Variable neighborhood search heuristics for a test assembly design problem

Jordi Pereira[a,b,*], Mariona Vilà[b]

[a] *Departamento de Ingeniería Industrial, Universidad Católica del Norte, Av. Angamos 0610, Antofagasta, Chile*
[b] *Escola Universitària d'Enginyeria Tècnica Industrial de Barcelona, Universitat Politècnica de Catalunya. C. Comte Urgell, 187 1st. Floor, 08036, Barcelona, Spain*

Abstract

Test assembly design problems appear in the areas of psychology and education, among others. The goal of these problems is to construct one or multiple tests to evaluate some criteria. This paper studies a recent formulation of the problem known as the one-dimensional minimax bin-packing problem with bin size constraints (MINIMAX_BSC). In the MINIMAX_BSC, items are initially divided into groups and multiple tests need to be constructed using a single item from each group, while minimizing the difference among the tests. We first show that the problem is NP-Hard, which remained an open question. Second, we propose three different local search neighborhoods derived from the exact resolution of special cases of the problem, and combine them into a Variable Neighborhood Search (VNS) metaheuristic. Finally, we test the proposed algorithm using real-life-based instances. The results show that the algorithm is able to obtain optimal or near-optimal solutions for instances with item pools with up to 60.000 items. Consequently, the algorithm is a viable option to design large-scale tests, as well as to provide tests for online small-sized situations such as those found in e-learning platforms.

Keywords: Bin Packing, Test Assembly Design, Test Splitting, Variable Neighborhood Search.

1. Introduction

Test assembly design studies the problem of selecting the optimal subset of items (questions) among those available to define one or several tests (questionnaires) subject to specific constraints and objectives. These problems appear in multiple areas, specifically in test design (van der Linder, 2005) both for edu-

*Corresponding author
Email addresses: jorgepereira@ucn.cl, jorge.pereira@upc.edu (Jordi Pereira), mariona.vila.bonilla@upc.edu (Mariona Vilà)

cation (Porter et al., 2013; Sun et al., 2008) and psychology studies (Veldkamp, 2005).

Based on the specific objective of the test, as well as the characteristics of the questions and questionnaires, different problems and formulations may appear. One of these formulations arises when considering the problem as a packing problem, specifically as a one dimensional Bin Packing Problem (BPP) with additional constraints (see Dyckhoff (1990) for a classification of packing problems, and Wäscher et al. (2007) for an update of the previous classification). In the BPP formulation, the questions and questionnaires are interpreted as the items and the bins of the BPP, respectively, and the weight of the items corresponds to some metric (e.g., the difficulty of the question) that needs to be considered during the test design. The objective of the problem is to find disjoint subsets of all the items so that the sum of weights (i.e., the total difficulty) of each subset is as evenly matched as possible.

In this work we consider the formulation introduced in Brusco et al. (2013). This formulation is known as the one-dimensional minimax bin-packing problem with bin size constraints (MINIMAX_BSC). To solve the problem, the authors propose a mixed zero-one integer linear programming model that is then solved using the CPLEX commercial software. For large real-life problems, the authors propose a Simulated Annealing (SA) algorithm that outperforms the quality of the solutions provided by CPLEX.

While Brusco et al. (2013) do not address the complexity of the problem, the analysis of the results showed that all of the algorithms proposed in the paper were able to optimally solve large problems with 2 questionnaires, leading the authors to hypothesize that the special case of 2 questionnaires may be solvable in polynomial time (Brusco et al., 2013, p. 623). Furthermore, the quality of the solutions provided by the SA deteriorates as the number of questionnaires increases, which may identify possible improvements if different solution methods were used.

1.1. Contributions of this work

In this paper, we address the complexity of the MINIMAX_BSC and demonstrate that the MINIMAX_BSC is NP-Hard when two questionnaires are to be constructed and strongly NP-Hard when more than two questionnaires are required. We also show that the case with two sets of questions can be optimally solved in polynomial time and that the case with two questionnaires can be efficiently solved as a subset sum knapsack problem (Kellerer et al., 2004).

Based on these special solvable cases, we derive a new constructive heuristic and three local search neighborhoods. These methods are then combined in a Variable Neighborhood Search (VNS) metaheuristic (Mladenović & Hansen, 1997) to provide a combined approach to solve the problem. VNS is a popular metaheuristic approach (Mladenović & Hansen, 1997; Hansen et al., 2010) based on the use and exploration of different local neighborhoods of an incumbent solution and a mechanism to escape from local optima by restarting the search on random neighbor solutions. Both mechanisms provide a method to systematically explore the solution space.

The results of the method clearly outperform those provided by previous algorithms, such as the SA method proposed in Brusco et al. (2013). More specifically, the proposed method is able to routinely reach a BPP-based lower bound in instances with a large number of questions per questionnaire (i.e., over ten questions per questionnaire) with characteristics of real-life problems, and still manages to obtain small deviations from the theoretical bound in other cases.

1.2. Paper outline

The remainder of the paper is structured as follows. In Section 2, we study the problem, describe its mathematical formulation, address the issue of complexity and identify special cases that can be efficiently solved. We also outline some of the work in the literature devoted to test design and other related problems, such as the BPP. In Section 3, we describe the proposed local searches and a constructive heuristic. We also propose their combination in a VNS-based method. In Section 4, we describe the results of a computational experiment to assess the quality of the proposed algorithm. Finally, in Section 5, we provide conclusions of the present work and identify some future areas of research.

2. Problem description

2.1. Literature review

Test assembly design was early identified as a combinatorial optimization problem. According to Van der Linden (van der Linden, 1998), an initial work in the field (Birnbaum, 1968) proposed a three step process to design and construct tests: (1) identification of the goal of the test under construction; (2) identification of a target function in accordance with the goal; and (3) selection of items based on the target function and the fulfillment of some constraints. These steps clearly mimic the formulation and resolution of any other combinatorial optimization problem.

These three steps are preceded by a preliminary one in which the item pool is constructed. Item pool construction requires the estimation of the information function of the items, a step that is performed using and Item Response Theory (IRT), see Veldkamp (2013) for a basic description and a critique of IRT, and Lu (2014) for a description of an estimation method. The information function is then discretized into specific ability levels of interest (such as the pass/fail ability level for an examination test) that provide coefficients for the objective and/or the constraints of the combinatorial optimization model.

The present paper only considers the item selection process. Even when we limit our attention to the selection step, the literature of the field is very broad. We can divide previous work into three groups: (1) offline single test construction, in which the objective is to design a single test, which is completely constructed before its use; (2) online single test construction, in which the objective is to design a single test, which is sequentially constructed using the information provided by the answers to previous items; and (3) multiple test construction, in which multiple tests are constructed together in order to evaluate examinees with different, but equivalent, test forms.

Note that a method to solve any of these problems can be used to solve other problems after some modifications, and consequently we offer a review of recent procedures for each of these three methods.

Offline single test construction is known in the literature as static test generation (Nguyen & Fong, 2013), or automated test assembly (Veldkamp, 2013). A general reference on the area (van der Linder, 2005) provides a general integer programming formulation for the problem. The proposed objective is to maximize the minimum amount of discrimination index on any ability level of interest, while satisfying some constraints on the different requirements of the test. Veldkamp (2013) considers the model in van der Linder (2005) under robustness considerations on coefficients provided during item pool construction. The effect of using a robust formulation is then verified by constructing a test using a 306 items pool, and comparing the solutions provided by both models. The robust solution is shown to be less sensitive to errors in the evaluation of the information function.

Nguyen & Fong (2013) identifies the problem as a multidimensional knapsack problem (Kellerer et al., 2004), in which the test tries to maximize the discrimination degree of the test (how good the question is at recognizing user proficiency) under constraints on number of questions, time to perform the test, average difficulty and number of questions per topic. The model is then solved using a branch and cut based procedure and the solutions are compared to previous methods found in the literature on instances with up to 50.000 items.

Online single test construction corresponds to the construction of the test along with its resolution and it is usually known in the literature as CAT (computerized adaptive test). Due to the variable nature of the problem, the methods are usually constructive and item selection makes use of greedy rules. Furthermore, CATs try to take into account that multiple tests are to be generated and, thus some randomization is introduced to avoid excessive use of a subset of questions.

In He et al. (2014), four different item-selection methods are evaluated. These methods take into account multiple particularities, such as side constraints, content specifications, time requirements, or item formats, among others. The heuristics are compared in terms of their use of items (how good are they providing multiple different tests) and their accuracy to measure the information function.

Edmonds & Armstrong (2009) considers a hybrid between offline and online single test construction, denoted as Multiple Stage adaptive Test (MST). The hybrid divides items into groups, each belonging to a different stage of the test. When an item is to be selected, the method randomly takes an item from the corresponding group by taking into account the stage and the level shown by the examinee on previous items. The definition of each group is modeled and solved using a commercial solver (CPLEX) for a 1.336 items pool.

While IRT is the predominant method to evaluate the information provided by the items, alternatives exist. In Smits & Finkelman (2014) a health-oriented alternative is put forward. The objective is now to minimize the time required to administer a diagnosis. Consequently, the method minimizes the number of questions required to identify the expected total score of the test. The method applies an ordinal regression on the previous answers in order to stop administering questions once a required degree of precision on the final result of the test has been reached. The applicability of the model is then tested on a real data set provided by a screener for depression.

Multiple test construction, or parallel test design, is used when multiple interchangeable tests need to be constructed. An example in which multiple tests are needed is the evaluation of different candidates at different time frames, or the assignment of different tests to different students in order to avoid cheating. A basic multiple test construction method builds tests sequentially until the desired number of tests have been generated. Chen et al. (2012) proposes one of these sequential methods. The method first divides the item pool into groups (referred as cells) containing questions with similar characteristics. Then, a specified number of items from each group are randomly selected in order to generate different tests in each execution. Chen (2015) provides an improved method that incorporates a content-balancing element similar to those found on CAT test construction methods. Both works use a real 540 items pool to test their proposed method.

Chang & Shiu (2012) proposes a method to construct multiple tests based on CLONALG, an algorithm based on the clonal selection principle in a biological immune system. The authors report results with a 4.000 items pool used to construct 5 parallel tests.

Hwang et al. (2008) proposes a tabu search approach that generates multiple tests subject to time constraints and identical number of questions per test. The method was tested in simulated 50.000 items pools, which represents the largest sized item pool used in the literature.

The previous references try to focus on constructing a predefined number of tests while minimizing deviations among the tests and/or maximizing the discrimination power of each test. Another possible objective aims at maximizing the utilization of the item pool, by constructing the maximum number of tests under some constraints on the repeated use of each item.

