BUDAPESTI CORVINUS EGYETEM

Doctoral School of Economics, Business and Informatics

Thesis Synopsis

Gábor Kondor

One-sided matching markets in balanced clustering approach

Ph.D. Dissertation

Advisor:

Ágnes Vidovics-Dancs Ph.D.

Associate Professor

Budapest, 2022

Department of Finance

Thesis Synopsis

Gábor Kondor

One-sided matching markets in balanced clustering approach

Ph.D. Dissertation

Advisor:

Ágnes Vidovics-Dancs Ph.D.

Associate Professor

Contents

Chapter 1

Research and topic selection background

Economics study how markets allocate resources. On matching markets, there is no or there is only partially a price system which determines allocations. In this case, the matching formed is determined by the mechanisms that govern the market. (Nobel Prize, 2012b)

The foundations of the theory of matching markets were established by Gale and Shapley (1962)[1], who proposed the concept of *stability* to solve the matching problem. According to this, in case of pairs, the task is to create a matching such that there are no two people in different pairs who would prefer each other to their actual partner. One of the basic models of one-sided matching markets is the *stable roommates problem* defined by Gale and Shapley (1962), in which students have strict preference order on the others and the goal is to form a stable matching, if it exists.

An important application of one-sided matching markets is the kidney paired donation (see Biró (2006)). In this case market participants are donor-recipient pairs, that is a pair of a patient with renal disease and a willing live donor who are incompatible with each other, and thus, the renal transplantation cannot be executed. The aim is to match donor-recipient pairs, such that renal transplantation between the pairs are viable - unfortunately, because of some medical tests taken later it might turn out that any of

[1]Later, the theory served as a base for several applications, for example the assignment of new doctors to hospitals, students to schools, and human organs for transplant to recipients. In 2012, the Nobel Memorial Prize in Economic Sciences was awarded to Alvin E. Roth and Lloyd Shapley for for the theory of stable allocations and the practice of market design (Nobel Prize, 2012a).

the matched pairs is actually incompatible, and the transplantation in that pair cannot be carried out.

Morrill (2010) suggested an alternative approach, the Pareto efficiency, to solve one-sided matching problems. He claims that stability ignores the key physical constraint that roommates require a room and, therefore, it is too restrictive. Since once a roommate assignment is formed, two students in different rooms cannot in a unilateral manner push their roommate out of the room, and thus, the new pair cannot be created. It is especially important in case of kidney exchange where after an assignment has been made, subsequent tests may determine that a patient and donor are incompatible.

The approaches mentioned so far aim to create matching of pairs, however, there are applications in which larger groups are considered. For example, some kidney exchange programs allow for groups of three (Biró, 2006), and in case of roommates there are often rooms of 3 or 4 places. The version of roommates problem in which three students have to be assigned to a room is called 3-dimensional roommates problem (3D-SR). According to our knowledge, with the exception of 2-stable matching of Arkin et al. (2009), all of the 3-dimensional approaches lead to NP-complete problems. It means that for these problems, from a complexity theory perspective, there is no efficient way to determine the solution, and so, in practice for instances of larger sizes we are not able to solve the problems. Also, we note that we know of only one related approach in the literature (the result of Lam and Plaxton (2019) for complete cyclic lists) which considers higher dimensions.

For kidney exchanges, Segev et al. (2005) studies an approach, also applied in practice, which is analogous to a weighted matching problem (Biró, 2006). The authors, based on simulations, claim that their method applied on the country level performs better than the "first accept" scheme used by some centers and regions.

Our goals in the dissertation are the following:

- Extend the weighted matching approach in the context of one-sided matching to problems of arbitrary group sizes. In particular, we would like to introduce a framework in which we can study both minimization and maximization goals for the problem.
- Derive complexity results for the defined problems, that is investigate whether the problems can be solved efficiently from a theoretical point of view.

- Examine the practical tractability and properties of the problem through extensive simulations and evaluating the results of heuristics from the literature and algorithms defined by us.

We utilize the m-dimensional matching problem to extend the weighted matching problem to arbitrary group size of m. This views the task of creating groups as a graph partitioning problem, which also results in a Pareto efficient solution. In the dissertation, we focus on a special version of this, in which participants are represented by points in an Euclidean space and their relations are described by the Euclidean distances between them. The aim is to create groups of size m, such that, depending on the goal, the sum of squared distances within groups is either minimal or maximal.

We treat the problem described in the Euclidean space as matching roommates, or alternatively, assigning students to rooms in a college and hence, call it *m-roommates problem*[2,3]. Dimensions of the Euclidean space represent the characteristics of students and the Euclidean distances indicate whether the students would be *good* roommates to each other. The aim is to assign students of a college to rooms of equal sizes, such that it is the best for the students.

For the goal of assignment we consider two vastly different approaches based on different views. In the first, we assume creating groups with similar students is desirable. This approach is supported by the possibly better connections between people with similar interests and characteristics. In this case we consider a minimization problem and we form homogeneous groups, or clusters. In the second approach, emphasizing the role of diversity which is becoming more and more important nowadays, we wish to assign roommates with varying characteristics as much as possible. In this case we face a maximization problem in order to create heterogeneous groups.

A cluster is a group of similar elements. This means that, strictly speaking, in our case we may call only the minimization versions of the m-dimensional matching and m-roommates problems as balanced clustering problems, that is a problem in which the aim is to create groups of equal sizes with similar elements. However, there are approaches

[2]Note that the m-roommates problem might look similar to the stable roommates problem of Gale and Shapley (1962), but in our case doesn't involve preference orders and the goal is not to create stable matchings.

[3]We called the same problem k-roommates problem in my conference lecture Kondor (2018) and my thesis submitted to the Scientific Students' Associations Conference (TDK, Tudományos Diákköri Konferencia). However, to be consistent with problems with the broader literature we changed the parameter in the name to m to denote the uniform size of groups.

in the literature (see, e.g. the k-partitioning problem considered by Feo and Khellaf (1990) and Feo et al. (1992)) where the distances between the elements are interpreted as similarity scores, and hence, clustering is achieved trough maximization. Because of this, and also for the sake of simplicity, in the dissertation we treat the notion *clustering* in a more flexible way, and we look at *clusters* as groups formed with the aim of solving a well-defined optimization goal be it either minimization or maximization. According to this, we refer to the problems of creating groups of equal sizes as *balanced clustering problems* in the dissertation.

In Chapter 2 we review the applications related to balanced clustering.

In Chapter 3 we first illustrate the complexity of the m-roommates problem by calculating the number of possible roommate assignments. Then, we give an introduction to the theory and definitions of computational complexity and review the complexity of related grouping problems. Finally, we extend the existing complexity results in general dimension.

In Chapter 4 we review the stable roommates problem, its various versions for 3-dimensional matching and the complexity of these approaches. Then, we give a formal definition of the m-roommates problem and discuss the benefits and drawbacks of this model compared to the stable roommates problem.

In Chapter 5 we present algorithms, which provide feasible solutions, have polynomial running times and provide some kind of guarantees for the suboptimality of the solution. Two kind of approaches are considered, approximation methods and cone optimization.

In Chapter 6 we review the most important heuristic approaches in the literature and we also construct new methods. We consider two groups of algorithms. In the first one we review algorithms related to conventional cluster analysis which are not guaranteed to provide clusters of equal sizes. For being able to apply these to the m-roommates problem we introduce 6 heuristics to make the clusters balanced, which are tested in Chapter 7. In the second group of methods we review methods which provide balanced clusters by default. Some of these share the property that they try to improve the solution by swapping elements in different groups. We contribute to this group by reviewing the possibility of larger swaps, and we introduce three heuristics which try to make improvements by swaps of three elements. These are also tested in Chapter 7.

