Creating similar item sets using integer linear programming with an application to questionnaire design in experimental psychology

Martin Papenberg

Bachelor thesis

Department of Computer Science

Algorithmic Bioinformatics

Submission:	25.10.2018
Supervisor:	Prof. Dr. Gunnar Klau
Second Assessor:	Prof. Dr. Stefan Conrad
Advisor:	Eline van Mantgem

Declaration

I hereby confirm that this thesis is my own work. I have documented all sources and tools used. Any direct or indirect quote has been marked as such clearly with specification of the source.

Düsseldorf, Oktober 25, 2018 Martin Papenberg

Contents

Abstract

A method is presented to partition n items into p subsets with the aim of creating item sets that are as similar as possible. A measure of set similarity is introduced that is based on the pairwise distances between items. Pairwise item distance can be computed as the Euclidean distance of numeric item features. It is shown that the similarity of item sets can be maximized using an integer linear programming (ILP) formulation that was developed in the context of clustering research and has been used to solve the weighted cluster editing problem optimally. In this approach, establishing similar sets of items is the direct opposite of cluster editing and only differs in the direction of the optimization (minimization versus maximization). An implementation in the programming language `R` is presented that generates the ILP and calls one of three solvers to obtain the optimal assignment. Initial tests on random data show that the quality of set assignment is high; item sets obtain very similar feature distributions. However, the running times of the ILP solvers do not allow to process moderately large instances. To improve running time, a heuristic approach is proposed. In the heuristic, elements are first partitioned into $\frac{n}{p}$ homogeneous clusters. Elements of each cluster are forbidden to be assigned to the same set, which improves the running time of the ILP solvers considerably. Because only similar items are forbidden to be part of the same set, this approach preserves a high quality of the solution. In most cases, the heuristic identifies the optimal solution and, on average, the deviation from the optimal objective is less than 0.1% whenever the optimal solution has not been found. Two case studies confirm that both the exact and the heuristic approach are well-suited to establish similar sets of items in real applications. The proposed methods can therefore be applied to solve practical problems in the fields of experimental psychology and psychometrics, among others.

1 Introduction

Common research designs in experimental psychology involve the presentation a set of items to research participants who have to respond to these items in some way [1]. For example, in research on human memory, participants might be presented with a list of words they have to memorize. In a later stage of the experiment, they are presented with a mix of old words – that they know from the beginning – and some new words. It is their task to indicate whether they know the words from the initial learning phase (e.g., [2]). To investigate whether there is a memory-independent threshold influencing when people categorize a word as old rather than new, one might present one group of participants with a high frequency of old words; a different group of participants would be presented with a low frequency of old words [3]. Participants tend to respond "old" more often when they are in doubt about their response if there is a high frequency of old words. This difference in response criterion has been termed "probability matching" and occurs for old as well as new words [3].

Oftentimes, different experimental conditions are not varied between different research participants, but within the same person. It is for example possible to vary the relative frequency of old versus new words within the same participant in two consecutive experimental sessions. The within-participants approach is popular because it is statistically more efficient than the between-participants approach [4]. Due to memory effects, however, it is usually not possible to present the same set of words to the same person more than once. Instead, it is required that different words are employed in different conditions [1]. However, for the conditions to be comparable, the different word sets have to be as similar as possible; only the experimental manipulation should vary systematically between conditions. Therefore, word sets should be matched with regard to all features that may influence participants' responses. Such features may include the perceived familiarity of the words or average word length. Assigning items to different sets while maximizing similarity between sets presents researchers with a difficult problem [5]. Throughout this thesis, this problem of establishing similar sets of words or other elements is called item assignment. The term item may refer to any element that has some quantifiable features. In item assignment, it is assumed that a full set of items is partitioned into p subsets in such a way that each item is assigned to exactly one subset and that each subset has the same cardinality. The restriction that each subset has the same size follows from the full balancing that is usually conducted in psychological experiments: in each condition, the same number of items is employed to make all conditions as parallel as possible.

Lahl and Pietrowsky (2006) argued that the assignment of items to sets should be based on the mathematical distances between item features, for example by computing pairwise Euclidean distances [1]. To help researchers with the task of assigning items to sets, the authors provided a Windows computer program that takes as input a table of item features and outputs a table of item pairs. The output is sorted by item distance, such that pairs of items that are most similar appear at the top of the table. From top to bottom, the researcher is advised to assign the items that are closest to each other to different sets. The authors argue that this procedure results in item sets that are truly equivalent in a mathematical sense. However, while offering a reasonable algorithmic approach to establishing similar items

sets, there are several problems associated with their approach that preclude a mathematically optimal assignment. First, the authors argue that based on the table of item distances, "creating equivalent word lists is simply done by selecting the pairs with the lowest distance coefficients" (p. 146). However, it is not that simple. One problem is that the closest neighbor relationship is not symmetric; if one item has another item as its closest neighbor, this second item might be closer to a different, third item. Thus, when pairing two items, it is possible that a third item is not paired with its preferred partner. The second-closest neighbor for this third item may be a bad match, leading to a bad grouping of items. Hence, the greedy approach of connecting nearest neighbors is not guaranteed to find the global best pairing of items. Finding global optima in partitioning problems usually corresponds to computationally hard problems [6]. Second, even if the initial pairing of items were optimal, the authors do not explain how to assign the paired items to the two item sets. The assignment could be based on a criterion that optimizes set similarity, but such a criterion is not provided. The lack thereof can introduce discrepancies between the two sets since the assignment does not strive for feature similarity. Third, because the assignment is based on pairs of items, the procedure only allows to create two sets of items. However, the creation of more sets is necessary when an experiment employs more than two conditions within participants, which is in fact quite common. Forth, as will be shown in the present thesis, while establishing sets of items based on a pre-grouping may lead to satisfying results, it is sometimes necessary to assign very similar items to the same set to obtain the best global partitioning (see Section 4).

1.1 Contributions

This thesis presents a new method to optimize item set similarity based on an objective developed in Section 2. The objective is based on measures of group similarity developed in the context of clustering research. After formalizing item assignment as a graph problem, it will be shown that an integer linear programming (ILP) approach can be applied to solve item assignment optimally. This ILP was developed to solve the NP-complete cluster editing problem [7]–[10]. It will be shown that item assignment is the direct opposite of cluster editing and only differs in the direction of the optimization (minimization versus maximization). Section 3 presents an implementation in the programming language `R` [11] that was built to solve the ILP. It is shown that the quality of the solution is very satisfactory. In Section 4, a heuristic algorithm is developed that often finds the optimal solution and decreases running time substantially. Finally, Section 5 presents two case studies applying item assignment to real data. In the first application, sets of portrait photos are created that are parallel with regard to trustworthiness, perceived threat, attractiveness, happiness, angriness. In a second application, different sets of general knowledge questions are created that are parallel with regard to difficulty. The applications on real data show that both the exact and the heuristic approach yield satisfying assignments, encouraging their further use.

2 Formalization

In item assignment, a set of n items $I = \{i_1, ..., i_n\}$ has to be partitioned into p subsets $S_1, ..., S_p$, $p \leq \frac{|I|}{2}$. An item i is represented by a vector of l numeric features, $i = (f_1, ..., f_l)$. The partitioning has to satisfy the following restrictions:

(1) $\bigcup_{k=1}^{p} S_k = I$
(2) $|S_1| = |S_2| = ... = |S_p|$
(3) $S_k \cup S_j = \emptyset \ \forall k, j \in \{1, ..., p\},\ k \neq j$

Restriction (1) ensures that each item is assigned to a group; restriction (2) ensures that each subset contains the same number of items; restriction (3) ensures that each item is assigned to only one subset. It follows that $|S_k| = \frac{|I|}{p} \ \forall k \in \{1, ..., p\}$. The objective is to select a partitioning that maximizes the similarity of the different subsets, i.e., that leads to the most similar distribution of item features.

2.1 Cluster analysis and item assignment

Generally, there is a close semantic connection between cluster analysis and item assignment. In both problems, elements have to be partitioned into distinct groups. When doing so, however, item assignment tries to accomplish the direct opposite of cluster analysis: in item assignment, elements are assigned in such a way that groups resemble each other as much as possible. In cluster analysis, elements in each group (that is, each *cluster*) are supposed to be similar only to each other, but should be as different as possible from elements in other clusters [12]. Given the diametral meaning, establishing similar groups has been coined "anticlustering"; the term was also chosen because in the case of k-means clustering, establishing similar sets as opposed to clusters is accomplished by changing the direction of the problem from minimizing to maximizing the objective function [13], [14]. In the present thesis, a new objective of item set similarity is defined that will be shown to be the twin problem of a different clustering problem, i.e., the cluster editing problem. This objective is derived directly from measures of cluster similarity and may be more apt for the item assignment problem than the k-means objective, as will be discussed below.

