Deplano, I, Yazdani, D and Nguyen, TT

The Offline Group Seat Reservation Knapsack Problem with Profit on Seats

The Offline Group Seat Reservation Knapsack Problem with Profit on Seats

Igor Deplano, Danial Yazdani, Trung Thanh Nguyen

***Abstract*—In this paper we present the Group Seat Reservation Knapsack Problem with Profit on Seat. This is an extension of the the Offline Group Seat Reservation Knapsack Problem. In this extension we introduce a profit evaluation dependant on not only the space occupied, but also on the individual profit brought by each reserved seat. An application of the new features introduced in the proposed extension is to influence the distribution of passengers, such as assigning seats near the carriage centre for long journeys, and close to the door for short journeys. Such distribution helps to reduce the excess of dwelling time on platform. We introduce a new GRASP based algorithm that solves the original problem and the newly proposed one. In the experimental section we show that such algorithm can be useful to provide a good feasible solution very rapidly, a desirable condition in many real world systems. Another application could be to use the algorithm solution as a startup for a successive branch and bound procedure when optimality is desired. We also add a new class of problem with five test instances that represent some challenging real-world scenarios that have not been considered before. Finally, we evaluate both the existing model, the newly proposed model, and analyse the pros and cons of the proposed algorithm.**

***Index Terms*—heuristic, GRASP, knapsack 2D, Two-dimensional packing problem, Seat reservation.**

I. Deplano are with Liverpool Logistics, Offshore and Marine Research Institute, Department of Maritime and Mechanical Engineering, Liverpool John Moores University, Liverpool L3 3AF, United Kingdom. email: igor.deplano@gmail.com

D. Yazdani are with Shenzhen Key Laboratory of Computational Intelligence, University Key Laboratory of Evolving Intelligent Systems of Guangdong Province, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. email: danial.yazdani@yahoo.com

T.T. Nguyen are with Department of Maritime and Mechanical Engineering, Liverpool John Moores University, Liverpool L3 3AF, United Kingdom. email: T.T.Nguyen@ljmu.ac.uk

Corresponding author: T.T. Nguyen (e-mail: T.T.Nguyen@ljmu.ac.uk).

Manuscript received XXX; revised August XXX.

I. Introduction

In this paper we extend the Offline Group Seat Reservation Knapsack Problem (GSR-KP) presented in Clausen et al. (2010). In the original formulation, a train with W seats stops in H stations. It is required to allocate n reservations. Each reservation i occupy a set of contiguous seats for w_i people from one initial station y_i to a final one h_i. The profit is identified as to maximise the space occupied during the journey. In our extension the value of the profit of the reservation is dependent also on the profits assigned to seats in which the reservation is eventually allocated. Our extension makes the problem more realistic, allowing the modelling of scenarios that were not possible to model with the original formulation. The new scenarios cover the cases where the ideal position of an item is affected by how long the item must be kept in its position. We exploit the original naming style and call the new extension Group Seat Reservation Knapsack Problem with Profit on Seat (GSR-KPPS). Moreover, solving realistically sized instances is challenging for a general solver and often having a good solution rapidly may be better than having an optimal solution later, e.g. when there are fixed time constraints. Thus, we suggest a new GRASP procedure that solves GSR-

KP and GSR-KPPS. Eventually, we adapt and improve the original instances considered in Clausen et al. (2010) adding a random profit on seats and proposing five new problems.

The GSR-KP is the problem of maximising the use of seats in a train during its journey. In the offline version, the passengers reserve a seat from a departing station until their arrival station. Each reservation is known in advance and before the train departs. A reservation can occupy on one or more than one seats. Groups of people are considered to be willing to sit on close seats, this is true especially for long journey, e.g. business trips or family trips with children. For example, according to a survey by Transport Focus (Transport Focus, 2016) in a typical busy UK station that is used for both long and short/commuting journeys, 29% of passengers were travelling in groups, and 7% of passengers were travelling with children.

GSR-KP belongs to the family of the packing problems in two dimensions. In the packing terminology, the reservations are the items, and the train is the bin. The bin and items are rectangles. Packing rectangles into a rectangle is a strongly NP-Complete class of problems (Leung et al., 1990). Regarding the bin, the dimension of the side parallel to the horizontal axis represents the number of seats, while the dimension of the vertical side represents the journey length of the train. For each item, the dimension of the horizontal side represents the number of people in the reservation, while the dimension of the vertical side represents the journey length of the reservation. The dynamic of a reservation consists of reserving a seat, or a group of seats, from a departing station to an arrival station. This special behaviour is modelled by a special constraint which forces the vertical position of the item.

The GSR-KP has similarities with the Dynamic Storage Allocation Problem (DSA) (Garey and Johnson, 1990), but they can be considered two different problems. One of the differences is on the definition of the bin. Both problems consider a fixed height for the bin. However, while the bin in the GSR-KP has a maximum width, the bin in the DSA is generally an open bin. The second difference between the two problems is their objective. The GSR-KP maximises the space occupied, or equivalently minimises the wasted space. The DSA on the other hand has the same objective as a bin packing problem, i.e. it tries to compress the allocation of memory space toward one side to ensure that the allocated areas are contiguous.

Thus, a solution of the DSA can be a solution of the GSR-KP, but a solution of the GSR-KP may not be a solution of the DSA. A DSA problem only has the same solution as the GSR-KP if it is a special instance with an upper bound on the available storage space, and if the amount of memory space to be allocated is strictly greater than the available storage space.