Songmuang & Ueno (2011) consider a mixed problem in which both the number of constructed tests and quality of each individual test are considered. The problem is then solved using a Bee Algorithm. The computation is performed using parallelization and multiple tests are obtained for item pools with up to 20.000 items.

Ishii et al. (2014) considers the maximization of the number of tests using a maximum clique formulation. Their work builds upon the study by Belov & Armstrong (2006), in which the problem is assimilated to a set packing problem and solved using a constraint programming approach. The proposal of Ishii et al. (2014) constructs a graph in which a maximal subset of tests that share none or few items (the clique) is sought. The method first constructs multiple tests with the required conditions like the discrimination power and number of items used (see also Belov, 2008). The tests define the vertices of the graph and edges among vertices are included if the number of shared items is deemed small. Once the graph has been built, a classical method to ascertain the maximum clique is used.

A different approach, which also links test assembly design to a classical combinatorial optimization problem was proposed in Brusco et al. (2013). In this case, the problem is formulated as a constrained Bin Packing Problem (BPP) and it is named the one-dimensional minimax bin-packing problem with bin size constraints (MINIMAX_BSC). The authors consider that the item pool has been previously divided into groups of items with similar characteristics, as seen in van der Linden & Boekkooi-Timminga (1988), Chen et al. (2012) or Chen (2015), and then they consider the generation of multiple tests each containing an item from each group.

As pointed out by van der Linden & Boekkooi-Timminga (1988) and Brusco et al. (2013), the division of items into groups has multiple advantages as it can be used to obtain tests with similar composition as well as to enforce constraints which may be difficult to specify. The objective of the problem is then to obtain tests with similar discrimination index on the ability level of interest. The authors solve the problem using a commercial software (CPLEX) and a Simulated Annealing approach on item pools with up to 6.000 items.

In this paper, we extend the work of Brusco et al. (2013) by considering its complexity and by providing a new efficient solution method for the MINIMAX_BSC. The computational experiment shows that we are able to solve instances with up to 60.000-item pools within less than a minute in a modern commodity computer, and the solutions obtained show very small optimality gaps when compared to a theoretical lower bound on the maximum discrimination index. For smaller sized instances, the method provides solutions of the same quality within seconds. These results open the door to their use in online single test construction methods using the shadow test approach (He et al., 2014). The shadow test approach is a online single test construction method in which a complete test is constructed in each CAT decision step and the final item used is selected from the constructed test.

Additionally, this work also verifies the applicability of the proposed method on different conditions to those proposed in Brusco et al. (2013). According to Brusco et al. (2013), items are grouped according to their discrimination indexes. We conduct an experiment in which items are randomly grouped (so we can consider that they have been grouped by subject topic or some other constraint). The results of the additional experiment show that the method is still capable of obtaining optimal or near-optimal solutions, highlighting the possible application of this method to other multiple test construction problems in which the number of tests is known in advance and the constraints can be adapted into the multiple initial groups method required by the MINIMAX_BSC formulation.

As the problem under study is a special case of the BPP, we also provide a brief bibliography on the methods available for packing problems. We refer the reader to the classifications of packing problems provided in Dyckhoff (1990); Wäscher et al. (2007) and the references included in these papers for a more extensive and comprehensive literature review. We will only highlight that multiple approaches have been proposed for the resolution of the one-dimensional case, both exact (i.e., usually based on column generation Kallrath et al., 2014; Brandão & Pedroso, 2014) and heuristic, such as genetic algorithms (Quiroz-castellanos et al., 2015), variable neighborhood search (Fleszar & Hindi, 2002; M'Hallah et al., 2013) or hyperheuristics (López-Camacho et al., 2014) methods.

Our proposal belongs to the second group and its more distinguishing feature is the use of special neighborhood structures, rather than the common swap or exchange neighborhoods. The proposed neighborhood structures allow us to explore larger neighborhoods and to find the best move in these neighborhoods within very short running times. The neighborhood structures are then combined in a Variable Neighborhood Search (VNS) method, and allow us to obtain optimal or near-optimal solutions in very short running times. While the proposed method is not directly applicable to other Bin Packing problems, it offers an example on how to apply the resolution of special cases of the problem into the definition of effective heuristics.

2.2. Mathematical formulation, lower bounds and special cases

The MINIMAX_BSC considers the problem of sorting a number of questions into questionnaires of equal length (i.e., each is composed of the same number of questions). The questions are initially divided into different homogeneous sets, and a solution corresponds to the assignment of one question from each set to each questionnaire so that all questions are assigned, and some measure of fairness among the resulting questionnaires is optimized (e.g., minimizing the maximum difference between the total difficulty of the questions assigned to two different questionnaires).

These constraints (e.g., multiple sets and assigning a question of each set to each questionnaire) are particularly common in applications pertaining to the splitting of a set of items to form parallel tests of equal length. For example, in the research of van der Linden & Boekkooi-Timminga (1988), a first stage divides the questions into homogenous sets. Then, a second stage tries to find a solution in which questionnaires may only use one question from each set. While the first and second stage problem could be solved as a single problem, the first stage allows for the inclusion of additional constraints, (e.g., the grouping of questions with comparable difficulty or similar content) and avoids the selection of highly unbalanced solutions (e.g., one test is only composed of easy and difficult questions, while another contains a combination of moderately difficult questions).

The formulation proposed in Brusco et al. (2013) is described as follows: T disjoint sets ($1 \leq t \leq T$) of questions exist, and each set is composed of B questions ($1 \leq r \leq B$) with weights w_{rt} ($1 \leq r \leq B$, $1 \leq t \leq T$). These weights represent the discrimination index of the question (i.e., the percentage of recipients that are capable of correctly answering the question). The questions need to be grouped into B ($1 \leq b \leq B$) groups (i.e., questionnaires) in such a way that each questionnaire contains exactly one item from each one of the sets; and the total weight of the questions assigned to a questionnaire identifies the

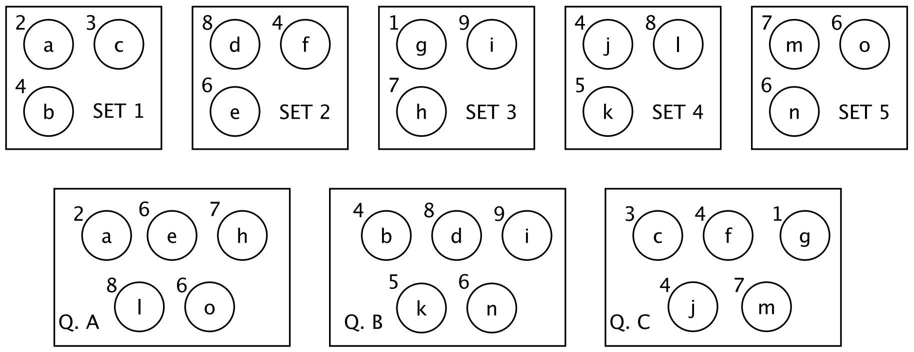

Figure 1: Example of a MINIMAX_BSC instance and one of its possible solutions. The instance has five sets and three questions per set. The questions must be divided into three questionnaires, which are pictured in the solution. Each question is named with a lowercase letter, and its weight w_{rt} is indicated above (e.g., question c has a weight of 3). In the solution pictured, questionnaires are identified with uppercase letters, and the total weight of the questionnaires can be calculated by summing the weights of all the questions assigned to each questionnaire (e.g., questionnaire A, which contains questions a, e, h, l and o, has a total weight of 29). The value that needs to be optimized is the maximum total weight for any questionnaire. In the solution pictured, the objective value is 32, which corresponds to questionnaire B.

difficulty of the questionnaire. The objective is then to obtain questionnaires with similar difficulty. An example of both an instance and a solution for this problem is detailed in Figure 1.

One of the possible methods to achieve the previous objective and the one used in the MINIMAX_BSC is to minimize the maximum difficulty of any questionnaire; this is known as a minimax objective and it is similar to the objective of some Bin Packing related problems, like the unrelated parallel machine scheduling problem with maximum completion objective (Dell' Amico & Martello, 1995).

Equations (1) to (5) are a valid integer programming (IP) formulation for the problem, which uses binary decision variables x_{rbt} to identify the assignment of different questions to the questionnaires. The variable is equal to 1 when item r from set t is assigned to group b, and 0 otherwise. Finally, a real-valued variable Z is used to identify the maximum weight among the questionnaires.

$$[\text{MIN}] Z \tag{1}$$

subject to:

$$Z \geq \sum_{r=1}^{B} \sum_{t=1}^{T} w_{rt} \cdot x_{rbt}, \qquad 1 \leq b \leq B; \tag{2}$$

$$\sum_{r=1}^{B} x_{rbt} = 1, \qquad 1 \leq b \leq B, 1 \leq t \leq T; \tag{3}$$

$$\sum_{b=1}^{B} x_{rbt} = 1, \qquad 1 \le r \le B, 1 \le t \le T; \tag{4}$$

$$x_{rbt} \in \{0,1\}, \qquad 1 \le r \le B, 1 \le b \le B, 1 \le t \le T \tag{5}$$

Equation (1) corresponds to the objective function (i.e., the minimization of the weight of the questionnaire with the largest sum of weights). Z is calculated in the constraint set (2). Equation set (3) verifies that each questionnaire has one question from each set, while equation set (4) checks that each question of each set is assigned to one questionnaire. Finally, constraint set (5) defines the domain of the variables. Note that constraint set (4) corresponds to the traditional BPP constraint set, forcing each item to be assigned to one bin, while constraint set (3) corresponds to the set of additional constraints over the traditional BPP constraint set.

While the mathematical formulation can be used to obtain the optimal solution to small-sized problems and to obtain lower bounds on the optimal solution of larger-sized ones, alternative methods exist in some cases:

1. Let W denote the sum of all of the weights of the questions ($W = \sum_{i \le b \le B, 1 \le t \le T} w_{rt}$). Then, equation (6) provides a valid lower bound for Z. This lower bound is used in the computational experiments to verify the optimality of the solutions.

$$Z_{LB} = \lceil W/B \rceil. \tag{6}$$

2. When T=2, the optimal solution can be obtained as follows: first, order the questions of one set in non-decreasing order of weights and the questions of the other set in non-increasing order of weights; let $(a_1, a_2, ..., a_B)$ and $(a'_1, a'_2, ..., a'_B)$ be such sequences. Then, build a solution so that the b-th questionnaire $(1 \le b \le B)$ is composed by questions $\{a_b, a'_b\}$.

Claim 1. The previous algorithm provides an optimal solution.