In cluster analysis, the basic unit underlying a measure of group similarity is a distance d_{ij} quantifying the pairwise dissimilarity between two items i and j based on their respective feature values [15], [16]. Sometimes, measures of item similarity are used as well, in which case larger values indicate higher similarity [17]. Measures of item dissimilarity include the common Euclidean distance. The product-moment correlation coefficient can be used as a measure of similarity [18]. In this thesis, d_{ij} always represents a distance measure, meaning that larger values indicate higher dissimilarity.

To quantify the similarity of complete sets rather than pairs of items, measures of intra-group homogeneity and inter-group separability have been developed in the context of clustering research [12]. Intra-group homogeneity indicates how homogeneous the elements of a cluster are. In clustering, homogeneity within clusters should be high. For example, the k-means

objective is to minimize the feature variance within clusters, thus maximizing within-cluster homogeneity [19]. In item assignment, however, intra-group homogeneity does not need to be considered. It matters only that the different groups are equally homogeneous; the ground level of homogeneity itself is unimportant and is determined by the input data. In contrast, inter-group separability is a more important criterion in item assignment. Inter-group separability indicates how distinct clusters are and is computed as the distance between the different clusters [20]. To quantify inter-group separability, i.e., the distance between two sets A and B, the following measures have been used [21], [22]:

$$d_{min}(A,B) = \min_{i \in A,\, j \in B} d_{ij} \tag{1}$$

$$d_{max}(A,B) = \max_{i \in A,\, j \in B} d_{ij} \tag{2}$$

$$d_{avg}(A,B) = \frac{1}{|A|\,|B|} \sum_{i \in A} \sum_{j \in B} d_{ij} \tag{3}$$

The measures d_{min} and d_{max} quantify the distance between two item sets as the minimum and maximum of pairwise item distances d_{ij}, respectively. Thus, set similarity is measured on the basis of only a single distance. The measure d_{avg} quantifies the distance between two item sets as the average distance between all items that are assigned to different sets. It takes into account the overall characteristics of all elements in each set, and it is low when item sets as a whole are similar to each other. Therefore, it is less susceptible to outliers and potentially preferable to measures based on only a single distance [16]. In particular, d_{avg} has been argued to be preferable to d_{min} and d_{max} when comparing sets of items in psychological tests [22]. Hence, the objective function to be optimized in item assignment is developed by extending d_{avg}.

2.2 The optimal solution of item assignment

In item assignment, p sets have to be created where p can be larger than 2. Thus, d_{avg} – that is used to compare two sets – has to be extended to the comparison of p sets. First, we recognize that for a given item set I and a subset number p, each created subset has the same cardinality $\frac{|I|}{p}$. Therefore, the normalizing factor in d_{avg} has the same value for each possible assignment and does not need to be considered when comparing the quality of different assignments. Hence, without losing any information, we construct the distance measure d_{sum} as a substitute of d_{avg} in the context of item assignment:

$$d_{sum}(A,B) = \sum_{i \in A} \sum_{j \in B} d_{ij} \tag{4}$$

In the special case of two sets, d_{sum} is the sum of all pairwise distances of items that are not

assigned to the same set. To extend this measure to p sets, we let x_{ij} denote whether two items i_i and i_j have been assigned to the same item set:

$$x_{ij} = \begin{cases} 1 & \text{if } i_i \in S_m \wedge i_j \in S_m \\ 0 & \text{otherwise} \end{cases} \quad i \neq j,\ m \in \{1, ..., p\} \tag{5}$$

Then, we define d_{psum} as the summed distance between items that are not part of the same item set in the case of p item sets:

$$d_{psum} = \sum_{1 \leq i < j \leq n} d_{ij}\,(1 - x_{ij}) \tag{6}$$

Larger values of d_{psum} indicate a lower total similarity of all p item sets. Hence, equation (6) defines the objective function in item assignment: minimizing d_{psum} while ensuring that (a) the required number of sets is created, (b) all sets are of equal cardinality, and (c) all sets are mutually disjoint is defined as the optimal solution of item assignment.

Note that minimizing d_{psum} is equivalent to maximizing the sum of all distances between items that are assigned to the same set. To appreciate this equivalence, consider the total sum of paired distances of all items. This total sum can be split by whether items are assigned to the same set or not. Let d_{ssum} denote the sum of distances of items that have been assigned to the same set:

$$d_{ssum} = \sum_{1 \leq i < j \leq n} d_{ij}\, x_{ij} \tag{7}$$

Then, let d_{tsum} denote the total sum of all item distances:

$$d_{tsum} = \sum_{1 \leq i < j \leq n} d_{ij} = d_{psum} + d_{ssum} \tag{8}$$

The total sum of all item distances d_{tsum} is not influenced by the assignment of items to sets. Hence, it is clear that minimizing d_{psum} simultaneously maximizes d_{ssum}. Therefore, instead of minimizing d_{psum}, we can also assign items with the objective of maximizing d_{ssum}. Both procedures will lead to the same optimal item assignment x_{ij}.

The following two sections explain how to obtain the optimal solution to item assignment based on a graph formulation that can be solved using ILP. In this formulation, the optimal solution will be obtained by maximizing d_{ssum} from equation (7).

2.3 Item assignment as a graph problem

In the graph formulation of item assignment, a problem instance is an undirected complete graph $G = (V, E)$ [23]. V is the set of vertices and each vertex represents an item $i \in I$. E is the set of edges and represents the relationships between items. Edges are unordered pairs $\{i, j\}$ of vertices. The short form ij will be used to refer to edges. A cost function $w : E \to \mathbb{R}^+$ is defined that assigns a weight to each edge. In item assignment, the weight of each edge is simply the distance between the two items that are connected by the edge, i.e., $w(ij) = d_{ij}$. This is because the objective function in equation (7) directly works on item distances.

To solve item assignment, we have to select a subgraph $G' = (V, E')$ of the complete supergraph G. According to the objective function in (7), the optimal solution maximizes the total sum of pairwise distances of items in the same group. Therefore, in graph terms, the optimal solution is a subgraph G' that contains a subset $E' \subset E$ that maximizes the sum of the weights of the edges. Additionally, the solution must consist of disjoint cliques; in a graph clique, all vertices are connected by an edge, but there are no edges connecting different cliques. Hence, cliques in the subgraph E' represent item sets. One has $ij \in E' \Leftrightarrow x_{ij} = 1$ and $ij \notin E' \Leftrightarrow x_{ij} = 0$ where x_{ij} is defined as in (5); thus, in the graph formulation of item assignment, we will also use variables x_{ij} to indicate whether two elements are part of the same clique. When setting the vertex set $V = \{1, ..., n\}$, we obtain the following form of the objective:

$$\text{maximize} \sum_{1 \le i < j \le n} w(ij)\, x_{ij} \tag{9}$$

2.4 Item assignment and cluster editing

The objective function in equation (9) has a striking resemblance to the objective function used in the weighted cluster editing problem [9], [24]. In weighted cluster editing, one is given a graph $G = (V, E)$ and a similarity function $s : \binom{V}{2} \to \mathbb{R}^+$ describing the relationships between elements [8]. In this graph, two elements are connected by an edge if and only if the connected elements are deemed to be similar to each other; this is usually the case if the similarity exceeds some threshold t, i.e., if $s(uv) > t$ for $u, v \in \binom{V}{2}$ [8], [25]. The problem definition is to "clean" the initial graph, that is, to construct a graph consisting of disjoint cliques that are interpreted as clusters. Weighted cluster editing and its unweighted counterpart have also been studied under different names such as correlation clustering [26], the clique partition problem [9], and transitivity clustering [25]; the unweighted version of this problem has been shown to be NP-complete several times (e.g., [26]; see [27]).

To transform the initial graph G into a cluster graph $G' = (V, E')$, edges are inserted and deleted in such a way that the cost of inserting and deleting edges is as small as possible. The editing function $cost(G \to G')$ defines the costs associated with transforming the initial graph into a cluster graph as follows (cf. [8]):

$$cost(G \rightarrow G') = \sum_{uv \in E \setminus E'} s(uv) + \sum_{uv \in E' \setminus E} s(uv) \tag{10}$$

If an edge uv is part of the initial graph, its deletion is associated with a cost of $s(uv)$; if an edge uv is not part of the initial graph, its insertion is associated with a cost of $s(uv)$. Thus, the cost function punishes the insertion of non-existing edges (i.e., inserting an edge that connects dissimilar elements) and the deletion of existing edges (i.e., deleting an edge that connects similar elements). Minimizing the total cost of insertions and deletions while creating a graph of disjoint cliques is the objective in weighted cluster editing.