We propose to extend GSR-KP to create a new model that can distribute the allocation of passengers based on their journey length and the profit of the seats, e.g. allocate reservations of long journeys or groups in the centre of the carriage, and reservations of short journeys or unitary groups near doors, reducing the excess of friction during the boarding/alighting phases (Yazdani et al., 2019). Another notable application is in the events industry, e.g different stands may cost differently depending on their location and size. Applications as such can also be modelled using this newly proposed model, considering the lending requests as reservations with time and size, while the price paid to the lender is dependant on the position in which the request will be placed. A similar problem exists also in the tourism industry, for example in the booking system of an hotel, different room may have a different profit.

Our work can be especially meaningful for the United Kingdom (UK) rail industry (Hatano, 2004; UK, 2016). The UK rail industry is an open market, Train Operators are private, or a mix of private and public, companies in competition on the main corridors. In longer journeys, i.e. from Liverpool to London, booking a seat in advance is the common rule of thumb to avoid standing up for the whole journey. Train Operators are interested in reducing delays to improve the Public Performance Measure and gain a competitive advantage over competitors.

The first novelty of this paper is the introduction of a new problem extension, the GSR-KPPS, that binds seat-based profit with the length of the reserved journey in a mixed integer programming (MIP) model.

The second novelty is a GRASP based algorithm that is suitable for both GSR-KP and GSR-KPPS. Such algorithm is useful when the time to achieve a solution is fixed.

The third novelty is the adaptation and extension of the instances used in Clausen et al. (2010). We introduced a new group of problems that better represents some challenging real world scenarios than the ones suggested in the original paper.

The paper is organised as follows. In section II we outline the consistent work found in the literature. Definitions and terminology follows in section III along with the MIP model in detail, section IV outlines the proposed algorithm. The new instances are explained in section V, section VI shows computational results and the paper ends with conclusions in section VII.

II. Previous work

To the best of our knowledge, since the original publication of the problem in Clausen et al. (2010), none of the follow-up study on the Group Seat Reservation Problem has shown to be better than the original work. An online version of the seat allocation problem was first published in Boyar and Larsen (1999), and further analysis was made in Goyal (2018). A real-time algorithm that aims to reduce the boarding/alighting time by maintaining a uniform load on carriages through systematic distribution of passengers with flexible tickets has been recently proposed by the authors in Yazdani et al. (2019).

Many papers have been published in the more general packing problems context, some examples of new approximation approaches are genetic algorithms (Gonçalves and Resende, 2011b; Gupta et al., 2017; Jegadeshwari and Jaisree, 2014; Wang and Chen, 2010) and their biased versions (Gonçalves and Resende, 2011a, 2013), divide and conquer algorithms (in which the solution space is partitioned and searched independently) (Wei et al., 2013), neuro-genetic approaches that mix neural networks and genetic algorithms (Deane and Agarwal, 2013), GRASP algorithms (Resende and Ribeiro, 2019) and GRASP/Path relinking (Alvarez-Valdés et al., 2013), Tabu search (Ceschia and Schaerf, 2013; Crainic et al., 2009) and other greedy randomized heuristics (Crainic et al., 2012; Perboli et al., 2011).

The GSR is a specialised version of the bin packing problem in the two dimensional case, so every algorithm that has been designed for orthogonal two dimensional rectangular packing will work on a GSR problem. The difference in our contribution is that none of them can exploit the nature of the problem: in a two dimensional problem both dimensions are free, while in a GSR problem the allocation toward one dimension is constrained.

III. Definitions, terminology and MIP model

Using the usual terminology of the packing problems and utilising as much as possible the terminology used in Clausen et al. (2010), a train contains W seats and stops at H stations. Let $N = \{1, \dots, n\}$ be the set of reservations. Each reservation asks to reserve w_i seats from station y_i to station $y_i + h_i$. Without any loss of generality, we can assume that $w_i \leq W$.

First, we briefly describe the original GSR-KP as shown in Clausen et al. (2010). The active stations are represented as $Y := \{y_i, | i \in N\} \bigcup \{y_i + h_i, i \in N\}$, and $N_y := \{i \in N \mid y_i \leq y < y_i + h_i\}$ is the set of requests using a seat at station $y \in Y$. Associated with each station $y \in Y$ there is a "height" H_y that represents the distance from station y to the next active station in Y. Let $\delta_i = 1$ if request i is selected. Let x_i be the first seat (from the left, horizontal axis) of request i. Let $E = \{(i, j)\}$ be the set of rectangle pairs which in some way share a station (vertical axis). Finally, let $l_{ij} = 1$ iff request i is located left of request j.

The original model is shown in Eq: (1)-(8), the item profit is identified with the item area $(w_i \cdot h_i)$, the objective (1) is to maximise the profit. Constraint (2) enforces that the number of allocated seats must not exceed the train capacity in any station. Constraint (3) makes sure that two requests i and j are selected, than only one must be on the left of the other. Constraint (4) enforces that two selected items must not overlap. Constraint (5) ensures that an item must be allocated inside the train. The remaining constraints (6)-(8) define the domains of the model variables. We refer the reader to the original paper for a further explanation.

$$
\begin{aligned}
\text{maximize} \sum_{i \in N} & h_i \cdot w_i \cdot \delta_i && (1)\\
\text{s.t.} \sum_{i \in N_y} & w_i \cdot \delta_i \leq W, y \in Y, && (2)\\
& \delta_i + \delta_j - l_{i,j} - l_{j,i} \leq 1, (i,j) \in E, && (3)\\
& x_i - x_j + W \cdot l_{i,j} \leq W - w_i, (i,j) \in E, && (4)\\
& 0 \leq x_i \leq W - w_i, i \in N, && (5)\\
& l_{i,j} \in \{0,1\}, (i,j) \in E, && (6)\\
& \delta_i \in \{0,1\}, i \in N && (7)\\
& x_i \geq 0, i \in N. && (8)
\end{aligned}
$$

Our model extends the original one by considering also a profit value associated to the seat. From the modelling perspective, it translates in a two-dimensional knapsack problem where the item profit is dependant on a combination of its area and its position inside the bin. The distribution of the passengers among and along the carriages can be modelled by assigning profits on the seats, i.e. considering the seats of each carriage, the central seats have higher profit than the seats near to the doors (this profits assignment will allocate reservation with longer journey or more people in the center of the carriage).