In Chapter 7 we investigate the problem in a broad framework through multiple steps.

In the literature, usually only one version of the optimization problem is considered, either the minimization or the maximization, and only a smaller set of heuristics is tested. In the dissertation we take a joint view, and we include a broad set of algorithms which we set on smaller instances, on real data, and on instances of large sizes.

In Chapter 8 we conclude.

Chapter 2

Methods used in the dissertation

2.1 Defining m-dimensional matching problems and proof by reduction

In Chapter 3 of the dissertation we extend the existing general dimension hardness results of creating balanced groups. For this, we use balanced MSSC as a starting point. According to Huygen's theorem[1] the squared distances of points in a cluster from the centroid of the cluster is equal to the squared distances of the points from each other divided by two times the cardinality of the cluster. In a simple manner, formally for clusters $C_1, \dots, C_k$ of equal size m

$$\sum_{s=1}^{k} \sum_{x_i \in C_s} \left\| x_i - \Big(\frac{1}{m} \sum\nolimits_{x_j \in C_s} x_j\Big) \right\|^2 = \frac{1}{2m} \sum_{s=1}^{k} \sum_{x_i, x_j \in C_s} \|x_i - x_j\|^2.$$

Based on this, we can reformulate the problem so that in the objective function we use Euclidean distances between the points. To generalize the problem, we replace the Euclidean distance by a distance measure based on the ℓ_p norm, and besides the minimization version of the problem we also consider the maximization version. We formally define these problems by Optimization problem 2.1.1.

Optimization problem 2.1.1 pMin-mDM (pMax-mDM).

Input: $C = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ *m, p and $k = n/m$ positive integers.*

Output: A partition of C into disjoint sets $C_1, \dots, C_k$, such that $|C_i| = m, 1 \leq i \leq k$ and

[1]Huygen's theorem was proved by Edwards and Cavalli-Sforza (1965) first (Novick, 2009).

the objective function

$$\sum_{s=1}^{k} \sum_{x_i \in C_s} \sum_{x_j \in C_s} \|x_i - x_j\|_p^p$$

is minimized (maximized).

To show the NP-hardness of the optimization problem we show that the decision version of it is NP-complete (Ausiello et al., 2003). For this, a reduction from another NP-complete decision problem is used. The idea behind this proof technique is the following. Given an NP-complete decision problem $\mathcal{A}$, if we would like to prove that another decision problem $\mathcal{B}$ is NP-complete, we have to show the following to things. First, that $\mathcal{B}$ is in NP. Second, we have to show that there is a polynomial-time reduction from any instance of $\mathcal{A}$ to an instance of $\mathcal{B}$. In this way, if we were able to solve $\mathcal{B}$ then we could also solve $\mathcal{A}$. However, we know that $\mathcal{A}$ is NP-complete, and thus, $\mathcal{B}$ is at least as hard as $\mathcal{A}$.

2.2 Constructing new heuristic approaches

In the dissertation we discuss the practical tractability of the m-roommates problem. To this aim, we compare a variety of heuristic approaches in Chapter 7 in terms of solving the problem. We also contribute to this set of methods with some own heuristics constructed to find a feasible solution, which we describe in the followings.

2.2.1 Equalizer methods `eq1-6`

Neither of agglomerative hierarchical clustering with a suitable cut (`cluster`, see Dr. Kovács et al. (2011)), k-means++ algorithm (`kmeans`, see Arthur and Vassilvitskii (2007)), or fuzzy c-means clustering with equi-sized clusters (`eqFCM`, see Höppner and Klawonn (2008)) guarantees that the produced clusters are balanced. Hence, we construct six heuristic approaches to equalize the groups. All of these in Step 0 check whether the number of clusters equals k. If not, then from the largest cluster it chooses the element being furthest from the center of the cluster and creates a new group having one point from it. This step is iterated until no empty clusters left. After this, the methods of the equalizer algorithms are the following:

1. Equalizer method 1 (`eq1`)

It equalizes the groups one after another starting with the largest cluster. For a given cluster it moves as many elements out of the cluster as needed to meet the desired cardinality. In each step the point closest to the center of another cluster is reassigned to that cluster. After the group size of m is reached we block the given cluster so that elements won't get back to it and then we consider the next largest cluster. We iterate this method until the clusters are balanced. This process requires at most $k(k-1)(m-1)/2$ steps. The pseudocode of the method is described by Algorithm 2.1.[2]

Equalizer method 1 (`eq1`)

```
Input $C = \{C_1, \ldots, C_k\}$ non-empty clusters with cluster centers $c = \{c_1, \ldots, c_k\}$, where $c_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$ is the center of cluster $C_i$, and $m$ as the desired size of balanced clusters
(1)  $i \leftarrow \arg\max_i \{|C_i| : C_i \in C\}$    [index of the largest cluster]
(2)  while ($|C_i| > m$)
         $j, l \leftarrow \arg\min_{j,l} \{d(x_j, c_l) : x_j \in C_i, c_l \in c, l \neq i\}$
         $C_i \leftarrow C_i \backslash x_j,\ C_l \leftarrow C_l \cup \{x_j\}$, updating $c_l$
     end
(3)  $C \backslash C_i,\ c \backslash c_i,\ i \leftarrow \arg\max_i \{|C_i| : C_i \in C\}$    [updating sets and indexes]
(4)  if ($|C_i| > m$)
         goto Step (2)
     end
```

Algorithm 2.1: Equalizer method 1 (`eq1`)

2. Equalizer method 2 (`eq2`)

   Makes the excess 'flow down' starting from the largest cluster until the clusters become balanced. In each step it moves one element of a given cluster over to another cluster based on the distances from the centers of the other clusters. Once

[2]The pseudocode of the other equalizer methods are not shown, as those can be derived from this one with small modifications.

a point is reassigned we block the given cluster and move to the next largest one. We iterate this process as long as there are clusters with more than the desired number of elements. If there isn't any and the clusters are not balanced we make available all the clusters again and restart the process from the largest cluster. The required number of steps is at most $k(k-1)(m-1)/2$.

3. Equalizer method 3 (`eq3`)

   It equalizes the clusters starting from the smallest one in a way similar to that of `eq1`. We move as many elements into a given cluster as it is needed to reach the balanced cluster size. The reassignment is done based on the distance of the given cluster's center and the points in other clusters with at least two elements. We always move the point closest to the cluster center and after the given cluster misses no more elements we block it. The algorithm terminates after at most $(k-1)(m-1)$ steps.

4. Equalizer method 4 (`eq4`)

   Makes the excess flow down starting from the smallest cluster. In each step the distances of the center of the cluster being equalized and the points in other clusters with at least two elements are considered. We reassign the point with the smallest distance, then we block the cluster and move to the next smallest cluster. We iterate this process as long as there are clusters with fewer elements than needed. If there isn't any and the clusters are not balanced we make all of the clusters available again and restart the process. The algorithm terminates after at most $(k-1)\left((m-2)k/2+1\right)$ step.

5. Equalizer method 5 (`eq5`)

   Combines methods `eq1` and `eq3` with alternating between the largest and smallest clusters. The algorithm terminates after a finite number of steps.

6. Equalizer method 6 (`eq6`)

   Combines methods `eq2` and `eq4` with alternating between the largest and smallest clusters. The algorithm terminates after a finite number of steps.

