Department of Computer Science
Algorithmic Bioinformatics

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

Using anticlustering to maximize diversity and dispersion: Comparing exact and heuristic approaches

Martin Breuer

Bachelor Thesis

Submission:	20.08.2020
Supervisor:	Univ.-Prof. Dr. G. Klau
Second Assessor:	Dr. M. Papenberg

Abstract

The problem of partitioning a pool of elements arises with different goals. We deal with different approaches to anticluster these elements. This means we seek groups that are similar to each other, and not the elements within those groups. We analyze the bicriterion iterated local search (BILS) heuristic, which combines two objective functions for anticlustering, namely diversity and dispersion. Diversity is the sum of pairwise dissimilarities of all elements within a group. Dispersion is the minimal pairwise dissimilarity between two elements, which exists within one group of a partition. The BILS heuristic seeks solutions, that are Pareto efficient regarding both objectives. Using a heuristic is necessary, as we show NP-hardness of both mentioned objectives. We go into the advantages and limitations of this heuristic. Further, we compare the BILS heuristic with the exchange method, another heuristic approach. The exchange method focuses on the contrary only on one objective, that can be specified. The BILS heuristic performs better in terms of computation and solution quality for small instances. Finally, we evaluate the quality of solutions of the BILS heuristic with exact solutions. For small instances, solutions are optimal or near-optimal.

Contents

1 Introduction

The problem of partitioning a pool of elements arises with different goals. Often, it is useful to have a partition with homogenous and well-separated groups. This means elements within the same group should be as similar as possible (low intra-group dissimilarity), whereas elements from different groups should be as different as possible (high inter-group dissimilarity). These properties are representative of a clustering problem. To solve problems of this kind, there already exists a variety of algorithms [15]. Clustering methods differ with regard to their respective objective functions, which define how intra-group dissimilarity and inter-group dissimilarity are measured.

Sometimes, the opposite of clustering is required. For this thesis, we look for a partition, in which the groups are similar, but not the elements within the groups. Thus, high intra-group dissimilarity and low inter-group dissimilarity are the goals of what we refer to as anticlustering, a term independently coined by Späth [12] and Valev [13]. Anticlustering has a variety of use cases in different areas. Creating groups out of students, assigning stimulus objects to subgroups, and consumers to focus groups are anticlustering problems arising in psychology [3]. Apart from the psychological background, anticlustering is used for sub-typing of viruses, the analysis of microarray data, large-scale integration in circuitry and the location of facilities [3].

As well as for clustering, there are different objective functions for anticlustering. In this thesis, we will focus on two objective functions, called diversity and dispersion. Diversity is the sum of all dissimilarities between pairs of elements in the same group. It is a measure of total dissimilarity [3]. An anticlustering example, in which maximizing diversity is useful, is the formation of groups in an academic context. The problem consists of a number of groups and a number of students, where each student is determined by gender, age, current grades, etc. as possible attributes. Dividing students into groups such that the groups are similar and each group provides a good representation of the classroom population is desirable [5]. Forming similar groups of equal size is equivalent to maximizing the total dissimilarity [6]. So we seek a partition of the students, that maximizes the sum of all dissimilarities between pairs of students in the same group.

The second objective function is dispersion. Dispersion is the minimum difference between two elements within one cluster. We interpret this as the worst-case pairwise dissimilarity across all groups [3]. An anticlustering application, in which maximizing dispersion is useful, is the creation of multiple versions of an exam from a given pool of questions. Suppose a teacher has prepared questions from different topics and wants to determine the general knowledge of their students. When they assign the questions to different versions of their exam, focussing on dispersion leads to exams, with a lower risk of two similar questions contained in one exam. Just one exam with two similar questions lets this exam become less informative about the knowledge of the affected students.

The problem of finding a partition with a maximum value in diversity or dispersion is NP-hard, when considering the underlying decision problem. For this reason, we mainly examine the bicriterion iterated local search heuristic by Brusco et al. [3], which can approximate optimal solutions for both of these problems. In fact, this heuristic considers the optimization of diversity and dispersion at the same time. We further investigate the limitations of this heuristic and compare it to the exchange method, another heuristic

approach. At the end, we evaluate our results in comparison with optimal solutions.

2 Problem Formalization

2.1 Preliminaries

In this section, we describe the concepts used in this thesis. For illustration, we use a simple example with 6 different shaped rectangles, which we want to anticluster. Firstly, we describe the required data. We assume that there is an $N \times M$ information matrix and the number of subsets K given for every anticlustering problem. Each row in this matrix represents one of the N elements (e.g., the different rectangles in Table 1) and each of the M columns corresponds to a numeric attribute (e.g., length and width of a rectangle in Table 1). Thus, there are M attributes assigned to every element. The N elements need to be partitioned into K subsets. We use $k \in \{1, ..., K\}$ as an index of a subset. A partition of the N elements is indicated by $\pi = \{c_1, ..., c_K\}$, where c_k contains a subset of all given elements. The set Π contains all possible partitions.

rectangle	length	width
1	6cm	5cm
2	2cm	2cm
3	3cm	3cm
4	1cm	6cm
5	5cm	4cm
6	4cm	1cm

Table 1: $N \times M$ information matrix of the rectangles.

In the following, we use a graph structure to refer to the anticlustering problem. Let $G = (V, E)$ be an undirected graph with vertex set $V = \{v_1, ..., v_N\}$, representing the elements of the problem. For simplification, we will refer to a vertex v_i by using only a positional index $i = 1, ..., N$ henceforth. In our example, each rectangle is represented by a vertex. We visualize the rectangles as points on a coordinate system with length on the x-axis and width on the y-axis in Figure 1.

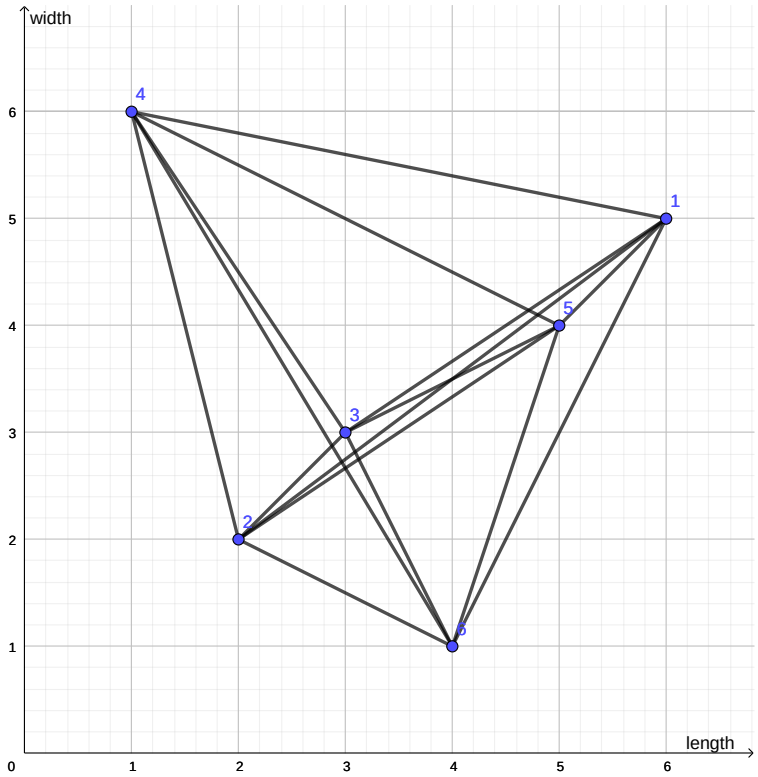

Figure 1: Rectangles represented as vertices in a graph.