Let $Q := \{1, \dots, W\}$ be the set of seats, and $p_k, k \in Q$ be the profit p_k associated to the seat k. Let $\gamma_{i,k}, i \in N, k \in Q$ be 1 iff reservation i occupies seat k.

The new formulation is shown in Eq: (9)-(19).

$$
\begin{aligned}
\text{maximize} \sum_{i \in N} \sum_{k \in Q} & h_i \cdot \gamma_{i,k} \cdot p_k && (9)\\
\text{s.t.} \sum_{i \in N_y} & w_i \cdot \delta_i \leq W, y \in Y, && (10)\\
& \delta_i + \delta_j - l_{i,j} - l_{j,i} \leq 1, (i,j) \in E, && (11)\\
& x_i - x_j + W \cdot l_{i,j} \leq W - w_i, (i,j) \in E, && (12)\\
& w_i \cdot \delta_i \leq \sum_{k \in Q} \gamma_{i,k} \leq w_i \cdot \delta_i, \forall i \in N && (13)\\
& -(1 - \gamma_{i,k}) \cdot 2W + x_i \leq \gamma_{i,k} \cdot k \leq \\
& x_i + w_i \cdot \delta_i, \forall i \in N, k \in Q && (14)\\
& \gamma_{i,k} \in \{0,1\}, i \in N, k \in Q && (15)\\
& 0 \leq x_i \leq W - w_i, i \in N, && (16)\\
& l_{i,j} \in \{0,1\}, (i,j) \in E, && (17)\\
& \delta_i \in \{0,1\}, i \in N && (18)\\
& x_i \geq 0, i \in N. && (19)
\end{aligned}
$$

The differences between the models are on the objective (9), which now includes the profit associated on the seat, and in three additional constraints (13)-(15). Considering an unitary profit we will produce the same results of the original model, thus, the proposed model can be considered as an extension of the original problem. Constraint (13) represents the total allocation of a reservation. If the reservation i is booked, then w_i seats must be allocated, otherwise none has to be assigned. Constraint (14) enforces the contiguity of the allocated seats k, for the reservation i. Constraint (10) represents the knapsack constraint, which enforce allocation inside the train. Constraints (11) and (12) represent the fact that two items i,j must not overlap.

IV. Proposed algorithm

In this section we describe the proposed algorithm. The algorithm is a GRASP procedure (Feo and Resende, 1989, 1995; Resende and Ribeiro, 2016, 2019) that exploits a percentage of the best bound found by the continuous relaxation of the problem (relaxing integer and boolean variables to real variables) for enforcing a stopping condition. The rationale to use a GRASP based method is to produce a simple algorithm that produces good solutions in a very limited time. Such algorithm can be used as a startup for a successive branch and bound procedure, or can be used directly when achieving a solution in the timelimit is more important than achieving absolute optimality.

The main procedure, Algorithm 1 (Algorithm will be abbreviated as Alg from now on), is composed by the following steps: create a random candidate solution, evaluate the candidate and update the best solution if the candidate improves the current best solution. If the solution is not improved then pick a uniformly random number $c \in [0, 1]$ and if $c \leq 0.5$ try to improve the current candidate, otherwise try to improve the best solution found so far.

The stopping criteria of the main procedure are met when at least one of the following conditions is met. First, the maximal number of iterations *max_iterations* has been achieved. Second, a time threshold *timelimit* has been reached. Third, a threshold has been reached on the best candidate evaluation c_{best}. The last threshold is calculated as the fraction *bratio* of the objective value *relaxed_bound* of the continuous relaxation of the problem. The combination of these three stopping criteria has been chosen to keep the running time of the algorithm balanced in borderline conditions.

Algorithm 1 $main\_procedure(relaxed\_bound, bratio, timelimit, max\_iterations)$

```
c_best ⇐ epoch ⇐ 0, start ⇐ current_time(), best ⇐ ∅
while (current_time() − start ≤ timelimit and
relaxed_bound · bratio > c_best and
epoch < max_iterations do
  candidate, limit, bound ⇐ generate_candidate()
  if bound > c_best then
    c_best, limit_best, best ⇐ bound, limit, candidate
  else
    if uniform(0, 1) ≤ 0.5 then
      candidate, limit, bound ⇐
      local_search(candidate, limit)
      if bound > c_best then
        c_best, limit_best, best ⇐ bound, limit, candidate
      end if
    else
      candidate, limit, bound ⇐
      local_search(best, limit_best)
      if bound > c_best then
        c_best, limit_best, best ⇐ bound, limit, candidate
      end if
    end if
  end if
  epoch ⇐ epoch + 1
end while
return best, limit, c_best
```

The return values of the main procedure are *best*, *limit* and c_{best}. *best* is the sequence of indexes that represents the best solution, *limit* is the position of the last fitting reservation index in *best*, c_{best} is the evaluation of the profit totalised in the feasible part of the best solution.

The evaluation procedure, Alg 2, requires as input the *candidate* list. We remind that a *candidate* solution is a permutation of the n indices that represent the reservations, the evaluation procedure "cuts" the candidate up to the last feasible element *limit*. There are two ideas behind the evaluation: firstly to exploit the corner point concept presented in Martello and Toth (1990) and secondly to to exploit the problem structure, and reduce the positions to evaluate along the horizontal axis only. The evaluation procedure keeps a list of candidate positions in *corner*. The algorithm tries to place the items in the first feasible candidate position. *corner* is initially initialised with the position 0. After an item i fits in a position $x \in corner$, the candidate positions list is updated with the corner of the item i, $corner := corner \bigcup \{w_i + x\}$. The evaluation procedure is a constructive first fit heuristic (Zhu et al., 2012).

