\boldsymbol{x}}$ and define the following functions to simplify the notations:

$$f_k(\hat{\boldsymbol{x}})=\begin{cases}\sum_{j\in\mathcal{K}}\sum_{i\in\mathcal{N}}\left(w_1u_ip_{i,j}+w_2u_iq_{i,j}+w_3d_iq_{i,j}-w_3d_ip_{i,j}\right) & \text{if } k=0,\\ w_4\left(\sum_{i\in\mathcal{N}}d_ix_{i,k}\left(m_ku_i-\sum_{j\in\mathcal{N}}u_jx_{j,k}\right)\right) & \text{if } k\in\mathcal{K},\end{cases}\tag{13}$$

$$g_k(\hat{\boldsymbol{x}})=\begin{cases}1 & \text{if } k=0,\\ \sum_{i\in\mathcal{N}}u_ix_{i,k}\left(m_ku_i-\sum_{j\in\mathcal{N}}u_jx_{j,k}\right) & \text{if } k\in\mathcal{K}.\end{cases}\tag{14}$$

Define $\mathcal{K}^+=\{0,1,\dots,K\}$, then the SBDP can be equivalently expressed as:

$$\max\ S(\hat{\boldsymbol{x}})=\sum_{k\in\mathcal{K}^+}\frac{f_k(\hat{\boldsymbol{x}})}{g_k(\hat{\boldsymbol{x}})},\tag{15}$$

$$s.t.\ \hat{\boldsymbol{x}}\in\mathcal{T},\tag{16}$$

where $\mathcal{T}$ is the feasible region defined by constraints (3) to (12).

In the main context, we restrict the SBDP to be the problem in which activities and dates are already scheduled with each other. We also consider the case that activities and dates are not assigned with each other, and formulate the SBDP with activity-date-bundle assignment decision in Appendix D.

3.2. An overview of the algorithms

The SBDP is a 0–1 SQRP, which is a 0–1 fractional optimization problem. In this paper, we consider four methods of solving the problem: The MIQCP solver with alternative formulation, the linearization technique, the GB&B algorithm, and the GB&B-based heuristic algorithm. The key concepts of are summarized as follows:

- MIQCP solver with alternative formulation
  The SBDP can be formulated as an MIQCP with additional variables and constraints as shown in Appendix B.1. The alternative formulation requires $O(K)$ additional variables and constraints.
- The linearization technique
  This method linearizes the SBDP into a MILP with additional variables and constraints. We introduce two techniques (ia) and (ib) in Appendix B.2.2. By the numerical study in (i) of Appendix B.2.3, technique (ia) is the most efficient one, which requires $O(NK)$ additional variables and constraints.
- The GB&B algorithm
  The GB&B algorithm is a modification of the classical B&B algorithm. The algorithm divides the SBDP into several dimensions and strengthens the bounds of each dimension by solving a series of 0–1 quadratic optimization subproblems. The framework of the algorithm is presented in Algorithm 2 in Section 4.5.
- The GB&B-based heuristic algorithm
  As shown by the numerical study in Section 6.3.2 and Section 6.4.1, the efficiency of the GB&B algorithm can be enhanced if the weight of sequence effects and target gap are set as certain values. Based on this property, we modify the GB&B algorithm as a heuristic algorithm presented in Algorithm 3 in Appendix C.2.

As seen by the numerical study in Section 6.3, our GB&B algorithm outperforms both the MIQCP solver and the linearization technique. Thus, we focus on the GB&B algorithm in the rest of the paper and describe it in detail in Section 4.

4. The GB&B algorithm

The GB&B algorithm is based on two main operations: *branching* and *bounding*. In the branching operation, the algorithm decomposes the SBDP into several subproblems with reduced feasible space. In the bounding process, the algorithm conducts bounding procedures by solving subproblems on the selected searching dimension.

In this section, we first introduce some preliminary concepts in Section 4.1, and then describe the general idea of the algorithm in Section 4.2. We present the detailed branching and bounding procedures in Section 4.3 and 4.4, and present the framework of the GB&B algorithm in Section 4.5.

4.1. Preliminaries

In the GB&B algorithm, we refer to each fractional term in the objective (15) as a *dimension*. Then there are K fractional dimensions and one linear dimension (i.e., when $k = 0$) in problem (15)-(16).

Given that there are multiple fractional dimensions in the objective, it is difficult to estimate the bounds directly. We then introduce a new bounding method by computing a number of subproblems. To achieve this, we will first introduce the concept of a “box”.

Let V be a *box* or *rectangular region*, given by:

$$V = \left\{ \boldsymbol{v} \in \mathbb{R}^{K+1} : L_k(V) \le v_k \le U_k(V), \forall k \in \mathcal{K}^+ \right\}, \tag{17}$$

where $L_k(V)$ and $U_k(V)$ are the lower bound and upper bound of dimension k over box V respectively.

Let $\gamma(V)$ be the upper bound of the box V, which is given by:

$$\gamma(V) = \sum_{k \in \mathcal{K}^+} U_k(V). \tag{18}$$

By the definition of the trend effect (A4) in Appendix A, $g_k(\hat{\boldsymbol{x}}) > 0$ for any $\hat{\boldsymbol{x}}$. If we know that the optimal solution is located in a box V, then the optimal solution of the following problem must be the optimal solution of the original problem (15)-(16):

$$\max\ S_V(\hat{\boldsymbol{x}}) = \sum_{k \in \mathcal{K}^+} \frac{f_k(\hat{\boldsymbol{x}})}{g_k(\hat{\boldsymbol{x}})}, \tag{19}$$

$$\begin{aligned} s.t.\ & f_k(\hat{\boldsymbol{x}}) \ge L_k(V) g_k(\hat{\boldsymbol{x}}), \forall k \in \mathcal{K}^+, \\ & f_k(\hat{\boldsymbol{x}}) \le U_k(V) g_k(\hat{\boldsymbol{x}}), \forall k \in \mathcal{K}^+, \\ & \hat{\boldsymbol{x}} \in \mathcal{T}. \end{aligned} \tag{20}$$

We then introduce the subproblems on the dimension k over box V. We refer to the maximization problem as the SDOP-k:

$$\max\ \ S_{k,V}(\hat{\boldsymbol{x}}) = \frac{f_k(\hat{\boldsymbol{x}})}{g_k(\hat{\boldsymbol{x}})}, \tag{21}$$

$$\begin{aligned} s.t.\ \ & f_i(\hat{\boldsymbol{x}}) \ge L_i(V) g_i(\hat{\boldsymbol{x}}), \forall i \in \mathcal{K}^+ \\ & f_i(\hat{\boldsymbol{x}}) \le U_i(V) g_i(\hat{\boldsymbol{x}}), \forall i \in \mathcal{K}^+, \\ & \hat{\boldsymbol{x}} \in \mathcal{T}. \end{aligned} \tag{22}$$

The SDOP-0 is a 0–1 quadratic constrained problem (0–1 QCP), and the SDOP-k ($k > 0$) is a 0–1 quadratic constrained quadratic ratio problem (0–1 QCQRP). In our study, the GB&B algorithm computes the bounds by solving the subproblems. For a given dimension k over box V, we can compute its upper bound by solving the SDOP-k. The upper bound of the objective is the maximum of $\gamma(V)$ for all the feasible boxes V.

4.2. General idea of the algorithm

Our GB&B algorithm differs from the B&B algorithm in both branching and bounding procedures. We describe the general idea of our algorithm using an example with two dimensions, as shown in Fig. 1.