The objective of weighted cluster editing as defined in equation (10) is often given in a different, but equivalent form that can be used directly as the objective function in an ILP formulation to solve the problem optimally [9], [24], [28]. In this problem formulation, the input graph is a complete graph $G = (V, E)$; edges that were missing in the previous formalization are now represented by negative edge weights. Thus, a weight function $w' : E \rightarrow \mathbb{R}$ assigns a negative weight to an edge uv if the two elements u and v are categorized as dissimilar. Positive edge weights are assigned if two elements are categorized as similar. If we assume that the categorization into similar and dissimilar elements depends on a threshold $t \in [0, \infty)$, the weight function w' is defined as follows [25], [28]:

$$w'(uv) = s(uv) - t \tag{11}$$

When defining variables $x_{uv} = 1$ if $uv \in E'$ and $x_{uv} = 0$ if $uv \notin E'$, and $V = \{1, ..., n\}$, the optimal solution for weighted cluster editing is then given by the following objective that also has to be optimized under the restriction that the output graph $G' = (V, E')$ is a union of disjoint cliques [9]:

$$\text{maximize} \sum_{1 \leq i < j \leq n} w'(ij)\, x_{ij} \tag{12}$$

The close relationship between item assignment and weighted cluster editing is now apparent: The objective function (12) actually has the same form as the objective that was derived for the item assignment problem in (9). The function only differs with regard to the definition of the weight function; in weighted cluster editing, we have positive and negative edge weights, whereas in item assignment we only have positive distance values as edge weights. As an additional correspondence, the objective has to be maximized under the restriction that the created subgraph is a union of disjoint cliques. In item assignment, there are some additional restrictions that have to be considered, however. First, it is required that a pre-defined number of cliques is created; in the standard cluster editing problem, the number of cliques is returned as part of the mathematically optimal solution [29]. Second, each clique has to

contain the same number of vertices.

2.5 Solving item assignment using integer linear programming

In 1989, Grötschel and Wakabayashi [9] introduced an ILP formulation to solve the weighted cluster editing problem. The objective of their ILP is given by equation (12). To ensure that the output graph is a union of disjoint cliques, they introduced so called triangular constraints as part of their ILP. More recently, in 2017, Bulhões, de Sousa Filho, Subramanian and Lucídio dos Anjos [10] developed p-cluster editing, an extension of Grötschel and Wakabayashi's ILP that enforces the creation of a user-specified number p of clusters (also see formulation $F1$ in [30] for a concise version of the ILP proposed in [10]). They used decision variables $\{y_1, ..., y_n\} \in \{0, 1\}$, denoting whether each vertex is a "cluster leader". Their constraints ensure (a) that there are exactly p cluster leaders and (b) that each vertex is only connected to one cluster leader. The restrictions used in their ILP are given as follows:

$$-x_{ij} + x_{ik} + x_{jk} + y_k \leq 1, \qquad \forall i, j, k \in V, i < j < k, \tag{13}$$
$$x_{ij} - x_{ik} + x_{jk} \leq 1, \qquad \forall i, j, k \in V, i < j < k, \tag{14}$$
$$x_{ij} + x_{ik} - x_{jk} \leq 1, \qquad \forall i, j, k \in V, i < j < k, \tag{15}$$
$$y_1 = 1, \tag{16}$$
$$y_j \leq 1 - x_{ij}, \qquad \forall i, j \in V, i < j, j \geq 2, \tag{17}$$
$$y_j \geq 1 - \sum_{i<j} x_{ij} \qquad \forall j \in V, j \geq 2, \tag{18}$$
$$\sum_{j \in V} y_j = p, \tag{19}$$
$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in V, i < j, \tag{20}$$
$$y_j \in \{0, 1\}, \qquad \forall j \in V. \tag{21}$$

Here, (13) - (15) are the triangular constraints originally developed by Grötschel and Wakabayashi. The constraints (16) - (19) ensure that exactly p cliques are created; constraints (20) - (21) ensure that the decision variables are binary. To solve item assignment, we additionally have to make sure that the same number of items $C := \frac{|V|}{p}$ is assigned to each of the p clusters. The following constraint enforces this restriction by ensuring that each item is connected to exactly $C - 1$ other items:

$$\sum_{1 \leq i < j \leq n} x_{ij} = (C - 1), \qquad \forall i \in \{1, ..., n - 1\} \tag{22}$$

2.5.1 Differences between cluster editing and item assignment

Formally, the formulation of item assignment as presented here is very similar to the weighted cluster editing problem: the same objective function is used, and the ILP only adds one additional constraint in comparison to the ILP presented in [10]. To prevent any confusion as to why the same formulations leads to semantically different results – in the one case, group similarity is maximized, and in the other case, group separation is maximized –, it is appropriate to clearly name the differences between the problem formulations of item assignment and cluster editing.

In weighted cluster editing, edge weights are based on a similarity measure, such that larger values indicate a stronger connection between elements. Hence, maximizing a similarity measure leads to a clustering of elements, but maximizing a distance measure establishes similar groups. This leads to an interesting insight that will be employed in Section 4: when minimizing objective function (9) with the restrictions posed in (13) - (22), we obtain a clustering of the elements, such that elements within each cluster are homogeneous and different from elements in other clusters. Thus, item assignment is not only the semantic opposite of clustering, but also the mathematical opposite of cluster editing.

One additional difference between item assignment and classical cluster editing is that negative edge weights are used in cluster editing if two elements are thought to be dissimilar to each other. This ensures that a reasonable clustering is found even when the number of clusters is not specified a priori. If there were only positive edge weights, all elements would be assigned to the same large cluster because such an assignment would maximize the objective. In item assignment, it is possible to work only with positive distance values because the restrictions in (13) - (22) ensure that the desired number and size of clusters is created.

3 Implementation

The solution to the item assignment problem was implemented using the statistical programming language `R` [11]. It can be retrieved as an installable software package `anticlust` (version 0.1.0) from https://github.com/m-Py/anticlust. The highest level function `item_assignment` receives a data frame of item features – where rows represent items and columns represent features – and solves the problem instance by solving the ILP that was derived in the previous section. To this end, the following steps are taken:

1. Based on the item features, a quadratic matrix of between-item distances is computed. The default distance measure is the standard Euclidean distance. When a different distance measure is preferred, the user can circumvent the automatic computation and directly pass a matrix of distance measures to a lower level function (see [18] for an overview of some common distance measures used in clustering). When using the default distance measure, it is possible to request a standardization of all features to ensure an equal weight of each feature in the resulting distance [18]. When features are standardized, each feature is transformed in such a way that the feature mean is 0 and the standard deviation is 1. In this case, the Euclidean distance is not dominated by features that have the largest numeric values.
2. The distance matrix is transformed into a vector of item distances d_{ij}. This vector is needed as the objective function in the ILP formulation; its length is $\frac{n(n-1)}{2}$, where n is the number of items.
3. The ILP is constructed. It solves the system of inequalities $\boldsymbol{A}x \sim b$ for the vector x representing the binary decision variables. Decision variables encode the connection between each pair of items ($x_{ij} \in \{0, 1\}$), or the leadership status of each item ($y_j \in \{0, 1\}$), yielding $x = (x_{12}, x_{13}, ..., x_{n-1\,n}, y_1, ..., y_n)$. The relationship $\sim$ either indicates equality, lower equal, or greater equal, depending on the direction of a constraint. In the matrix $\boldsymbol{A}$, each row represents a constraint and each column represents a decision variable. Each cell of the matrix is the coefficient of a decision variable for a given constraint. Since most constraints only affect few variables – in almost all cases, a single constraint only affects a maximum of four decision variables – most coefficients are zero and the matrix $\boldsymbol{A}$ is very sparse. For a problem size of 20 items, the constraints in (13) - (22) are realized in a 3651×210 matrix. In this matrix, only 12,390 of 766,710 entries are non-zero (2%). The vector b in the righthand-side of the equation $\boldsymbol{A}x \sim b$ represents the objective. It contains the distance values d_{ij} and n zeros. Inserting zeros ensures that the decision variables representing the leadership status of each item do not affect the objective function. So we have $b = (d_{12}, d_{13}, ..., d_{n-1\,n}, 0, ..., 0)$.
4. The ILP is solved for x using one of the commercial solvers gurobi or IBM CPLEX, or by using the open source GNU linear programming kit. The solvers are treated as black boxes that are fed with the ILP and return the optimal assignment for x. They are called by the package `anticlust` using one of the interface packages `gurobi`, `Rcplex`, or `Rglpk`, respectively.
5. Based on the optimal vector x that contains information of the pairwise connectivity of items, the assignment of items to groups is deduced.