An anticlustering problem is called M-dimensional Euclidean if the vertices are points of the M-dimensional Euclidean space, and the edges are all straight lines between those points [2]. There are other ways of distance measuring [3], but we only use Euclidean distances in this thesis. Thus, we refer to the weight d_{ij} of an edge i, j as the distance between the vertices i and j, where the distance measures the dissimilarity between two vertices. The $N \times N$ matrix D is the distance matrix containing all weights of the edges.

$$D = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0.00 & 5.00 & 3.61 & 5.10 & 1.41 & 4.47 \\ 5.00 & 0.00 & 1.41 & 4.12 & 3.61 & 2.24 \\ 3.61 & 1.41 & 0.00 & 3.60 & 2.24 & 2.24 \\ 5.10 & 4.12 & 3.60 & 0.00 & 4.48 & 5.83 \\ 1.41 & 3.61 & 2.24 & 4.48 & 0.00 & 3.16 \\ 4.47 & 2.24 & 2.24 & 5.83 & 3.16 & 0.00 \end{bmatrix} \end{array}$$

Figure 2: $N \times N$ distance matrix of the rectangles in cm.

2.2 Feasible Anticlustering Solutions

An anticlustering partition is considered feasible if it satisfies the two following restrictions:

$$\bigcup_{i=1}^{K} c_i = V \tag{1}$$

$$c_i \cap c_j = \emptyset, \forall i, j \in \{1, ..., K\}, i \neq j \tag{2}$$

Restriction (1) ensures that every element is contained in at least one of the clusters. Restriction (2) on the other hand requires pairwise disjoint clusters. Thus, no pair of two clusters can contain the same element.

Because many anticluster-applications desire clusters of equal size [11], we give an additional third restriction for our purposes.

$$|c_i| = |c_j|, \forall i, j \in \{1, ..., K\} \tag{3}$$

Assuming that the number of elements N is a multiple of the number of clusters K, restriction (3) ensures that every cluster contains $\frac{N}{K}$ elements. The case that N is not divisible by K would lead to groups differing by one in their size and is not covered in this thesis.

We use the term feasible solutions for partitions that satisfy all of the three restrictions. Figure 3 shows a feasible anticlustering partition $\pi_r = \{\{1, 3, 4\}, \{2, 5, 6\}\}$ for our rectangle example.

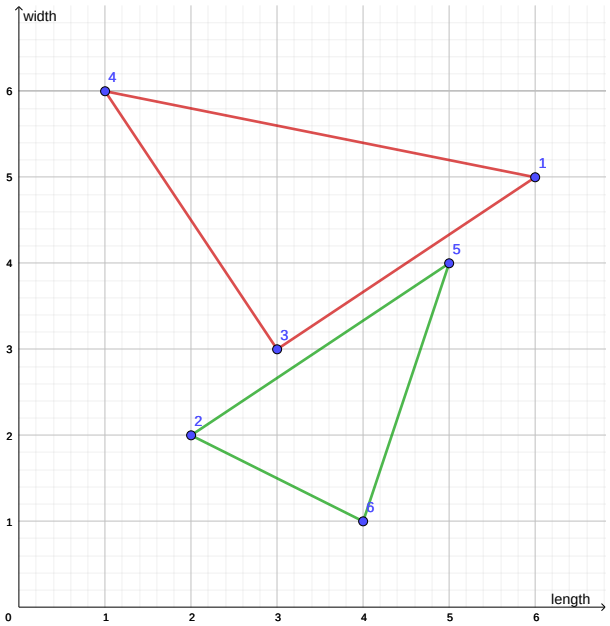

Figure 3: Feasible solution π_r.

2.3 Objective Functions

In this thesis, we will focus on two objective functions, called diversity and dispersion. An objective function $f : \Pi \to \mathbb{R}^+$ associates a positive real number $f(\{c_1, ..., c_K\})$ with each partition. An anticlustering problem may be defined as follows: Find a feasible partition π^* such that $f(\pi^*) = \max\{f(\pi)|\forall\pi \in \Pi\}$.

The objective function diversity measures the sum of within-group distances. We compute the diversity value as the sum of $\{S_1, ..., S_K\}$, where S_k is the sum of all the weights of the edges whose endpoints are both vertices in cluster c_k:

$$diversity(\pi) = \sum_{k=1}^{K} \sum_{\substack{i,j \in c_k \\ i<j}} d_{ij} \tag{4}$$

The diversity of the feasible partition π_r from Figure 3 is computed in the following way: $diversity(\pi_r) = (d_{1,3} + d_{1,4} + d_{3,4}) + (d_{2,5} + d_{2,6} + d_{5,6})$.

While diversity-anticlustering considers all edges within all clusters, dispersion considers only one edge. It is the minimum distance, that appears within one of all clusters. We compute the dispersion value as $\min\{M_1, ..., M_K\}$, where M_k is the minimum distance of an edge whose endpoints are vertices in cluster c_k:

$$dispersion(\pi) = \min_{k \in \{1,...,K\}} \min_{\substack{i,j \in c_k \\ i<j}} d_{ij} \tag{5}$$

The dispersion of the feasible partition π_r from Figure 3 is computed in the following way: $dispersion(\pi_r) = min(min(d_{1,3}, d_{1,4}, d_{3,4}), min(d_{2,5}, d_{2,6}, d_{5,6})) = d_{2,6}$.

3 Complexity

In this thesis, we discuss heuristics and exact solution methods. Heuristics on the one hand only approximate a solution and thus, do not always find an optimal solution. The exact solution methods on the other hand find an optimal solution, but require exponential running time for the calculation of this problem. That is because finding the partition with the maximum diversity value is an NP-hard problem for $K \geq 2$ subsets and an arbitrary M-dimensional Euclidean problem [6]. Thus, they are impractical for big problem instances.

Further, we show that finding a partition with a maximum dispersion value, is also an NP-hard problem. We do this by showing that the corresponding decision problem is NP-complete. The decision version, which we refer to as MaxDP, is defined as follows:

Given are an instance (D, K) of a partitioning problem and a value Δ. Is there a partition of the N items into K equally-sized parts such that $dispersion \geq \Delta$?

MaxDP is in NP because one can non-deterministically guess the cluster assignments and then check in polynomial time whether the minimum distance between any two

elements, that are in the same cluster, is larger than or equal to Δ.

To show NP-hardness, we reduce the decision version of a known NP-complete problem to MaxDP. One problem, that we know to be NP-hard is a clustering problem. We define clustering problems similar to anticlustering problems, but we seek a partition with the minimum objective value. Find a feasible partition π^* such that $f(\pi^*) = \min\{f(\pi) \mid \forall \pi \in \Pi\}$. The specific problem we want to use has the following objective function:

$$f(\pi) = \max_{k \in \{1,...,K\}} \max_{\substack{i,j \in c_k \\ i<j}} d_{ij} \tag{6}$$

We compute the value of this function as $\max\{A_1, ..., A_K\}$, where A_k is the maximum distance of an edge whose endpoints are vertices in cluster c_k.