Algorithm 2 *evaluate_candidate*(*candidate*)

Require: *candidate* ordered list of indexes
Require: n number of items
Require: $N \Leftarrow$ set of items

```
corner ⇐ {0}
positioned ⇐ ∅
end = n
bound ⇐ 0
for i ∈ candidate do
  test ⇐ false
  if h_i + y_i ≤ H then
    for x ∈ corner do
      if x + w_i ≤ W then
        if isNotOverlapping(x, i, positioned) then
          positioned ⇐ positioned ⋃{(x, i)}
          corner ⇐ corner ⋃{x + w_i}
          test ⇐ true
        end if
      end if
    end for
  end if
  if test = true then
    bound ⇐ bound + ∑_{j∈Q} p_j * h_i
  else
    end ⇐ position of i in candidate
    return candidate, end, bound
  end if
end for
return candidate, end, bound
```

The candidate generation, Alg 3, makes use of a $shuffle(x)$ function, where x is the set to shuffle. $shuffle(x)$ returns a random permutation of x. After that, the candidate is evaluated with the procedure in Alg 2.

The local search procedure, Alg 4, swaps half of the positions of the feasible region with positions picked randomly. The method exploits the solution structure: items that belong to feasible regions are located in the initial part of the solution array. So swapping half of the items forces the method to evaluate new solutions while keeping parts of the solution. An example is shown in Fig 1. The local search procedure is in fact reducible to the 2-opt local search (Croes, 1958), with the identification of the sets of the candidates to swap with the feasible and unfeasible region.

Algorithm 3 *generate_candidate*()

Require: n number of items

```
candidate ⇐ shuffle([1, ..., n])
return evaluate_candidate(candidate)
```

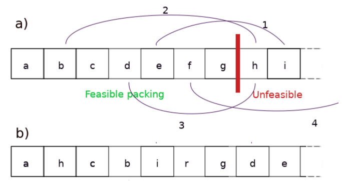

Fig. 1. An example of a local search procedure. a) shows the swap sequence between the feasible and unfeasible region, b) reports the array consequence of the swapping sequence, the limit of the feasible region is removed because the new array requires a new evaluation. The reader should note that since the swapping sequence is random, the combination of multiple swaps may result in a swap of elements in the feasible region.

The Alg 1-4 are designed to be a very fast procedure that can be used to determine lower bounds for a branch and bound framework. The component with varying computational cost is the evaluation procedure Alg 2, since Alg 3 and Alg 4 have a constant number of operations.

Let n be the number of items and W be the maximal number of seats. The worst case scenario for Alg 2 is having groups of one element, the procedure will make $\frac{n\cdot(n+1)}{2}$ $isNotOverlapping(x, i, positioned)$ operations. $isNotOverlapping(x, i, positioned)$ can be implemented as a loop with a check if the new item overlaps with the others already placed. In the worst case it has to compare N items. Summarising, the evaluation procedure

Algorithm 4 $local\_search(candidate, limit)$

Require: $n, numberofavailableitems$

Require: $candidate$ ordered list of indexes from 1 to n

Require: $limit$ index of the candidate list that indicate the first element that does not fit in the bin

```
s ⇐ round(limit/2)
if s = 0 then
  s ⇐ 1
end if
counter ⇐ 0
while counter ≤ s do
  source ⇐ uniform(1, limit)
  target ⇐ uniform(1, n)
  if source ≠ target then
    swap(candidate_source, candidate_target)
  end if
end while
return evaluate_candidate(candidate)
```

Alg 2 has a worst case scenario with a time complexity of $O(N^3)$.

This complexity can be reduced to $O(log_2(n) \cdot N^2)$ by using a balanced binary tree to represent the already placed objects, and a dichotomy search for the $isNotOverlapping(x, i, positioned)$ procedure.

V. Class instances

The original paper (Clausen et al., 2010) considers problem instances used in the literature of the two-dimensional packing, in a total of 190 experiments in five main classes, namely CGCUT (Christofides and Whitlock, 1977), WANG (Wang, 1983), GCUT (Beasley, 1985), OKP (Fekete et al., 2007), GXON and GXOU (Clausen et al., 2010). The instances can be downloaded from the author's repository [1].

Table I shows a comparison between the main features. For each feature we report the minimum and maximum values for each parameters to show a broad picture of the problem class.

Experiments number shows the number of instances available in the class, *stations* reports the journey length measured as number of stations, *seats* is the number of seats available in the train, *journey* represents the journey length for the reservations, *reservation* stands for the number of reservations for each instance and *groups* are the group dimension in the reservations.

DEPL is the new class of instances that we propose, considering also the recent work of Smith-Miles and Lopes (2012). The idea behind is to provide a challenging problem class inspired by a real-world scenario that can be hard to solve. DEPL has an high number of reservations to represent a busy connection between cities, and a range of limits for the other features compatible with a broad range of railway journeys. Most of the details of the proposed instances are based on the real statistics, facts or observations from the UK industry. Specifically:

- The reason for consider up to 50 stations only is that a journey lasting 50 stations is unlikely to happen bar exceptional cases. An example of an exceptional case could be the Trans-Siberian Railway (the longest in the world), which stops in 157 stations during the journey from Moscow to Vladivostok.
- The range of available seats (400-750) is based on the actual number of seats available on the dominant intercity trains in Great Britain West coast line, such as the Virgin Trains fleet.
- The large number of reservations is based on statistics of the Rail Executive (Rail Executive, 2015). This statistics showed that for the InterCity West Coast lines (the major north-south rail connection in Great Britain) with over 100 miles journey distance, at least 60% of tickets purchases are reservations (these are advanced tickets, with which seat reservation is mandatory). This 60% figure is just the lower bound. The actual percentage of reservation should be much higher,

[1] http://hjemmesider.diku.dk/~pisinger/codes.html

because the rest of ticket purchases (near 40% for this type of journeys) are peak and off-peak tickets, which also offer optional or default seat reservations. For peak and off-peak tickets, passengers normally choose reservations to make sure that they have a seat.