2.2.2 Fuzzy c-means with equi-sized clusters (`eqFCMv2`)

The fuzzy c-means clustering with equi-sized clusters (c denotes the number of clusters, see Höppner and Klawonn (2008)) returns a membership matrix $U = [u_{ij}] \subset \mathbb{R}^{c\times n}$ which determines for each point $x_j, 1 \leq j \leq n$ the degree of belongingness into each cluster $i, 1 \leq i \leq c$. Based on this, we can construct an assignment with equal number of points in each cluster.

Through the process we consider the values $u_{ij}, 1 \leq i \leq c, 1 \leq j \leq n$ in ascending order. For a given value u_{ij} we assign point x_j to cluster i if it has fewer elements than the desired uniform cluster size. Otherwise, we move on to the next u_{ij} value. We continue this process until every element is assigned to a cluster.

2.2.3 Algorithm LCW with swaps of size three (`LCWv2`, `LCWv3` és `LCWv4`)

Algorithm LCW (see Weitz and Lakshminarayanan (1996)) tries to find a feasible solution for the optimization problem with starting from a randomized initial assignment and making improvements in the objective function through swaps of elements in different groups. If it can no longer make any improvement the algorithm halts. However, from the perspective of the optimization problem the solution found by LCW is not necessarily optimal. It might be the case swapping more than two elements, with elements coming from different groups, can still make an improvement. In the dissertation, we consider extending the LCW method with swaps of size three or even greater.

The LCW method iterates through the points and in each step it considers a given point and evaluates how much the value of objective function would improve if we were to swap this element with another one from a different group. This implies $(k-1)m$ comparison in each step and if the value of the objective function can be improved the corresponding swap is executed. If we were to apply the method of LCW and we considered a swap of size s then we would have to compare

$$\binom{k-1}{s-1} m^{s-1} D_s$$

values in each step, where $D_s = sD_{s-1} + (-1)^s, D_0 = 1$ is a recursive function which gives the number of permutations of order s without a fixpoint, i.e. a reassignment where no elements remains assigned to the initial group ($D_5 = 44, D_6 = 265, D_7 = 1854$; see Király and Tóth (2011)). The number of possibilities quickly becomes huge.

Because of the argument above, we consider only swaps of size three in addition to the swaps of pairs. We do this in three different ways, thus, we construct three different heuristic. The first one (LCWv2), described by Algorithm 2.2, first runs LCW, and then tries to make an improvement focusing only on swaps of size three.

LCW Algorithm, version 2 - swaps of size three (LCWv2)

```
(1) Initializing an arbitrary $X = [x_{ip}]$ feasible solution, where
    $x_{ip} = 1$, if $i = (p-1)*S+1, (p-1)*S+2, \ldots, (p-1)*S+S$ and
    $p = 1, 2, \ldots, G$, and 0 otherwise.
(2) $R \leftarrow DX$, where $D = [d_{ij}]$ is the distance matrix of the elements.
    Flag ← false    [used to indicate if there was a swap]
    $i \leftarrow 0$
(3) *We check if we can make an improvement by swapping i with an item
    from another group.*
    $i \leftarrow i+1$
    **if** $(i \leq N)$
        $t \leftarrow$ group of element $i$, for which $x_{it} = 1$
        $k \leftarrow \arg\max_{j \in J} w_j$, where $w_j = (r_{iq} - r_{it}) + (r_{jt} - r_{jq}) - 2d_{ij}$ and
            $J = \{j | x_{jq} = 1, 1 \leq q \leq G, q \neq t\}$
        **if** $(w_k > 0)$
            $x_{kq} \leftarrow 0, x_{kt} \leftarrow 1, x_{iq} \leftarrow 1, x_{it} \leftarrow 0$    [swap the items]
            $R \leftarrow DX$    [update matrix $R$]
            Flag ← true    [we indicate a swap has been made]
        **end if**
        **goto** Step (3)
    **end if**
(4) *If there has been a swap in the latest iteration, we iterate through all of
    the points again.*
    **if** (Flag == true)
        Flag ← false, $i \leftarrow 0$
        **goto** Step (3)
    **end if**
```

```
(5) We check if we can make an improvement by swapping i and two other
    items coming from different groups.
    $i \leftarrow i+1$
    if ($i \leq N$)
        $t \leftarrow$ group of element $i$, for which $x_{it}=1$
        $(k_1,k_2) \leftarrow \operatorname*{arg\,max}_{(j_1,j_2)\in J\cdot J} W(j_1,j_2)$, where
            $W(j_1,j_2) = r_{iq_2}+r_{j_1r}+r_{j_2q_1}-(r_{it}+r_{j_1q_1}+r_{j_2q_2}+d_{ij_1}+d_{ij_2}+d_{j_1j_2})$
            és
            $J\cdot J = \{(j_1,j_2) \mid x_{j_1q_1}=1, 1\leq q_1\leq G, q_1\neq t,$
                                $x_{j_2q_2}=1, 1\leq q_2\leq G, q_2\neq t, q_1\neq q_2\}$
        if ($W(k_1,k_2)>0$)
            [swap the three items]
            $x_{k_1q_1}\leftarrow 0, x_{k_1t}\leftarrow 1, x_{k_2q_2}\leftarrow 0, x_{k_2q_1}\leftarrow 1, x_{iq_2}\leftarrow 1, x_{it}\leftarrow 0$
            $R \leftarrow DX$    [update matrix $R$]
            Flag $\leftarrow$ true    [indicate a swap has been made]
        end if
        goto Step (5)
    end if
(6) We check if there has been a swap in the latest iteration, and if not, the
    algorithm terminates.
    if (Flag == false)
        stop    [cannot make further improvements]
    else
        Flag $\leftarrow$ false, $i \leftarrow 0$
        goto Step (5)
    end if
```

Algorithm 2.2: LCW Algorithm, version 2 - swaps of size three (`LCWv2`)

The second version of the extended LCW algorithm (`LCWv3`) is a repeated run of `LCWv2` until no improvement can be made with either swap of pairs or swap of three items.

Finally, in the third version (`LCWv4`) in case of each point we consider improvements by both swaps of pairs and swaps of three items. From the available improvements we choose the one with the highest value. In case of ties we prefer a swap of size two. The algorithms terminates if no improvement can be made by a swap of pair or swap of size three.

2.3 Simulations to compare heuristic approaches

In Chapter 7 of the dissertation we use simulations to investigate the practical tractability of the problem. We generate samples of students, run a selected set of algorithms on the samples and compare the results of the algorithms. We divide the algorithms into three subsets and present them in Table 2.3.

In the analysis we assume 3 attributes for the students and each attribute is randomly generated according to discrete uniform distribution on the interval $[0, 10]$. We used MATLAB for the implementations and simulations, and the analysis was made on a personal computer with Core i5-8600K 3.60GHz processor and 16,0 GB RAM.

Constructive methods

`cluster`	Agglomerative hierarchical clustering with suitable cut, which uses squared Euclidean distances and Ward method (for details see Dr. Kovács et al. (2011))
`eq1-6`	Heuristic methods which equalize the clusters based on the distances of items and cluster centers

Methods which alternate computing cluster centers and assignments

`kmeans`	k-means++ method, which alternates computing cluster centers and assigning the points to the cluster center closest to them (Arthur and Vassilvitskii, 2007)
`eqFCM`	Fuzzy c-means method with equi-sized clusters, which alternates computing prototypes and belongingness degrees (Höppner and Klawonn, 2008)

`eqFCMv2`	After running `eqFCM` we assign the points to clusters in an ascending order of the belongingness degrees
`MalinenFranti`	An alternating approach based on `kmeans`, where the assignment step usis the Hungarian method (Malinen and Fränti, 2014)
`JittaKlami`	A probabilistic clustering method, which alternates assigning points and estimating cluster parameters (Jitta and Klami, 2018)

Heuristics which apply local search

`LCW`	Starts from an arbitrary solution and tries to make improvements by swapping pairs (Weitz and Lakshminarayanan, 1996)
`LCWv2-4`	These methods try to make improvements by swapping pairs and groups of size three in different ways
`TLCW`	The `LCW` method with greedy construction and tabu memory, where after finding a local optimum we make the best non-improving change and restart the search (Gallego et al., 2013)
`SO`	Method `TLCW` with strategic oscillation, where the search is allowed to take not feasible solutions too (Gallego et al., 2013)
`Costaetal`	Applies LIMA-VNS method, which, after choosing a feasible solution by random swaps from an increasing neighbourhood, restarts the `LCW` method (Costa et al., 2017)

Table 2.3: Summary table of the heuristic approaches included in the analysis.