We refer to the decision version of this problem as MinMMC (minimum-maximum-maximum clustering). The goal is to decide whether there is a partition of N items into K subsets, such that $f \geq \Delta$ for a given instance (D, K) of a partitioning problem and a value Δ. For this problem, the subsets do not need to be from equal size, but they must contain at least one element.

Given an instance of MinMMC, we now transform it into an instance of the MaxDP-problem in polynomial time. Let $K \cdot N \times K \cdot N$ be the distance matrix D' with entries d'_{ij} defined as follows:

$$d'_{ij} = \begin{cases} -d_{ij} & \text{if } 1 \leq i < j \leq N \\ \max(d_{ij})_{1 \leq i < j \leq N} & \text{otherwise.} \end{cases} \tag{7}$$

In other words, we multiply the distance matrix D by -1 and add $(K-1)N$ vertices. All edges connected to these vertices have the maximum distance. These additional nodes do not change the dispersion, as they have either a greater or equal distance to all other elements in a cluster compared to the minimum distance in the cluster. They just act as "padding material", such that all clusters have the same amount of elements. An algorithm for MaxDP can thus be used to solve MinMMC, showing the NP-hardness of MaxDP.

MinMMC is NP-hard for $K \geq 3$ and an M-dimensional Euclidean instance with $M \geq 2$ [2]. If the elements from a given instance have only one attribute or need to be partitioned in less than three groups, there are algorithms, that can compute the optimal solution in polynomial time [2]. Our proof is therefore only valid for $K \geq 3$ and $M \geq 2$. Whether the algorithms for MinMMC in the mentioned cases are also useable for MaxDP requires further investigation.

4 Test Data

In the following sections, we use test data with the following properties: Every test set consists of a number of elements N and a number of groups K. We specify this information, that defines the instance size, in every context it is used. Every element in the test data has 3 attributes t_1, t_2 and t_3. These are uniformly distributed numbers, with $t \in [0, 100]$, $t_2 \in [0, 50]$ and $t_3 \in [0, 10]$.

All tests are run on a computer with an Intel® Core™i5-5300U CPU (Intel Corporation, Santa Clara, CA, USA) at 2.3 GHz with 8 GB of RAM.

5 Bicriterion Heuristic

5.1 Multiobjective Anticlustering

In this section, we introduce the bicriterion iterated local search (BILS) heuristic by Brusco et al. [3]. Firstly, we discuss multi-objective anticlustering. The BILS heuristic falls into this category, as it considers the optimization of diversity and dispersion at the same time. In general, there are no solutions for multiobjective problems that are optimal according to all considered objectives. For example, it is unclear, whether a partition π_1 with $diversity(\pi_1) = 1$ and $dispersion(\pi_1) = 4$ is better or worse than a partition π_2 with $diversity(\pi_2) = 2$ and $dispersion(\pi_2) = 3$.

One approach is to combine all objectives to one function, such that optimal solution verification is possible. But this requires applications, for which the exact priority of each objective is known. For the case of diversity and dispersion, Brusco et al. [3] have established a bicriterion combinatorial optimization model. In the following, we refer to it as bicriterion. The bicriterion $Z(\pi)$ defines w_1 and w_2 as the priority-weightings for $diversity(\pi)$ and $dispersion(\pi)$ as follows:

$$Z(\pi) = w_1\, diversity(\pi) + w_2\, dispersion(\pi)$$
$$\text{with } w_1, w_2 \in [0, 1] \text{ and } w_1 + w_2 = 1$$

Since two anticlustering objectives are combined, the goal is to find the partition π, that maximizes $Z(\pi)$. In this context, fixed values (knowledge of exact priorities) for w_1 and w_2 allow us the identification of optimal solutions regarding the bicriterion. Often these values are not fixed. Thus, we aim for a set of partitions, that is better than all other possible partitions. To compare partitions we use the definition of Pareto efficiency. For multiobjective anticlustering with q considered objectives $O = \{o_1, ...o_q\}$ Pareto efficiency is defined as a set containing only undominated partitions. Partition π_1 dominates π_2 if and only if

$$o_t(\pi_1) \geq o_t(\pi_2) \forall t \in \{1, ..., q\}$$

and for at least one $i \in \{1, ..., q\}$ the strict inequality $o_i(\pi_1) > o_i(\pi_2)$ holds. Thus, a

partition is contained in the Pareto efficient set, that considers diversity and dispersion has to satisfy one of the following conditions:

$$diversity(\pi_1) > diversity(\pi_2) \wedge dispersion(\pi_1) \geq dispersion(\pi_2)$$
$$diversity(\pi_1) \geq diversity(\pi_2) \wedge dispersion(\pi_1) > dispersion(\pi_2)$$

Although the BILS heuristic calculates the Pareto efficient set and not a single optimal solution, it uses the bicriterion with fixed values to detect local maxima and evaluate these according to the dominance relation.

5.2 Multistart Bicriterion Pairwise Interchange Heuristic

The BILS heuristic is divided into two parts. The first part is a multiple restart bicriterion pairwise interchange (MBPI) heuristic, which is useable as a stand-alone program. This heuristic approximates the Pareto efficient set for a multiobjective anticlustering problem, that considers diversity and dispersion. The required input is

- a distance matrix D, that contains all distances between the N elements, or an $N \times M$ information-matrix with the attributes of the elements, from which the distance matrix is derivable
- the number of groups K, to which the elements get assigned to. K is only valid, if the number of elements N is divisible by K
- the number of restarts R, which is decisive for the running time. More restarts lead to an improvement of the approximated solution
- (optional) a list W of priority-weightings w_1 for diversity. The entries of this list determine if the focus of the search lays on diversity or dispersion. An entry $w_1 = 1$ means a focus solely on diversity, while an entry $w_1 = 0$ means a focus solely on dispersion. Multiple entries in this list allow splitting the focus of the search. The standard is $W = \{.0000001, .00001, .0001, .001, .01, .1, .5, .99, .999, .999999\}$, which we use in all our tests.

The algorithm proceeds as follows. First, it initializes an empty Pareto set P. Then, for every restart, priority-weights are decided by picking a diversity weight w_1 with equal probability from W and computing the corresponding weight for dispersion as $w_2 = 1 - w_1$. Also, a random feasible partition π is generated by creating K different group labels, with $\frac{N}{K}$ occurrences each, which are randomly assigned to the elements. Now the bicriterion value of π is computed with the chosen weights. Via local search we determine the closest local maximum to π (based on the bicriterion with the used weights). This is accomplished by pairwise exchanging the group assignments of each element with each other element. If one of those exchanges improves the current bicriterion value, this exchange remains in π. Otherwise, this exchange is undone. If at least one exchange improved the bicriterion value of π during the search process, the search process starts again with this new, changed π. For all examined partitions,

including those whose exchange was undone, P is updated. This means if an examined partition is not dominated by any partition included in the current Pareto set, it is added. Furthermore, a partition $\pi' \in P$, that is dominated by the newly added partition, is excluded from P. The algorithm repeats R times with a new selection of priority-weights and a newly generated partition. At the end P, which has been updated in R iterations, is returned. In the following, we give pseudocode for the MBPI heuristic.

```
function MBPI(D, K, R, W)
    P = ∅
    for r = 1 to R do
        pick a random weight w_1 ∈ W
        w_2 = 1 − w_1
        initialize random partition π = {c_1, ..., c_K}
        Z* = w_1 diversity(π) + w_2 dispersion(π)
        exchanged = True                          ▷ start of local search component
        while exchanged = True do
            exchanged = False
            for n = 1 to N − 1 do
                for j = i + 1 to N do
                    set k : i ∈ c_k                ▷ k and l are set on the group-
                    set l : j ∈ c_l                ▷ indices of the goups of i and j
                    if k ≠ l then
                        c_k = c_k ∪ {j}\{i}
                        c_l = c_l ∪ {i}\{j}
                        update P with current π
                        Z = w_1 diversity(π) + w_2 dispersion(π)
                        if Z > Z* then
                            Z* = Z
                            exchanged = True
                        else
                            c_k = c_k ∪ {i}\{j}
                            c_l = c_l ∪ {j}\{i}
                        end if
                    end if
                end for
            end for
        end while                                 ▷ end of local search component
    end for
    return P
end function
```