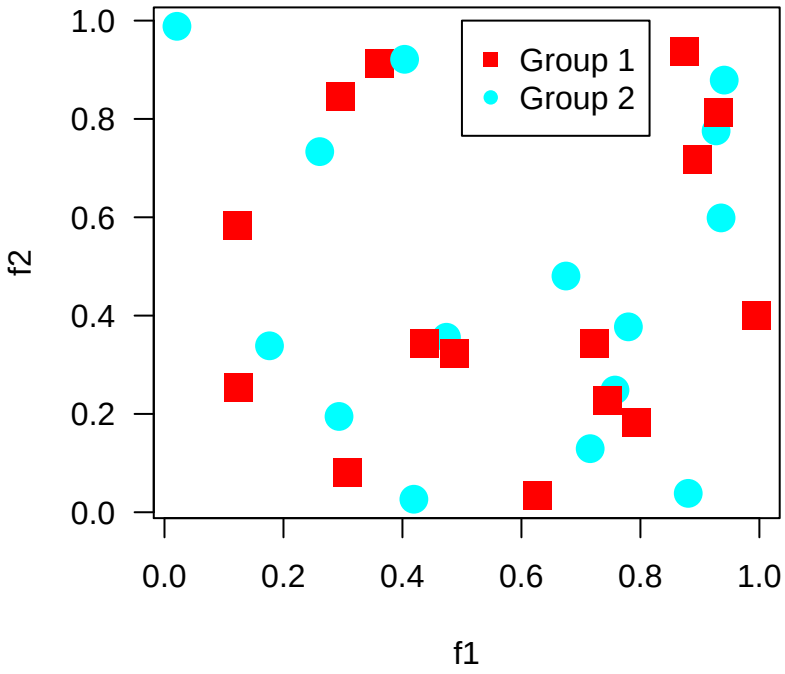

Figure 1: Illustrates an optimal group assignment in a 30-to-2 problem instance. Each item has two randomly generated features $f1$ and $f2$.

3.1 Quality of the solution

A first test illustrates the working of the algorithm. In this example, 30 items were assigned to two groups (referred to as a 30-to-2 problem in the following) using IBM CPLEX. Each item was represented by two features; features were generated as random variates from a uniform distribution in [0, 1]. Figure 1 illustrates the item-to-group allocation of the items. A visual inspection suggests that the distribution of item features is rather similar across the two groups. Table 1 quantifies group similarity using the most common descriptive statistics employed in psychology [31]. We can see that measures of centrality (mean, median) and dispersion (sd = standard deviation, range, minimum and maximum) are indeed very close to each other. This initial test therefore illustrates the usefulness of optimizing objective function that was derived in the previous section.

Table 1: Descriptive statistics by group in a 30-to-2 problem.

	f1		f2	
	Group 1	Group 2	Group 1	Group 2
mean	0.58	0.58	0.47	0.47
sd	0.3	0.3	0.31	0.33
median	0.63	0.67	0.34	0.38
min	0.12	0.02	0.03	0.03
max	1	0.94	0.94	0.99
range	0.87	0.92	0.9	0.96

3.2 Running time

The ILP developed by Grötschel and Wakabayashi [9] is known to effectively solve the cluster editing problem despite its NP-complete nature. In a lot of cases, their ILP is already solved optimally by its linear relaxation [32]. When using preprocessing techniques, it is even possible to solve instances that have more than 1,000 vertices [7], [33]. Unfortunately however, solving item assignment using the ILP described in the previous section does not run efficiently even for relatively moderate problem sizes. Figure 2 illustrates the running time for problem sizes of up to 32 items. As in the previous example, items had two features that were generated as random variates from a uniform distribution in [0, 1]. IBM CPLEX was used to solve the instances on an Intel Core i7-7700 computer (3.60GHz x 8) with 8 GB RAM running Ubuntu 16.04 LTS. The instances described in the following sections were also solved using the same setting.

Figure 2 shows that in the case of the two group problem, instance sizes up to 20 items can be solved exactly within a few seconds. However, the 30-to-2 problem already took almost an hour to finish (2975 seconds), and the 32-to-2 problem took almost 6 hours to finish. Larger problem instances did not finish at all on the test computer because the problem tree that was generated by the solver consumed too much memory. Creating three groups was even slower in comparison to the 2-group case. The factor by which the 3-group case was slower is not immediately clear from inspecting Figure 2 because the y-axis is on a log-scale. However, solving the 18-to-2 problem was ten times faster than solving the 18-to-3 problem. The 24-to-3 instance already failed on the test computer.

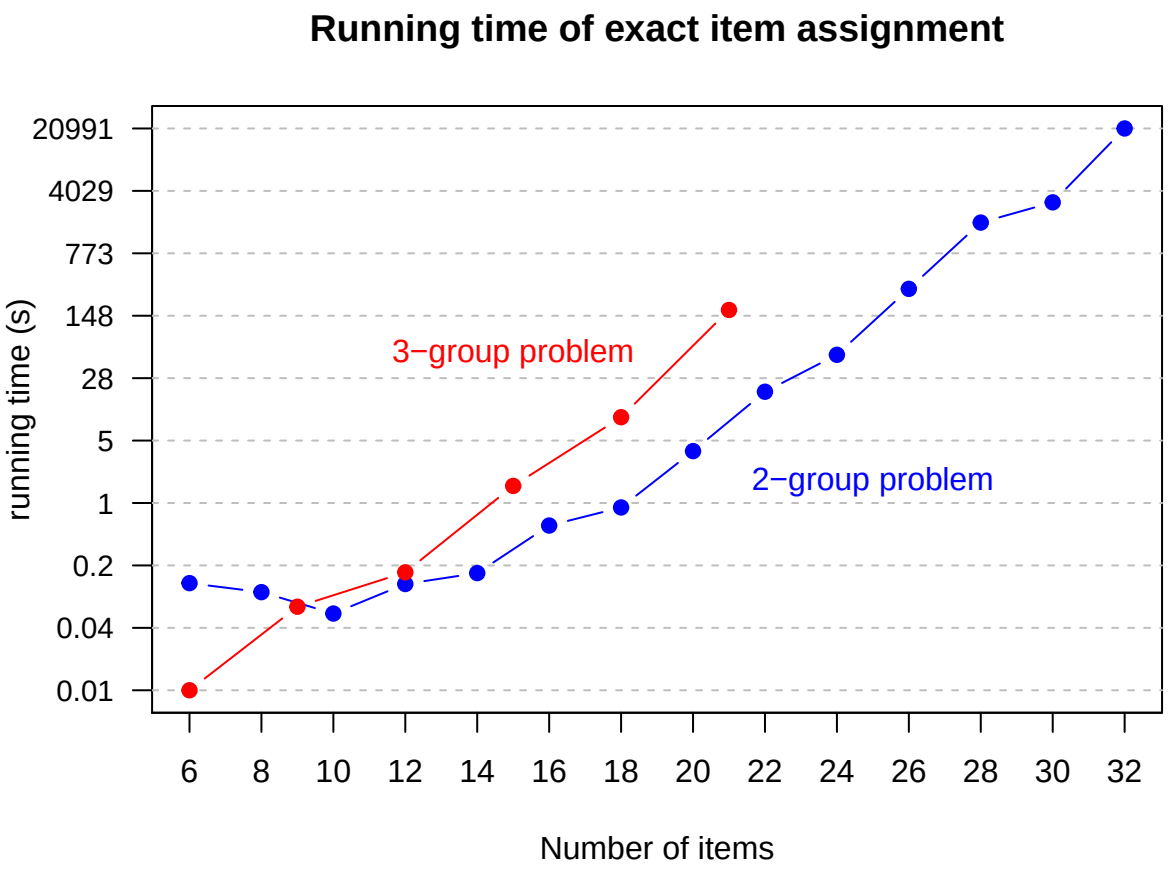

Figure 2: Illustrates the running time of the ILP in the 2-group and the 3-group case. The y-axis is on a log-scale due to the exponential increase in running time with increasing problem size.

4 Heuristic

Whereas the item assignment algorithm produces results of high quality, the running time of the ILP solver does not allow to process large problem instances in a reasonable time. Even though instances of several hundred items may not be expected in experimental psychology, it would still be desirable to solve instance sizes of up to 50 or 60 items. For this reason, a heuristic framework was developed to solve larger instances while preserving a high quality of the solution. The heuristic is based on the following considerations:

- Given that item sets should be as similar as possible, it is reasonable to assign very similar items to different sets.
- Disallowing very similar items to be assigned to the same set can be accomplished by setting the weight of their connecting edge to $-\infty$ in the ILP framework [7].
- Setting some edge weights to a large negative value may improve the running time of the ILP because an uneven distribution of edge weights tends to decrease running time in weighted cluster editing [7], [27].