The scheme of the algorithm is as follows:

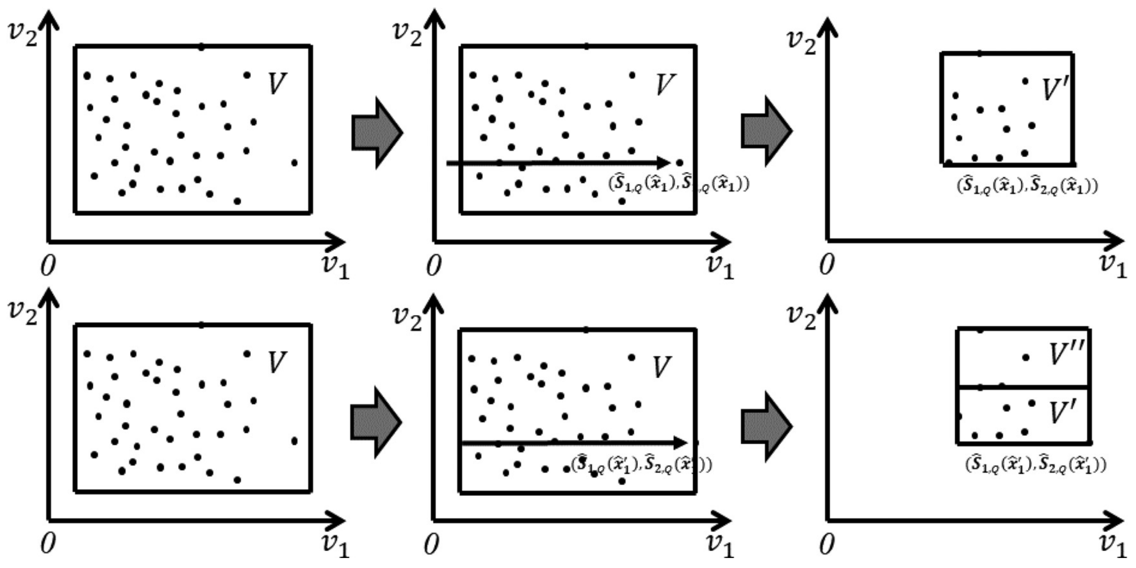
(i) Initialization

The GB&B algorithm starts with initialization computation of the feasible space. It computes the bounds of each dimension, and formulates a box V with these bounds. Let $\boldsymbol{L}(V)$ and $\boldsymbol{U}(V)$ be the bounds of box V, and $\underline{S}$ and $\bar{S}$ be the bounds of the objective function. Then the optimal solution must be located in box V, which is saved in list $\mathcal{V}$.

(ii) Dimension selection and box selection

In this step, the algorithm selects the dimension and box in which to conduct the search. As illustrated in Fig. 1, dimension v_1 and box V are selected in this iteration.

(iii) Branching



v_2
v_1
0
V
V'
V''
$(\hat{S}_{1,Q}(\hat{x}_1), \hat{S}_{2,Q}(\hat{x}_1))$
$(\hat{S}_{1,Q}(\hat{x}'_1), \hat{S}_{2,Q}(\hat{x}'_1))$

Fig. 1. General Idea of the GB&B Algorithm.

The algorithm takes the branching procedure by dividing the box into one or two smaller sub-boxes. After the bounding procedure in the iteration, if a solution is found in the interior, the GB&B will replace V with a smaller box V', as illustrated in Fig. 1. Otherwise, the GB&B will divide box V into two smaller sub-boxes V' and V''.

(iv) Bounding

Given a box V and a dimension k, our algorithm updates the bounds by computing the subproblems (i.e., SDOP-k). Then it turns to another box V' and dimension k' for the bounding procedure. The upper bound of the objective is the maximum upper bound over all the boxes in the list $\mathcal{V}$. The lower bound of the objective is the remembered utility of the incumbent solution.

(v) Pruning

In this step, boxes that cannot provide an optimal solution will be deleted from the list $\mathcal{V}$.

4.3. Branching

There are different methods of splitting the box. As found by Ursulenko et al. (2013), it is more efficient to divide the box into two sub-boxes when there are multiple dimensions (i.e., $K > 1$). We consider two methods of selecting the dimension to do the branching: rotation-dimension search (RDS) and fixed-dimension search (FDS).

(i) RDS

Let $K^+ = K + 1$. In this method, the algorithm searches along each of the K^+ dimensions with one dimension at each iteration. With a given dimension k and box V, if the solution of the SDOP-k is found at the interior (i.e., $\hat{S}_{k,V}(\hat{\boldsymbol{x}}^*) < U_k(V)$), then the algorithm updates the upper bound of this box by setting $U_k(V) = \hat{S}_{k,V}(\hat{\boldsymbol{x}}^*)$. Otherwise, we adopt the splitting method suggested by Ursulenko et al. (2013). This method splits box V into two smaller sub-boxes in the direction, for which the distance between the incumbent solution and the upper bound on the dimension is the longest (i.e., dimension j such that $j = \arg\max\{U_i(V) - \hat{S}_{i,V}(\hat{\boldsymbol{x}}^*) | i \in \mathcal{K}$ and $i \neq k\}$). The algorithm continues selecting another dimension k' and box V' to search.

(ii) FDS

In this method, the algorithm always searches along a fixed dimension k at each iteration. The algorithm splits box V into two sub-boxes once a feasible solution is found. The algorithm continues by selecting another box V' along dimension k.

The fixed-dimension search is a special case of the RDS. The advantage is that it may perform much more efficiently if the bounds of a dimension can be solved more efficiently than others. One possible weakness is that it may take more iterations for the algorithm to converge, as it can only update the bounds on one fixed dimension. To take full advantage of the FDS method, we set the algorithm searches along the linear dimension (i.e., $k = 0$), as the bounds of the linear dimension can be computed more efficiently, because the SDOP-0 is less complex than the SDOP-k, when $k > 0$.

In our study, we consider two types of FDS methods: K^+-dimension-fixed-dimension search (KDFDS), and 2-dimension-fixed-dimension search (2DFDS).

(iia) KDFDS

In this configuration, there is one linear dimension and K fractional dimensions as defined in Section 4.1. The algorithm searches along the linear dimension by solving a series of linear subproblems (i.e., the SDOP-0), and it follows the same box splitting method as the RDS.

(iib) 2DFDS

In this configuration, we merge the fractional dimensions $\{1, \ldots, K\}$ into one single fractional dimension F, so that there are two dimensions 0 and F in box V. Let the SDOP-0 in the 2DFDS be:

$$\max \quad S_{F,V}(\hat{\boldsymbol{x}}) = \frac{f_0(\hat{\boldsymbol{x}})}{g_0(\hat{\boldsymbol{x}})}, \tag{23}$$

$$\begin{aligned} s.t. \quad & \sum_{k\in\mathcal{K}} \frac{f_i(\hat{\boldsymbol{x}})}{g_i(\hat{\boldsymbol{x}})} \geq L_i(V), \forall i \in \{0, F\}, \\ & \sum_{k\in\mathcal{K}} \frac{f_i(\hat{\boldsymbol{x}})}{g_i(\hat{\boldsymbol{x}})} \leq U_i(V), \forall i \in \{0, F\}, \\ & \hat{\boldsymbol{x}} \in \mathcal{T}. \end{aligned} \tag{24}$$

In this configuration, the algorithm searches along the linear dimension by solving a series of SDOP-0 in the 2DFDS, which is a 0–1 quadratic fractional constrained problem (0–1 QFCP). The algorithm always splits the box along dimension F. According to our numerical test, the FDS performs more efficiently than the RDS. We will discuss the performance of different branching methods in Section 6.2.

4.4. Bounding

Our algorithm conducts the bounding process by solving the subproblems. There are two kinds of subproblems: the linear subproblem (i.e., the SDOP-0) and the fractional subproblem (i.e., the SDOP-k, $k > 0$). We now describe the methods of solving each kind of subproblem respectively.

4.4.1. The fractional subproblem

The fractional subproblem refers to the SDOP-k ($k > 0$) over box V. As defined in (21), it is a 0–1 QCQRP. The problem can be reformulated by the linearization technique presented in (i) of Appendix B.2.2 and resolved by the MILP solver. However, the efficiency can be rather low, because it introduces a large number of additional variables and constraints (Kapoor & Arora, 2009). To solve the problem, we develop a parametric approach inspired by Ibaraki (1983).