5.3 Bicriterion Iterated Local Search

Multistart local search heuristics commonly fail to identify globally optimal (or even near-globally optimal) solutions as problem size increases [9] [10]. This fact also applies to iterated local search (ILS), but an ILS approach performs significantly better in comparison to a multistart approach [10]. Instead of generating a random feasible solution, an ILS approach uses an existing solution, changes it to some degree, and starts a local search. It is advantageuos to use an approximated solution as the existing solution [10] [3]. Thus, the BILS heuristic approximates a Pareto set with the MBPI heuristic as the first step. It requires the same input as the MBPI heuristic and an optional range factor $\xi = [\xi_1, \xi_2]$, decisive for the rate of change. The standard values are $\xi_1 = 0.05$ and $\xi_2 = 0.10$. A change of an existing partition with the use of ξ works as follows. For every pair of elements from different groups, their group assignment is exchanged with a probability between ξ_1 and ξ_2 (for example is the standard exchange rate 5-10%). The exact percentage is picked as a random uniform number in the range of $[\xi_1, \xi_2]$. So for every restart, we pick a random partition $\pi \in P$ (initially P is obtained from MBPI) and change it by use of ξ. After these changes on π, we use the local search component from the MBPI heuristic to find a local maximum according to π. Again we use all examined partitions to update P as we have seen in the MBPI heuristic. Half of the entered restarts $R/2$ are used for an initial approximation of the Pareto set computed by the MBPI heuristic and the remaining $R/2$ restarts are used for the iterative local search. Thus, at least 2 restarts are required and the number of restarts has to be even. A restart of the MBPI heuristic takes slightly less computation time because used partitions do not have to be perturbed first. As well as the MBPI heuristic the BILS heuristic returns P as the resulting Pareto set. In the following, we give pseudocode for the BILS heuristic.

function BILS(D, K, R, W, ξ_1, ξ_2)
  run MBPI with $R/2$ restarts and let P be the approximated Pareto set
  for $r = 1$ to $R/2$ **do**
    pick a random weight $w_1 \in W$
    $w_2 = 1 - w_1$
    pick a random neighborhood size ξ in range $[\xi_1, \xi_2]$
    pick a random partition π from P
    for $n = 1$ to $N - 1$ **do**
      for $j = i + 1$ to N **do**
        set $k : i \in c_k$
        set $l : j \in c_l$
        if $k \neq l$ **then**
          pick a uniform random number ρ in range $[1, 2]$
          if $\rho < \xi$ **then**
            $c_k = c_k \cup \{j\} \backslash \{i\}$
            $c_l = c_l \cup \{i\} \backslash \{j\}$
          end if
        end if
      end for
    end for
    local search(D, w_1, w_2, π) ▷ local search component from MBPI
  end for
  return P
end function

5.4 Implementation Details

A majority of the computing time in the BILS heuristic is needed to evaluate all bicriterion values. For this, the heuristic computes diversity and dispersion for every examined partition. Without external information about a partition, both of these computations consider all edges of an anticlustering instance. Thus, for the partition, that is randomly chosen at each restart, these computations are unavoidable. All other examined partitions differ only by one pairwise exchange in comparison to the partition examined directly before. So the information we use to compute the bicriterion for a new partition π_n, is the bicriterion value of the previously examined partition π_p, as well as the elements x and y involved in the pairwise exchange. We assume x is assigned to group c_x and y is assigned to group c_y in partition π_p and are exchanged in partition π_n. Then the computation of diversity is as follows:

$$diversity(\pi_n) = diversity(\pi_p) - \sum_{i \in c_x} d_{ix} - \sum_{i \in c_y} d_{iy} + \sum_{i \in c_x} d_{iy} + \sum_{i \in c_y} d_{ix}$$

We use the old diversity value and subtract all edge-weights from edges who have $x \in c_x$ or $y \in c_y$ as one endpoint and a vertex in the same group as the other endpoint. Then, we add all edge-weights of $x \in c_y$ and $y \in c_x$ analogous. Instead of considering all edges of the anticlustering instance, we now only look at edges connected to two of the elements.

For our computation of dispersion, we consider two questions. Firstly we examine the partition before the exchange and ask: Does one of the edges connected to x or y have the distance $dispersion(\pi_p)$? We refer to this statement as *Before*. After the pairwise exchange, we ask: Does one of the edges connected to x or y have the distance $dispersion(\pi_p)$ or a smaller distance? We refer to this statement as *After*. To satisfy *After*, the equation $dispersion(\pi_n) \leq dispersion(\pi_p)$ must hold. Moreover, all newly considered edges are connected to x and y. Thus, if *After* is true, we receive the new dispersion value, while investigating this question. Regarding both questions there are four different cases to examine for the computation:

1. *Before* and *After* are both false
   x and y were not responsible for the dispersion value in the exchange process, leading to $dispersion(\pi_n) = dispersion(\pi_p)$.

2. *Before* is false and *After* is true
   We find the new value $dispersion(\pi_n)$ in an edge from x or y to the other elements in their groups, after the exchange, because *After* is true.

3. *Before* is true and *After* is false
   In this case, we need to consider all edges of the anticlustering instance. Beforehand, x and y were responsible for the dispersion value. As they were not responsible for the value after the exchange, every edge within a cluster could be responsible for the new dispersion value.

4. *Before* and *After* are both true
   Case 4 is analogous to case 2.

Only case 3. can possibly improve the dispersion value, but requires the most computation time. Using test data with 20, 30, 40, 50 and 100 elements and 10 groups, we determined the occurrence rate of this case. In Table 2 we can see, that an increased element size leads to a smaller occurrence rate of Case 3. A cause for this is, that the probability of the exchanged elements x and y being responsible for the dispersion value drops with an increasing amount of elements. The cases 1., 2. and 3. require only the examination of the edges connected to x and y. In comparison to considering all edges for the computation of the bicriterion, the computation time has decreased significantly.

Elements	Case 1.	Case 2.	Case 3.	Case 4.	Total	Case 3. in %
20	180 468	214 235	17 601	78 746	491 050	3.58
30	445 435	1 306 920	45 645	215 894	2 013 894	2.27
40	906 141	2 845 433	51 242	337 496	4 140 312	1.24
50	1 313 856	5 387 835	57 992	483 836	7 243 519	0.80
100	7 453 104	24 457 118	169 006	1 068 157	33 147 385	0.51

Table 2: Case occurrence rates of the BILS heuristic with 1000 restarts and 10 groups.