Proof The claim is verified using exchange arguments. Let us consider a solution constructed with the previous method, and consider the case when the sum of weights from the first questionnaire equals the objective function. Then, to reduce the objective value, one of the questions in $(a_2, ..., a_B)$ must be assigned together with question a'_1; however, because the weight of a_1 is smaller than or equal to the weights of the questions in $(a_2, ..., a_B)$, the objective value would not improve. Hence, the current solution is optimal.

Now suppose that the sum of weights from the second questionnaire equals the value of the objective function. Then, to reduce the objective value, a_1 can only be assigned together with a'_2, effectively reducing the sum of weights of the second questionnaire. However, the best possible assignment for a'_1 among the remaining questions is thus a_2 and $w_{a'1} + w_{a2} \ge w_{a2} + w_{a'2}$. Hence, the exchange would increase the objective value.

The previous exchange argument can be extended to the remaining questionnaires. Note that we must exchange the questions between a questionnaire and an earlier questionnaire, but this would not improve the global solution unless the newly exchanged question is exchanged with the question of an earlier questionnaire. Eventually, the question from the first or the second questionnaire is to be considered, and the previous arguments state that the exchange is not going to improve the solution, thus proving this claim. $\square$

A computer implementation to solve the MINIMAX_BSC with $T = 2$ requires the ordering of two sets with complexity of $O(B \cdot logB)$. This is a clear improvement over the $B!$ permutations that would need to be explored if a brute force method were used.

2.3. Complexity of the problem

In this section, we first reduce the Partition Problem to the MINIMAX_BSC problem in which $B = 2$. Then, we show that the 3-Partition Problem reduces to a general MINIMAX_BSC, ascertaining the theoretical difficulty of the problem. Finally, we identify the relationship between the MINIMAX_BSC problem and the subset sum knapsack problem (Kellerer et al., 2004), which will be used in the proposed VNS as one of the neighborhood evaluation methods.

Theorem 1. The MINIMAX_BSC problem with B=2 is NP-Hard.

Proof. We use a reduction for the Partition problem that is known to be NP-Complete (Garey & Johnson, 1979, p. 223).

Problem PARTITION. Given a finite set A and a size $s(a) \in Z^+$ for each $a \in A$, is there a subset $A' \subset A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A'-A} s(a)$?

To ease the explanation of the reduction procedure, we assume that A is an ordered set, and thus, the t-th element of A can be referred to as a_t.

Reduction: Given an instance of the problem PARTITION, the corresponding MINIMAX_BSC instance with $B = 2$ is constructed as follows: Let T be the cardinality of A ($T = |A|$), and let each set ($1 \leq t \leq T$) be composed by two items with weights $w_{t1} = a_t$ and $w_{t2} = 0$.

The optimum objective value for this instance is equal to $\sum_{a \in A} s(a)/2$ if and only if the answer to the original instance of the Problem PARTITION is "Yes" $\square$

Theorem 2. The general MINIMAX_BSC problem is strongly NP-Hard.

Proof. We use a reduction for the 3-Partition problem that is known to be strongly NP-Complete (Garey & Johnson, 1979, p. 96).

Problem 3-PARTITION. Given a set A of $3 \cdot m$ elements, a bound $U \in Z^+$, and a size $s(a) \in Z^+$ for each $a \in A$ such that $U/4 < s(a) < U/2$ and such that $\sum_{a \in A} s(a) = m \cdot U$, can A be partitioned into m disjoint sets $A_1, A_2, ..., A_m$ such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = U$?

Reduction: Given an instance of problem 3-PARTITION, the corresponding instance of MINIMAX_BSC is constructed as follows: $B = m$ and $T = 3 \cdot m$. Then, let $w_{t1} = a_t$, $w_{tb} = 0$ for $2 \leq b \leq B$ and $1 \leq t \leq T$.

The optimum objective value for this instance of the MINIMAX_BSC is equal to U if and only if the answer to the original instance of Problem 3-PARTITION is "Yes" $\square$

The reduction of the MINIMAX_BSC with $B = 2$ to the Partition Problem allows us to solve this special case using efficient methods found in the literature for the subset sum knapsack problem. While the subset sum knapsack problem is NP-Hard (Garey & Johnson, 1979, p. 65), the problem is known to be easily solvable in many practical circumstances.

The subset sum knapsack problem can be defined as follows: let there be n $(1 \leq i \leq n)$ items, each with weight w_i, and a maximum capacity W. The objective is to find the subset of items such that the total weight is maximized without surpassing the maximum capacity W. The mathematical formulation of the problem using a set of variables $x_i \in \{0, 1\}$, $(1 \leq i \leq n)$ corresponds to maximizing $\sum_{1 \leq i \leq n} w_i \cdot x_i$ subject to $\sum_{1 \leq i \leq n} w_i \cdot x_i \leq W$.

In this case, the subset sum knapsack problem considers the optimal assignment of questions to one of the questionnaires (i.e., implicitly assigning the remaining questions to the other questionnaire). Let w_i be the additional difficulty associated to selecting the most difficult question in the set i, which implies $w_i = |w_{i1} - w_{i2}|$ for each i $(1 \leq i \leq T)$. Let $x_i \in \{0, 1\}$, $1 \leq i \leq T$ be a binary variable that equals 1 if the most difficult question from set i is assigned to the questionnaire under construction and 0 if the easy question from set i is assigned instead. Finally, let W be $\lfloor \sum_{i \leq i \leq T} |w_{i1} - w_{i2}|/2 \rfloor$.

Claim 2. The optimal solution to the previously defined subset sum knapsack problem provides an optimal assignment of questions to questionnaires.

Proof. To prove this claim, first note that each question must belong to one questionnaire; then, the unselected question of each set defines the second questionnaire. Let Z_1 and Z_2 be the total difficulty of each questionnaire, which can be obtained using (7) and (8):

$$Z_1 = \sum_{i=1}^{T} x_i \cdot |w_{i1} - w_{i2}| + \sum_{i=1}^{T} \text{MIN}(w_{i1}; w_{i2}) \tag{7}$$

$$Z_2 = \sum_{i=1}^{T} (1 - x_i) \cdot |w_{i1} - w_{i2}| + \sum_{i=1}^{T} \text{MIN}(w_{i1}; w_{i2}) \tag{8}$$

Then, the ideal weight for the non-constant part of Z_1 is $\lfloor \sum_{i \in B} |w_{i1} - w_{i2}|/2 \rfloor$, which would lead to a non-constant part of Z_2 equal to $\lceil \sum_{i \in B} |w_{i1} - w_{i2}|/2 \rceil$. Hence, if Z_1 decreases, Z_2 increases. As the subset sum knapsack problem maximizes Z_1, and the maximum possible value for Z_1 is $\lfloor \sum_{i \in B} |w_{i1} - w_{i2}|/2 \rfloor$, the optimal subset sum problem minimizes the non-constant part of (8), effectively minimizing the MINIMAX_BSC problem.$\square$

Note that during the previous proof, the definition of the problem imposes w.l.o.g. that $Z_1 \leq Z_2$.

3. Description of the algorithm

The Variable Neighborhood Search (VNS) approach is a metaheuristic used as a framework to develop heuristics for specific problems. The primary concepts of the VNS approach are based on the following observations (Hansen et al., 2010): (1) local optimality with respect to a neighborhood does not imply local optimality in other neighborhood structures; (2) an optimum is a local optimum for any neighborhood structure; and (3) usually a local minimum for one neighborhood is similar to those in other neighborhoods.

Based on these observations, VNS uses multiple local searches, each with a different neighborhood structure. The algorithm starts from an initial solution and applies different local searches until a locally optimal solution to each of these neighborhoods is found. This technique is usually known as a variable neighborhood descent (VND).

While VND improves upon the use of traditional local search methods, it may still become trapped in a local (i.e., non-global) optimum. To escape from these optima and to diversify the search, an additional mechanism must be incorporated. This mechanism is known as a shake operator and introduces a random movement from the current solution to a neighbor solution in one of the neighborhood structures. The resulting algorithm is known as a general VNS.

This paper proposes a general VNS for the MINIMAX_BSC; the remaining subsections describe the primary components of the implementation. Subsection 3.1 describes a greedy constructive procedure, which is used to obtain an initial solution and also during the "shaking" procedure. Subsection 3.2 describes the neighborhood structures used in this implementation and the methods used to reach the best neighbor for each neighborhood structure. Subsection 3.3 documents the "shaking" procedure, and Subsection 3.4 details the conditions used to terminate the search.

3.1. Greedy Initialization heuristic

While the definition of a good initial solution is not an important aspect of an effective VNS heuristic, the quality of the initial solution helps reduce the time required to reach the first local optimum. Therefore, we propose the use of a fast $O(T \cdot B \cdot logB)$ heuristic, with an absolute performance guarantee equal to $R = \max_{1 \leq t \leq T} r_t$ ($r_t = \max_{1 \leq r \leq B} w_{rt} - \min_{1 \leq r \leq B} w_{rt}$). Algorithm 1 depicts the algorithm.

Algorithm 1. Greedy heuristic.

```
for each set $1 \leq b \leq B$ do
  $Z_b$=0
end for
for each set $1 \leq t \leq T$ do
  Order the questions of the set into a non-decreasing order of the weights.
  Order the questionnaires into a non-increasing order of accumulated weights.
  for each $t$, $1 \leq t \leq T$ do
    Assign the $t$-th question to the $t$-th questionnaire.
  end for
end for
```

An example depicting the process of Algorithm 1 can be found in Figure 2.