We define a parametric optimization problem (POP-k-λ) with parameter $\lambda \in \mathbb{R}$ on dimension $k > 0$ over box V. The POP-k-λ is given by:

$$\max \hat{S}_{k,V}\left(\hat{\boldsymbol{x}}|\lambda\right) = f_k\left(\hat{\boldsymbol{x}}\right) - \lambda g_k\left(\hat{\boldsymbol{x}}\right) \tag{25}$$

$$\begin{aligned} s.t.\ & f_i\left(\hat{\boldsymbol{x}}\right) \geq L_i(V) g_i\left(\hat{\boldsymbol{x}}\right), \forall k \in \mathcal{K}^+, \\ & f_i\left(\hat{\boldsymbol{x}}\right) \leq U_i(V) g_i\left(\hat{\boldsymbol{x}}\right), \forall i \in \mathcal{K}^+, \\ & \hat{\boldsymbol{x}} \in \mathcal{T}. \end{aligned} \tag{26}$$

The POP-k-λ is a 0–1 quadratic constrained quadratic problem (QCQP). We can apply the linearization technique introduced in Appendix B.3.2 to convert the problem into a MILP, and resolve it with the commercial solver.

We then prove the following lemmas for the parametric approach.

Lemma 4.1. *The optimal value $\hat{S}^*_{k,V}\left(\hat{\boldsymbol{x}}|\lambda\right)$ for the POP-k-λ is non-increasing in λ.*

Lemma 4.1 enables us to search with the parametric approach. We then prove when $\hat{S}^*_{k,V}\left(\hat{\boldsymbol{x}}|\lambda\right) = 0$, λ is the optimal value of the subproblem SDOP-k.

Lemma 4.2. *When $\hat{S}^*_{k,V}\left(\hat{\boldsymbol{x}}|\lambda\right) = 0$, we have $\hat{S}^*_{k,V}\left(\hat{\boldsymbol{x}}\right) = \lambda$.*

Therefore, we can solve the SDOP-k ($k > 0$) by solving the POP-k-λ repeatedly, until we find a λ satisfying $\hat{S}^*_{k,V}\left(\hat{\boldsymbol{x}}|\lambda\right) = 0$. Then λ is the optimal value.

The common methods of solving the parametric approach are the Newton method and the binary search. In our testing, the Newton method is the fastest to converge; therefore we develop a parametric approach based on the Newton method to optimize the fractional dimension, as Algorithm 1 illustrates.

We prove the convergence property of the parametric approach in Proposition 4.1.

Proposition 4.1. *The parametric approach converges in a finite number of iterations.*

Thus, we can incorporate the parametric approach in our GB&B algorithm to solve the fractional subproblems. The parametric approach is an efficient method of solving the single ratio optimization problem, yet it is found to be biased if we use it to solve the sum-of-ratios problem directly (Falk & Palocsay, 1992). Hence, we only apply the parametric approach to solve the single ratio problem.

Algorithm 1: The Parametric Approach.

```
Input: ~
    The parameters $\mathcal{N}$, $\mathcal{K}$, $\boldsymbol{u}$, $\boldsymbol{d}$, weights $\boldsymbol{w}$, box $V$, and the dimension $k > 0$ and $\varepsilon$;
Output: ~
    The optimal fractional value $\lambda$, and the solution $\hat{\boldsymbol{x}}^*$.
1:  Set $\lambda = U_k(V)$, solve the POP-$k$-$\lambda$ over box $V$;
2:  if the POP-$k$-$\lambda$ is feasible then
3:      Let $\hat{S}_{k,V}\left(\hat{\boldsymbol{x}}^*|\lambda\right)$ be the optimal value of the POP-$k$-$\lambda$ over box $V$;
4:      while $|\hat{S}_{k,V}\left(\hat{\boldsymbol{x}}^*|\lambda\right)| > \varepsilon$ do
5:          Set $\lambda = \frac{f_k(\hat{\boldsymbol{x}}^*)}{g_k(\hat{\boldsymbol{x}}^*)}$;
6:          Solve the POP-$k$-$\lambda$ over box $V$ and set $\hat{S}_{k,V}\left(\hat{\boldsymbol{x}}^*|\lambda\right)$ be the optimal value;
7:      end while
8:  else
9:      The problem is infeasible;
10: end if
11: return Solution $\hat{\boldsymbol{x}}^*$ and $\lambda$.
```

4.4.2. The linear subproblem

The linear subproblem refers to the SDOP-0 over box V. Given that we already know $\boldsymbol{L}(V)$ and $\boldsymbol{U}(V)$ for a given box V, where $\boldsymbol{L}(V)$ and $\boldsymbol{U}(V)$ are the vectors of the lower bounds and upper bounds over box V respectively, the SDOP-0 in a K^+D search is a 0–1 QCP. We can use the linearization technique introduced in Appendix B.3.2 to convert the problem into a MILP and resolve it with a commercial solver.

For the SDOP-0 in the 2DFDS, we introduce two linearization techniques (iia) and (iib) in Appendix B.2.2 to convert the problem into a MILP. By the numerical study in (ii) of Appendix B.2.3, technique (iia) is more efficient. Therefore, we will refer to the algorithm using technique (iia) in the following paper, when we mention the GB&B method with 2DFDS, if there is no ambiguity.

4.5. The framework of the algorithm

As we consider different configurations of the GB&B algorithm, we first define some notations to express the variables under different configurations.

Let $\mathcal{D}_0$ be the set of searching dimensions in the initialization. Let $\mathcal{D}_0 = \{0\}$ if we consider a 2DFDS, and $\mathcal{D}_0 = \mathcal{K}^+$ otherwise. Let $\mathcal{D}_1$ be the set of searching dimensions in main iterations. Let $\mathcal{D}_1 = \{0\}$ if we consider either type of fixed-dimension search, and $\mathcal{D}_1 = \mathcal{K}^+$ otherwise. Let $\mathcal{D}_2 = \{0, F\}$ if we consider 2DFDS, and $\mathcal{D}_2 = \mathcal{K}^+$ otherwise. Let D be the number of dimensions in the GB&B algorithm. Let $D = 2$ if we adopt a 2DFDS, and $D = K^+$ otherwise. In addition, let S_L be the overall remembered utility of the K service bundles from the linear effects, and let S_F be the overall remembered utility of the K service bundles from the trend effect. Let $\bar{S}_L$ and $\bar{S}_F$ be the upper bound of S_L and S_F respectively.

Algorithm 2 shows the detailed steps of the GB&B algorithm. Our GB&B starts with the initialization procedure in steps 1 to 4. It computes the bounds of each dimension and saves the value in box V_0, which is stored in a list $\mathcal{V}$. When we consider a K^+-dimension search, we will obtain the upper bound of dimension k by solving the SDOP-k; when we consider a 2DFDS, we will set the upper bound of dimension 0 by solving the SDOP-0, and the dimension F as $\bar{S}_F$. We differentiate the 2DFDS from the others in this step because it is rather complex to compute the upper bound of dimension F directly. As dimension F's upper bound does not affect