The following heuristic is based on these considerations:

Heuristic framework to solve item assignment
Given: n items that have to be assigned to p groups

1. Determine the pairwise distances between all items
2. Create a clustering of similar items; each cluster has p elements, resulting in $\frac{n}{p}$ clusters
3. Set the between-item distance of items in the same cluster to $-\infty$
4. Solve item assignment on the revised distances as described in Section 2

Figure 3 illustrates the working of the heuristic for $n = 26$ and $p = 2$. Similar items that are part of the same *precluster* on the left side are never part of the same group on the right side. Given that only similar items are forbidden to be part of the same group, this procedure likely preserves a good solution. However, the optimal item assignment might require to group items together that are part of the same precluster; therefore, this heuristic is not guaranteed to find the optimal solution.

4.1 Equal-sized cluster editing

The heuristic presented above may make use of different clustering algorithms to create the initial preclusters. Depending on the instance size, exact or heuristic clustering algorithms could be employed. For the example in Figure 3, exact weighted cluster editing was used. As stated in Section 2, item assignment is the mathematical opposite of cluster editing. Therefore, the same ILP that was used to solve item assignment can be used to obtain a clustering while ensuring the same number $\frac{n}{p}$ of elements per cluster. However, instead of maximizing the objective function, the objective function has to be minimized. For the example in Figure 3, this procedure created an optimal clustering of 13 clusters with 2 elements per cluster. Note

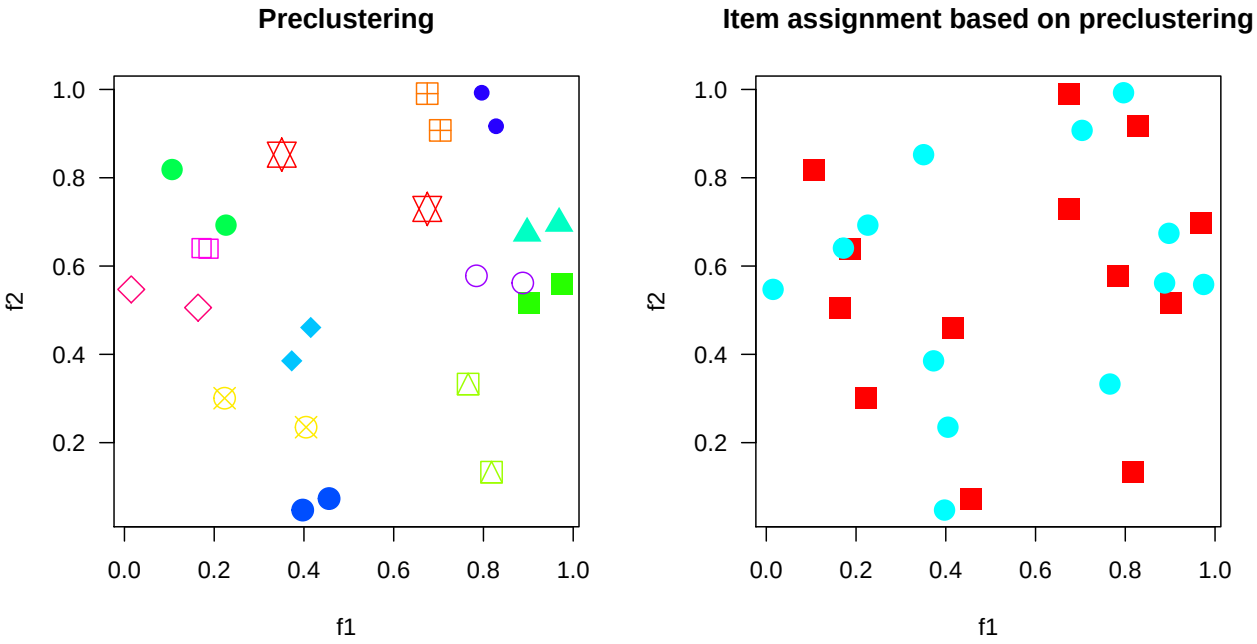

Figure 3: Illustrates the proposed heuristic to solve item assignment for a 26-to-2 problem instance. Based on a preclustering into 13 clusters, pairs of similar items belonging to the same precluster are assigned to different groups subsequently.

that this procedure creates useful clusters even though there are no negative edge weights in the problem instance. In contrast, classical cluster editing – that does not prescribe a cluster number and cluster size – relies on the presence of negative edge weights to identify dissimilar vertices. In this *equal-sized cluster editing*, external constraints enforce the cluster number as well as the cluster size; there is hence no need to incorporate negative edge weights. Solving equal-sized cluster editing can be accomplished using the `R` package `anticlust`.

Equal-sized cluster editing seems to work rather efficiently in comparison to the running time of item assignment. Figure 4 shows that instance sizes up to 60 items are solved in less than a minute. Hence, minimizing the objective function in (9) is much faster than maximizing it. Note that for larger instances of more than 50 items, the creation of the constraint matrix is actually the major contributor to the running time and the final matrix consumes quite a lot memory. The creation of instances larger than 60 items may even fail. This is, however, a limitation caused by the current implementation of the constraint matrix in `anticlust` and will be reworked in the future.

As shown in Figure 4, creating $\frac{n}{2}$ clusters is also efficient for instance sizes up to 60 items. Given that this n-to-$\frac{n}{p}$ case can be used as a preclustering step for the proposed heuristic for solving item assignment, this result is encouraging. If disallowing items of the same precluster to be assigned to the same group decreases the running time of the subsequent item assignment ILP, larger instances of item assignment should be solvable as well.

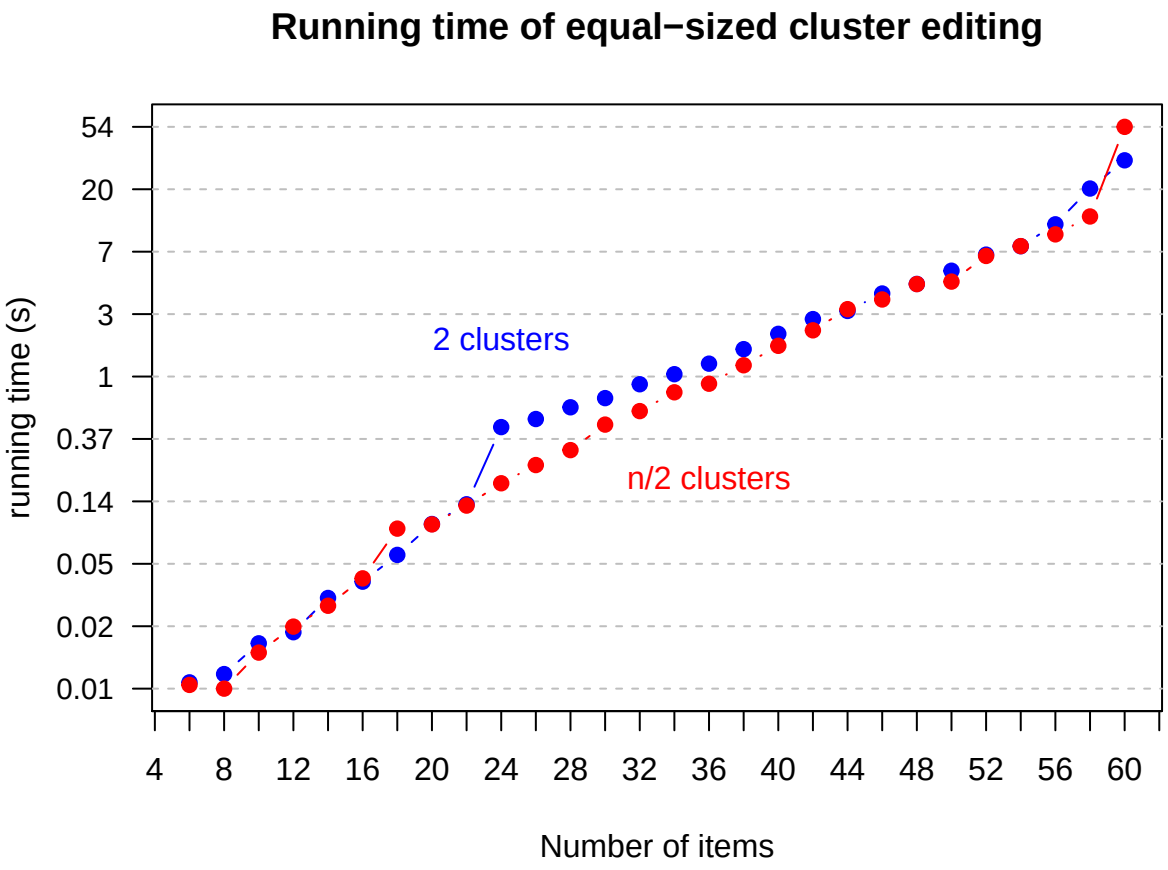

Figure 4: Illustrates the running time of the ILP solving the equal-sized weighted cluster editing problem. The y-axis is on a log-scale. The items consisted of two features that were drawn from a uniform distribution in [0, 1]. The running time is shown for the case of two groups and the case of $\frac{n}{2}$ groups; the latter may be used as a preprocessing step for a n-to-2 item assignment problem.