Note that the method is based on considering the questionnaires of a partial solution as single questions and finding the best assignment combining these

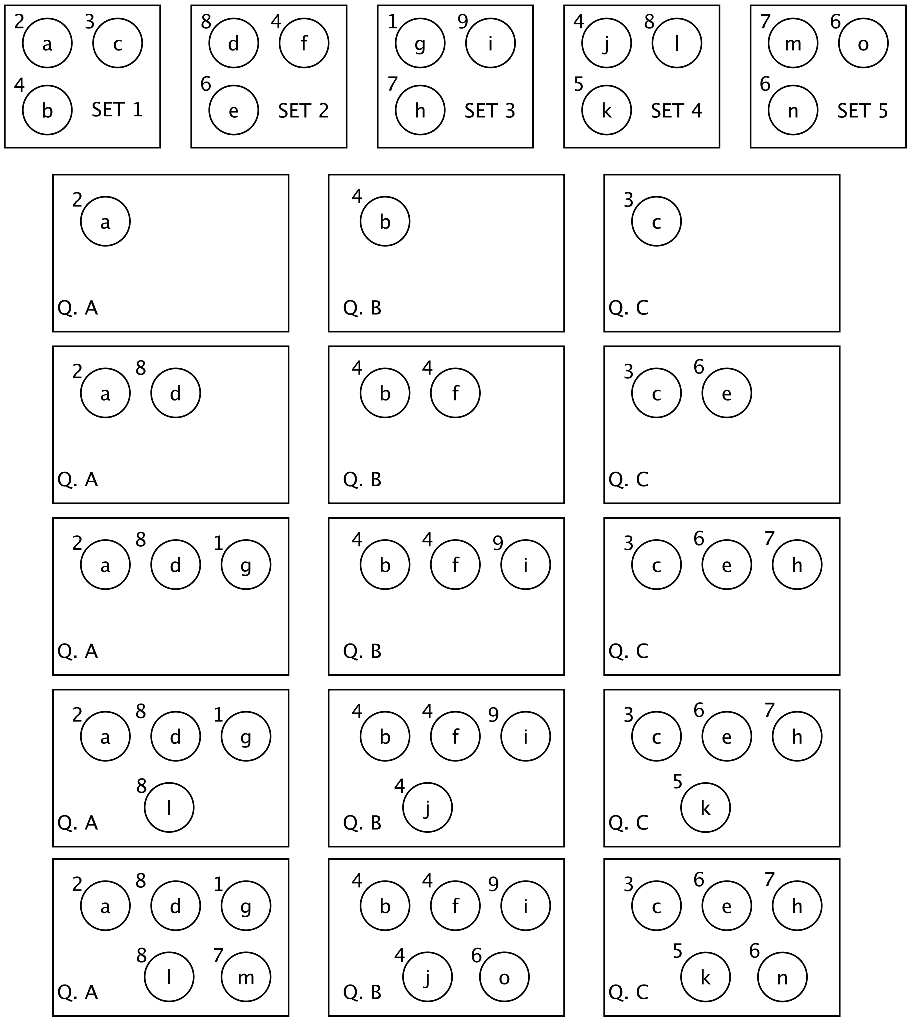

Figure 2: Illustration of the constructive heuristic. The instance is illustrated in the first row of the figure. First, set 1 is assigned to the questionnaires at random, as shown in the second row of the figure. Then, set 2 is assigned following the heuristic logic described as follows: the question with the lowest weight is assigned to the partially filled questionnaire with the highest weight; the question with the second lowest weight is assigned to the questionnaire with the second highest weight; etc. until all of the questions are assigned, as shown in the third row of the figure. Afterwards, the process is repeated for the following sets until all questions from all sets have been assigned. The last row in the figure shows the final solution.

questions with an additional set, which is equivalent to finding the optimal solution of a MINIMAX_BSC problem when $T = 2$ (see Subsection 2.2).

To analyze the computational complexity of the algorithm, first note that the computationally expensive part of the algorithm is ordering the questions and questionnaires. This step is executed T times, and each ordering has $O(B \cdot logB)$ complexity. Hence, the final computational complexity of the algorithm is $O(T \cdot B \cdot logB)$.

Claim 3. Algorithm 1 provides a solution with an absolute performance guarantee of R.

Proof. First, we show that the maximum difference between then accumulated weights of any two questionnaires is equal to R. To simplify the explanation, let d denote the maximum difference in the accumulated weights between two questionnaires.

Initially, d is set to 0. Accordingly, after the first assignment of questions to questionnaires, $R \geq d$ holds. During subsequent assignments, d is bounded by $\max\{d, r_t\}$ with r_t equal to the range of the assigned set (if $r_t \leq d$, then $d - r_t$ is bounded by d, and if $r_t \geq d$, then $r_t - d$ is bounded by r_t). Because both $d \leq R$ and $r_t \leq R$ hold, the difference between any two groups will always be smaller than R.

Second, note that for any valid solution, the minimum total weight of any questionnaire ($\min_{1 \leq b \leq B}\{Z_b\}$) is a lower bound on the optimal solution. Because R is a bound on the difference between any two questionnaires obtained using Algorithm 1, the difference between the solution provided by Algorithm 1 and a lower bound for the problem is also bounded by R. This construct proves the claim that Algorithm 1 provides a solution that has an absolute performance guarantee equal to R.$\square$

The solution offered by this constructive method depends on the order in which the sets are considered. A preliminary computational experiment showed that the experimental efficiency of Algorithm 1 is improved when the sets are assigned in non-increasing values of r_t order. Accordingly, the initial solution of the VNS uses this ordering.

3.2. Neighborhood structures

Neighborhood structures are the most important part of the definition of a successful VNS implementation. In this work, we propose three different neighborhood structures that try to improve the incumbent solution in different ways. The search in each of the neighborhoods is conducted by solving special cases of the problem (i.e., a MINIMAX_BSC with either $T = 2$ or $B = 2$) until the solution is locally optimal for all of the explored neighborhoods.

We now proceed to define each of the three neighborhoods:

Neighborhood 1. This neighborhood considers the reassignment of the questions of one set to the questionnaires.

This neighborhood corresponds to the exploration of all possible solutions that can be obtained by removing all the questions of a set from the incumbent solution and their reassignment to the questionnaires. For all of the sets, the following three steps are performed: (1) remove the questions that correspond to that set from the questionnaires; (2) explore the reassignment of the extracted questions to the partially filled questionnaires; and (3) assign the questions to the questionnaires according to the least expensive solution obtained in step (2).

Step (2) can be time consuming, as all possible permutations of the assignments of questions to questionnaires need to be considered in the worst case. Note that the weight of the questionnaires in the partial solution can be considered as a unified contribution to the total weight of the final questionnaires. Consequently, each partially filled questionnaire can be considered as a single question from a different set, and the optimal reassignment of the extracted questions can be obtained by solving the MINIMAX_BSC with $T = 2$ (see Subsection 2.2).

This neighborhood is implemented in a first descent fashion, and as such, the ordering of sets may modify the final solution. Therefore, to reduce the effect of a specific ordering, whenever neighborhood 1 is explored, a random order of sets is generated and the sets are considered in accordance with the proposed order. Algorithm 2 depicts the final algorithm.

Algorithm 2. Neighborhood structure 1 search.

```
Define a random order of the T sets
repeat until no further improvement is possible
  for each $t$ ($1 \leq t \leq T$) do
    Remove the questions from ordered set $t$ from the questionnaires
    Solve the MINIMAX_BSC problem with $T = 2$.
    if the new solution improves upon the incumbent solution then
      Update the incumbent solution
    end if
  end for
end repeat
```

An example of this neighborhood is depicted in Figure 3.

Note that the neighborhood considered in Brusco et al. (2013) can be seen as a special case of this neighborhood. Their neighborhood studies the reassignment of two questions from the same set to their respective questionnaires, which is a limited version of neighborhood 1. Therefore, the limited neighborhood is not used in the final VNS implementation.

Neighborhood 2. This neighborhood considers the division of the solution into two partial solutions that are then optimally combined to possibly form an improved solution.

Let T' and T'' be a partition of the sets in T. For any given solution S, let S' and S'' be the partial assignment of the questions to questionnaires defined in S for the questions of subsets T' and T'', respectively. Then, consider each questionnaire in S' and S'' as a single question (i.e., all questions assigned to a partial questionnaire are to be assigned together when the partial questionnaires are combined). Based on the previous definition, neighborhood 2 corresponds

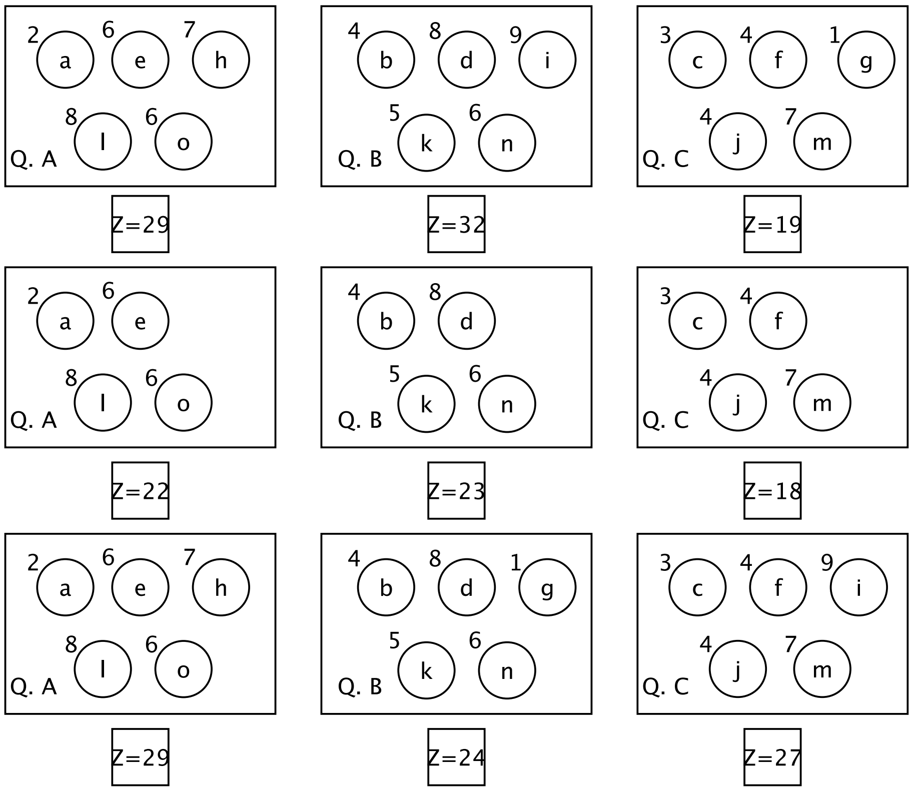

Figure 3: Example of the application of the neighborhood 1. The figure depicts an initial solution in the first row (taken from the example in Figure 1), from which the first neighbourhood is created. First, all of the questions that correspond to set 3 are removed from the solution (second row in the figure). Underneath the partial solution, the total weight of each questionnaire is depicted in a box (for example, the total weight for the partial questionnaire A is 22). Then, the questions corresponding to set 3 are optimally reassigned (third row) providing an improved solution.

to the exploration of all possible partitions of S into S' and S'', as well as all of their respective assignments combining S' and S'' to obtain a new solution.

The previous neighborhood definition has two issues that must be addressed to obtain an efficient implementation:

1. The number of partitions in the previous definition is not bounded by a polynomial function to the number of sets in the instance, and thus only a limited number of partitions can be considered. The implementation uses the following technique; each time neighborhood 2 is explored, we define a random order of the sets and we explore the $t-1$ ($1 \leq t < T$) partitions in which T' is composed of the t first sets of questions according to the random ordering, and T'' is composed of the remaining sets of questions.
2. Once T' and T'' have been defined, the optimal combination of the partial questionnaires in S' and S'' can be solved as a MINIMAX_BSC with $T = 2$, which is solvable in polynomial time (see Subsection 2.2). Algorithm 3 shows the final algorithm.