- The group size is shaped considering the most likely number of people in a group in a train journey. Based on observations, we have limit the typical number of people in a group to 9 or less, since bigger groups may prefer to reserve directly a bus or a private driver especially in long journeys like the case considered here.

VI. Experimental results

The experiments are divided in two main groups, group one considers unitary profits while group two considers random profit. For each group we evaluated all the instances of the classes in Table I: a total of 195 problem instances per group.

In each experiment we reported the gap, defined as the best bound of the continuous relaxation of the model. The formula is reported in Eq: (20), where *bestbound* is the best bound of the continuous relaxation of the model, *bestinteger* is the best solution found so far, and $\epsilon = \frac{1}{10^{10}}$ is a small numeric constant to avoid a division by zero error. We applied the same formula to calculate the gap in the heuristic results as well.

$$gap = \frac{|bestbound - bestinteger|}{\epsilon + |bestinteger|} \tag{20}$$

The model in Eq:(9)-(19) has been implemented in OPL and solved using IBM CPLEX® 12.7. The proposed Algorithm 1-4 have been implemented in Python 3.6.7. The machine used for running the experiments is an Intel® Core™ i7-7700HQ @ 2.80GHz, 16GB DDR4 RAM. The operating system is Ubuntu™ 18.04. The time limit for solving each instance is 20 minutes. This choice is reasonable if we consider 1) a booking system that accepts reservation up to one hour before departure and 2) the seats must be comunicated before people arrive at platform, e.g. by email, by printing the seat number at the station gate, or by a smarthphone application. The proposed algorithm has been run three times for each instance to avoid potential bias due to a lucky initialisation. The stopping criteria for Algorithm 1-4 are: $bratio = 0.95$ (95% of the result of the continuous relaxation), $max\_iterations = 15000$, and $timelimit = 1200$ seconds. In most of the experiments the *max_iterations* and the *bratio* are the triggering stopping criteria, while in the DEPL class, since the massive number of items considerably slow down the evaluation procedure, the *timelimit* becomes the only stopping criterion.

Table II and Table III report the experiments of group one. Table IV and Table V report the experiments of group two. For CPLEX, the tables report average and standard deviation of the gap, calculated using as baseline the continuous relaxation, and time (reported in seconds). For the algorithm, the tables report the average and standard deviation of computational time (seconds), and average, standard deviation, minimum and maximum. The last column is the difference between the mean gap achieved by CPLEX and the mean gap achieved by the algorithm. This column highlights the degradation of the objective. We highlight any gap degradation lower than 10% in **bold** font.

In all the instances solved in Tables II-III-IV-V, our algorithm achieved a maximum running time of 6.35 seconds and a minimum of 5.67E-05 seconds. The average running time in the first group is 1.95 seconds, while in the second group is 2.32 seconds.

The experiments of group one, apart from G40U20, G50U20 and GCUT13, are relatively easy to solve for CPLEX: 50% of the experiment classes in the first group have an average gap difference lower than 10%.

The second group is more difficult to solve for CPLEX: 57.89% of the classes ran an average of more than 16 minutes, while the objective degradation was averagely less than 10% in 26.31% of the experiment classes.

DEPL experiments with random profit are reported in

TABLE I

Main features of the original instances compared with the proposed one

class	experiments	stations		seats		journey		reservations		groups	
name	number	min	max	min	max	min	max	min	max	min	max
CGCUT	15	15	40	10	70	2	33	16	62	1	43
GXON	60	100	100	100	100	1	45	20	50	1	45
GXOU	20	100	100	100	100	1	35	20	50	6	37
GCUT	65	250	3000	250	3000	62	970	10	50	63	1890
OKP	25	100	100	100	100	1	100	30	97	1	99
WANG	5	70	70	40	40	11	43	42	42	9	33
DEPL	5	6	50	400	750	1	48	2000	2500	1	9