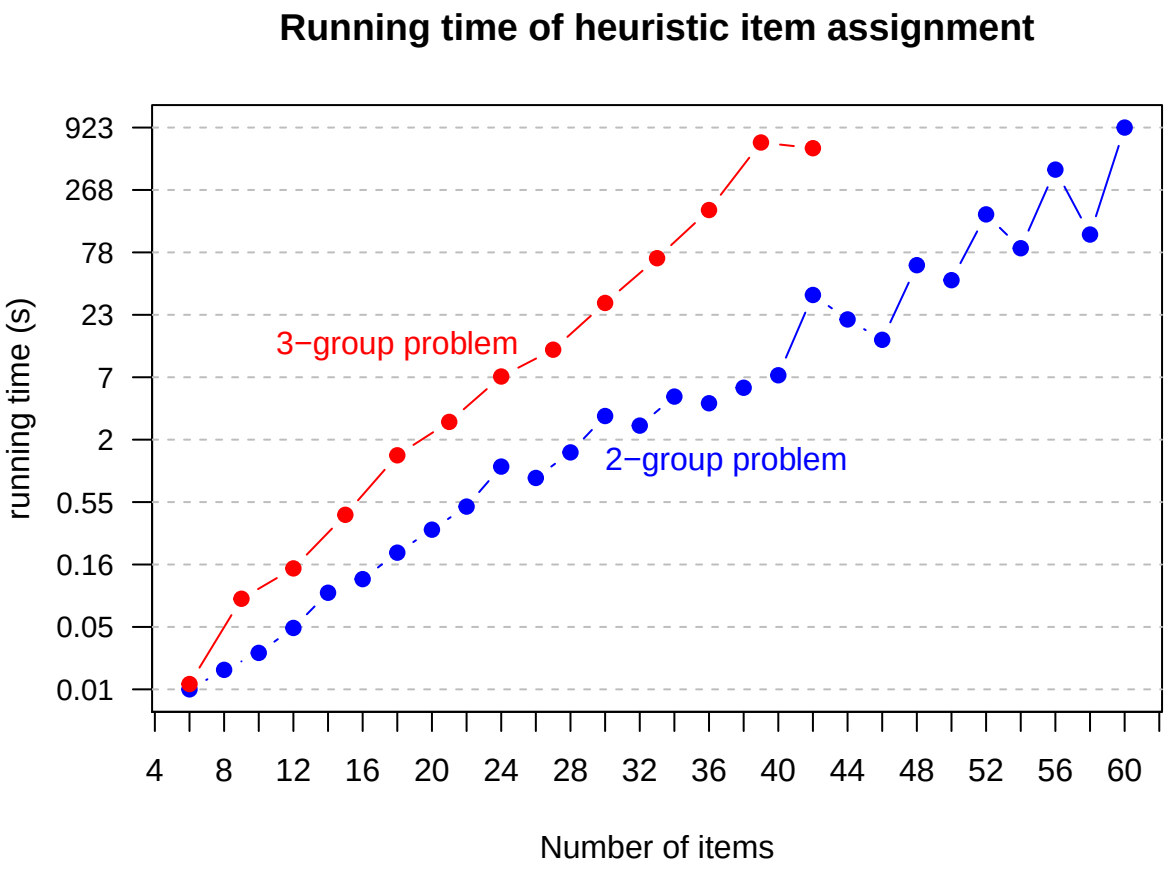

Figure 5: Illustrates the running time of the proposed heuristic to solve item assignment for the 2-group and the 3-group case. Again, items consisted of two features that were drawn from a uniform distribution in [0, 1]. Due to the preclustering, larger instances can be processed in a reasonable time. The running times incorporate both solving the preclustering as well as the item assignment ILP.

4.2 Running time of the heuristic

Solving item assignment based on a preclustering leads to a significant increase in efficiency as compared to solving it without these additional constraints (see Figure 5). Instances up to 50 items can be solved in less than a minute in the 2-group case, and instance sizes up to 60 items can be solved within acceptable time. In contrast, solving a 30-to-2 instance almost took an hour when the unadjusted edge weights were used. Again, the 3-group case is slower, but preclustering allows to process more items for this case as well.

4.3 Quality of the heuristic solution

To compare the objective value of the exact solution and the heuristic approach, 340 tests were run. Instance sizes varied between 6 and 26 items and items were assigned to 2 or 3 groups. For each run, items were generated as two features created from a random uniform

distribution in [0, 1]. Table 2 illustrates how often the heuristic found the optimal solution by item number and group size. Each combination of group size and items was tested in 20 runs.

Table 2: Illustrates the heuristic's success rate in finding the optimal solution.

	Items	6	8	9	10	12	14	15	16	18	20	21	22	24	26
Groups															
2		1	.95		1	.95	.95		.90	.90	.95		.85	.30	.65
3		.95		.90		.85		.90		.65		.55			

Across all 340 runs, the heuristic algorithm found the optimal solution 284 times (83.53%). In the 2-group case, the optimal solution was identified in 85.45% of all runs; in the 3-group case, the optimal solution was identified in 80% of all runs. Table 2 also shows that the heuristic's success rate tended to decrease with increasing instance size. When the optimal solution was not identified, the heuristic objective was 99.92% of the optimal objective—on average. In the worst case, the heuristic found a solution whose objective value was 98.94% of the optimal solution. Thus, the heuristic performed very close to optimum on the tested instances.

5 Case studies

The previous tests of the item assignment algorithm were all performed on artificial data. The present section illustrates two case studies where the quality of the solution can also be evaluated on the basis of real applications.

5.1 Case study 1: Matching portrait photos

The creation of similar stimulus sets for psychological experiments was the primary motivation for the item assignment algorithm. As an example of this application, the first case study is concerned with assigning portrait photos to different sets. Ma, Correll and Wittenbrink (2015) released a data base with 158 images to researchers of psychology [34]. Their images include photos of persons of different skin colors that have been rated by test persons on several dimensions such as the persons' trustworthiness.

Imagine we wish to study how people's evaluation of a crime is affected by skin color and gender. Based on previous research, we might hypothesize that punishment for crimes is most pronounced for male black persons who commit violent crimes (see [35] and [36]). To investigate this hypothesis, we might present the photos to two groups of research participants. In both groups, several photos of white and black men and women are presented. Each photo is accompanied by a description of a hypothetical crime the person is said to have committed. In the first group, each photo is paired with a story informing the research participant that the person on the photo has committed a violent crime, like robbing a bank or beating up a person. In the second group, each photo is paired with a story informing the participant that the person on the photo has committed a financial crime like avoiding taxes. In dependence of skin color, gender and crime type, we might analyze the perceived negativity of the persons and how much punishment they deserve for committing the crimes.

Instead of presenting distinct groups of research participants with the two kinds of crimes, a statistically more efficient approach is to present each participant with both kinds of crimes [4]. In this case, it is desirable that photos in both conditions are similar with regard to any features that might influence people's evaluations of the persons in the photos. The photos in the data base by Ma et al. (2015) contain ratings on several dimensions that should be parallel between sets; for this case study, six features were considered (trustworthiness, perceived threat, attractiveness, happiness, angriness, and age).

Table 3 shows that the data base by Ma et al. (2015) contains at least 36 photos for each combination of gender and skin color (black and white). To obtain the same number of photos in each item set, 36 photos were selected for each combination of gender and skin color; when more than 36 photos were available, some photos were discarded at random.

Table 3: Number of photos by gender and skin color in the data base by Ma et al. (2015).

	Black	White
Female	48	37
Male	37	36

A heuristic 36-to-2 assignment was conducted for each of the four combinations of gender and skin color. Prior to assignment, features were standardized. Table 4 shows that the feature means within each combination of skin color and gender are very similar. Thus, the heuristic algorithm successfully established similar sets of photos.

Table 4: Illustrates the results of the assignment in case study 1. Photos of were assigned to two groups within each combination of skin color [black, white] and gender [female, male]. Each group of photos consists of 18 photos. Cells display the group average for each feature that was used in item assignment.