5.5 Limitations of the BILS Heuristic

An ILS approach can become a state of the art algorithm for a specific problem if the local search component and factor of perturbation (in the BILS heuristic this factor is ξ) are correctly chosen. Nevertheless, an ILS approach is always a heuristic and thus, not reliably able to find optimal solutions. Besides, the BILS heuristic has no approximation factor. This means there is no guarantee, that a result has at least a fixed proportion of the value of the optimal solution. Furthermore, the probability of finding a near-optimal solution with an ILS approach decreases, as instance size increases [10]. In section 7 we discuss, how the solutions of the BILS heuristic perform on average in comparison to optimal solutions.

With an increase of the instance size, the computation time of the BILS heuristic also increases. The number of feasible partitions grows exponentially with the number of elements. To be specific, it is equal to the Stirling number of the second kind [3]. Therefore there are in general more partitions, that improve a currently examined partition during the local search. This leads to a more frequent use of the local search component during one restart of the heuristic.

The computation time of the BILS heuristic does not solely depend on the instance size. The users can affect the computation time by setting the number of restarts, which specifies the "thoroughness" of the search. We demonstrate the differences, by running the BILS heuristic with 2, 10, 100, 1000, and 10000 restarts on test data. The test set consists of 100 elements, which have to be divided into 10 groups. Figure 4 shows the diversity and dispersion values of the resulting Pareto sets, by plotting each entry of these sets.

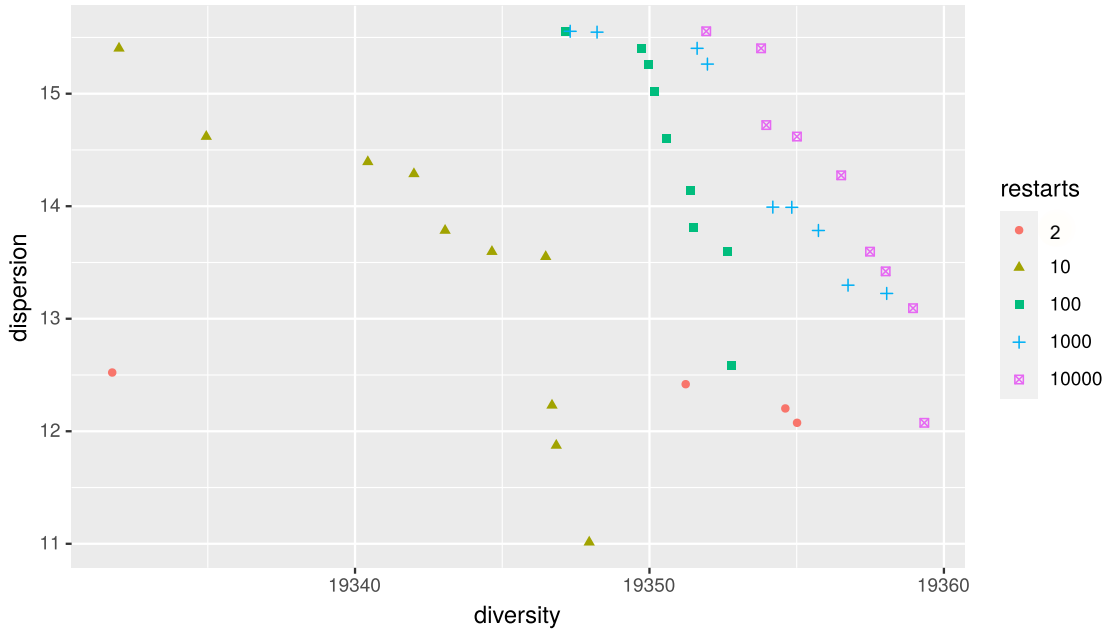

Figure 4: BILS heuristic using different numbers of restarts.

In general, the solution of the BILS heuristic gets closer to the optimal solution, when more restarts are used [3]. In Figure 4 you can see, that partitions of the Pareto set obtained by the BILS heuristic with 2 restarts dominate partitions obtained with 10 restarts. So the number of restarts provides only a higher probability of better solutions.

The average time of a single restart does not depend on the total number of restarts. A restart requires time dependent on how distant an initial partition is to the next local maximum and the size of the growing and shrinking Pareto set. A lower number of restarts is crucial for reasonable computation time for bigger instances but comes with a loss of solution quality.

5.6 BILS on 1-Dimensionial Instances

We face two problems using the BILS heuristic as instance size increases. The quality of a resulting approximation decreases and the computation time increases. Another special case are 1-dimensional instances (1-dimensional Euclidean problem). The BILS heuristic is unsuitable for even small instances of this kind. The crux of this problem is, that the heuristic returns, and more importantly works, with a Pareto containing a large number of elements. The size of the Pareto set affects the computation time in the updating process. Every newly examined partition is compared with all partitions in the current Pareto set.

For an 1-dimensional Euclidean problem, most of the partitions in the corresponding Pareto set are not unique. They share the same diversity and dispersion value with other partitions. To better understand why the partitions are not unique in diversity, we look at the example anticlustering problem in Figure 5. The problem consists of 4 elements and 2 groups. The elements have one attribute with an arbitrary value (w.l.o.g these values are in ascending order of the vertex indices). For this example there are three feasible partitions π^1, π^2 and π^3. Their groups are recognizeable by connected edges.

If we combine the edges of each partition by superimposing them, we observe that partition π^2 and π^3 both have a combined edge from vertex 1 to vertex 4. In the process of combining, we also notice, that for both of these partitions, their edges only overlap in the area from vertex 2 to vertex 3. Thus, the combined edge distances of these partitions are equal: $d_{1,3} + d_{2,4} = d_{1,4} + d_{2,3}$. Per definition of diversity, this is equivalent to $diversity(\pi_2) = diversity(\pi_3)$. This applies to arbitrary distances between the vertices in Figure 5. With an increase of the problem size, partitions with the same diversity value occurre more frequently. We evaluate 10 tests with 12 elements and 4 groups. For an instance with these properties there exist 15400 feasible partitions. All unique diversity values occurre 25 times on average. While this would not be problematic, the highest diversity values tend to occurre more often. The maximum diversity value appears 576 times in all of the tests. Occurrence rates of other diversity values were not fixed in these tests. We presume that the occurrences of the highest diversity value are a substantial proportion for all 1-dimensional anticlustering instances.