When we combine any of the equalizer methods with an algorithm which might not provide balanced clusters we indicate this by combining their names, e.g. `cluster_eq3`. As the equalizer methods are able to create balanced clusters on their own, we include them in the analysis as individual methods too.

For some methods, if it can be done through straightforward changes, we include both minimization and maximization versions of the algorithms. For example, in case of `LCW` we consider `LCWmin` and `LCWmax` algorithms. Similarly, we do the same for the extensions of `LCW`, `TLCW`, `SO`, and `Costaetal`. Furthemore, we also consider running the methods that apply swaps of size three with using the results of other heuristics as initial solutions in them to see if we can further improve the value of the objective function.

2.3.1 Simulations of the equalizer methods

We use Monte-Carlo (MC) simulations to compare equalizer methods `eq1-eq6`. The comparison is always done in case of a fixed initial clustering method which might be either `cluster`, `kmeans`, or `eqFCM`.

In each scenario, we generate a random sample of students, we apply the base clustering method, for the resulting clusters we run all of the equalizer methods (`eq1-eq6`), and then we evaluate their results compared to each other. After this, we compare the methods using different measures. In each MC simulation we generate 10000 samples, thus, the measures reflect the aggregated results of the scenarios. To assess the stability of the measures we repeat this process 100 times and calculate the following statistics of the measures: median, minimum, maximum, 25% and 75% percentiles.

To be able to tell in a given scenario the relative performance of the equalizer methods, we introduce the *relative points*. Let $c_1, c_2, \ldots, c_6$ denote the cost returned by equalizer methods `eq1`, `eq2`, ..., `eq6`, respectively. Let

$$c_{range} = \max\{c_1, \ldots, c_6\} - \min\{c_1, \ldots, c_6\}$$

denote the range of the costs. Given this, the relative point

$$p_i = \begin{cases} \sum_{\substack{1 \le j \le 6 \\ j \ne i}} \frac{c_j - c_i}{c_{range}}, & \text{if } c_{range} > 0, \\ 0, & \text{otherwise} \end{cases}$$

shows the relative performance of the ith method compared to the other ones. Scaling with the range is recommended because the difference of minimum and maximum costs might vary a lot.

We introduce the following approaches to compare the methods:

p0 : Number of times when the algorithm returned the best solution (ties included), divided by the number of repetitions.

p1 : We give points from 6 to 1 for the algorithms based on their performance, and take the mean of the points over the iterations. The best method gains 6 point, the worst gets 1. In case of ties the methods gain the average of points they would gain without ties. E.g. in case of a tie of two methods at the first place they both receive 5.5 points.

p2 : Average of p_i relative points.

T : Average of running times.

To display the statistics of the 100 run on the figures we use candle-like shapes. These illustrate the maximum (upper horizontal line), the 75% percentile (upper edge of the box in the middle), the median (yellow horizontal line), the 75% percentile (bottom edge of the box in the middle), and the minimum (bottom horizontal line) as shown on Figure 2.1.

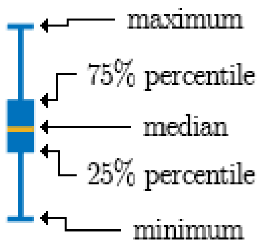

Figure 2.1: Illustration of the candle-like shape used to display the statistics of the equalizer methods.

2.3.2 Optimality points for large student sizes

We compare the heuristics that apply local searches in case of samples with $n = 600$ students for different m room sizes too. In this case, with the exception of $m = 2$, we are most probably not able to find the optimal solution. Thus, we introduce the *optimality point*, which allows us to evaluate the performance of either minimization or maximization methods compared to each other, and to do a comparison for both minimization and maximization methods in the same framework.

Definition 2.3.1 *Let M denote the number of student samples. Let x_i^{max} and x_i^{min} be the minimum and maximum of objective function values of the heuristics, respectively, in case of the ith sample. Let's assume that $x_i^{max} > x_i^{min}$ $\forall i$. Finally, let x_i^{alg} denote the objective function value of an arbitrary algorithm 'alg' for the ith sample. Then, the* optimality point p^{alg} *of algorithm 'alg' is defined as*

$$p^{alg} = \frac{1}{M} \sum_{i=1}^{M} \frac{x_i^{alg} - x_i^{\min}}{x_i^{\max} - x_i^{\min}}.$$

The idea behind the optimality point is to be able to compare the results on the same scale for different numbers and sizes of groups, as $p^{alg} \in [0, 1]$ always holds.

2.4 Constructing possible roommate assignments for small k and m

For small instances of the m-roommates problem we can determine the optimal solutions of problems 2MIN-mDM and 2MAX-mDM, so it can be used to evaluate the results of the algorithms. We find the optimums by constructing all of the possible roommate assignments, compute their costs according to the objective function, and search for the minimum and maximum values.

To construct the possible roommate assignments, we assign number from 1 to n to the students and we use a recursive algorithm to create the rooms step-by-step. We introduce the following definitions and notions before we give the pseudocode of the algorithm.

Definition 2.4.1 *Let $k > 1$ be the number of rooms, $m > 1$ be the (equal) size of the rooms, and thus, $n = k \cdot m$ be the number of students. Let $S = \{1, \ldots, n\}$ denote the set of students. Let $R \subset S, |R| = m$ be an assigned room (i.e. a set of students assigned to a room) and $\mathcal{R} = \{R | R \subset S, |R| = m\}$ be the set of possible assignments of a room. For given k and m we define the set of r-assignments $(0 \leq r \leq k)$ as*

$$\begin{aligned}\mathcal{P}_r = \Big\{ & (R_1, \ldots, R_r, s_1, \ldots, s_l) \mid \\ & 0 \leq r \leq k;\ \emptyset \neq R_i \in \mathcal{R}\ \forall i;\ R_i \cap R_j = \emptyset\ \forall i, j, i \neq j; \\ & 0 \leq l \leq n;\ l = n - r \cdot m;\ s_i \in S\ \forall i;\ s_i \notin R_j\ \forall i, j; \\ & s_i \neq s_j\ \forall i, j, i \neq j;\ (\cup_{i=1}^{r} R_i) \cup \left(\cup_{i=1}^{l} \{s_i\}\right) = S \Big\}.\end{aligned}$$

This is a set of partial assignments, where the number of rooms to which we have assigned m students is r. Hence, $\mathcal{P}_0$ has only one element which represents the state when no student is assigned to any room. Also, $\mathcal{P}_k$ is the set of possible complete assignments with elements where all of the students are assigned to a room.