Algorithm 3. Neighborhood structure 2 search.

```
Define a random order of the T sets
repeat until no further improvement is possible
  for each $t$ ($1 \leq t \leq T$) do
    Obtain the MINIMAX_BSC problem with $T = 2$ with partitions $T'$ and $T''$
    Solve the MINIMAX_BSC problem with $T = 2$
    if the new solution improves upon the incumbent solution then
      Update the incumbent solution
    end if
  end for
end repeat
```

An example of an improving move using neighborhood 2 is depicted in Figure 4.

Note that neighborhood 1 is a special case of neighborhood 2 if multiple orderings are considered. The previous statement holds if at least one of the considered orderings contains each set as its first or last studied set. Then, the exploration of neighborhood 2 using all of these orderings subsume neighborhood 1 as the algorithm considers the reassignment of each individual set of questions in addition to the joint reassignment of sets of questions.

Neighborhood 3. This neighborhood explores the optimal reassignment of questions to two questionnaires.

While neighborhood 1 considers changes of different sets of questions and neighborhood 2 considers changes of different groups of sets of questions, neighborhood 3 studies the effect of changing the questions assigned to two questionnaires. Neighborhood 3 is thus defined as the exploration of all possible solutions in which the reassignment of questions to exactly two questionnaires may vary.

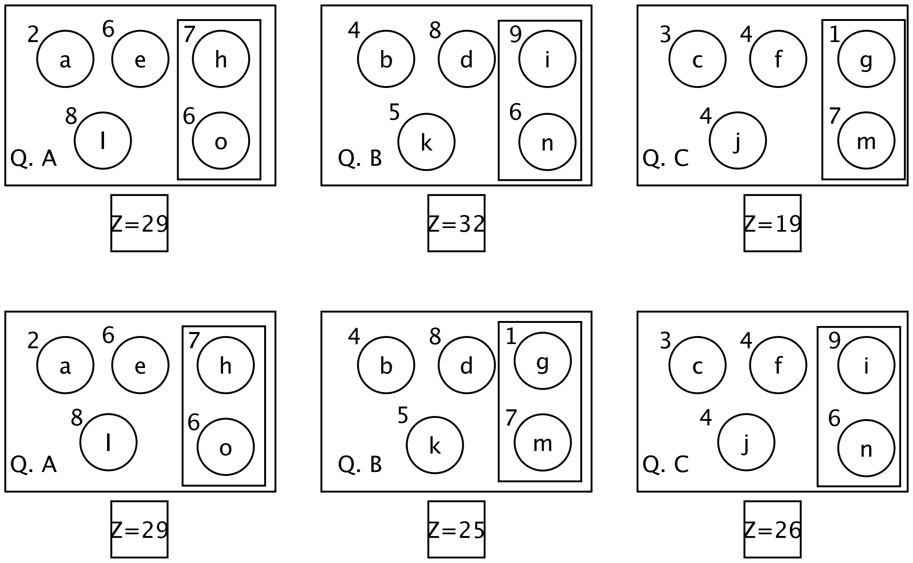

Figure 4: Example of the application of the neighborhood 2. The initial solution (first row, from the example in Figure 1) is divided by selecting two groups of sets (one is composed of sets 3 and 5 and is highlighted by boxing their respective questions; the other group corresponds to the remaining sets). The last row depicts the solution obtained after optimally solving the related MINIMAX_BSC problem with $T = 2$.

The previous definition naturally leads to multiple possible partial solutions, each requiring the exploration of a combinatorial problem (i.e., all possible assignments of questions to two questionnaires must be investigated). Consequently, an efficient exploration of the neighborhood must be based on some observations that may lead to an effective search for improving solutions. The following observations are used in the proposed neighborhood search method:

1. To improve the solution, some questions assigned to the questionnaire with the largest sum of weights must be exchanged.
2. The optimal assignment of two questionnaires can be found by solving a subset sum knapsack problem (see Section 2.3).

Combining both observations, we can establish that one of the questionnaires considered in the subset sum knapsack problem must be one of the questionnaires in which constraint (2) in the mathematical model is fulfilled in equality (i.e., the questionnaire with the largest weight, or one of them if several exist).

As in the previous neighborhoods, it is important to define the order in which the questionnaires are inspected. While the objective in the previous cases was to avoid a bias towards some sets, we opt for an ordering in this case that maximizes the probability of finding an improved solution; specifically, the questionnaires are ordered in non-decreasing order of the sum of weights. Then, for each questionnaire and in the order defined above, the algorithm solves a MINIMAX_BSC problem that combines the said and the last questionnaire in the list. Algorithm 4 illustrates this procedure.

Algorithm 4. Neighborhood structure 3 search.

```
Order the questionnaires in non-decreasing order to their respective weights
**for** each $r$ $(1 \leq r \leq B)$ **do**
  Solve the MINIMAX_BSC problem using the $r$-th and $B$-th questionnaires
  **if** the new solution improves upon the incumbent solution **then**
    Update the incumbent solution
  **end if**
**end for**
```

An example of neighborhood 3 is depicted in Figure 5.

3.3. Shaking procedure

The shaking procedure randomly chooses exchanges defined by one of the considered neighborhoods. In this case, we opt for the application of a neighborhood structure similar to neighborhood 1 with a variable parameter k that controls the distance from the current partial solution.

For any given k $(2 \leq k < T)$, the proposed procedure removes the questions associated to k distinct sets from the incumbent solution. Once these questions are removed, the questions of each set are introduced in the solution using the selection rule specified by neighborhood structure 1, which considers the partial

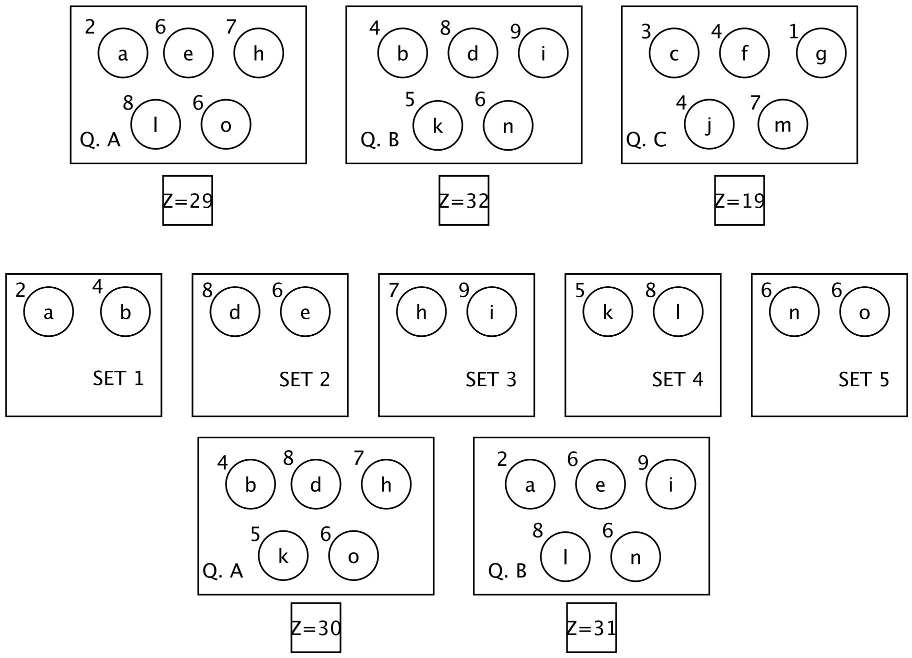

Figure 5: Example of the application of the neighborhood 3. The first row depicts an initial solution (from the example in Figure 1). A new instance of the problem is created by considering the questions assigned to questionnaires A and B; questionnaire B is selected because it is the questionnaire with the highest weight. The new instance is depicted in the second row and corresponds to an instance with 5 sets and 2 questions per set. The third row depicts the optimal reassignment of questions to questionnaires A and B that, with questionnaire C (which remains unchanged) provide an improved solution.

solution as a single set of questions and finds the optimal assignment of a new set of questions. Note that the consecutive introduction of multiple sets of questions can also be achieved using algorithm 1; thus, the shaking procedure can be described as the elimination of k sets from the solution and their reintroduction into the current solution using the constructive greedy heuristic; however, a discrepancy exists because the sets are reintroduced in random order rather than in the order proposed in Subsection 3.1. The shaking procedure highlights the relationship between the constructive procedure and the two first neighborhood structures, because all three structures are based on the optimal solution to a MINIMAX_BSC instance with $T = 2$.

The values of parameter k are controlled as follows: the initial value is set to $k = 2$ because $k = 1$ in the reintroduction scheme would return the solution to the same local optimum and thus it does not modify the solution. Whenever a local optimum for each of the three neighborhoods is reached, the shaking procedure is restarted, and k is increased by one. Whenever $k = T - 1$, the shaking procedure generates a random solution which can be considered as restarting the search; k is also restarted in this circumstance (i.e., k is set equal to 2).

3.4. Termination condition and outline of the complete algorithm

The termination condition is linked to a run time limit or the verification of an optimal solution (i.e., when the incumbent solution is equal to the lower bound defined in Subsection 2.2). The complete algorithm is depicted in Algorithm 5.

Algorithm 5. VNS method.

```
Find a heuristic solution using Algorithm 1, and initialize $k$.
Init:
repeat until termination condition is met
  Search neighborhood structure 1 until a locally optimal solution is found.
  Search neighborhood structure 2 until a locally optimal solution is found.
  If the solution was improved during the search, goto Init.
  Search neighborhood structure 3 until a locally optimal solution is found.
  If the solution was improved during the search, goto Init.
end repeat
```

Note that the application of the neighborhoods in the predefined order was decided based on run time considerations: the less-time-consuming algorithm is the first to be applied, while the most-time-consuming is the last to be applied.

4. Computational Experiments

The previous algorithm was implemented in C and compiled using gcc v.4.9.0 and run on computer with a 2.9 GHz Intel Core i5 processor and 8 GB of

RAM running the Mac OS X 10.9 operating system. To solve the subset sum knapsack problems, the expknap implementation (Pisinger, 1995) available at http://www.diku.dk/-pisinger/codes.html was used.

In addition to the algorithms described in this paper, we also implemented the SA algorithm proposed in Brusco et al. (2013). We did not report results for the IP formulation because Brusco et al. (2013) have already verified that this formulation is inefficient for large size instances.