To understand why dispersion values are not unique, we take a look at the definition of dispersion. Dispersion is the minimal distance of an edge, which is within one group of a partition. The number of different dispersion values is restricted to the number of

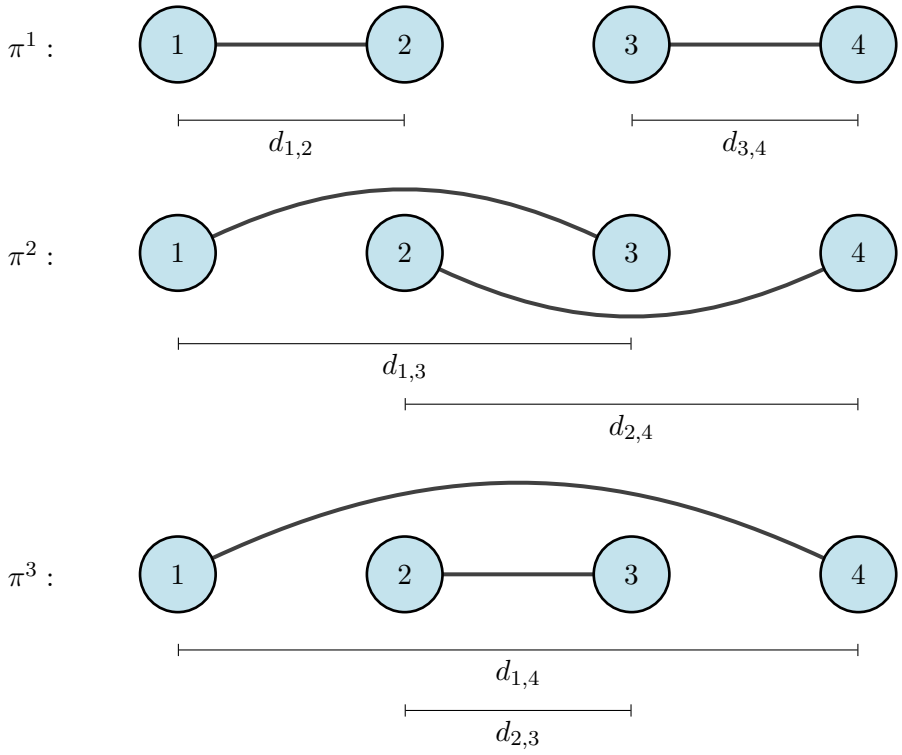

Figure 5: Feasible partitions π^1, π^2 and π^3.

$N(N-1)$ edges. To be more specific, it is restricted to the number of all unique edge distances. In the following, we assume that all edge distances are unique. In this case, some of these edges are excludable. For K groups, the group size of each group is $\frac{N}{K}$. So in a partition there are always $N((\frac{N}{K})-1)$ edges used. As the minimum distance of these edges is the dispersion value, the $N((\frac{N}{K})-1)-1$ edges with the longest distances can not be responsible for this. The test data with 12 elements and 4 groups has a problem graph consisting of 12 vertices and 132 edges. $12((\frac{12}{4})-1)-1 = 23$ edges can dispersion value, leading to 109 edges possibly responsible for this value.

Because of a high occurrence rate of the highest diversity value in different partitions, it is likely, that the BILS heuristic finds a partition with this value multiple times. Also, the number of different possible dispersion values is relatively low. Therefore, the occurrence rate of partitions sharing the same value in diversity and dispersion is still high. The BILS heuristic stores all those partitions in the Pareto set according to the definition of Pareto efficiency. One possibility to avoid this problem is the usage of a stricter dominance relation than the Pareto efficiency. If partitions with equal values dominate each other, only one of them is stored in the Pareto set of the BILS heuristic. As mentioned in section 3, it is unclear, if this specific problem is NP-hard for dispersion or if efficient algorithms for solving exist.

6 Exchange Method

In this section, we introduce the exchange method as another heuristic approach to our problem and compare it to the BILS heuristic. The exchange procedure was originally proposed by Weitz and Lakshminarayanan et al. [14]. This was later adapted to the exchange method by Papenberg et al. [11]. It is part of the anticlust package from Papenberg and is available as an open source software extension for the programming language R. This heuristic focusses solely on one objective, so we only return one optimal solution instead of a Pareto set. It can be used to optimize any objective function quantifying group similarity [11]. The approach is similar, but not the same, as one restart of the MBPI heuristic. Firstly, we generate a feasible partition in the same manner as in the MBPI heuristic. Then, we simulate pairwise exchanges for one of the elements of the partition with all elements from other groups. We carry out one of the simulated exchanges, that lead to the highest increase of the desired objective. It is possible, that no exchange improves the current objective value. In this case, the partition stays unchanged. The exchange method returns a partition as solution after this process is repeated for all elements.

6.1 Comparing Exchange Method and BILS Heuristic

To compare the exchange method with the BILS heuristic, we use 10 tests with 100 elements and 10 groups. In these tests, the average computation time of the exchange method with diversity as objective is approximately 1.61 seconds. A single restart of the BILS heuristic has a computation time of 0.1 seconds. Since at least 2 restarts are required to execute the BILS heuristic, we compare the exchange method with the 2-restart-BILS heuristic. For the sake of simplicity, we refer to the partition with the highest diversity value in the Pareto set obtained through the 2-restart-BILS heuristic, as BILS solution for the rest of this section.

Although the 2-restart-BILS heuristic requires only an eighth of the computation time in comparison to the exchange method, the 2-restart-BILS heuristic performs better. In the tests, all diversity values of the BILS solutions are greater than the diversity values of all exchange method solutions. On average the solutions of the exchange method have a diversity value of 20171. The average diversity value of a BILS solution is 20196. We demonstrate our test results by plotting one Pareto set and all solutions of the exchange method in Figure 6. For the sake of clarity, we did not plot all Pareto sets. Each entry of the Pareto set obtained through the BILS heuristic and each solution of the exchange method is a single point with a diversity and dispersion value.

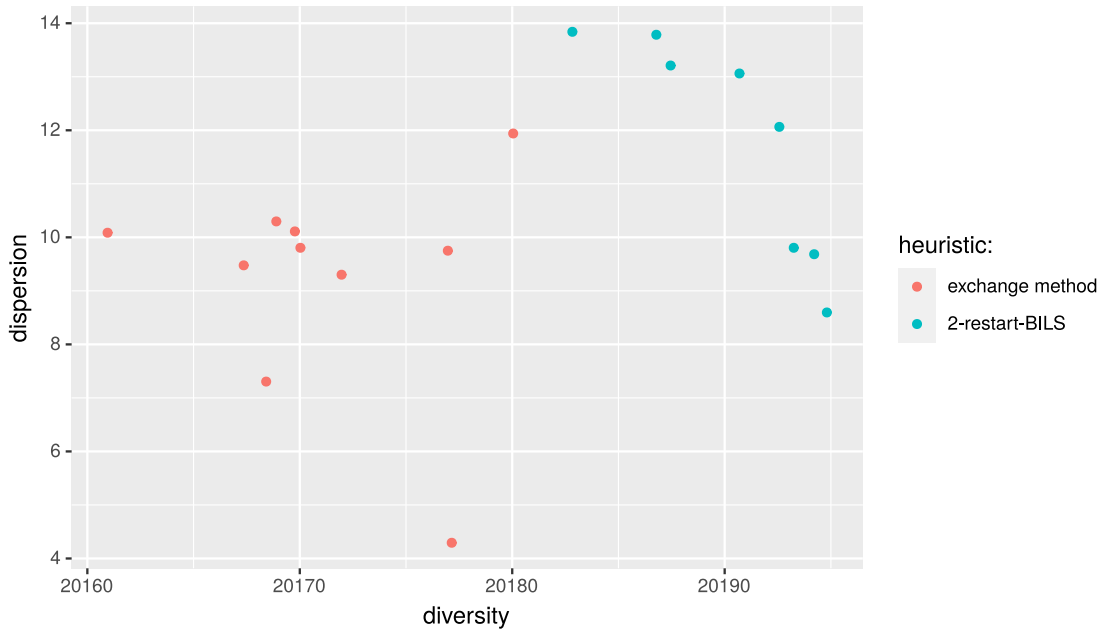

Figure 6: Diversity and dispersion values of solutions obtained through both heuristics.