Algorithm 2: The GB&B Algorithm.

```
Input: ~
    Parameters $\mathcal{N}$, $\mathcal{K}$, $\boldsymbol{u}$, $\boldsymbol{d}$, weights $\boldsymbol{w}$, and $\varepsilon$;
Output: ~
    The optimal assignment $\underline{\hat{\boldsymbol{x}}}$ and the optimal value $\underline{S}$.
1:  Let $V_0$ be a box and set $U_k(V_0)=\bar{S}_F$ for all $k \in \mathcal{D}_0$ and $k \neq 0$;
2:  Solve the SDOP-$k$ over box $\mathbb{R}^D$, and obtain the optimal solution $\hat{\boldsymbol{x}}_k$ for all $k \in \mathcal{D}_0$;
3:  Let $\mathcal{V}$ be a list of boxes, set $U_k(V_0) = S(\hat{\boldsymbol{x}}_k)$ for all $k \in \mathcal{D}_0$, and add $V_0$ to $\mathcal{V}$;
4:  Set $\underline{\hat{\boldsymbol{x}}} = \arg\max\{S(\hat{\boldsymbol{x}}_k)|k \in \mathcal{D}_0\}$, $\bar{S} = \gamma(V)$, and $\underline{S} = S(\underline{\hat{\boldsymbol{x}}})$;
5:  while $\bar{S} - \underline{S} > \varepsilon\bar{S}$ and $\mathcal{V} \neq \emptyset$ do
6:      for $k \in \mathcal{D}_1$ do
7:          Select box $V$ from $\mathcal{V}$ and remove it from the list. Set $\boldsymbol{L}(V) = \underline{S} - \bar{S} + \boldsymbol{U}(V)$;
8:          Solve the SDOP-$k$ over box $V$ and obtain $\hat{S}_{k,V}(\hat{\boldsymbol{x}}^*)$;
9:          if $S(\hat{\boldsymbol{x}}^*) > \underline{S}$ then
10:             Set $\underline{\hat{\boldsymbol{x}}} = \hat{\boldsymbol{x}}^*$ and $\underline{S} = S(\hat{\boldsymbol{x}}^*)$;
11:         end if
12:         if $U_k(V) = \hat{S}_{k,V}(\hat{\boldsymbol{x}}^*)$ then
13:             Introduce new boxes $V'$ and $V''$, where $V' = V'' = V$;
14:             Let $j = \arg\max\{U_i(V) - \hat{S}_{i,V}(\hat{\boldsymbol{x}}^*)|i \in \mathcal{D}_2$ and $i \neq k\}$;
15:             Set $L_j(V') = (U_j(V) + \hat{S}_{j,V}(\hat{\boldsymbol{x}}^*))/2$ and $U_j(V'') = (U_j(V) + \hat{S}_{j,V}(\hat{\boldsymbol{x}}^*))/2$;
16:             Add boxes $V'$ and $V''$ to list $\mathcal{V}$;
17:         else
18:             Set $U_k(V) = \hat{S}_{k,V}(\hat{\boldsymbol{x}}^*)$ and add $V$ to list $\mathcal{V}$;
19:         end if
20:         Remove boxes $\{V \in \mathcal{V}|\gamma(V) < \underline{S}\}$ from list $\mathcal{V}$;
21:     end for
22: end while
23: return $\underline{\hat{\boldsymbol{x}}}$, and $\underline{S}$.
```

the optimal solution, we just set it as $\bar{S}_F$, which is the maximum possible value. The algorithm sets the upper bound of the objective $\bar{S}$ as the summation of the upper bounds of the box V. The lower bound of the objective $\underline{S}$ is set as the maximum satisfaction with the initial solutions.

In the main algorithm, there are five major steps in each iteration: box selection, dimension selection, bounding, branching, and pruning. In the dimension-selection step (i.e., step 6), the algorithm selects a dimension $k \in \mathcal{D}_1$ to start the bounding process. If the algorithm adopts the fixed-dimension searching method, the algorithm always selects dimension 0; otherwise, the algorithm selects one dimension from $\mathcal{K}^+$. In the box-selection step (i.e., step 7), it selects a box V in the list, such that its corresponding upper bound value is the maximum among all the boxes in list $\mathcal{V}$. It updates the lower bound of the box as $\boldsymbol{L}(V) = \underline{S} - \bar{S} + \boldsymbol{U}(V)$. The algorithm then begins the bounding procedure, and the upper bound is computed by solving the SDOP-k in step 8. The algorithm then updates the lower bound of the objective $\underline{S}$ if the solution has higher satisfaction $S(\hat{\boldsymbol{x}}^*)$ in step 10. In the branching process (i.e., steps 12 to 19), the algorithm divides the current box into two smaller sub-boxes V' and V'', or updates the upper bound of the box V. We choose dimension j to do the splitting as it is suggested to be of high efficiency (Ursulenko et al., 2013). In the pruning procedure, all the boxes whose upper bound is smaller than $\underline{S}$ will be removed from the list $\mathcal{V}$ in step 20. We repeat the above steps until the optimality gap is met or there is no box left in the list.

We show in Proposition 4.2 that the GB&B algorithm can identify the global optimal solution within a finite number of iterations. Given enough time, the algorithm can provide the optimal assignment for the service designer.

Proposition 4.2. *The GB&B algorithm converges in a finite number of iterations.*

5. Algorithm improvement

In this section, we discuss several methods to further improve the efficiency of the algorithm. We describe how we speed up the algorithm in the branching and bounding procedures respectively.

5.1. Branching

In each iteration, the algorithm implements the branching process after the bounding procedure. However, we may remove the bounding process and jump straight to the branching procedure directly when we apply a fixed-dimension search.

In the FDS, given that the algorithm always optimizes along dimension 0, it may find exactly the same solution repeatedly when solving the subproblems. Given that $L_j(V') = U_j(V'') = (U_j(V) + \hat{S}_{j,V}(\hat{\boldsymbol{x}}^*))/2 \geq \hat{S}_{j,V}(\hat{\boldsymbol{x}}^*)$, the feasible solution that we find in the bounding solution must lie in the sub-box V''. We can create a list $\mathcal{V}'$, and save the sub-box V'' in the list. If a fixed-dimension search is conducted again on box V'', we can simply omit the bounding procedures and divide V'' into two sub-boxes directly.

5.2. Bounding

In Section 4.4, we state the bounding procedures of the GB&B algorithm for different subproblems. We can improve their efficiency by reducing the symmetry in the constraints, as suggested by Fischetti, Monaci, & Salvagnin (2012). Given that there is no constraint limiting activity bundle assignments, multiple optimal solutions with identical activity-bundle assignments may exist. Breaking the symmetry in the constraints can rule out such a repeated solution, and hence improve the efficiency. For example, we can add certain constraints onto the end effect, as Lemma 5.1 illustrates.

Lemma 5.1. *The following constraint can rule out the solutions whose end effects are the same but ordered in different sequences.*

$$\sum_{i \in \mathcal{N}} u_i q_{i,k} \leq \sum_{j \in \mathcal{N}} u_j q_{j,k+1}, \quad \forall i, j \in \mathcal{N} \text{ and } \forall k \in \{1, K-1\}. \tag{27}$$

Given that (27) limits the end effect to being sorted in an ascending sequence, solutions whose end effects are the same but are ordered in different sequences must be ruled out. For example, there may be feasible solutions $\hat{\boldsymbol{x}}_1$, and $\hat{\boldsymbol{x}}_2$, whose remembered utilities, and peak, spread, and trend effects are the same. The end effects are $(v_1, v_2)^\top$ and $(v_2, v_1)^\top$, where $v_1 < v_2$. Constraint (27) will rule out $\hat{\boldsymbol{x}}_2$ and hence improve the performance of the algorithm by reducing the size of feasible space.

6. Numerical studies

In this section, we report on the numerical studies of our proposed algorithm. We begin by introducing the experiment setup in Section 6.1. We analyze the performance of the GB&B algorithm under different configurations in Section 6.2, and compare the GB&B algorithm with other algorithms in Section 6.3. We discuss the performance of our algorithm in more general configurations in Section 6.4, and the performance of a GB&B-based heuristic algorithm in Section 6.5.

Table 1
Runtime of the GB&B Algorithm under Different Configurations (s).

(N, K)		(60, 8)	(60, 10)	(60, 12)	(90, 8)	(90, 10)	(90, 12)
GB&B with RDS	Max.	387.23	891.51	1832.71	503.98	2099.05	-
	Min.	65.10	100.31	78.21	131.07	178.91	254.31
	Avg.	199.68	377.80	1291.28	290.06	1257.83	1976.16
	Unsolved	0	0	0	0	0	25%
GB&B with KDFDS	Max.	410.82	931.03	1721.30	742.86	2311.09	-
	Min.	72.16	85.77	136.97	167.84	201.07	1805.
	Avg.	179.03	453.02	1456.11	264.98	1363.70	2290.16
	Unsolved	0	0	0	0	0	35%
GB&B with 2DFDS	Max.	311.11	380.93	834.90	422.06	1261.89	-
	Min.	160.53	92.10	107.01	88.29	502.21	701.89
	Avg.	185.24	253.08	504.26	277.73	855.53	1865.91
	Unsolved	0	0	0	0	0	15%

6.1. Experiment setup

In the numerical study, we consider different sizes of the SBDP. The utility of each event is drawn from an exponential distribution with a mean of 50, closely matching the actual distribution of event utility collected in Dixon & Verma (2013). We apply all the improvement techniques proposed in Section 5.