The parameters of the SA were identical to those proposed in Brusco et al. (2013). Furthermore, if the SA terminates before the specified running time limit, the algorithm is restarted with a different randomly generated initial solution. This VNS implementation is parameter-free, and thus no parameter tuning was required. Both algorithms were run for at most 600 s, which was measured as wall time.

The two algorithms were initially tested on a set of instances constructed by Brusco et al. (2013). The characteristics of both sets correspond to those found in previously simulated data found in the literature (Veldkamp, 2005; van der Linder, 2005; van der Linden & Boekkooi-Timminga, 1988).

Instances with varying number of questions (e.g., 300, 600, 3000 and 6000 questions) were generated. These questions were divided into different number of questionnaires (e.g., $B = 2$, 3, 4, 5, 10, 20, 30, 60, 120, 300, 600 and 1200 questionnaires). For each number of questions, the abovementioned number of questionnaires is only used if the number of questions is at least five times bigger than the number of questionnaires (i.e., each questionnaire will be formed by five or more questions). Finally, the weights of the questions in each instance were generated using the following method, which was proposed in Brusco et al. (2013), and based on van der Linden & Boekkooi-Timminga (1988). The method uses formula (9) to compute the weight of each question based on two parameters π_{rt} and ρ_{rt}, which represent the difficulty (i.e., percentage of examinees who obtained the correct answer) and the discrimination effect of the question, respectively.

$$w_{rt} = .5 \cdot \pi_{rt} + .5 \cdot \pi_{rt} \cdot (1 - \pi_{rt}) \cdot \rho_{rt} \tag{9}$$

To generate sets of instances with similar weights, the method initially obtains the difficulty π_{1t} and discrimination effect ρ_{1t} from the first question of each set t. These values are drawn from a uniform distribution with interval $[.3, .8]$ for the difficulty effect (π) and from the interval $[.25, .60]$ for the discrimination effect (ρ). For the remaining questions π_{rt} and ρ_{rt} ($2 \leq r \leq B$), the values are randomly drawn from uniform distributions with interval $[\pi_{1t} - .1, \pi_{1t} + .1]$ and $[\rho_{1t} - .1, \rho_{1t} + .1]$. These weights are then used in combination with (9) to generate the final weights.

Ten instances per combination of number of questions, questionnaires and distribution of weights were constructed to create a total of 400 instances. Note that while the weights were obtained from uniform distributions and thus are real values, the method works with integer-valued weights to avoid rounding issues. To achieve this objective, we decided to multiply the obtained values by a constant (10^6) and round to the nearest integer. Consequently, this algorithm is working using real values with six digits of precision.

Table 1 provides a summary of the experimental results. For each number of questions and questionnaires, the table provides a comparison between the heuristic method depicted in Subsection 3.1, the VNS method proposed in this paper and the SA algorithm proposed by Brusco et al. (2013). For each algorithm, we provide the average absolute gap between the lower bound (LB), calculated using equation (6), and the upper bound (UB) provided by the algorithm (that is, $UB - LB$). For the metaheuristic approaches, the number of optimal solutions (out of 10 instances) and the running time required to reach the best solution (measured in wall time) are also provided.

An analysis of the table shows that the gap between the solutions provided by both metaheuristics and the lower bound are small in all cases. Furthermore, both methods are able to dramatically improve the initial solutions provided by the heuristic with a performance guarantee.

Furthermore, the VNS was able to obtain the optimal solution for most of the instances, including all of the instances in which the ratio between the questions and questionnaires was higher than 15 within negligible running times (i.e., within one second). Hence, we can conclude that those instances in which the ratio of questions per questionnaire is larger than or equal to 15 are to be considered easy for the proposed method. The running time increased for the instances in which the algorithm was not able to verify optimality, but were still reduced.

Note that table 1 reports the running time to reach the best solution for all methods rather than the total computing time required. According to the termination condition, if the optimal solution is not verified, both algorithms continue until the time limit is reached.

Based on the results provided above, we test the algorithm with larger instances (e.g., with T=30.000 and 60.000, and different values of B ranging from 2 to 12.000). Similar results to those reported in table 1 were obtained, and the VNS method was able to optimally solve all instances with a ratio of questions to questionnaires larger than 10 within a maximum of 5 seconds of computing time. Therefore, we limit this report to the results for those groups of instances in which the VNS fails to verify the optimal solution for every instance with the given ratio of questions per questionnaire ("challenging" instances). Table 2 reports these results.

The results from table 2 show that the algorithm is still effective for larger-sized instances. Furthermore, the number of verified optimal solutions increases due to the existence of more alternatives to assign the questions. Consequently, we conclude that for the VNS method, increasing the number of questions does not affect the quality of the solutions, while the running times increase in accordance with the requirement of solving larger problems, specifically larger knapsack instances.

The results provided in tables 1 and 2 indicate that the proposed VNS offers high quality results, but do not identify which components of the algorithm are responsible for the good performance. Therefore, an additional experiment

Table 1: Results of the proposed algorithms. For each combination of characteristics, we provide the number of optimal solutions verified by each of the metaheuristic algorithms, the average absolute gap (i.e., the average difference between the solution and the lower bound, provided by equation 6) for each algorithm and the average time required to reach the best solution by the metaheuristic methods; for the SA algorithm, we report the time required for replication that provided the solution under consideration.

Questions	B	Heuristic Gap	VNS Opt	VNS Gap	VNS t	SA Opt	SA Gap	SA t
300	2	152.8	10	0	0.0	10	0	0.0
300	3	2092.4	10	0	0.0	5	0.7	0.0
300	4	3990	10	0	0.0	0	11.4	0.1
300	5	5184.9	10	0	0.0	0	25.6	0.1
300	10	10424.8	10	0	0.0	0	107.6	0.1
300	20	14789.5	9	0.1	0.1	0	164.4	0.0
300	30	14425	0	9	0.1	0	237.3	0.0
300	60	26446.1	0	201.7	0.0	0	368.7	0.0
600	2	23.9	10	0	0.0	10	0	0.0
600	3	1253.9	10	0	0.0	4	0.6	0.2
600	4	2598.8	10	0	0.0	0	4.4	0.2
600	5	4292.7	10	0	0.0	0	12.9	0.2
600	10	12199.9	10	0	0.0	0	50.6	0.1
600	20	14419.5	10	0	0.0	0	85.8	0.1
600	30	17964.2	10	0	0.0	0	116.6	0.1
600	60	13006.9	0	6.6	0.2	0	183.4	0.1
600	120	28216.4	0	133.9	0.1	0	290.3	0.0
3000	2	14.3	10	0	0.0	10	0	0.0
3000	3	453	10	0	0.0	10	0	1.4
3000	4	2093.4	10	0	0.0	5	0.6	1.2
3000	5	2614.6	10	0	0.0	0	2.1	1.0
3000	10	9293.1	10	0	0.0	0	10.2	0.6
3000	20	13998	10	0	0.0	0	19.5	0.4
3000	30	15172.7	10	0	0.0	0	23.9	0.3
3000	60	13931.3	10	0	0.0	0	35.5	0.2
3000	120	17909.3	10	0	0.1	0	58.5	0.2
3000	300	5742.3	0	2.5	1.1	0	120	0.1
3000	600	30084.8	0	42.2	0.6	0	174.9	0.2
6000	2	3.7	10	0	0.0	10	0	0.0
6000	3	252.6	10	0	0.0	10	0	2.7
6000	4	1297.2	10	0	0.0	8	0.2	2.5
6000	5	2510.4	10	0	0.0	4	0.8	2.1
6000	10	8054.3	10	0	0.0	0	5.2	1.3
6000	20	10717.1	10	0	0.0	0	9.1	0.8
6000	30	14209.7	10	0	0.0	0	11.1	0.6
6000	60	13686.2	10	0	0.0	0	17.3	0.5
6000	120	11105.7	10	0	0.0	0	29.8	0.3
6000	300	7902.1	10	0	0.1	0	59.7	0.3
6000	600	5324.9	0	1.6	0.9	0	90.3	0.4
6000	1200	29043.9	0	25.7	1.1	0	290.9	0.4

Table 2: Results for larger instances. Only those combinations of the number of questions and questionnaires that are deemed to be ”challenging” are reported. For each combination of characteristics, we provide the same results that in table 1.

Questions	B	Heuristic Gap	VNS Opt	VNS Gap	VNS t	SA Opt	SA Gap	SA t
30000	3000	4208.8	6	4	60.8	0	351.1	2.6
30000	6000	24234.8	0	11.8	10.3	0	2155	2.4
60000	6000	12963.3	3	0.7	24.6	0	1055.3	4.8
60000	12000	38837.5	0	4	48.3	0	2757	5.7

geared towards evaluating the contribution of each of the specific components was conducted.

The experiment considers different configurations of the VNS algorithm in which some of the neighborhoods or the shaking procedure are not used. While specific neighborhood can be removed from the algorithm without modifying its global structure, it is necessary to include some kind of restarting procedure to escape from local optimality, continue the search and effectively use the allotted computing time. Consequently, when the shaking procedure is not used, it is substituted by the greedy initialization procedure proposed in Subsection 3.1 in which the order of the sets is randomly determined before executing algorithm 1.

A total of 14 different configurations of the VNS algorithm were tested (all possible combinations of use/not use for each of the three neighborhoods and the shaking procedure, with the exception of the two combinations that do not use any of the three neighborhoods) on the large and “challenging instances (the 40 instances reported in table 2). Each configuration was run for at most 600 seconds of wall time, and three values were recorded: (1) the running time required to reach the best solution; (2) the absolute gap between the solution found by the configuration and the best known solution for the instance; and (3) the absolute gap between the solution found by the configuration and the theoretical bin packing lower bound. Furthermore, and in order to verify if the restarting procedure was efficient, we also run the procedure as a simple descent method with all of the neighborhood structures and recorded the same information.

Table 3 summarizes the results obtained for the experiment. The examination of table 3 highlights the effectiveness of neighborhood 3 (gaps are much smaller if neighborhood 3 is used). Note that the gap reduction is accompanied by larger computational times. The shaking procedure, as well as neighborhood 1 and neighborhood 2 do not affect the solution quality but they do seem to reduce the running time required to reach the best solution. Also note that the gap difference among the VNS configurations using neighborhood 3 is minimal.

If we compare the single descent method with the VNS algorithm, the results from table 3 show that the VNS is capable of providing improvements on the solution provided by the simple descent method, at the cost of larger computational times. Please note that the gap reduction is small as the solu-

Table 3: Results for different configurations of the proposed algorithms. For each combination of use/not use of each of three neighborhoods (columns N1, N2 and N3) and shaking procedure (column "Shake"), the average absolute gap between the solution found by the configuration and the theoretical bin packing lower bound (column Gap$_{bound}$); the average absolute gap between the solution found by the configuration and the best known solution for the instance (column Gap$_{best}$); and the average running time required to reach the best solution (column t) are reported. Additionally, the results of the simple descent algorithm (no restart used) are also reported (row with "Shake" entry equal to "Descent").