Skin color	Gender	Group	Trustworthy	Threat	Attractive	Happy	Angry	Age
Black	Female	1	3.50	2.30	3.03	2.65	2.55	26.92
Black	Female	2	3.51	2.32	2.99	2.63	2.58	26.89
Black	Male	1	3.55	2.55	3.21	2.61	2.55	27.57
Black	Male	2	3.54	2.54	3.18	2.56	2.58	27.47
White	Female	1	3.59	2.19	3.41	2.51	2.66	26.00
White	Female	2	3.58	2.20	3.32	2.51	2.64	26.31
White	Male	1	3.28	2.57	2.98	2.39	2.67	26.05
White	Male	2	3.28	2.59	2.98	2.38	2.69	25.98

5.2 Case study 2: Parallel tests

Another interesting application of item assignment is the creation of parallel tests [37]. The second case study therefore applies item assignment to 30 multiple-choice items from a general knowledge test [38]. These 30 items were presented to 1,142 test-takers, allowing us to determine empirical item characteristics that can be used as features in item assignment [39]. In this case study, item assignment is used to obtain two and three similar test sets, respectively. Such an application is of interest when creating several versions of a university exam that are presented to different cohorts of students. Due to potential item sharing, it might not be desirable to present the same test items in consecutive examinations, but it might be desirable that the different versions of the test do not differ too much in difficulty. The difficulty of a multiple-choice item is simply the proportion of test-takers who solved the item correctly. Hence, its range is in [0, 1] and larger values indicate easier items. In

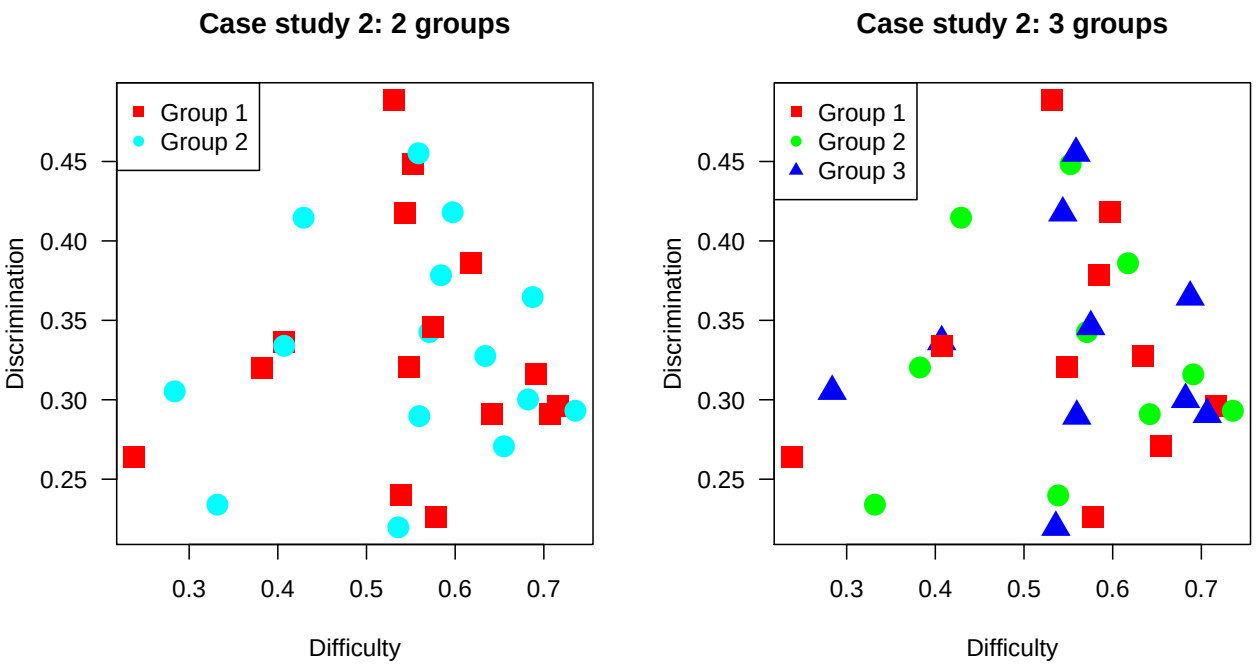

Figure 6: Illustrates the results of the assignment in case study 2 with item difficulty and item discrimination as assignment criteria. The 3-group problem was solved using the heuristic detailed in Section 4.

addition to item difficulty, item discrimination is another property of test items that is often considered when constructing parallel tests. Item discrimination is an indicator of item quality. It is defined as the correlation between the item (coding: 1 = solved; 0 = not solved) and the total test score. As item discrimination is simply a correlation coefficient, its range is in [-1, 1]. Semantically, item discrimination indicates how well the item distinguishes between test-takers of high and low ability. The higher the discrimination, the better. If item discrimination is near zero, it does not offer any information on a test-taker's ability. If item discrimination is negative, high ability test-takers tend to answer the item incorrectly more often than low ability test-takers; items of negative discrimination should be avoided.

The 2-group and the 3-group problems were solved for the 30 multiple-choice items, thus creating two or three parallel tests, respectively. Item features were standardized before the assignment was conducted. The 2-group case was solved exactly in about 90 minutes on the test computer; the heuristic also identified the optimal solution in 2 seconds. The 3-group case had to be solved using the heuristic. Figure 6 illustrates the assignments in two scatter plots. The results are very satisfying: features are distributed very evenly in two and three groups, respectively (see Table 5 and Table 6).

Table 5: Item difficulty and item discrimination by group (Case study 2, 30-to-2 problem, optimal assignment).

	Difficulty		Discrimination	
	Group 1	Group 2	Group 1	Group 2
mean	.55	.55	.33	.33
sd	.13	.13	.07	.07
median	.55	.57	.32	.33
min	.24	.28	.23	.22
max	.72	.74	.49	.46
range	.48	.45	.26	.24

Table 6: Item difficulty and item discrimination by group (Case study 2, 30-to-3 problem, heuristic assignment).

	Difficulty			Discrimination		
	Group 1	Group 2	Group 3	Group 1	Group 2	Group 3
mean	.55	.55	.55	.33	.33	.33
sd	.14	.13	.13	.08	.07	.07
median	.58	.56	.56	.32	.32	.32
min	.24	.33	.28	.23	.23	.22
max	.72	.74	.71	.49	.45	.46
range	.48	.40	.42	.26	.21	.24

6 Discussion

This thesis presents a method to assign elements to sets with the aim of creating sets that are as similar as possible. The problem was termed item assignment and has its roots in establishing experimental conditions in experimental psychology. To solve item assignment, an ILP was employed that was developed in the context of clustering research and has been used to solve the weighted cluster editing problem optimally [9], [10]. In contrast to previous algorithmic approaches to establish similar sets of items (e.g., [1]), the present method introduced and optimized an objective of set similarity. The approach led to very satisfying results on random and real data. Since the running time of the exact solution did not allow to process moderately large instances, a heuristic approach was developed that was much faster and often found the optimal solution. When the heuristic did not find the optimal solution, it nevertheless identified an objective that was very close to optimal; on average, the heuristic objective missed the optimal objective by less than 0.1% whenever it was not found. Additionally, the heuristic performed well on real data in two case studies, encouraging its use to solve real-life problem instances.

Beyond creating similar sets in experimental psychology, there are additional potential applications of item assignment in psychology or other research areas. As was suggested in case study 2, creating parallel tests is an interesting application. By correlating two parallel test versions that are as similar as possible, it is possible to obtain an estimate of test score reliability, i.e., the precision with which a test measures knowledge or ability [37]. In machine learning, item assignment may be used to create validation sets in the case of k-fold cross-validation (see [40]). In k-fold cross-validation, it is desirable that the different validation sets are as similar as possible with regard to the distribution of the predictor features, because observed predictive performance does not only depend on "true" predictive accuracy, but also on the range of the predictors' features.

6.1 Limitations

Several limitations have to be considered with regard to the present work. The first limitation concerns the problem formulation itself. First, it became apparent that optimizing the item assignment objective function was not done efficiently. This was somewhat disappointing because the ILP algorithm that was employed had previously been used to efficiently solve the cluster editing problem [7], [33]. Given that equal-sized cluster editing was solved much faster than item assignment, we can deduce that the difference in efficiency was caused by the different objective: maximizing group similarity obviously worked less well than maximizing group dissimilarity. For moderately large instances, the problem tree that the ILP solver generated became very large. Apparently, it was not possible to cut a lot of branches of the tree when searching the problem space. For cluster editing, the tree-cutting approach seems to work much better. However, given that the heuristic also produced very satisfying results for instance sizes up to 60 items, it is likely that the implementation presented here will support researchers with problems that arise in practice. In the end, as in cluster analysis, the quality of the assignment has to be evaluated by the user and an exact solution may not

always be required.