We consider different setups in our numerical study. In Section 6.2 and 6.3, we follow Dixon & Thompson (2016) as closely as possible. We generate 20 instances for each case and use the normalized weights, such that the weighted effects make equal contributions to the objective. The SBDP does not allow for overlap assignment, and we consider the case that each bundle has $m_k = 4$ activities. We assume there is no activity bundle assignment constraint, and that the duration between every two consecutive activities in a bundle is the same, which is $\tau = N/10$.

In Section 6.4, we extend our test to more general features. The basic configuration is the same as Section 6.2 and 6.3, but we modify one parameter at a time to study the performance of the GB&B algorithm in different situations. In Section 6.5, we test the GB&B-based heuristic algorithm with modified weights and target gap.

In our study, all the algorithms are implemented in the MATLAB R2020b environment. We rely on the commercial MILP and MIQCP solver Gurobi to solve the subproblems. The experiments are performed on a computer with Intel Core i7-10875H 2.30GHz CPU and 16GB of RAM. The target gap of the objective function is set to be $\varepsilon = 0.01\%$, and the computation time is limited to one hour. Any runtime exceeding one hour is noted as '-'. When we report the average runtime, we only consider the instances shorter than one hour and calculate their average runtime.

6.2. Efficiency of the GB&B algorithm

In this section, we present the performance of the GB&B algorithm using different branching methods as discussed in Section 4.3. We report on the performance of each configuration in Table 1.

We can see from the table that the performance of the GB&B algorithm depends on the number of service bundles and activities. The runtime grows as K and N increases. As expected, the number of bundles affects the runtime more than the number of activities. We can resolve all the instances with $(N, K) = (60, 12)$ and $(N, K) = (90, 10)$, but only 85% of the instances with $(N, K) = (90, 12)$ within one hour. The GB&B with 2DFDS not only improves the maximum and average computation time, but also reduces the number of unsolved instances within one hour. The KDFDS can only improve the efficiency when the number of bundles is small.

6.3. Comparison with other algorithms

In this section, we compare the performance of the GB&B algorithm with the classical methods. We compare the GB&B algorithm with the exact methods in Section 6.3.1 and with the SA algorithm in Section 6.3.2. Because the GB&B algorithm with 2DFDS is the most efficient one according to the analysis in Section 6.2, we will refer to the GB&B with 2DFDS when we mention the GB&B algorithm in this section.

6.3.1. Comparison with the exact methods

In this section, we compare the GB&B algorithm with the MIQCP solver using the alternative formulation presented in Appendix B.1, and with the linearization technique (ia) introduced in Appendix B.2.2, as it has higher efficiency by the numerical study in (i) of Appendix B.2.3. The results are presented in Table 2.

From the above table, we can see that all of the algorithms take more time to solve the problem as both K and N increase. The number of bundles affects the efficiency of the algorithm more than the number of activities. The GB&B algorithm is the most efficient one, and it can efficiently reduce the percentage of instances exceeding the time limit. The linearization technique is more efficient only when N and K are small. The MIQCP solver with alternative formulation is the least efficient one among the three algorithms having the most unsolved instances.

6.3.2. Comparison with the SA algorithm

In this section, we compare the performance of our GB&B algorithm with the SA algorithm. We analyze the convergence condition between the algorithms, because the number of instances discussed in Dixon & Thompson (2016) is quite limited. We adopt the data of the SA algorithm presented in Table 4 of Dixon & Thompson (2016), and compare it with five randomly generated instances in size $(N, K) = (90, 10)$ solved by the GB&B algorithm.

We report the incumbent solution as the percentage of total iteration increases, because the algorithms take a different number of iterations to stop. Let the number of iterations be n, and the maximum number of iterations be $n_{\max}$, then the percentage of the total iterations is given by:

$$p_n = \frac{n}{n_{\max}} \cdot 100\%. \tag{28}$$

As we study the performance with different instances, we present the percentage of the optimal value instead of the value of the incumbent solution. Let the incumbent solution be S, and the optimal value be $S_{\max}$. The percentage of optimal value is given by:

$$p_S = \frac{S}{S_{\max}} \cdot 100\%. \tag{29}$$

We compare the convergence condition in Fig. 2.

Table 2
Efficiency Comparison Using the Exact Methods.

(N, K)		(60, 8)	(60, 10)	(60, 12)	(90, 8)	(90, 10)	(90, 12)
MIQCP solver with Alternative Formulation	Max.	310.28	760.27	1821.92	1451.26	-	-
	Min.	141.22	186.41	621.07	120.31	1521.52	1917.79
	Avg.	278.56	560.31	1130.85	734.06	1958.31	2574.09
	Unsolved	0	0	0	0	20%	45%
Linearization Technique	Max.	276.51	489.35	790.25	506.46	2071.98	-
	Min.	89.53	296.07	434.11	444.03	1103.63	1840.17
	Avg.	210.94	403.86	686.74	468.54	1658.77	2250.21
	Unsolved	0	0	0	0	0	30%
GB&B Algorithm	Max.	311.11	380.93	834.90	422.06	1261.89	-
	Min.	160.53	92.10	107.01	88.29	502.21	701.89
	Avg.	225.24	253.08	504.26	277.73	855.53	1865.91
	Unsolved	0	0	0	0	0	15%

Table 3
Runtime of the GB&B Algorithm with Different Weights (s).

(N, K)		(60,10)				(90,10)			
Weights		Max.	Min.	Avg.	Unsolved	Max.	Min.	Avg.	Unsolved
$\boldsymbol{w}/\hat{\boldsymbol{w}}$	100%	380.93	92.10	253.08	0	1261.89	502.21	855.53	0
$w_1/\hat{w}_1$	50%	502.13	113.01	320.63	0	2145.36	234.06	910.21	0
	75%	402.71	109.12	273.53	0	1980.13	225.72	901.14	0
	125%	360.19	93.36	240.01	0	1643.87	189.60	765.00	0
	150%	411.20	79.52	275.81	0	1077.53	408.90	874.39	0
$w_2/\hat{w}_2$	50%	531.74	116.69	338.61	0	-	306.89	1350.65	10%
	75%	409.03	116.60	301.22	0	1732.57	188.65	844.09	0
	125%	336.76	135.27	249.51	0	2151.05	195.61	948.79	0
	150%	278.56	87.45	238.56	0	1474.86	272.83	830.97	0
$w_3/\hat{w}_3$	50%	372.44	86.50	204.28	0	1004.02	212.50	537.34	0
	75%	400.21	139.35	225.65	0	994.94	249.68	618.00	0
	125%	480.77	116.00	332.59	0	1340.91	538.26	802.66	0
	150%	626.50	107.80	362.96	0	1453.35	503.14	865.23	0
$w_4/\hat{w}_4$	50%	342.73	60.83	218.22	0	1293.28	145.24	750.17	0
	75%	366.66	83.80	234.35	0	1416.60	496.19	857.56	0
	125%	591.69	113.18	326.88	0	2580.06	613.55	1190.33	0
	150%	924.49	108.98	428.69	0	-	532.86	1346.61	5%

Table 4
Runtime of the GB&B Algorithm on the SBDP Allowing for Bundle Overlap (s).

Overlap	(N, K)	(60, 8)	(60, 10)	(60, 12)	(90, 8)	(90, 10)	(90, 12)
$b = 1$	Max.	311.11	380.93	834.90	422.06	1261.89	-
	Min.	160.53	92.10	107.01	88.29	502.21	701.89
	Avg.	185.24	253.08	504.26	277.73	855.53	1865.91
	Unsolved	0	0	0	0	0	15%
$b = 2$	Max.	285.19	634.70	1280.20	516.66	2285.45	-
	Min.	102.49	77.49	125.07	78.74	466.87	1352.81
	Avg.	204.67	346.23	606.70	378.34	1161.79	1957.15
	Unsolved	0	0	0	0	0	20%
$b = 3$	Max.	332.04	600.55	1473.96	588.66	3005.05	-
	Min.	119.67	186.48	126.26	87.48	684.06	1483.46
	Avg.	221.69	410.11	770.97	432.18	1492.18	2259.72
	Unsolved	0	0	0	0	0	35%

The solid line depicts the incumbent solution of the SA algorithm, and the dashed lines illustrate the incumbent solution of the GB&B algorithm. We can tell from Fig. 2 that the GB&B is able to find the optimal solution with less percentage of iterations than the SA algorithm. Usually the GB&B algorithm finds the optimal solution within 50% of the total iterations.