Definition 2.4.2 *Let $k > 1$ be the number of rooms, $m > 1$ be the (equal) size of the rooms, and thus, $n = k \cdot m$ be the number of students. Let $D \in \mathbb{R}^{n \times n}$ be the matrix of squared Euclidean distances of the students. Let $0 \leq r \leq k$ be the number of rooms already assigned, and according to this, let $P = (R_1, \ldots, R_r, s_1, \ldots, s_l) \in \mathcal{P}_r$ be an r-assignment. Let $d : \mathcal{P}_r \to \mathbb{R}_0^+$ be the cost of assignment function, which, for a given r-assignment,* *gives the distances of students within the rooms which are already assigned:*

$$d(P) = \sum_{m=1}^{r} \sum_{\substack{i,j \in R_m, \\ i<j}} D_{ij}.$$

Also, let $\mathcal{D} = \{(P, d(P))\,|\, P \in \mathcal{P}_r, 0 \le r \le k\}$ *be the set of all possible assignment-cost pairs, and let* $2^{\mathcal{D}}$ *denote the set of subsets of* $\mathcal{D}$*. Finally, let* $h : \mathcal{P}_r \times \mathbb{R}_0^+ \to 2^{\mathcal{D}}$ *be a 'set of rooms'-assignment (or rooms-assignment for short), where for an* r*-assignment* $P \in \mathcal{P}_r$

$$\begin{aligned} h\,(P, d(P)) = \Big\{\, & (P', d(P'))\,\big|\,(P', d(P')) \in \mathcal{D}, \\ & P' = \big(R_1, \ldots, R_r, R_{r+1}, s'_1, \ldots, s'_{l-m}\big) \in \mathcal{P}_{r+1}, \\ & \min\{s_1, \ldots, s_l\} \in R_{r+1}\Big\}. \end{aligned}$$

Algorithm 2.4 presents the pseudocode of the recursive method, which constructs every possible roommate assignment, computes their costs, and finds the minimum and maximum of the values.

Recursive method to construct the possible roommate assignments

```
(1)  *Initializing variables.*
     $C = \emptyset$    [global variable: set of the costs]
     $Cmax = NaN, Cmin = NaN$    [global variables: extreme values of $C$]
     $r = 0$    [assigned rooms]
(2)  *Running the* **kroomcases**$(r, P, d(P))$ *recursive function.*
     **if** $(r < k)$
         $H \leftarrow h\big(P, d(P)\big)$    [generating $r+1$-assignments]
         **for** $\big(P', d(P')\big)$ in $H$
             **kroomcases**$(r+1, P', d(P'))$
         **end for**
     **else**    [updating the set of costs and extreme values]
         $C \leftarrow C \cup d(P)$
         **if** (isnan($Cmax$)) **or** ($Cmax < d(P)$)
             $Cmax \leftarrow d(P)$
         **end if**
         **if** (isnan($Cmin$)) **or** ($Cmin > d(P)$)
             $Cmin \leftarrow d(P)$
```

```
        end if
    end if
```

Algorithm 2.4: Recursive method to construct the possible roommate assignments, compute their costs and determine the minimum and maximum of the costs

Figure 2.2 illustrates the logic of Algorithm 2.4 for $n = 9$ students and $m = 3$ roommates.

Proposition 2.4.3 *Algorithm 2.4 constructs all of the possible roommate assignments, and it constructs all of them exactly once.*

$$\begin{tikzcd}
 & & \{1,2,3\},\{4,5,6\},7,8,9 \\
 & \{1,2,3\},4,5,6,7,8,9 \arrow[ur] \arrow[r] \arrow[dr] \arrow[dddr] & \{1,2,3\},\{4,5,7\},6,8,9 \\
 & \{1,2,4\},3,5,6,7,8,9 & \{1,2,3\},\{4,5,8\},6,7,9 \\
1,2,3,4,5,6,7,8,9 \arrow[uur] \arrow[ur] \arrow[r] \arrow[dddr] & \{1,2,5\},3,4,6,7,8,9 & \vdots \\
 & & \{1,2,3\},\{4,8,9\},5,6,7 \\
 & \vdots & \vdots \\
 & \{1,8,9\},2,3,4,5,6,7 &
\end{tikzcd}$$

Figure 2.2: Illustrating the recursive steps of Algorithm `kroomcases` for constructing all of the possible roommate assignments for $n = 9$ students and $m = 3$ roommates.

Chapter 3

Main results

3.1 NP-hardness of m-dimensional matching problems

In the followings we extend the existing general dimension hardness results of creating balanced groups. We show explicitly that if the size m of the groups is at least 3, then the balanced MSSC problem is NP-hard for various distance measures, and for both minimization and maximization of the objective function. With this result we also give an alternative proof for the NP-hardness of the optimization problem corresponding to the decision problem mDM defined by Feo and Khellaf (1990). For this optimization problem from now on we refer to as *maximally weighted m-dimensional matching* (Max-mDM) and we consider it with $m \geq 2$. Furthermore, we prove that the minimization version of the previous problem, referred to as *minimally weighted m-dimensional matching* (Min-mDM) is also NP-hard for $m \geq 3$. The importance of the latter is given by that it can be viewed as the generalization of the balanced MSSC problem, in which arbitrary distances may be considered between the points. Problems Max-mDM and Min-mDM are referred to as *m-dimensional matching problems.* Our results are also presented in the working paper Kondor (2022a). The source of tables which summarize the hardness results of m-dimensional matching problems is Kondor (2022b).

Note that balanced MSSC is equivalent to the 2Min-mDM problem. In what follows, we prove the NP-hardness of pMin-mDM and pMax-mDM for $m \geq 3$ and different values of p.

Theorem 3.1.1 (Kondor (2022a)) *pMin-mDM is NP-hard for integers $m \geq 3$ and $p \geq 1$.*

Proof. We show that the decision version of the optimization problem is NP-complete. The formal definition of the problem is as follows.

Decision problem 3.1.2 pMin-mDM decision problem.
Instance: $C = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$, *$m$ and p positive integers, such that* $k = n/m$ *and* $W \in \mathbb{Q}^+$.
Question: Is there a partition of C into disjoint sets $C_1, \dots, C_k$, such that $|C_i| = m, 1 \leq i \leq k$ and

$$\sum_{s=1}^{k} \sum_{x_i \in C_s} \sum_{x_j \in C_s} \|x_i - x_j\|_p^p \leq W$$

holds?

The problem is in NP. The rest of the proof follows the steps of Pyatkin et al. (2017). The reduction is done from the decision problem m-dimensional matching without edge weights which is given by Definition 3.1.3. This problem was shown to be NP-complete by Feo and Khellaf (1990) as a part of their NP-hardness proof of Max-mDM.

Decision problem 3.1.3 m-dimensional matching without edge weights (mDM-$\{0, 1\}$) decision problem.
Instance: A graph $G = (V, E)$ with $|V| = km, m \geq 3$, k, l positive integers.
Question: Is there a partition of V into disjoint sets $V_1, \dots, V_k$ such that $|V_i| = m, 1 \leq i \leq k$ and the number of edges that have both endpoints in the same set V_i is greater than l?

For the reduction, consider an arbitrary instance of mDM-$\{0, 1\}$ with $|V| = km$ and $|E| = q$. Fix the value of p to be a positive integer, make $d = q$ and $W = 4q(m-1) - 4(l+1)$. Denote by $C \in \{0, 1\}^{km \times d}$ the incidence matrix of G, i.e. $r_{it} = 1$ if vertex $v_i \in V$ is incident to edge e_t, and $x_{it} = 0$ otherwise. The rows of C may be considered as points in $\mathbb{R}^d$ and each balanced partition of these points corresponds to a balanced partition of V.