N1	N2	N3	Shake	Gap$_{bound}$	Gap$_{best}$	t
No	No	Yes	No	4.4	0.5	36.3
No	No	Yes	Yes	4.4	0.6	39.5
No	Yes	No	No	17.8	13.9	26.5
No	Yes	No	Yes	26.6	22.7	16.1
No	Yes	Yes	No	4.2	0.3	31.7
No	Yes	Yes	Yes	4.3	0.4	34.0
Yes	No	No	No	25.3	21.5	6.0
Yes	No	No	Yes	16.7	12.8	2.8
Yes	No	Yes	No	4.2	0.3	32.5
Yes	No	Yes	Yes	4.3	0.4	22.7
Yes	Yes	No	No	10.4	6.5	22.6
Yes	Yes	No	Yes	12.3	8.4	21.1
Yes	Yes	Yes	No	4.2	0.3	34.9
Yes	Yes	Yes	Yes	4.0	0.1	21.6
Yes	Yes	Yes	Decent	6.6	2.7	10.0

tions provided by the descent heuristic are already very good and show close to nonexistent optimality gaps.

In addition to the previous qualitative analysis, we conducted an ANOVA test on two response variables (Gap$_{best}$ and running time) and a MANOVA test using both response variables (see Christensen, 2011, for a general reference on ANOVA and MANOVA). The tests reported differences among VNS configurations, but an analysis of the residuals showed severe violations of the normality assumptions and heteroscedasticity issues. As a consequence, we chose to reexamine the results using a non-parametric MANOVA alternative.

The permutational MANOVA using distance matrices proposed in Anderson (2001) was selected. The method does not require traditional assumptions; it is based on a geometric interpretation of the analysis of variance; and it uses permutations in order to obtain p-values. Furthermore, the code is readily available on the R statistical package (Oksanen et al., 2015). The test was run using four factors (the three neighborhood searches, and the shaking procedure) and a blocking factor (the instances) that tries to block the variability caused by specific characteristics of each instance.

The results show that neighborhood structures 1 and 3 are significant with a p-value < 0.001, and the shaking procedure is significant with a p-value < 0.01. Neighborhood structure 2 is not significant, leading us to conclude that it could be removed from the algorithms without modifying its results (both in terms of running time and solution quality).

If we combine the p-values with the analysis of table 3, we conclude that the different components of the proposed VNS method are able to improve the solution obtained by the final algorithm (neighborhood structure 3); to reduce the required running time to reach the aforementioned solution(neighborhood structure 1 and shaking procedure); or do not significatively modify running time or solution quality.

A final experiment was performed to evaluate the behavior of the proposed method if the sets were based on some other constraints (e.g., length, time required to answer the question) and not on similar objective function weights. A new instance set in which the weights of the questions are uniformly distributed is considered, see for example Nguyen & Fong (2013). For these instances, each weight is randomly selected form a uniform distribution on the interval $[.1, .9]$ maintaining the remaining characteristics from the instances.

The results, see table 4, show that the performance of the VNS method is significantly better than its SA counterpart. Hence the VNS depends less on a grouping of questions by similar weights. The relation between the number of questions and questionnaires to ascertain the difficulty of the instance still holds. However, the instances are harder to solve for both metaheuristics, and larger absolute gaps appear for those instances in which optimality is not verified (even if they are still small). Note that the heuristic constructive method is clearly affected by changes in structure and generates significantly worse initial solutions. As similar results in terms of optimally solved instances are obtained, we conclude that the distribution of weights among the questions in each set does not greatly affect the performance of the VNS method.

Table 4: Results for the instances generated according to a uniform distribution as in Nguyen & Fong (2013). For each combination of characteristics, we provide the number of optimal solutions verified by each algorithm; the average absolute gap, which is measured as the average difference between the solution and the lower bound using equation 6; and the average time required to reach the best solution; for the SA algorithm, we report the time required for replication that provided the solution under consideration.

Questions	B	Heuristic	VNS	VNS	VNS	SA	SA	SA
		Gap	Opt	Gap	t	Opt	Gap	t
300	2	761.3	10	0	0.0	10	0	0.0
300	3	11312.1	10	0	0.0	0	17.5	0.0
300	4	33248.2	10	0	0.0	0	94.4	0.1
300	5	52034.2	10	0	0.0	0	263.6	0.1
300	10	121717.5	10	0	0.0	0	803.7	0.0
300	20	151533.7	0	2.7	0.1	0	1303.5	0.0
300	30	152080.8	0	77.6	0.1	0	1863.4	0.0
300	60	276051.8	0	1715.4	0.0	0	2853.1	0.0
600	2	227.1	10	0	0.0	10	0	0.0
600	3	7769.4	10	0	0.0	0	6.8	0.2
600	4	24895.5	10	0	0.0	0	42.7	0.2
600	5	41638.3	10	0	0.0	0	117.8	0.2
600	10	133000	10	0	0.1	0	355.5	0.1
600	20	151155.8	10	0	0.1	0	710.3	0.1
600	30	198745.9	10	0	3.2	0	873.9	0.1
600	60	137782.5	0	55.6	0.4	0	1441.3	0.0
600	120	318178	0	1090.6	0.1	0	2331.8	0.0
3000	2	156.3	10	0	0.0	10	0	0.0
3000	3	3727	10	0	0.0	3	0.7	1.6
3000	4	15952.3	10	0	0.0	0	8.7	1.2
3000	5	25568.9	10	0	0.0	0	23.9	1
3000	10	92353.9	10	0	0.0	0	77.1	0.6
3000	20	138538.2	10	0	0.1	0	141.1	0.4
3000	30	162414.9	10	0	0.2	0	182.9	0.3
3000	60	179236.1	10	0	0.3	0	287	0.2
3000	120	217096.8	10	0	0.5	0	442.7	0.2
3000	300	76249	0	20	3.8	0	987.5	0.1
3000	600	365729	0	358.5	0.2	0	1325.4	0.2
6000	2	29.5	10	0	0.0	10	0	0.1
6000	3	2497.9	10	0	0.0	6	0.4	3.6
6000	4	10685.3	10	0	0.0	0	5.2	2.6
6000	5	31974.9	10	0	0.0	0	11.4	2.1
6000	10	72561.1	10	0	0.1	0	40.2	1.3
6000	20	125779.9	10	0	0.1	0	70.2	0.8
6000	30	175586.3	10	0	0.1	0	91.7	0.6
6000	60	192118.3	10	0	0.3	0	134.7	0.4
6000	120	177719.9	10	0	0.5	0	229.5	0.3
6000	300	108666	10	0	1	0	482.5	0.3
6000	600	53802.5	0	11.4	7.9	0	667.9	0.4
6000	1200	366379.5	0	209.3	0.8	0	2186	0.5

5. Conclusions

In this paper, we have investigated a Bin Packing-based formulation of the test assembly design problem known as the one-dimensional minimax bin-packing problem with bin size constraints (MINMAX_BSC). We first show that the problem in NP-Hard when the number of questionnaires is 2 and strongly NP-Hard if more questionnaires are considered. Second, we develop a constructive heuristic with an absolute performance guarantee, and we derive three neighborhood structures for the problem that we can explore by optimally solving special cases of the problem. Third, we combine these procedures into a variable neighborhood search in order to efficiently search the solution space. Finally, the quality of the proposed procedures is tested in computational experiments on instances with up to 60.000 item pools, up to 12.000 questionnaires, and different discrimination effect structures (see Brusco et al., 2013; Nguyen & Fong, 2013). The proposed method shows significant improvements over previous algorithms proposed in the literature (Brusco et al., 2013), and the optimality of the solutions provided for instances with a 60.000 items pool and up to 3.000 tests has been verified.

Our complexity results follow the line of previous works (like Belov & Armstrong, 2006; Nguyen & Fong, 2013; Ishii et al., 2014) in which other formulations of the test assembly design problem were identified as classical strongly NP-hard combinatorial optimization problems. Our approach differs when we consider the resolution considerations, as Belov & Armstrong (2006); Nguyen & Fong (2013); Ishii et al. (2014) apply classical algorithms of the identified optimization problem to the resolution of the test design problem, while our Bin Packing-based formulation significantly differs from the classical problem, thus forcing us to develop specially tailored procedures for its resolution.

The proposed algorithm corresponds to a neighborhood exploration heuristic. If compared to previous neighborhood-based approaches for test assembly design (Hwang et al., 2008; Brusco et al., 2013), which are based on traditional swap or exchange neighborhoods, our neighborhoods are based on the optimal exploration of special structures. These neighborhoods are derived from theoretical results for the MINMAX_BSC, and successfully use the structure of the problem to reach high quality results within very reduced running times.

The quality of the solutions provided by the method is demonstrated by our experimental results with instances larger than those previously considered in the literature (see Hwang et al., 2008; Nguyen & Fong, 2013). Please note that the results are not directly comparable as there are strong differences among different formulations for the test assembly design problem. Nevertheless, we can make comparisons with a Bin Packing lower bound, which shows the high quality of the solutions found. The proposed method routinely finds the optimal solution when the ratio between items per questionnaire is above 10, and obtains very low gaps if the aforementioned ratio is 10 or below. These results apply under varying conditions, such as when the items are grouped according to similar weights, as in Brusco et al. (2013), or when the items are randomly grouped. Results for randomly grouped items consider cases in which additional constraints are to be enforced (He et al., 2014; Ishii et al., 2014; Nguyen & Fong, 2013), or cases in which a cell approach (Chen et al., 2012; Chen, 2015) is used to maintain similar characteristics not only among questionnaires, but also among the item composition of each questionnaire.

In addition to the quality of the results, we would like to highlight the reduced running times required to solve the instances, which are below the 60 seconds mark even for the largest item pools. These results lead us to conjecture that the proposed procedure could also be used for on-line test construction methods as in van der Linden & Veldkamp (2004) or He et al. (2014).

The possibility of efficiently solving large-sized instances leads us to conclude that any further research should address the inclusion of additional characteristics into the formulation. Among these characteristics, we would like to highlight the joint evaluation of multiple ability levels of the recipients on a single unified test (van der Linder, 2005; Chang & Shiu, 2012; Nguyen & Fong, 2013) or the inclusion of additional constraints, such as incompatibilities among questions of different sets or limiting the number of questions with specific attributes (Hwang et al., 2008; Yao, 2014; Yao et al., 2014). These additional characteristics are usually present in many real situations and would impose challenging changes to both the bin-packing-based formulation as well as to the neighborhood structures proposed in this work.