A second limitation concerns the implementation presented here. Solving item assignment was implemented as a package for the statistical programming language `R` that is used increasingly often by researchers in psychology and other fields. A new `R` implementation was created because it is most likely useful to the target users. However, note that efficient and mature implementations of cluster editing already exist for other programming languages [7], [28], whereas the new `anticlust` package has to be developed further. For example, the current version (v0.1.0) has a limitation on the instance sizes that can be processed because the constraint matrix of the ILP becomes very large for moderately large instances. This is because all constraints are enumerated and for large instances, adding all $3 \cdot \binom{n}{3}$ triangular constraints alone becomes infeasible [9]. More efficient implementations only add inequalities if they are violated [7]. Even more importantly, the constraints matrix is not yet implemented as a sparse matrix in `anticlust` even though most entries are zero, wasting a lot of memory. This was not a problem when solving instances exactly because the exact approach was already infeasible for more than 32 items. However, if we want to process more than 60 items using the heuristic, it is important to employ a sparse matrix.

6.2 Outlook

Apart from making `anticlust` more efficient and less memory consuming, several other factors should be considered in the future. First, to solve even larger instances, it may be useful to extend the heuristic proposed in Section 4. This heuristic actually combines two exact approaches and both steps may be replaced by less exact approaches to gain efficiency. For example, assuming that the preclustering alone already helps to create rather similar groups – in fact, the approach proposed in [1] relies entirely on a pairing of similar items –, it may not be necessary to apply exact item assignment on the edited distances to obtain satisfying results. Instead, a simple possibility would be to employ repeated random assignment under the restriction not to assign preclustered items to the same set, while optimizing the item assignment objective function. For larger instances, even the preclustering may be done heuristically.

Another factor that should be considered in the future is the implementation of other distance measures in the `anticlust` package. The current implementation allows to compute Euclidean distances automatically; other measures can only be employed if the user passes a self-computed matrix of inter-item distances, which may be a nuisance. The automatic computation of other distance measures should be made possible in future releases. For example, distance measures for categorical features are desirable as well as distance measures that incorporate the correlation between features, such as the Mahalanobis distance.

Finally, the item assignment algorithm proposed here should be systematically compared to other approaches that have been used to solve similar problems. These include heuristic approaches to establish similar validation sets in k-fold cross validation (e.g., [40]) and heuristic anticlustering methods based on k-means clustering [13], [14].

References

[1] O. Lahl and R. Pietrowsky, "EQUIWORD: A software application for the automatic creation of truly equivalent word lists," *Behavior research methods*, vol. 38, no. 1, pp. 146–152, 2006.

[2] J. Kantner and D. S. Lindsay, "Response bias in recognition memory as a cognitive trait," *Memory & cognition*, vol. 40, no. 8, pp. 1163–1177, 2012.

[3] T. E. Parks, "Signal-detectability theory of recognition-memory performance." *Psychological review*, vol. 73, no. 1, pp. 44–58, 1966.

[4] D. Lakens, "Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for t-tests and anovas," *Frontiers in psychology*, vol. 4, p. 863, 2013.

[5] A. Cutler, "Making up materials is a confounded nuisance, or: Will we able to run any psycholinguistic experiments at all in 1990?" *Cognition*, vol. 10, pp. 65–70, 1981.

[6] P. Arabie and L. J. Hubert, "An overview of combinatorial data analysis," in *Clustering and classification*, P. Arabie, L. J. Hubert, and G. De Soet, Eds. 1996, pp. 5–63.

[7] S. Böcker, S. Briesemeister, and G. W. Klau, "Exact algorithms for cluster editing: Evaluation and experiments," *Algorithmica*, vol. 60, no. 2, pp. 316–334, 2011.

[8] S. Rahmann, T. Wittkop, J. Baumbach, M. Martin, A. Truss, and S. Böcker, "Exact and heuristic algorithms for weighted cluster editing," in *Computational systems bioinformatics: (Volume 6)*, World Scientific, 2007, pp. 391–401.

[9] M. Grötschel and Y. Wakabayashi, "A cutting plane algorithm for a clustering problem," *Mathematical Programming*, vol. 45, nos. 1-3, pp. 59–96, 1989.

[10] T. Bulhões, G. F. de Sousa Filho, A. Subramanian, and F. C. Lucídio dos Anjos, "Branch-and-cut approaches for p-cluster editing," *Discrete Applied Mathematics*, vol. 219, pp. 51–64, 2017.

[11] R Core Team, *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing, 2018.

[12] R. Sharan, A. Maron-Katz, and R. Shamir, "CLICK and expander: A system for clustering and visualizing gene expression data," *Bioinformatics*, vol. 19, no. 14, pp. 1787–1799, 2003.

[13] H. Späth, "Anticlustering: Maximizing the variance criterion," *Control and Cybernetics*, vol. 15, no. 2, pp. 213–218, 1986.

[14] V. Valev, "Set partition principles revisited," in *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, 1998, pp. 875–881.

[15] F. Kovács, C. Legány, and A. Babos, "Cluster validity measurement techniques," in *6th international symposium of hungarian researchers on computational intelligence*, 2005.

[16] O. Fujita, "Metrics based on average distance between sets," *Japan Journal of Industrial*

and Applied Mathematics, vol. 30, no. 1, pp. 1–19, 2013.

[17] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica, "Correlation clustering in general weighted graphs," *Theoretical Computer Science*, vol. 361, nos. 2-3, pp. 172–187, 2006.

[18] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*, Springer, 2005, pp. 321–352.

[19] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[20] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz, "Internal versus external cluster validation indexes," *International Journal of computers and communications*, vol. 5, no. 1, pp. 27–34, 2011.

[21] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," in *ACM sigmod record*, 1998, vol. 27, pp. 73–84.

[22] D. R. Bacon, "An evaluation of cluster analytic approaches to initial model specification," *Structural Equation Modeling*, vol. 8, no. 3, pp. 397–429, 2001.

[23] S. E. Schaeffer, "Graph clustering," *Computer science review*, vol. 1, no. 1, pp. 27–64, 2007.

[24] A. Miyauchi and N. Sukegawa, "Redundant constraints in the standard formulation for the clique partitioning problem," *Optimization Letters*, vol. 9, no. 1, pp. 199–207, 2015.

[25] T. Wittkop *et al.*, "Partitioning biological data with transitivity clustering," *Nature methods*, vol. 7, no. 6, pp. 419–420, 2010.

[26] N. Bansal, A. Blum, and S. Chawla, "Correlation clustering," *Machine learning*, vol. 56, nos. 1-3, pp. 89–113, 2004.

[27] S. Böcker and J. Baumbach, "Cluster editing," in *Conference on Computability in Europe*, 2013, pp. 33–44.

[28] P. Spohr, "Developing and evaluating a cytoscape app for graph-based clustering," Bachelor thesis, University of Düsseldorf, 2017.

[29] R. Röttger *et al.*, "Density parameter estimation for finding clusters of homologous proteins—tracing actinobacterial pathogenicity lifestyles," *Bioinformatics*, vol. 29, no. 2, pp. 215–222, 2012.

[30] T. Bulhões, A. Subramanian, G. F. Sousa Filho, and F. C. Lucídio dos Anjos, "Branch-and-price for p-cluster editing," *Computational Optimization and Applications*, vol. 67, no. 2, pp. 293–316, 2017.

[31] W. Revelle, *Psych: Procedures for psychological, psychometric, and personality research.* Evanston, Illinois: Northwestern University, 2018.

[32] A. Mehrotra and M. A. Trick, "Cliques and clustering: A combinatorial approach,"

Operations Research Letters, vol. 22, no. 1, pp. 1–12, 1998.

[33] L. H. N. Lorena, M. G. Quiles, A. C. P. de L. F. de Carvalho, and L. A. N. Lorena, "Preprocessing technique for cluster editing via integer linear programming," in *Intelligent computing theories and application*, 2018, pp. 287–297.

[34] D. S. Ma, J. Correll, and B. Wittenbrink, "The chicago face database: A free stimulus set of faces and norming data," *Behavior research methods*, vol. 47, no. 4, pp. 1122–1135, 2015.

[35] D. Steffensmeier, J. Ulmer, and J. Kramer, "The interaction of race, gender, and age in criminal sentencing: The punishment cost of being young, black, and male," *Criminology*, vol. 36, no. 4, pp. 763–798, 1998.

[36] J. Hurwitz and M. Peffley, "Public perceptions of race and crime: The role of racial stereotypes," *American journal of political science*, pp. 375–401, 1997.

[37] R. D. Armstrong, D. H. Jones, and L. Wu, "An automated test development of parallel tests from a seed test," *Psychometrika*, vol. 57, no. 2, pp. 271–288, 1992.

[38] R. Hossiep and M. Schulte, *BOWIT: Bochumer Wissenstest*. Göttingen: Hogrefe, 2008.

[39] M. Papenberg and J. Musch, "Of small beauties and large beasts: The quality of distractors on multiple-choice tests is more important than their quantity," *Applied Measurement in Education*, vol. 30, no. 4, pp. 273–286, 2017.

[40] X. Zeng and T. R. Martinez, "Distribution-balanced stratified cross-validation for accuracy estimation," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 12, no. 1, pp. 1–12, 2000.