6.4. SBDP with general features

In this section, we study the performance of the GB&B algorithm on the SBDP with general features. We conduct a study on the SBDP with different weight of sequence effects in Section 6.4.1, and the SBDP allowing for bundle overlap in Section 6.4.2. In Section 6.4.3, we test the GB&B algorithm on the SBDP with non-identical bundles.

6.4.1. SBDP with different weight of sequence effects

In this section, we consider the situation where normalized weight $\hat{\boldsymbol{w}}$ is no longer used. We conduct a study of the GB&B algorithm with different values of the actual weight $\boldsymbol{w}$. We control three of the actual weights to be the same as the normalized weights, and study the performance of the GB&B algorithm as

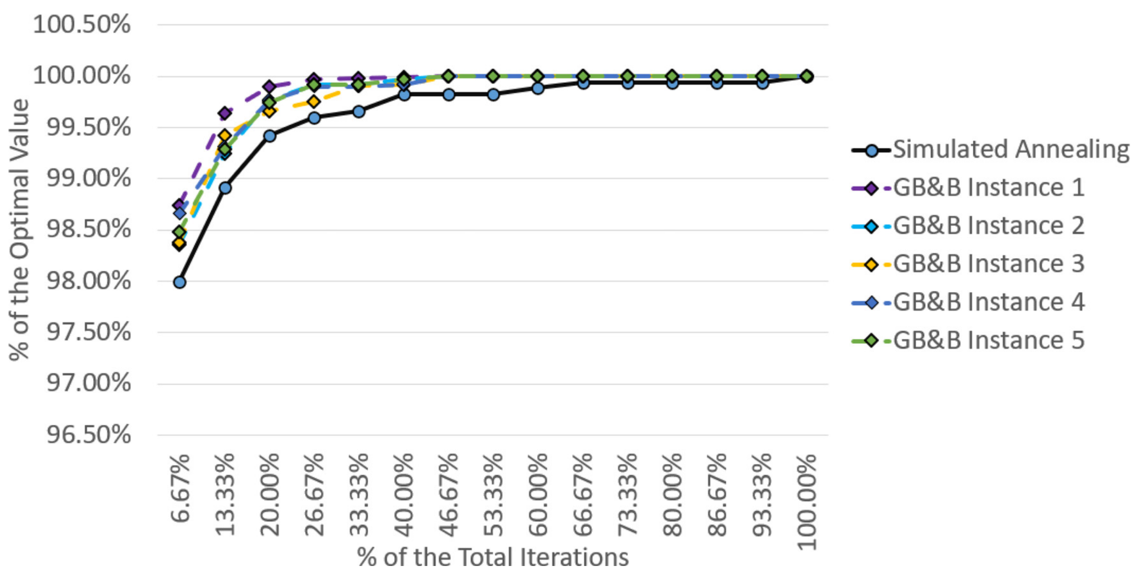


Fig. 2. Comparison of Convergence Condition Using the SA Algorithm.

the remaining weight changes. We test the GB&B algorithm on the SBDP with sizes $(N, K) = (90, 10)$ and $(N, K) = (60, 10)$. We report the performance of the GB&B algorithm in Table 3.

We can tell from the table that the weights have different effects on the performance of the algorithm. The runtime of the algorithm increases as $w_3/\hat{w}_3$ and $w_4/\hat{w}_4$ grows. In addition, the computation time is slightly reduced if $w_1/\hat{w}_1$ is slightly larger than the normalized weight and if $w_2/\hat{w}_2$ is large.

As the peak effect does not conflict with other effects, the runtime is minimized as long as $w_2/\hat{w}_2$ is in the medium level. Because both end and trend effects suggest a late peak, as the weight of end effect $(w_2/\hat{w}_2)$ increases, or the weight of spread effect $(w_3/\hat{w}_3)$ decreases, the runtime decreases as end and trend effects jointly dominate the bundle design. Finally, as the weight of trend effect $(w_4/\hat{w}_4)$ decreases, the SBDP goes closer to a MILP, which makes the runtime decrease.

6.4.2. SBDP allowing for bundle overlap

In this section, we consider the situation where overlap is allowed (i.e., activity can be assigned to $1 < b < K$ bundles). We conduct a numerical study on performance of the GB&B algorithm with different sizes of overlap. The results are shown in Table 4.

From the table, we can see that it takes more time to solve the problem as b increases. In addition to N and K, b can also play an important role in the timing of the computation. Allowing repeat use of the activities enlarges the feasible space, therefore it takes more time for the algorithm to finish the searching.

6.4.3. SBDP ith non-identical bundles

In this section, we consider the situation where the bundles are non-identical. In practice, the number of activities in a bundle (i.e., the capacity of a bundle), the distance between two consecutive activities within a bundle, and the duration of a bundle can be quite different. Therefore, we consider three different configurations in our study with non-identical bundles, and these are presented in Table 5.

Configuration 1 is the identical bundle situation we study in Section 6.2 and 6.3. In configuration 2, we randomly generate the m_k from $\{2, \ldots, 6\}$ with 20 instances. In configuration 3, we relax the constraint on the distance between consecutive activities, but keep the constraint on the overall duration of the bundle as 4τ. In configuration 4, we randomly generate the bundle length from $\{8, 12, 16, 20, 24\}$ with 20 instances, and we keep the constraint that the distance between consecutive activities are the same, such that $\tau \in \{2, \ldots, 6\}$. We omit the case $(N, K) = (60, 12)$ in our numerical study, because the SBDP can be infeasible for some instances under the modified configurations. Performance of the GB&B algorithm is shown in Table 6.

We can see that the algorithm performs quite differently under different configurations. The runtime varies to a small degree when the capacity and duration are non identical, but the runtime increases dramatically if the distance is non-identical. The reason for this is that the constraint on the distance between consecutive activities is very tight, which can rule out many solutions.

6.5. GB&B as a heuristic algorithm

According to the analysis in Section 6.4.1, the efficiency of the algorithm can be enhanced if the weights of the spread and trend effects are smaller. In addition, by the analysis in Section 6.3.2, the algorithm is able to find the optimal solution earlier, before it satisfies the standard target gap ε. Our GB&B algorithm can be modified as a heuristic algorithm by setting the weight of spread or trend effects to be smaller values $(\boldsymbol{w}')$, and by setting the target gap to be a larger value (ε'). We present the GB&B-based heuristic algorithm in Algorithm 3 in the Appendix C.2.

In practice, the service designer can conduct a field experiment for the actual weights $\boldsymbol{w}$, and use the modified weights $\boldsymbol{w}'$ for the heuristic algorithm. Similarly, the service designer selects the target gap ε based on the practical requirement, and take the modified target gap ε' for the heuristic algorithm. In our paper, to test the performance of the GB&B-based heuristic algorithm, we consider the case that the normalized weights are the actual weights (i.e., $\boldsymbol{w} = \hat{\boldsymbol{w}}$), and $\varepsilon = 0.01\%$ is the standard target gap.

We now report on the gap of the heuristic algorithm in our study. Let the remembered utility under modified weights $\boldsymbol{w}'$ and modified target gap ε' be $S(\boldsymbol{w}', \varepsilon')$, and the remembered utility under normalized weights $\hat{\boldsymbol{w}}$ and standard target gap ε be $S(\hat{\boldsymbol{w}}, \varepsilon)$, then the optimal gap of the solution is given by:

$$p_h = \frac{S(\boldsymbol{w}', \varepsilon')}{S(\hat{\boldsymbol{w}}, \varepsilon)} \cdot 100\%. \tag{30}$$

We first study the performance of the heuristic algorithm with modified weights $\boldsymbol{w}'$ and standard target gap $\varepsilon' = \varepsilon = 0.01\%$. By our numerical study, the heuristic algorithm with a modified

Table 5
Configurations of the Numerical Study on the SBDP with Non-Identical Bundles.