For further investigation, we repeated the tests for 50, 200 and 300 elements and visualize the results in Table 3. In all tests, the 2-restart-BILS heuristic performs slightly better than the exchange method. As problem size increases the computation time of the exchange method increases faster than the computation time of 2-restart-BILS heuristic. We presume that this is generally the case and that there is a problem size at which the exchange method is faster than the 2-restart-BILS heuristic.

	computation time in seconds		average diversity value	
elements	exchange method	2-restart-BILS	exchange method	2-restart-BILS
50	0.19	0.02	4 980	4 988
100	1.61	0.20	20 171	20 196
200	10.83	1.91	82 885	82 935
300	41.46	11.05	198 907	199 003

Table 3: Computation time and average diversity values of both approaches.

As dispersion is, in this case, not the goal of the exchange method, we do not expect to find high values in this objective among the solutions. Nonetheless, it is interesting, that the average dispersion value of the solutions is at least 30% lower than the highest dispersion value contained in the Pareto set. We conclude that the 2-restart-BILS heuristic is, at least for a problem size with less than 300 elements, better suited for this problem.

7 Exact Solution Methods

7.1 Pareto Set Via Complete Enumeration

The simplest approach to achieve an exact Pareto set is to examine all feasible partitions via complete enumeration and save all partitions, that are not Pareto dominated by any other partition. Unfortunately, the number of all feasible partitions is enormous for practical values of N and K [3]. Thus, the computation time of complete enumeration is too high in praxis.

Running test anticlustering instances with 30 elements and 10 groups shows, that the BILS heuristic reliably finds exact solutions for this problem size. For 20 test sets, the BILS heuristic always returns the exact Pareto set by setting the number of restarts on 10000. The complete enumeration needs a computation time of nearly 7 hours to return the exact Pareto set. We observe that the BILS heuristic is not reliable on tests with 40 elements and 10 groups, because all solutions resulting from the same given test data are different. As computation time is too long, we do not use the complete enumeration on this problem size.

7.2 Relation of Diversity and Dispersion in an Exact Pareto Set

We reuse the test results from the complete enumeration approach with 30 elements and 10 groups to take a closer look at the relation between diversity and dispersion on an exact Pareto set.

Test	Diversity	Dispersion
1	1733.69	17.96
2	1648.70	23.45
3	1476.22	28.37
4	1417.87	28.63
5	1577.47	19.16
6	1481.63	25.09
7	1718.60	17.75
8	1520.68	16.96
9	1354.44	21.42
10	1543.58	29.27

Table 4: Objective values of partitions with the highest diversity value in the exact Pareto set.

Test	Diversity	Dispersion
1	1723.34	34.60
2	1634.24	38.70
3	1469.95	29.79
4	1411.31	33.76
5	1567.17	29.98
6	1480.28	29.78
7	1702.85	33.05
8	1508.11	28.59
9	1346.16	26.79
10	1531.81	36.73

Table 5: Objective values of partitions with the highest dispersion value, taking into account all partitions that are within 1% of the maximum diversity value of the corresponding exact Pareto set.

We observe in Table 4 and 5, that trading off 1% of the maximum diversity value can increase the dispersion value drastically. In this 1% range, we find partitions with an on average 46% higher dispersion value compared to the partition with the highest diver-

sity value. Test data with various values for N and K shows, that similar trade-offs are possible for non-exact Pareto sets. In these tests, we also observe, that the partition with the highest value in dispersion has, in the worst-case, only a 2% lower diversity value compared to the maximum. If the focus of an application does not solely lie on diversity, one may want to consider an exact Pareto set or an approximated Pareto set for a possible trade-off.

7.3 Integer Linear Programming

Another approach to receive optimal solutions is integer linear programming (ILP). It is a mathematical optimization model based on linear constraints and objectives. ILP approaches often improve the computation time in comparison to a complete enumeration [1] [8]. While complete enumeration would examine all solutions, ILP do not necessarily need to check all solutions in the solution space. To what extend the solution space can be restricted depends on the problem [4].

We need to choose suitable integer variables, with which we can formulate an ILP model consisting of an objective function and constraints. An ILP model represents the objective function as a linear function and the constraints as a system of linear inequalities [11]. An ILP solver acts as a "black box" that is guaranteed to return an optimal solution [11].

7.3.1 Diversity ILP

In this section, we introduce an ILP model for the problem of finding the maximum diversity. The ILP formulation was originally presented by Grötschel and Wakabayashi et al. [8] and later extended by Papenberg and Klau et al. [11].

Firstly we need to explain the integer variables, we use for the ILP formulation. The variables x_{ij} encode whether two elements i and j are connected by an edge and thus, are in the same group:

$$x_{ij} = \begin{cases} 1 & \text{if } v_i \in c_k \wedge v_j \in c_k \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

To find the partition with the maximum diversity value using these integer variables, we use the following ILP formulation:

$$max \sum_{1 \leq i < j \leq N} d_{ij} x_{ij} \tag{9}$$

$$\text{s.t.} \quad -x_{ij} + x_{ik} + x_{jk} \leq 1, \qquad \forall\, 1 \leq i < j < k \leq N, \tag{10}$$

$$x_{ij} - x_{ik} + x_{jk} \leq 1, \qquad \forall\, 1 \leq i < j < k \leq N, \tag{11}$$

$$x_{ij} + x_{ik} - x_{jk} \leq 1, \qquad \forall\, 1 \leq i < j < k \leq N, \tag{12}$$

$$\sum_{1 \leq i < j \leq N1}^{K} x_{ij} + \sum_{1 \leq k < i \leq N1}^{K} x_{ki} = \frac{N}{K} - 1, \qquad \forall i \in \{1, ..., N\}, \tag{13}$$

$$x_{ij} \in \{0, 1\}, \forall\, 1 \leq i < j \leq N$$

The objective in (9) is to maximize the sum of pairwise distances for a given anticlustering problem. Without any constraints, we would set every $x_{ij} = 1$ by assigning all elements to the same group. To ensure that the assignment is feasible, we use the constraints (10) to (13). Constraints (10) to (12) guarantee that there are no conflict triples in the graph [8]. Put simply, there are no edges between elements from different groups. Constraint (13) ensures equal group sizes by enforcing each element to have exactly $\frac{N}{K} - 1$ edges. The objective in (9) in combination with the constraints maximizes the diversity.

For Table 6 we computed an optimal solution regarding diversity with the introduced ILP. The test data consists of 50 elements and 10 groups. The compuatation exceeds 4 hours, so we do not investigate larger problem sizes. For this problem size BILS heuristic finds solutions with nearly optimal diversity values in 200 seconds.

BILS heuristic diversity	maximum diversity
4994.00	4995.64
4994.45	
4993.78	
4994.09	
4994.21	

Table 6: Diversity values obtained through the the BILS heuristic with 10000 restarts compared to an exact solution obtained through the ILP

7.3.2 Dispersion ILP

In this section, we discuss an ILP model for the problem of finding the maximum dispersion. Fernández et al. have introduced an ILP model for a more specific problem [7]. We adapt a more general approach of this model for our purposes.