Let $A_{st} = \sum_{i:x_i \in C_s} \sum_{j:x_j \in C_s} |x_{it} - x_{jt}|^p$ be the contribution into the objective function of subset C_s with respect to coordinate t. This allows to rewrite the objective function of pMin-mDM as

$$\sum_{t=1}^{d} \sum_{s=1}^{k} A_{st}.$$

Given two points x_i and x_j whose t-th coordinate is 1 only two cases can happen with respect to the partitioning: the points are either 1) in the same subset C_{s_1}, or 2) partitioned into different subsets C_{s_1} and C_{s_2}. Denote by $A_t = \sum_{s=1}^{k} A_{st}$ the contribution of the t-th coordinate into the objective function and let $A_t^{(1)}$ and $A_t^{(2)}$ denote its value in the two respective cases. Thus, the contribution in the first case is

$$\begin{aligned} A_t^{(1)} &= A_{s_1 t} = 2|1-1|^p + 4(m-2)|1-0|^p = \\ &= 4(m-2), \end{aligned}$$

while in the second case it is

$$A_t^{(2)} = A_{s_1 t} + A_{s_2 t} = 4(m-1)|1|^p = 4(m-1).$$

Finally, denote by b the number of edges whose both endpoints are in the same subset. Given this notation the value of the objective function can be expressed as a function of b, that is

$$A(b) = (q-b)A_t^{(2)} + bA_t^{(1)} = 4q(m-1) - 4b.$$

$A(b)$ is decreasing in b, which implies that $A(b) \le W$ if and only if $b \ge l+1$. $\square$

Similarly, Theorem 3.1.4 states the NP-hardness of pMax-mDM.

Theorem 3.1.4 (Kondor (2022a)) *pMax-mDM is NP-hard for integers $m \ge 3$ and $p \ge 2$.*

Proof. We follow the steps of the proof of Theorem 3.1.1 with the following changes. In the decision version of the optimization problem pMax-mDM the value of the objective function has to be greater than or equal to W.

The value of p has to be chosen such that $p \ge 2$ is also satisfied, and the target value of the objective function is defined as $W = 4q(m-1) + (l+1)(2^{p+1}-4)$. When we consider the incidence matrix of G we use a modified matrix in this case. In each column of R replace one of the 1's with -1 and keep the other one unchanged. This implies that

$$\begin{aligned} A_t^{(1)} &= 2^{p+1} + 4(m-2), \\ A_t^{(2)} &= 4(m-1), \text{ and} \\ A(b) &= 4q(m-1) + b(2^{p+1}-4). \end{aligned}$$

Given $p \geq 2$, $A(b)$ is increasing in b, which means $A(b) \geq W$ if and only if $b \geq l+1$. $\square$

From the NP-hardness of 2Min-mDM for any $m \geq 3$ an explicit NP-hardness result of balanced MSSC immediately follows.

Corollary 3.1.5 (Kondor (2022a)) *Balanced MSSC is NP-hard for any integer $m \geq 3$.*

Moreover, as for any p the optimization problem pMin-mDM is a special case of Min-mDM, Theorem 3.1.1 also implies the NP-hardness of the general problem.

Corollary 3.1.6 (Kondor (2022a)) Min-*mDM is NP-hard for any integer $m \geq 3$.*

The complexity results of m-dimensional matching problems are summarized by Table 3.1 and Table 3.2. In Table 3.2 we use the following numbered notation: (1) Bertoni et al. (2012), (2) Lin et al. (2016), (3) Kel'manov and Pyatkin (2016), (4) Pyatkin et al. (2017), (5) Kondor (2022a). Problems denoted by question marks are open problems according to our knowledge.

	Maximal weighted m-dimensional matching (Max-mDM)		
	general case	pMax-mDM	
		$p=1$	$p \geq 2$
$m=2$	polynomial (Edmonds, 1965)		
$m \geq 3$	NP-hard (Feo and Khellaf, 1990)	?	NP-hard (Kondor, 2022a)

Algorithm 3.1: Complexity results of maximal weighted m-dimensional matching problem and its special cases. m denotes the group size, p denotes the parameter of the ℓ_p norm. Source: Kondor (2022b).

		Minimal weighted m-dimensional matching (Min-mDM)			
		general case	pMin-mDM		
			$p=1$	$p=2$	$p \geq 3$
$m=2$		polynomial (Edmonds, 1965)			
$m=3$		NP-hard (5)	NP-hard (5)	NP-hard (4)	NP-hard (5)
$3 < m < n/2$		NP-hard (5)			
$m=n/2$	$d=1$	?	?	polynomial (1)	?
	$d=2$	?	?	polynomial (2)	?
	d arbitrary	NP-hard (5)	NP-hard (5)	NP-hard (3)	NP-hard (5)

Algorithm 3.2: Complexity results of minimal weighted m-dimensional matching problem and its special cases. m denotes the group size, d denotes the dimesion of the Euclidean space, and p denotes the parameter of the ℓ_p norm. Source: Kondor (2022b).

3.2 m-roommates problem, and its benefits and drawbacks compared to the stable roommates problem

The formal definition of the m-roommates problem is the following. Let n be the number of students in a college, and let m be the uniform size of rooms in the college. For the sake of simplicity, let $k = n/m$ be an integer, where k is the number of rooms. Let $C_1, \dots, C_k$ denote the rooms. Let's assume that the compatibility of two students can be described by certain attributes in the following way. Let d be the number of *attributes*, and let's assume that these can be expressed by real numbers[1]. Hence, every student can be represented by a point in the Euclidean space $\mathbb{R}^d$. Let $x_1, \dots, x_n \in \mathbb{R}^d$ denote the students and let $D \in \mathbb{R}^{d\times d}$ be the distance matrix of the students. Let $x_i \in C_s$ denote that student i is assigned to room s.

We assume that the compatibility of students x_i and x_j is defined by their Euclidean distance $d_{ij} = \|x_i - x_j\|$. For the *good* roommate assignment we take two different approaches. To create homogeneous groups we consider the minimization problem

$$\min \sum_{s=1}^{k} \sum_{x_i, x_j \in C_s} d_{ij}^2, \qquad \text{s.t. } |C_s| = m \ \forall s. \tag{2Min-mDM}$$

To create an assignment with heterogeneous groups we use the following maximization problem:

$$\max \sum_{s=1}^{k} \sum_{x_i, x_j \in C_s} d_{ij}^2, \qquad \text{s.t. } |C_s| = m \ \forall s. \tag{2Max-mDM}$$

The uniform objective functions guarantee that we can use a cohesive framework to investigate both the minimization and maximization versions of the problem.

From a theoretical aspect one of the most important differences of stable roommates and m-roommates problems is, while the former one provides a stable solution, the latter defines a Pareto efficient solution concept. Thus, the solution for the latter is not necessarily stable, but in some situations it might be a better model to describe the real-life

[1]Here we allow for arbitrary real numbers, but in the analysis we will use a restriction of integers from the interval $[0, 10]$.

problem (see Morrill (2010)).

For stable roommates, in case of pairs the set of solutions might be empty, and in case of weak preferences it is NP-complete to decide whether a weakly stable solution exists or not. The approaches in higher dimensions, with one exception, are related to groups of three, and again, with one exception, all of them defines an NP-complete problem.

On the other hand, for the m-roommates problem, being an optimization problem, the set of solutions is always non-empty. Furthermore, for pairs, that is in case of $m = 2$, we can always find an optimal solution in polynomial time using Edmonds' algorithm (Edmonds, 1965). However, for groups of at least three the problem becomes intractable in general dimension. Based on Theorems 3.1.1 and 3.1.4 we know problems 2Max-mDM and 2Min-mDM are NP-hard. According to this, for larger instances of the problem we are not able to find the optimal solution even if we know that it always exists for every cluster size (Kondor, 2022b).