No.	Configuration	Note
1	Identical Bundle	Identical capacity, distance, and duration
2	Non-Identical Capacity	Number of activities in a bundle is different
3	Non-Identical Distance	Distance between consecutive activities in a bundle is different
4	Non-Identical Duration	Duration of a bundle is different

Table 6
Runtime of the GB&B Algorithm on the SBDP with Non-Identical Bundles (s).

Configuration	(N, K)	(60, 8)	(60, 10)	(90, 8)	(90, 10)	(90, 12)
Identical Bundles	Max.	311.11	380.93	422.06	1261.89	-
	Min.	701.89	160.53	92.10	88.29	502.21
	Avg.	185.24	253.08	277.73	855.53	1865.91
	Unsolved	0	0	0	0	15%
Non-Identical Capacity	Max.	292.32	429.62	427.48	952.08	-
	Min.	79.65	62.87	106.69	410.86	580.27
	Avg.	195.65	292.27	309.16	892.61	1419.58
	Unsolved	0	0	0	0	15%
Non-Identical Distance	Max.	252.61	491.37	499.55	2638.04	-
	Min.	134.54	89.28	152.35	579.51	775.40
	Avg.	182.41	288.07	354.73	1206.12	2142.80
	Unsolved	0	0	0	0	30%
Non-Identical Duration	Max.	321.95	401.76	570.65	1065.87	-
	Min.	89.02	111.88	276.11	187.29	459.53
	Avg.	162.79	264.94	474.41	918.26	1698.36
	Unsolved	0	0	0	0	20%

Table 7
Performance of Algorithm 3 with $w_3'/\hat{w}_3 \in \{25\%, 50\%, 75\%, 100\%\}$ and $\varepsilon' = 0.01\%$.

(N, K)	$w_3'/\hat{w}_3$	ε'	Runtime (s)			Gap (%)		
			Max.	Min.	Avg.	Max.	Min.	Avg.
(60,8)	25%	0.01%	307.84	101.12	167.71	0.97	0	0.46
	50%		310.74	158.75	172.49	0.67	0	0.16
	75%		347.21	135.83	174.17	0.58	0	0.06
	100%		311.11	160.53	185.24	0	0	0
(60,10)	25%	0.01%	324.00	80.17	207.35	1.29	0	0.51
	50%		362.25	86.43	229.89	0.47	0	0.27
	75%		385.42	82.90	242.55	0.32	0	0.14
	100%		380.93	92.10	253.08	0	0	0
(60,12)	25%	0.01%	634.69	130.15	401.14	1.35	0.22	0.91
	50%		665.91	117.80	451.40	0.76	0	0.33
	75%		694.20	214.01	483.30	0.76	0	0.18
	100%		834.90	107.01	504.26	0	0	0
(90,8)	25%	0.01%	433.16	76.70	243.98	1.94	0	0.99
	50%		427.85	79.52	251.05	1.04	0	0.36
	75%		458.34	83.90	275.20	0.81	0	0.17
	100%		422.06	88.29	277.73	0	0	0
(90,10)	25%	0.01%	886.47	208.10	554.47	2.12	0.35	1.07
	50%		1044.24	214.19	605.26	1.22	0	0.60
	75%		1513.67	258.90	693.47	0.66	0	0.19
	100%		1261.89	502.21	855.53	0	0	0

weight of spread effect is the most efficient. Therefore, we report the runtime and gap in Table 7 when the weight w_3' is set as 25% to 100% of the normalized weight $\hat{w}_3$. We omit the case with $(N, K) = (90, 12)$, because we cannot compute the gap for some instances.

We can see from the table that, when the weight is 75% of the normalized weight, the average optimal gap is quite small. As the percentage decreases, computation time drops and the gap increases. The size of the SBDP can also affect the performance of the algorithm. The computation time and gap increase as the size of the SBDP increases.

The numerical study also suggests that, with 25% of the normalized weight, the solution is still close to the optimal solution with a gap less than 5%. The heuristic algorithm with $w_3'/\hat{w}_3 = 25\%$ and $\varepsilon' = 0.01\%$ can thus offer a solution with high accuracy.

We then study the performance of Algorithm 3 with modified weights $w_3'/\hat{w}_3 = 25\%$ and target gap $\varepsilon' \in \{0.01\%, 0.1\%, 1\%, 10\%\}$. The results are presented in Table 8.

We can see from the table that, when the modified target gap satisfies $\varepsilon' \in \{0.1\%, 1\%\}$, the average gap is stable, without great change. The average gap is relatively high if the target gap is set as 10%. Even with modified weight $w_3'/\hat{w}_3 = 25\%$ and target gap $\varepsilon' = 1\%$, the optimal gap is still less than 5%, but it takes less time to compute the solution. The heuristic algorithm can offer a solution with high accuracy as well.

Table 8
Performance of Algorithm 3 with $w_3'/\hat{w}_3 = 25\%$ and $\varepsilon' \in \{0.01\%, 0.1\%, 1\%, 10\%\}$.

(N, K)	$w_3/\hat{w}_3$	ε'	Runtime (s)			Gap (%)		
			Max.	Min.	Avg.	Max.	Min.	Avg.
(60,8)	25%	0.01%	307.84	101.12	167.71	0.97	0	0.46
		0.1%	148.58	20.63	85.49	0.97	0	0.46
		1%	84.19	11.02	48.36	0.97	0	0.46
		10%	27.65	7.81	17.46	8.17	0.77	3.02
(60,10)	25%	0.01%	324.00	80.17	207.35	1.29	0	0.51
		0.1%	170.30	31.99	102.95	1.29	0	0.51
		1%	101.54	17.06	57.77	1.29	0	0.51
		10%	38.05	9.98	18.85	8.26	2.82	4.21
(60,12)	25%	0.01%	634.69	130.15	401.14	1.35	0.22	0.91
		0.1%	450.74	223.63	308.59	1.35	0.22	0.91
		1%	275.01	99.79	185.87	1.79	0.22	0.97
		10%	72.41	32.16	53.89	7.94	3.25	5.42
(90,8)	25%	0.01%	433.16	76.70	243.98	1.94	0	0.99
		0.1%	435.68	41.22	211.98	1.94	0	0.99
		1%	245.45	22.00	112.18	1.94	0	0.99
		10%	65.69	16.86	39.06	5.43	0.91	3.44
(90,10)	25%	0.01%	886.47	208.10	554.47	2.12	0.35	1.07
		0.1%	777.30	258.10	497.81	2.12	0.35	1.07
		1%	437.08	94.10	258.10	2.12	0.35	1.27
		10%	170.95	42.41	99.54	8.03	0.58	5.82

Table 9
Performance of the Algorithm 3 with $w_3/\hat{w}_3 = 25\%$ and $\varepsilon' = 1\%$.

(N, K)	Runtime (s)			Unsolved
	Max.	Min.	Avg.	
(100,6)	69.68	28.99	51.52	0
(100,8)	287.03	35.19	146.32	0
(100,10)	466.82	72.91	244.93	0
(100,12)	-	84.90	1650.77	20%
(110,6)	84.00	59.88	73.67	0
(110,8)	349.27	36.94	200.26	0
(110,10)	722.91	127.01	350.52	0
(110,12)	-	162.95	1717.34	35%
(120,6)	188.21	38.85	89.01	0
(120,8)	1259.01	43.02	393.94	0
(120,10)	-	97.29	977.66	15%

Based on the previous results, we test our GB&B-based heuristic algorithm on the large-size SBDP with $w_3'/\hat{w}_3 = 25\%$ and $\varepsilon' = 1\%$. The results are presented in Table 9.

We can see from the results that the heuristic algorithm is more efficient than the GB&B algorithm, with reduced average running time. The GB&B-based heuristic algorithm is able to resolve the SBDP with size $(N, K) = (110, 10)$ and $(N, K) = (120, 8)$ within one hour. The runtime grows as K and N increases, and the number of bundles affects the runtime more than the number of activities.

Although we can also modify the algorithm into a heuristic algorithm by setting the weight of trend effect to be lower, the performance of this modification is not as efficient according to our test. Thus, we omit the discussion of this modification in the paper.