References

Anderson, M. J. (2001). A new method for non-parametric multivariate analysis of variance. *Austral Ecology*, *26*, 32–46.

Belov, D. I. (2008). Uniform test assembly. *Psychometrika*, *73*, 21–38. doi:10.1007/s11336-007-9025-0.

Belov, D. I., & Armstrong, R. D. (2006). A constraint programming approach to extract the maximum number of non-overlapping test forms. *Computational Optimization and Applications*, *33*, 319–332. doi:10.1007/s10589-005-3058-z.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinees ability. In F. Lord, & M. Novick (Eds.), *Statistical theories of mental test scores* chapter Some laten. (pp. 385–479). Reading, MA: Addison-Wesley.

Brandão, F., & Pedroso, J. a. P. (2014). Fast pattern-based algorithms for cutting stock. *Computers and Operations Research*, *48*, 69–80. URL: http://dx.doi.org/10.1016/j.cor.2014.03.003. doi:10.1016/j.cor.2014.03.003.

Brusco, M. J., Köhn, H. F., & Steinley, D. (2013). Exact and approximate methods for a one-dimensional minimax bin-packing problem. *Annals of Operations Research*, *206*, 611–626. URL: http://link.springer.com/10.1007/s10479-012-1175-5. doi:10.1007/s10479-012-1175-5.

Chang, T. Y., & Ke, Y. R. (2013). A personalized e-course composition based on a genetic algorithm with forcing legality in an adaptive learning system. *Journal of Network and Computer Applications*, *36*, 533–542. URL: http://dx.doi.org/10.1016/j.jnca.2012.04.002. doi:10.1016/j.jnca.2012.04.002.

Chang, T.-Y., & Shiu, Y.-F. (2012). Simultaneously construct IRT-based parallel tests based on an adapted CLONALG algorithm. *Applied Intelligence*, *36*, 979–994. URL: http://link.springer.com/10.1007/s10489-011-0308-x. doi:10.1007/s10489-011-0308-x.

Chen, P.-h. (2015). A sampling and classification item selection approach with content balancing. *Behavior research methods*, . doi:10.3758/s13428-014-0452-4.

Chen, P.-H., Chang, H.-H., & Wu, H. (2012). Item Selection for the Development of Parallel Forms From an IRT-Based Seed Test Using a Sampling and Classification Approach. *Educational and Psychological Measurement*, *72*, 933–953. doi:10.1177/0013164412443688.

Christensen, R. (2011). *Plane Asnwers to complex questions. The theory of linear models*. (4th ed.). Springer.

Dell' Amico, M., & Martello, S. (1995). Optimal Scheduling of tasks on identical parallel processors. *ORSA Journal of Computing*, *2*, 191–200. doi:10.1287/ijoc.7.2.191.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, *44*, 145–159. URL: http://www.sciencedirect.com/science/article/pii/037722179090350K. doi:10.1016/0377-2217(90)90350-K.

Edmonds, J., & Armstrong, R. (2009). A mixed integer programming model for multiple stage adaptive testing. *European Journal of Operational Research*, *193*, 342–350. URL: http://linkinghub.elsevier.com/retrieve/pii/S0377221707010752. doi:10.1016/j.ejor.2007.10.047.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, *2*, 5–30. URL: http://link.springer.com/10.1007/BF00226291. doi:10.1007/BF00226291.

Fleszar, K., & Hindi, K. S. (2002). New heuristics for one-dimensional bin-packing. *Computers & Operation Research*, *29*, 821–839. doi:10.1016/S0305-0548(00)00082-4.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability. A guide to the theory of NP-Completeness*. New York: Freeman.

Hansen, P., Mladenović, N., & Moreno Pérez, J. A. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, *175*, 367–407. URL: `http://link.springer.com/10.1007/s10479-009-0657-6`. doi:`10.1007/s10479-009-0657-6`.

He, W., Diao, Q., & Hauser, C. (2014). A Comparison of Four Item-Selection Methods for Severely Constrained CATs. *Educational and Psychological Measurement*, *74*, 677–696. URL: `http://epm.sagepub.com/cgi/doi/10.1177/0013164413517503`. doi:`10.1177/0013164413517503`.

Hwang, G. J., Chu, H. C., Yin, P. Y., & Lin, J. Y. (2008). An innovative parallel test sheet composition approach to meet multiple assessment criteria for national tests. *Computers and Education*, *51*, 1058–1072. doi:`10.1016/j.compedu.2007.10.006`.

Ishii, T., Songmuang, P., & Ueno, M. (2014). Maximum Clique Algorithm and Its Approximation for Uniform Test Form Assembly. *IEEE Transactions on Learning Technologies*, *7*, 83–95. URL: `http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6701350`. doi:`10.1109/TLT.2013.2297694`.

Kallrath, J., Rebennack, S., Kallrath, J., & Kusche, R. (2014). Solving real-world cutting stock-problems in the paper industry: Mathematical approaches, experience and challenges. *European Journal of Operational Research*, *238*, 374–389. URL: `http://dx.doi.org/10.1016/j.ejor.2014.03.027`. doi:`10.1016/j.ejor.2014.03.027`.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack Problems*. Springer.

van der Linden, W. J. (1998). Optimal Assembly of Psychological and Educational Tests. *Applied Psychological Measurement*, *22*, 195–211. URL: `http://apm.sagepub.com/content/22/3/195.abstract`. doi:`10.1177/01466216980223001`.

van der Linden, W. J., & Boekkooi-Timminga, E. (1988). A Zero-One Programming Approach to Guiliksen's Matched Random Subtests Method. *Applied Psychological Measurement*, *12*, 201–209. URL: `http://apm.sagepub.com/content/12/2/201.abstract`. doi:`10.1177/014662168801200210`.

van der Linden, W. J., & Veldkamp, B. P. (2004). Constraining Item Exposure in Computerized Adaptive Testing With Shadow Tests. *Journal of Educational and Behavioral Statistics*, *29*, 273–291. doi:`10.3102/10769986029003273`.

van der Linder, W. J. (2005). *Linear Models for Optimal Test Design*. Springer.

López-Camacho, E., Terashima-Marin, H., Ross, P., & Ochoa, G. (2014). A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, *41*, 6876–6889. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0957417414002668`. doi:`10.1016/j.eswa.2014.04.043`.

Lu, H.-y. (2014). Application of Optimal Designs to Item Calibration. *Plos One*, *9*, 1–8. doi:10.1371/journal.pone.0106747.

M'Hallah, R., Alkandari, A., & Mladenovic, N. (2013). Packing unit spheres into the smallest sphere using VNS and NLP. *Computers and Operations Research*, *40*, 603–615. URL: http://dx.doi.org/10.1016/j.cor.2012.08.019. doi:10.1016/j.cor.2012.08.019.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*, 1097–1100. URL: http://www.sciencedirect.com/science/article/pii/S0305054897000312. doi:10.1016/S0305-0548(97)00031-2.

Nguyen, M. L., & Fong, A. C. M. (2013). Large-Scale Multiobjective Static Test Generation for Web-Based Testing with Integer Programming. *IEEE Transactions on Learning Technologies*, *6*, 46–59. URL: http://www.computer.org/csdl/trans/lt/2013/01/tlt2013010046.html. doi:10.1109/TLT.2012.22.

Oksanen, J., Blanchet, F. G., Kindt, R., Legendre, P., Minchin, P. R., O'Hara, R. B., Simpson, G. L., Solymos, P., M., H., H., S., & Wagner, H. (2015). vegan: Community Ecology Package. URL: http://cran.r-project.org/package=vegan.

Pisinger, D. (1995). An expanding-core algorithm for the exact 0-1 knapsack problem. *European Journal of Operational Research*, *87*, 175–187. URL: http://www.sciencedirect.com/science/article/pii/0377221794000133. doi:10.1016/0377-2217(94)00013-3.

Porter, A., Polikoff, M. S., Barghaus, K. M., & Yang, R. (2013). Constructing Aligned Assessments Using Automated Test Construction. *Educational Researcher*, *42*, 415–423. URL: http://edr.sagepub.com/content/42/8/415.abstract. doi:10.3102/0013189X13503038.

Quiroz-castellanos, M., Cruz-reyes, L., Torres-jimenez, J., S, C. G., Fraire, H. J., & Alvim, A. C. F. (2015). A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operation Research*, *55*, 52–64. URL: http://dx.doi.org/10.1016/j.cor.2014.10.010. doi:10.1016/j.cor.2014.10.010.

Smits, N., & Finkelman, M. D. (2014). Variable length testing using the ordinal regression model. *Statistics in Medicine*, *33*, 488–499. doi:10.1002/sim.5936.

Songmuang, P., & Ueno, M. (2011). Bees Algorithm for Construction of Multiple Test Forms in E-Testing. *IEEE Transactions on Learning Technologies*, *4*, 209–221. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5560630. doi:10.1109/TLT.2010.29.

Sun, K.-T., Chen, Y.-J., Tsai, S.-Y., & Cheng, C.-F. (2008). Creating IRT-Based Parallel Test Forms Using the Genetic Algorithm Method. *Applied Measurement in Education*, *21*, 141–161. URL: http://www.tandfonline.com/doi/abs/10.1080/08957340801926151. doi:10.1080/08957340801926151.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, *86*, 565–594. doi:10.1007/s101079900096.

Veldkamp, B. P. (2005). *Encyclopedia of Social Measurement*. Elsevier. URL: http://www.sciencedirect.com/science/article/pii/B0123693985004473. doi:10.1016/B0-12-369398-5/00447-3.

Veldkamp, B. P. (2013). Application of robust optimization to automated test assembly. *Annals of Operations Research*, *206*, 595–610. URL: http://link.springer.com/10.1007/s10479-012-1218-y. doi:10.1007/s10479-012-1218-y.

Wäscher, G., Hauß ner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, *183*, 1109–1130. URL: http://www.sciencedirect.com/science/article/pii/S037722170600292X. doi:10.1016/j.ejor.2005.12.047.

Yao, L. (2014). Multidimensional CAT Item Selection Methods for Domain Scores and Composite Scores With Item Exposure Control and Content Constraints. *Journal of Educational Measurement*, *51*, 18–38. doi:10.1111/jedm.12032.

Yao, L., Pommerich, M., & Segall, D. O. (2014). Using Multidimensional CAT to Administer a Short , Yet Precise , Screening Test. *Applied Psychological Measurement*, *38*, 614–631. doi:10.1177/0146621614541514.