From a practical point of view a benefit of the m-roommates problem - and other metric approaches - over the stables roommates problem is that giving a preference order is not required in this case. It might be a more realistic approach in cases where some students have no preference order over the others at all. It might be the case when new students arrive to the college with no information about the others.

Another benefit of the model also comes from removing preferences. In case of preference orders there is no measurable difference between the items of the preference list. If for students x, y and z we have $y \succ_x z$ (x prefers y to z) we don't know how much more y is liked by x to z. On the other hand, distances give additional information and might be used to create better assignments.

One drawback of the m-roommates model is students now have symmetric views on each other, while in case of the stable roommates problem they could have asymmetric preferences.

Another shortcoming of the approach is the inability to express individual preferences. For example, there might be two or more students who already know each other and would like to be assigned to the same room. To be able to incorporate it into model we should use a more general approach, for example the m-dimensional matching, where arbitrary edge weights can be used between the vertices. Using this approach we can also define edge weights for the minimization problem which will result in two students always being assigned to different rooms. This is a property neither of the previously discussed models could achieve. However, discussion of the general model is not a subject of the dissertation, but might be an interesting topic for further research.

3.3 Results of comparing the equalizer methods

The results of the equalizer methods for parameters $k = 3$ and $m = 3$ are shown by Figures 3.1, 3.2, and 3.3.

As for the running times of the algorithms (measure T) method `eq3` proved to be the best. `eq5` is slightly behind, which is followed by `eq1` and `eq4`. Methods `eq2` and `eq6` produced the worst times. Based on the hight of candles the order is stable.

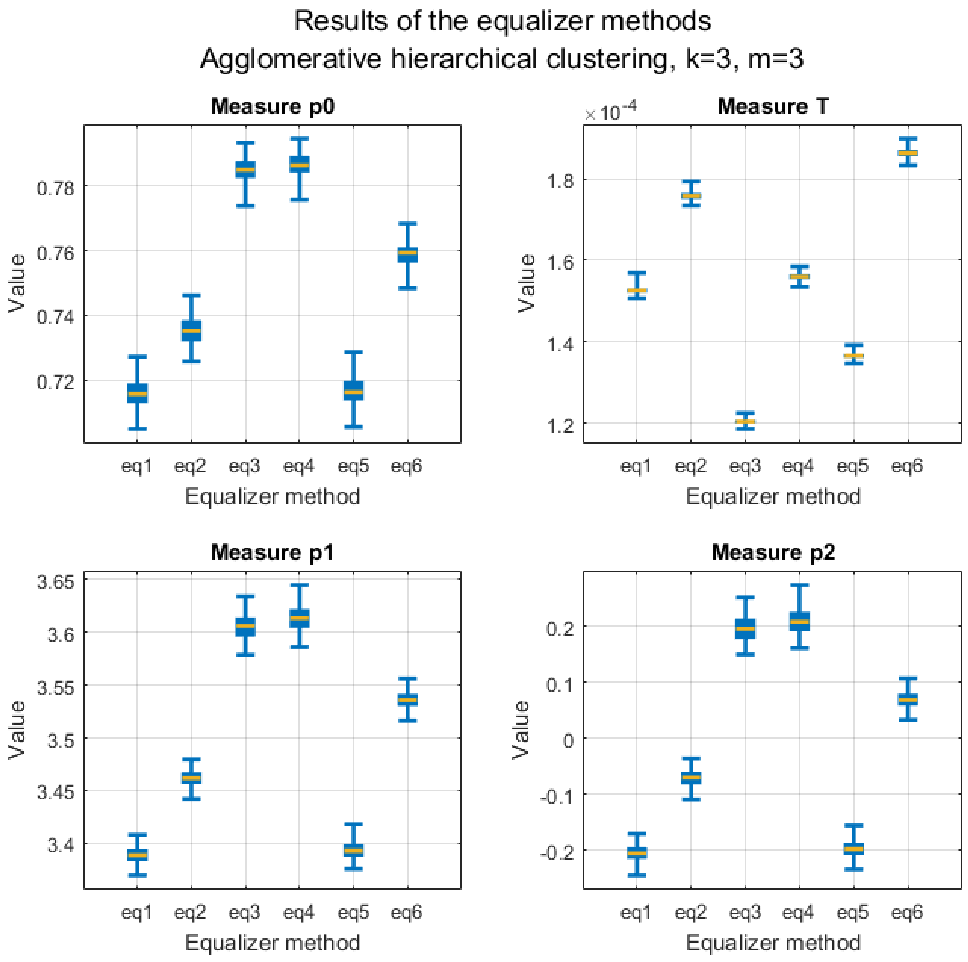

Figure 3.1: Results of the equalizer methods (`eq1`-`eq6`). Agglomerative hierarchical clustering, $k = 3, m = 3$.

According to the ratio of the number of times the procedure found the best result and the total number of scenarios (measure p0) methods `eq3` and `eq4` seems to be the best in all cases, which are followed with a larger lag by `eq6`. Note that for `eqFCM` the actual order might be different because of the overlapping candles.

Measures p1 and p2 are very similar to p0, which makes the evaluation easy. In each cases `eq3` and `eq4` are the best.

We executed the analysis for $k = 5$ groups and $d = 10$ dimensions too, and the results were mostly the same. As a result of the above analysis we consider `eq4` to be the best which is mediocre considering its running time, but superior according to all of the other measures. We also include `eq3` in some upcoming analysis for its favourable running time.

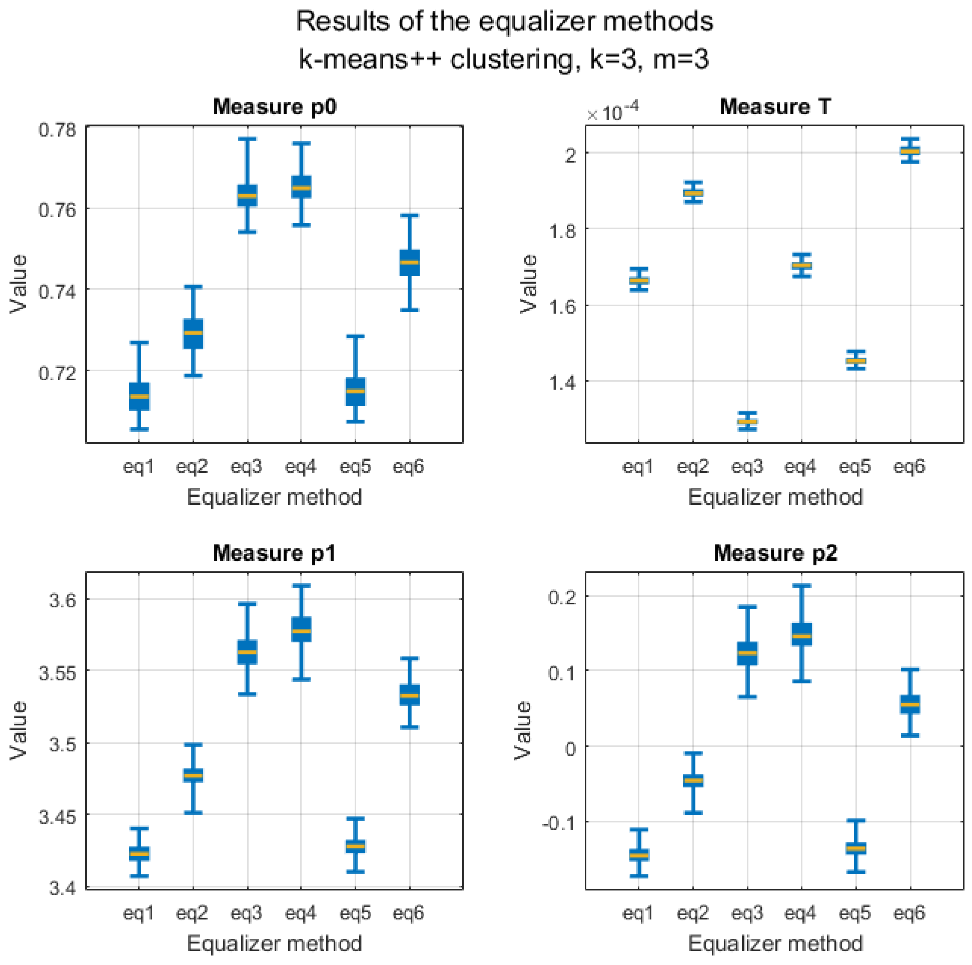

Figure 3.2: Results of the equalizer methods (`eq1`-`eq6`). k-means++ clustering, $k = 3, m = 3$.