Firstly we explain the variables, we use for our ILP model. The variables y_{ik} encode whether element i is contained in group c_k:

$$y_{ik} = \begin{cases} 1 & \text{if } v_i \in c_k \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The variables z_{ijk} encode whether element i and j are contained in group c_k:

$$z_{ijk} = \begin{cases} 1 & \text{if } v_i \in c_k \wedge v_j \in c_k \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

To find the partition with the maximum dispersion value using these integer variables, we use the following ILP formulation:

$$max \quad MinDis \tag{16}$$

$$\text{s.t.} \qquad \sum_{1 \leq K} y_{ik} = 1, \qquad \forall\, 1 \leq i \leq N, \tag{17}$$

$$\sum_{1 \leq N} y_{ik} = \frac{N}{K}, \qquad \forall\, 1 \leq k \leq K, \tag{18}$$

$$MinDis \leq d_{ij} z_{ijk} + M(1 - z_{ijk}), \qquad 1 \leq i < j \leq N, 1 \leq k \leq K \tag{19}$$

$$y_{ik} + y_{jk} \leq 1 + z_{ijk}, \qquad 1 \leq i < j \leq n, 1 \leq k \leq K \tag{20}$$

$$MinDis \in \mathbb{R},\ y_{ik}, z_{ijk} \in \{0, 1\}, \forall\, 1 \leq i < j \leq N$$

The objective in (16) is to maximize the variable $MinDis$, which represents the minimal distance of an edge within one partition. Constraint (17) ensures, that every element is assigned to exactly one group. Constraint (18) guarantees, that every group contains exactly $\frac{N}{K}$ elements. Constraint (19) ensures, that the variable MinDis is set on the shortest edge, that is used in the partition. If $z_{ijk} = 1$, then the edge is used and $MinDis \leq d_{ij} z_{ijk}$ holds. If $z_{ijk} = 0$, then $MinDis \leq d_{ij} z_{ijk} + M$ holds. We set the variable M on the maximum distance of all edges. Thus, this constraint does not affect $MinDis$ if $z_{ijk} = 0$. We use constraint (20) to prevent, that $z_{ijk} = 0$, while both $y_{ik} = y_{jk} = 1$. The objective in (9) in combination with the constraints maximizes the dispersion.

We evaluate this ILP on test data consisting of 50 elements and 10 groups. The compuatation exceeds 3 hours, so we do not investigate larger problem sizes. The BILS heuristic, with 10000 restarts, finds the optimal solution on such instances in about 200 seconds. On all tests the Pareto set obtained through the BILS heuristic, has a partition with the maximum dispersion.

8 Discussion

During this bachelor thesis, we implemented the BILS heuristic and evaluated it on simulated data. In contrast to other heuristics for anticlustering, it considers multiple objectives at the same time and returns not a single solution, but a list of possibly useful solutions.

Before considering the BILS heuristic as an approach for an anticlustering problem, these two cases should be taken into account.

- If elements should be divided into two groups and a high dispersion is desired, there might exist an efficient, exact algorithm. Brucker et al. [2] introduced such an algorithm for finding the maximum dispersion, without enforcing equal group sizes for the anticlustering problem. Optimal solutions for one objective are beneficial in order to determine a Pareto set, in which two objectives are considered. [3].
- If the elements of an anticlustering problem have only one attribute and a high dispersion is desired, there also might exist an efficient, exact algorithm. Again Brucker et al. [2] introduced such an algorithm, without enforcing equal group sizes. In this case, the BILS heuristic should definitely not be used, as it finds too many solutions, which affects the computation time drastically.

Except for these two specific cases, the BILS heuristic performs well. For small instances, it achieves better solutions than the exchange method with a lower computation time, which is adjustable through the number of restarts. Compared to exact solutions, the BILS heuristic returns near-optimal solutions for problem sizes with 50 elements. In regard to the problem size the computation time of the BILS heuristic increases faster than the computation time of the exchange method. As the BILS heuristic is an iterated local search approach, the probability of finding a near-optimal solution decreases, as problem size increases [10]. Furthermore, this heuristic has no approximation factor and thus, there is no guarantee on the quality of a solution. To better quantify the quality of solutions one could test our algorithm on real data and compare it to the optimal solutions.

Regardless of the choice of the algorithm used for anticlusering, it is often beneficial to investigate a Pareto set for a given problem. In general, a solution with the maximum value in one objective is not what is desireable for a problem [3]. It is nearly always possible to trade of a small amount of diversity for a huge amount of dispersion.

For further works on this subject, we suggest considering other anticlustering objectives, in addition to those presented in this thesis and evaluation on real data.

References

[1] Sebastian Böcker, Sebastian Briesemeister, and Gunnar W Klau. "Exact algorithms for cluster editing: Evaluation and experiments". In: *Algorithmica* 60.2 (2011), pp. 316–334.

[2] Peter Brucker. "On the complexity of clustering problems". In: *Optimization and operations research*. Springer, 1978, pp. 45–54.

[3] Michael J Brusco, J Dennis Cradit, and Douglas Steinley. "Combining diversity and dispersion criteria for anticlustering: A bicriterion approach". In: *British Journal of Mathematical and Statistical Psychology* (2019).

[4] Pei-Hua Chen. "Should We Stop Developing Heuristics and Only Rely on Mixed Integer Programming Solvers in Automated Test Assembly? A Rejoinder to van der Linden and Li (2016)". In: *Applied psychological measurement* 41.3 (2017), pp. 227–240.

[5] Jacques Desrosiers, Nenad Mladenović, and Daniel Villeneuve. "Design of balanced MBA student teams". In: *Journal of the Operational Research Society* 56.1 (2005), pp. 60–66.

[6] Thomas A Feo and Mallek Khellaf. "A class of bounded approximation algorithms for graph partitioning". In: *Networks* 20.2 (1990), pp. 181–195.

[7] Elena Fernández, Jörg Kalcsics, and Stefan Nickel. "The maximum dispersion problem". In: *Omega* 41.4 (2013), pp. 721–730.

[8] Martin Grötschel and Yoshiko Wakabayashi. "A cutting plane algorithm for a clustering problem". In: *Mathematical Programming* 45.1-3 (1989), pp. 59–96.

[9] David S Johnson and Lylle A McGeoch. "Local search in combinatorial optimization". In: *The Traveling Salesman Problem: A Case Study in Local Optimization* (1997), pp. 215–310.

[10] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. "Iterated local search". In: *Handbook of metaheuristics*. Springer, 2003, pp. 320–353.

[11] Martin Papenberg and Gunnar W Klau. "Using anticlustering to partition data sets into equivalent parts". In: *Psychological methods* ().

[12] H Späth. "Anticlustering: maximizing the variance criterion". In: *Control and Cybernetics* 15.2 (1986), pp. 213–218.

[13] Venceslav Valev. "Set partition principles". In: *Transactions of the Ninth Prague Conference on Information Theory, Statistical Decision Functions, and Random Processes,(Prague, 1982)*. 1983, p. 251.

[14] RR Weitz and S Lakshminarayanan. "An empirical comparison of heuristic methods for creating maximally diverse groups". In: *Journal of the operational Research Society* 49.6 (1998), pp. 635–646.

[15] Rui Xu and Don Wunsch. *Clustering*. Vol. 10. John Wiley & Sons, 2008.

A Source Code

The source code for this thesis is accessible at: https://github.com/ManaLama/anticlust