TABLE II

Experiment group one, comparison with unitary profit, first part

instance	CPLEX				algorithm						
	gap		time		gap				time		
name	mean	std	mean	std	mean	std	min	max	mean	std	difference
CGCUT01	1.38	3.08	0.04	0.01	28.52	11.61	18.18	42.11	1.92	0.09	27.14
CGCUT02	1.52	1.77	0.73	0.27	9.06	6.83	1.23	22.22	1.69	1.34	**7.54**
CGCUT03	4.66	4.49	1.09	0.3	20.38	16.02	1.83	47.87	1.92	0.83	15.72
G20N10	1.42	1.28	2.73	2.33	6.43	3.17	2.85	11.90	1.64	1.21	**5.01**
G20N20	0.18	0.24	1.54	1.41	4.88	3.39	0.43	12.37	1.26	1.52	**4.70**
G20N30	0	0	0.54	0.05	6.25	5.77	0.90	16.51	0.86	1.86	**6.25**
G20U20	1.73	2.63	3.14	2.85	34.61	18.87	10.39	62.57	2.27	0.48	32.88
G30N10	4.32	2.96	7.81	3.94	11.41	2.19	6.68	17.57	3.15	0.16	**7.09**
G30N20	0.79	0.71	5.52	2.91	11.23	4.19	2.40	18.37	3.05	0.63	10.44
G30N30	0	0	1.97	1.3	3.69	0.89	0.78	5.17	0.32	0.36	**3.69**
G30U20	0.9	0.83	40.3	43.59	42.21	5.52	33.67	55.40	2.55	0.38	41.31
G40N10	4.59	4.18	9.36	7.9	23.94	3.41	13.81	36.68	3.21	0.12	19.35
G40N20	0	0	13.9	10	10.03	5.26	3.09	23.81	2.90	1.11	10.03
G40N30	0	0	5.26	2.37	8.99	6.21	2.99	20.00	2.55	2.22	**8.99**
G40U20	2.01	0.95	70.19	71.48	53.11	11.08	28.18	73.76	2.72	0.15	51.10
G50N10	3.36	1.87	22.96	27.98	29.19	3.74	21.77	35.92	3.48	0.29	25.83
G50N20	1.69	2.36	12.1	3.96	19.45	10.18	8.46	38.86	3.49	0.11	17.76
G50N30	0	0	6.03	3.55	8.95	4.96	4.40	22.22	5.28	0.83	**8.95**
G50U20	0.99	1.11	195.36	303.41	64.08	12.06	43.03	84.65	2.66	0.20	63.09

TABLE III

Experiment group one, comparison with unitary profit, second part

instance	CPLEX				algorithm						
	gap		time		gap				time		
name	mean	std	mean	std	mean	std	min	max	mean	std	difference
GCUT01	27.62	5.23	0.19	0.02	76.96	75.95	23.32	257.14	0.87	0.79	49.34
GCUT02	11.45	8.7	0.84	0.1	74.73	42.07	17.65	185.71	0.34	0.75	63.28
GCUT03	13.17	11.46	1.78	0.96	69.64	77.38	1.42	259.12	0.84	0.90	65.47
GCUT04	1.71	1.59	4.75	2.76	12.78	7.60	2.64	31.25	1.54	0.90	11.07
GCUT05	5.16	7.77	1.11	0.5	5.29	7.67	0.45	18.46	0.69	0.93	**0.13**
GCUT06	10.37	12.67	2.3	1.27	11.49	11.67	1.27	24.53	0.70	0.90	**1.12**
GCUT07	8.49	5.63	3.94	1.45	8.84	5.26	4.60	18.06	1.43	0.71	**0.35**
GCUT08	1.06	1.09	14.05	3.46	4.82	1.53	0.20	9.41	1.02	0.84	**3.76**
GCUT09	10.62	10.64	1.82	0.8	11.10	10.11	0.35	24.53	0.97	0.88	**0.48**
GCUT10	1.99	1.6	3.63	0.78	2.62	1.21	0.81	4.44	0.06	0.08	**0.63**
GCUT11	7.31	7.75	14.04	10.55	11.56	6.29	0.60	17.79	1.46	0.68	**4.25**
GCUT12	1.36	0.7	18.67	6.94	5.77	3.31	0.70	18.98	0.86	0.65	**4.41**
GCUT13	21.3	22.02	864.27	0.16	14.98	2.59	9.64	21.63	3.60	0.11	**-6.32**
OKP01	6.13	2.48	1.13	0.41	38.46	9.86	17.33	68.38	2.07	0.03	32.33
OKP02	12.55	1.51	0.67	0.16	19.27	7.95	12.52	35.08	1.86	0.06	**6.72**
OKP03	6.26	1.3	0.39	0.06	7.95	2.26	5.41	22.64	1.83	0.05	**1.69**
OKP04	8.7	2.73	1.83	0.35	44.10	9.66	25.51	74.96	2.26	0.04	35.40
OKP05	17.57	2.95	6.99	2.13	75.39	18.04	36.19	123.60	2.90	0.03	57.82
WANG20	11.22	5.75	0.43	0.11	34.56	9.42	13.93	56.44	1.91	0.03	23.34

Table VI. CPLEX ran out of memory in all the experiment made. We were not able to provide the solution of the continuous relaxation, thus we were not able to calculate the gap between the relaxation and the best solution found. Consequently, we decided to run the instances with the proposed algorithm only at different time limits, one minute, three minutes and one hour. Table VI reports the average number of iterations made, maximum, minimum and average objective with standard deviation value found with three different time limits for the heuristic. The result shows that considering a much large number of items, the algorithm's chances to improve an already good solution by remixing part of the best solution or part of the actual solution are less. The evaluation process becomes much slower. For example, with the instance DEPL_0, tripling the time from 60 to 180 seconds only produced a gain in the average objective of 1.64%, and when we increase the time limit from 1 minute to 60 minutes, the gain in the objective was only 5.07%. A similar pattern can be seen in the other cases, where the best gain after an hour of computation was 12.17% in the objective value. To sum it up, the experimental results have shown that the proposed heuristic is a useful tool to provide good, feasible and quick solutions for the challenging instances that CPLEX fails. However, letting the heuristic run for an extended period will not improve performance significantly.