7. Conclusion

In this study, we consider a service design problem to optimize the customer's experience of multiple service bundles. We adopt the model in Dixon & Thompson (2016) on customer perception, which models the remembered utility as the linear sum of the sequence effects. The SBDP turns out to be a 0–1 SQRP, and we propose a GB&B algorithm to solve this optimally. In contrast to the classical B&B algorithm, the GB&B algorithm divides the objective function into several dimensions, and it computes the bounds of each dimension, rather than the bounds of the objective function.

The algorithm conducts the bounding process by solving two types of subproblems: the linear subproblem and the fractional subproblem. We solve the fractional subproblem by developing a parametric approach, which converts the 0–1 single ratio problem into a series of 0–1 polynomial optimization problems. For the linear subproblem, we introduce a linearization technique and convert it into a MILP. We consider two methods of choosing the dimension along which to perform the branching: RDS and FDS. The numerical study shows that the GB&B with 2DFDS is the most efficient. The GB&B algorithm outperforms other algorithms using our simulation, and we further test it on the SBDP with general features. The algorithm can also be modified as an adequate heuristic when the size of the problem does not allow it to be solved optimally.

Our work may be extended along several directions. One possible direction is to study the SBDP with activity-date-bundle assignment decision. In this case, the service provider has to make the activity-date-bundle assignment decision at the same time, which can become more complex. Another direction for future research is to consider customer heterogeneity in the service design. Customers are of different types, which may be reflected in the linear remembered utility model, so the solution method could be quite different from ours. Besides the exact algorithms, we may also introduce some methods to approximate the fractional terms. This may be achieved by modifying the parametric approach or the linearization technique.

Declaration of Competing Interest

None.

Acknowledgment

X. Qi is partially supported by Hong Kong RGC General Research Fund Project No. 16210617. Y. Li is partly supported by the IBSS Development Fund by the Xi'an Jiaotong-Liverpool University, Project No. IBSSDF0521-28.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2022.03.044.

References

Almogy, Y., & Levin, O. (1971). A class of fractional programming problems. *Operations Research, 19*(1), 57–67.

Baldacci, R., Lim, A., Traversi, E., & Wolfler Calvo, R. (2020). Optimal solution of vehicle routing problems with fractional objective function. *Transportation Science, 54*(2), 434–452.

Borrero, J. S., Gillen, C., & Prokopyev, O. A. (2016). A simple technique to improve linearized reformulations of fractional (hyperbolic) 0–1 programming problems. *Operations Research Letters, 44*(4), 479–486.

Chang, C.-T. (2001). On the polynomial mixed 0–1 fractional programming problems. *European Journal of Operational Research, 131*(1), 224–227.

Chase, R. B., & Dasu, S. (2001). Want to perfect your company's service? Use behavioral science. *Harvard Business Review, 79*(6), 78–84.

Conforti, M., Cornuéjols, G., Zambelli, G., et al., (2014). *Integer programming*. Cham, Switzerland: Springer.

Dias, S., Brito, P., & Amaral, P. (2021). Discriminant analysis of distributional data via fractional programming. *European Journal of Operational Research, 294*(1), 206–218.

Dixon, M., & Thompson, G. (2016). Bundling and scheduling service packages with customer behavior: Model and heuristic. *Production and Operations Management, 25*(1), 36–55.

Dixon, M., & Verma, R. (2013). Sequence effects in service bundles: Implications for service design and scheduling. *Journal of Operations Management, 31*(3), 138–152.

Dixon, M. J., & Thompson, G. M. (2019). The impact of timing and bundling flexibility on affect-based service package design. *Decision Sciences, 50*(5), 948–984.

Falk, J. E., & Palocsay, S. W. (1992). Optimizing the sum of linear fractional functions. In *Recent advances in global optimization* (pp. 221–258). Princeton, New Jersey: Princeton University Press.

Fernández, J., & Tóth, B. (2009). Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound methods. *Computational Optimization and Applications, 42*(3), 393–419.

Fischetti, M., Monaci, M., & Salvagnin, D. (2012). Three ideas for the quadratic assignment problem. *Operations Research, 60*(4), 954–964.

Hanukov, G. (2022). Improving efficiency of service systems by performing a part of the service without the customer's presence. *European Journal of Operational Research*. Online available https://doi.org/10.1016/j.ejor.2022.01.045

Ibaraki, T. (1983). Parametric approaches to fractional programs. *Mathematical Programming, 26*(3), 345–362.

Jiao, H.-W., & Liu, S.-Y. (2015). A practicable branch and bound algorithm for sum of linear ratios problem. *European Journal of Operational Research, 243*(3), 723–730.

Kahneman, D., Fredrickson, B. L., Schreiber, C. A., & Redelmeier, D. A. (1993). When more pain is preferred to less: Adding a better end. *Psychological Science, 4*(6), 401–405.

Kapoor, R., & Arora, S. (2009). Linearization of 0–1 multi-quadratic fractional programming problem. *Asia-Pacific Journal of Operational Research, 26*(01), 59–84.

Karmarkar, U. (2015). OM Forum—the service and information economy: Research opportunities. *Manufacturing & Service Operations Management, 17*(2), 136–141.

Li, Y., Dai, T., & Qi, X. (2021). A theory of interior peaks: Activity sequencing and selection for service design. *Manufacturing & Service Operations Management*. Online available https://doi.org/10.1287/msom.2021.0970

Li, Y., Ryan, C. T., & Sheng, L. (2021). Optimal level design in video games. *Available at SSRN 3883557*. https://ssrn.com/abstract=3883557

Loewenstein, G., & Sicherman, N. (1991). Do workers prefer increasing wage profiles? *Journal of Labor Economics, 9*(1), 67–84.

Page, K., Pérez, J., Telha, C., García-Echalar, A., & López-Ospina, H. (2021). Optimal bundle composition in competition for continuous attributes. *European Journal of Operational Research, 293*(3), 1168–1187.

Park, C. H., & Lim, H. (2021). A parametric approach to integer linear fractional programming: Newton's and hybrid-newton methods for an optimal road maintenance problem. *European Journal of Operational Research, 289*(3), 1030–1039.

Pfeuffer, F., Stiglmayr, M., & Klamroth, K. (2012). Discrete and geometric branch and bound algorithms for medical image registration. *Annals of Operations Research, 196*(1), 737–765.

Popat, N. (2018, June 7). The how-to: Delivering great customer service. Retrieved March 14, 2022, from https://www.entrepreneur.com/article/314557.

Prokopyev, O. A., Meneses, C., Oliveira, C. A., & Pardalos, P. M. (2005). On multiple-ratio hyperbolic 0–1 programming problems. *Pacific Journal of Optimization, 1*(2), 327–345.

Schaible, S., & Shi, J. (2003). Fractional programming: The sum-of-ratios case. *Optimization Methods and Software, 18*(2), 219–229.

Schöbel, A., & Scholz, D. (2014). A solution algorithm for non-convex mixed integer optimization problems with only few continuous variables. *European Journal of Operational Research, 232*(2), 266–275.

Scholz, D. (2011). *Deterministic global optimization: Geometric branch-and-bound methods and their applications*. New York: Springer Science & Business Media.

Shen, P., Zhu, Z., & Chen, X. (2019). A practicable contraction approach for the sum of the generalized polynomial ratios problem. *European Journal of Operational Research, 278*(1), 36–48.

Skiscim, C. C., & Palocsay, S. W. (2001). Minimum spanning trees with sums of ratios. *Journal of Global Optimization, 19*(2), 103–120.

Stancu-Minasian, I. (2019). A ninth bibliography of fractional programming. *Optimization, 68*(11), 2125–2169.

Thaler, R. H., & Johnson, E. J. (1990). Gambling with the house money and trying to break even: The effects of prior outcomes on risky choice. *Management Science, 36*(6), 643–660.

Ursulenko, O., Butenko, S., & Prokopyev, O. A. (2013). A global optimization algorithm for solving the minimum multiple ratio spanning tree problem. *Journal of Global Optimization, 56*(3), 1029–1043.

Wu, T.-H. (1997). A note on a global approach for general 0–1 fractional programming. *European Journal of Operational Research, 101*(1), 220–223.

Xia, Y., & Yuan, Y.-X. (2006). A new linearization method for quadratic assignment problems. *Optimisation Methods and Software, 21*(5), 805–818.