VII. Conclusions

In this paper we have developed a mixed integer programming model for the Group Seat Reservation Knapsack Problem with Profit on Seat. It is an extension of the Offline Group Seat Reservation Knapsack Problem that introduces a profit evaluation dependant on reservation profit, journey lenght, group size, and the profit of reserved

TABLE IV

Experiment group two, comparison with random profit, part one

instance	CPLEX				algorithm						
	gap		time		gap				time		
name	mean	std	mean	std	mean	std	min	max	mean	std	difference
CGCUT01	46.80	13.36	0.47	0.25	118.66	17.27	117.93	120.10	1.70	0.09	71.86
CGCUT02	58.84	12.23	1200.39	0.84	82.06	15.02	80.62	84.49	2.47	0.27	23.22
CGCUT03	64.91	14.20	580.02	409.21	106.61	27.69	101.06	114.82	1.98	0.08	41.70
G20N10	60.69	9.27	827.67	477.01	78.81	7.14	77.53	80.71	2.70	0.24	18.11
G20N20	60.46	9.04	946.67	347.33	88.37	12.94	86.22	90.43	2.65	0.30	27.91
G20N30	53.84	13.73	1141.04	131.85	91.99	11.63	91.68	92.19	4.57	0.78	38.15
G20U20	67.83	11.86	1042.28	216.89	133.10	28.98	127.70	139.59	1.97	0.45	65.28
G30N10	70.59	10.41	1029.90	252.47	93.82	6.27	87.39	99.70	2.80	0.16	23.23
G30N20	64.13	7.36	1008.56	337.22	95.74	12.98	90.78	100.44	2.82	0.17	31.61
G30N30	54.73	14.48	1200.01	0.00	82.83	9.72	81.74	83.61	4.87	1.16	28.10
G30U20	81.32	3.34	665.11	312.73	151.91	11.82	144.30	160.55	2.25	0.37	70.59
G40N10	68.56	8.22	1177.32	52.17	110.12	14.65	104.02	115.68	2.85	0.17	41.56
G40N20	75.04	10.48	1099.03	226.54	101.28	7.28	96.85	105.63	3.01	0.19	26.25
G40N30	57.90	13.72	1200.26	0.54	97.77	21.76	93.40	101.14	4.88	0.78	39.88
G40U20	157.29	97.61	924.14	275.48	182.74	25.11	168.71	193.63	2.32	0.15	25.45
G50N10	78.02	8.25	1121.18	177.59	128.50	18.86	119.13	140.83	3.03	0.31	50.48
G50N20	75.50	7.91	1149.88	112.31	120.58	17.31	113.41	128.85	3.11	0.10	45.08
G50N30	57.06	8.72	1092.70	239.96	99.18	11.33	95.14	103.16	4.81	0.55	42.12
G50U20	172.74	79.85	937.84	263.62	199.80	12.50	190.17	209.26	2.34	0.16	27.05

seats. The proposed extension covers situations where the ideal position of an item is affected by how long the item must be kept in its allocated position.

We have developed a new GRASP based algorithm that solves the original problem version and the newly proposed one.

We have improved the instances considered in the original paper with five new problems that better represent challenging real world scenarios and we have evaluated the limitations of the proposed algorithm.

In the experimental section we have shown that the proposed algorithm can be useful to provide a first lower bound very rapidly, which can be used as a startup for a successive branch and bound procedure. It can also be very useful in the cases where achieving a solution within a short time limit is more important than achieving an absolute optimality.

Acknowledgement

This research is supported by a LJMU PhD scholarship, a NRCP-funded project no NRCP1617-6-125 managed by the Royal Academy of Engineering, and an RSSB-funded project no COF-INP-05.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Alvarez-Valdés, R., Parreño, F., and Tamarit, J. M. (2013). A grasp/path relinking algorithm for two-and three-dimensional multiple bin-size bin packing problems. *Computers & Operations Research*, 40(12):3081–3090.

Beasley, J. (1985). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36(4):297–306.

TABLE V

Experiment group two, comparison with random profit, part two

instance	CPLEX				algorithm						
	gap		time		gap				time		
name	mean	std	mean	std	mean	std	min	max	mean	std	difference
GCUT01	98.60	19.95	18.74	7.33	223.77	186.94	201.86	252.10	0.76	0.69	125.17
GCUT02	76.81	7.21	318.76	460.77	105.62	31.45	97.83	112.42	0.59	0.80	28.81
GCUT03	80.59	11.51	684.56	547.00	162.21	96.29	145.38	176.55	0.94	0.85	81.62
GCUT04	74.59	7.79	1200.02	0.00	99.73	9.33	85.59	109.15	1.86	0.09	25.13
GCUT05	75.91	8.78	1077.14	121.75	80.96	7.82	80.96	80.96	1.45	0.07	**5.05**
GCUT06	75.38	3.87	934.90	429.82	78.44	5.58	78.44	78.44	1.46	0.01	**3.06**
GCUT07	88.10	4.04	1200.01	0.00	92.48	8.26	92.27	92.59	1.53	0.02	**4.38**
GCUT08	81.50	6.64	1168.73	70.00	83.73	6.66	80.18	85.66	1.85	0.03	**2.23**
GCUT09	76.22	11.59	1200.02	0.01	80.34	11.83	80.34	80.34	1.37	0.04	**4.12**
GCUT10	67.50	7.57	1001.04	273.37	69.89	9.37	69.89	69.89	1.43	0.02	**2.39**
GCUT11	83.63	11.37	1200.08	0.05	89.59	7.40	86.36	94.30	1.63	0.03	**5.96**
GCUT12	77.25	6.42	1200.13	0.16	82.05	6.29	78.14	89.62	1.81	0.02	**4.80**
GCUT13	111.15	19.27	1200.21	0.21	103.62	6.89	98.23	110.13	3.15	0.07	**-7.53**
OKP01	41.72	8.33	53.60	25.01	89.48	6.21	80.30	99.89	1.79	0.02	47.77
OKP02	56.61	11.81	17.76	4.98	74.03	15.99	70.20	79.53	1.62	0.08	17.42
OKP03	49.32	12.73	6.15	2.92	57.78	7.12	52.34	63.65	1.60	0.03	**8.46**
OKP04	46.60	7.83	51.68	70.13	100.59	24.30	82.61	119.71	1.94	0.04	53.99
OKP05	69.01	8.52	1135.56	144.74	153.13	10.19	144.65	162.06	2.52	0.06	84.13
WANG20	61.46	14.80	6.54	4.80	119.14	25.29	109.05	130.81	1.71	0.07	57.67

TABLE VI

Experiment group two, DEPL, with different time limits

instance	CPLEX	algorithm						
		iterations	obj				time	gain
name		#	mean	std	min	max	limit	%
DEPL_0	oom	7	10289	242.396	10055	10539	60	0
DEPL_0	oom	16	10458.33	160.051	10331	10638	180	1.64
DEPL_0	oom	284	10810.66	127.021	10685	10939	3600	5.07
DEPL_1	oom	20	9706.33	105.547	9599	9810	60	0
DEPL_1	oom	55	9895	152.302	9771	10065	180	1.94
DEPL_1	oom	1020	10654.66	219.62	10449	10886	3600	9.77
DEPL_2	oom	8	18436.67	303.216	18173	18768	60	0
DEPL_2	oom	19	19467	502.012	18967	19971	180	5.58
DEPL_2	oom	334	20342	172.6	20233	20541	3600	10.33
DEPL_3	oom	35	13006.67	114.988	12913	13135	60	0
DEPL_3	oom	103	13342	137.328	13195	13467	180	2.57
DEPL_3	oom	1858	14589.66	465.6	14271	15124	3600	12.17
DEPL_4	oom	64	20474.67	380.245	20060	20807	60	0
DEPL_4	oom	179	20837.67	447.474	20321	21101	180	1.77
DEPL_4	oom	3930	22723.66	582.85	22189	23345	3600	10.98

Boyar, J. and Larsen, K. S. (1999). The seat reservation problem. *Algorithmica*, 25(4):403–417.

Ceschia, S. and Schaerf, A. (2013). Local search for a multi-drop multi-container loading problem. *Journal of Heuristics*, pages 1–20.

Christofides, N. and Whitlock, C. (1977). An algorithm for two-dimensional cutting problems. *Operations Research*, 25(1):30–44.

Clausen, T., Hjorth, A. N., Nielsen, M., and Pisinger, D. (2010). The off-line group seat reservation problem. *European Journal of Operational Research*, 207(3):1244–1253.

Crainic, T., Perboli, G., and Tadei, R. (2012). A greedy adaptive search procedure for multi-dimensional multi-container packing problems.

Crainic, T. G., Perboli, G., and Tadei, R. (2009). Ts2pack: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, 195(3):744–760.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

Deane, J. and Agarwal, A. (2013). Neural, genetic, and neurogenetic approaches for solving the 0-1 multidimensional knapsack problem. *International Journal of Management & Information Systems (Online)*, 17(1):43.

Fekete, S. P., Schepers, J., and Van der Veen, J. C. (2007). An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55(3):569–587.

Feo, T. A. and Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA.

Gonçalves, J. F. and Resende, M. G. (2011a). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Gonçalves, J. F. and Resende, M. G. (2013). A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics*, 145(2):500–510.

Gonçalves, J. F. and Resende, M. G. C. (2011b). A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. *Journal of Combinatorial Optimization*, 22(2):180–201.

Goyal, S. (2018). Essays on the online multiple knapsack problem & the online reservation problem.

Gupta, I. K., Choubey, A., and Choubey, S. (2017). Clustered genetic algorithm to solve multidimensional knapsack problem. In *Computing, Communication and Networking Technologies (ICCCNT), 2017 8th International Conference on*, pages 1–6. IEEE.

Hatano, L. (2004). Complexity versus choice: Uk rail fares. *Japan Railway and Transport Review*, 37:26–34.

Jegadeshwari, S. and Jaisree, D. (2014). Heuristic algorithm for constrained 3d container loading problem: A genetic approach. *International Journal of Computing Algorithm*, 3(1):1016–1020.

Leung, J. Y., Tam, T. W., Wong, C. S., Young, G. H., and Chin, F. Y. (1990). Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Inc., New York, NY, USA.

Perboli, G., Crainic, T. G., and Tadei, R. (2011). An efficient metaheuristic for multi-dimensional multi-container packing. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pages 563–568. IEEE.

Rail Executive (2015). Intercity west coast overview and vision. https://www.gov.uk/government/publications/intercity-west-coast-overview-and-vision.

Resende, M. G. and Ribeiro, C. C. (2016). *Optimization by GRASP*. Springer.

Resende, M. G. and Ribeiro, C. C. (2019). Greedy randomized adaptive search procedures: Advances and extensions. In *Handbook of Metaheuristics*, pages 169–220. Springer.

Smith-Miles, K. and Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39(5):875–889.

Transport Focus (2016). London termini luggage and group travel survey. unpublished.

UK (2016). http://data.atoc.org/. Technical report, Rail Delivery Group.

Wang, H. and Chen, Y. (2010). A hybrid genetic algorithm for 3d bin packing problems. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, pages 703–707. IEEE.

Wang, P. (1983). Two algorithms for constrained two-dimensional cutting stock problems. *Operations research*, 31(3):573–586.

Wei, L., Oon, W.-C., Zhu, W., and Lim, A. (2013). A goal-driven approach to the 2d bin packing and variable-sized bin packing problems. *European Journal of Operational Research*, 224(1):110–121.

Yazdani, D., Omidvar, M. N., Deplano, I., Lersteau, C., Makki, A., Wang, J., and Nguyen, T. T. (2019). Real-time seat allocation for minimizing boarding/alighting time and improving quality of service and safety for passengers. *Transportation Research Part C: Emerging Technologies*, 103:158 – 173.

Zhu, W., Oon, W.-C., Lim, A., and Weng, Y. (2012). The six elements to block-building approaches for the single container loading problem. *Applied Intelligence*, 37(3):431–445.