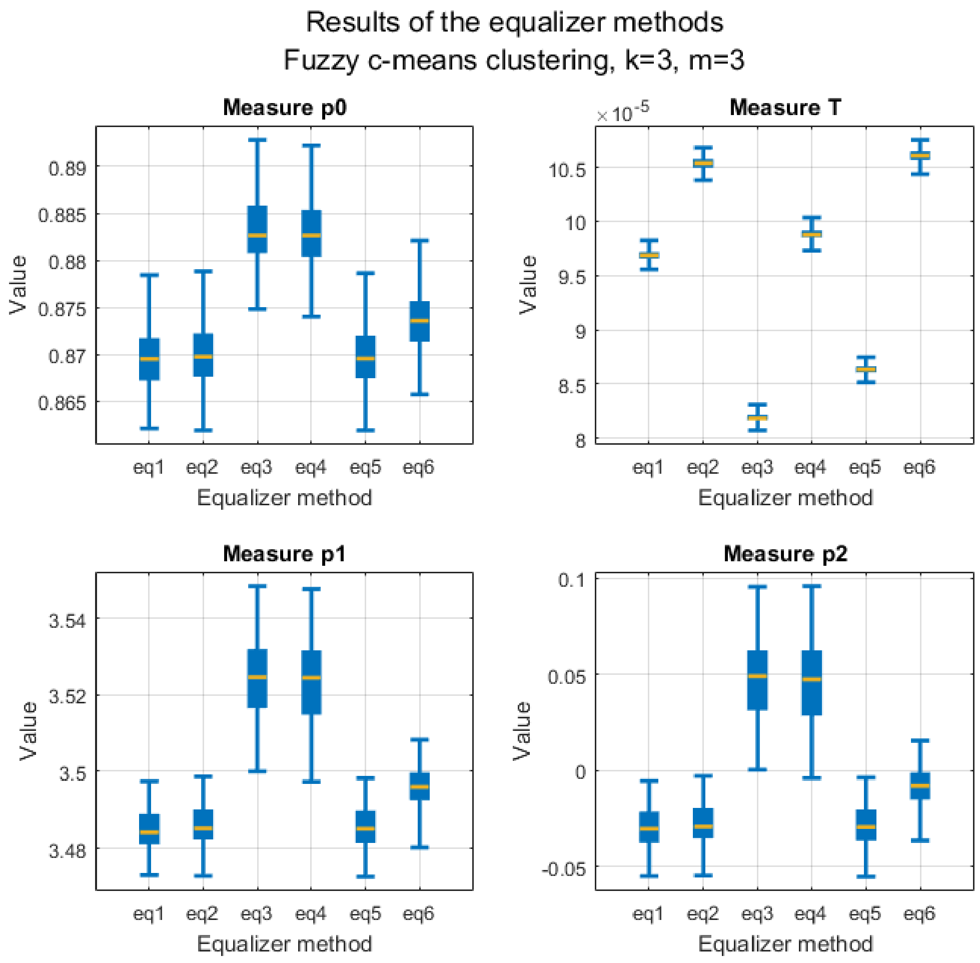

Figure 3.3: Results of the equalizer methods (`eq1`-`eq6`). Fuzzy c-means clusterin, $k = 3, m = 3$.

3.4 Experiments on real data

We compared the heuristics on real datasets 'Iris' and 'Seeds' too, which are frequently used in the literature (see Malinen and Fränti (2014); Costa et al. (2017); Rujeerapaiboon et al. (2019)). These are real-world datasets which contain the data of clusters with equal size. These are used in the literature in case of the minimization problem and the minimal objective function value is known for them. Hence, we also use these to compare the minimum searching algorithms.

For the analysis we run the algorithms on the data, measure the running times and evaluate the objective function values. The results are shown on Figures 3.4 and 3.5. Running times and objective function values are shown on the labels and the algorithms are sorted according to these.

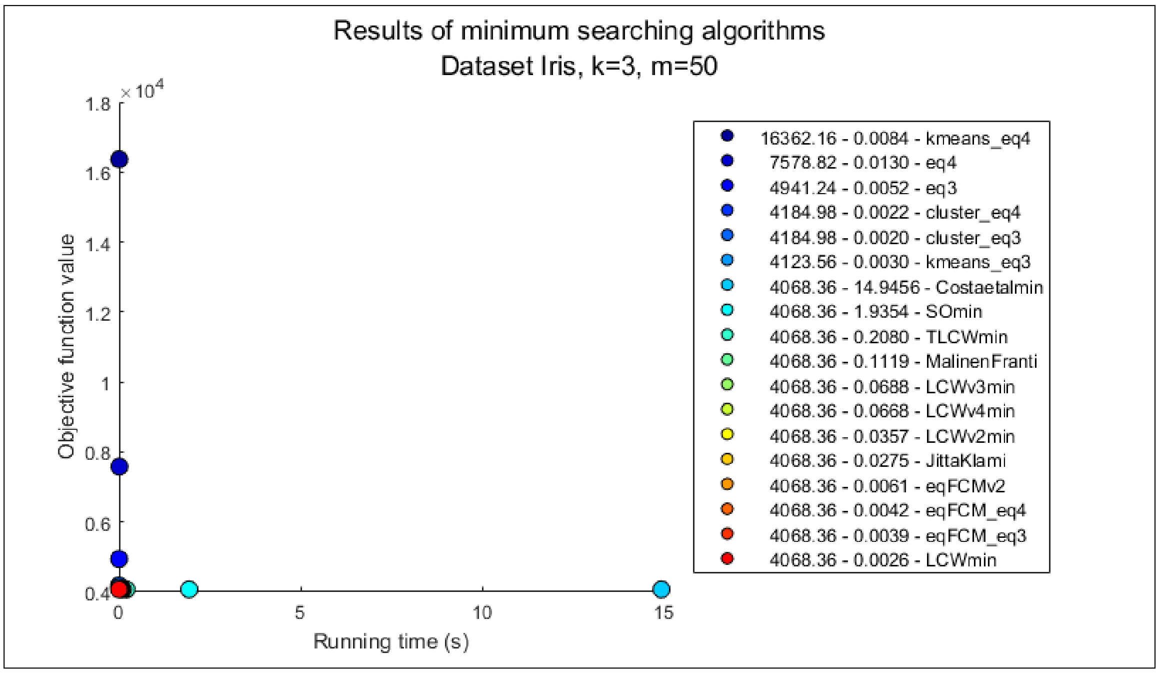

Figure 3.4: Results of minimum searching algorithms for dataset Iris. $k = 3, m = 50$. Description of labels: objective function value - running time (s) - name of the algorithm.

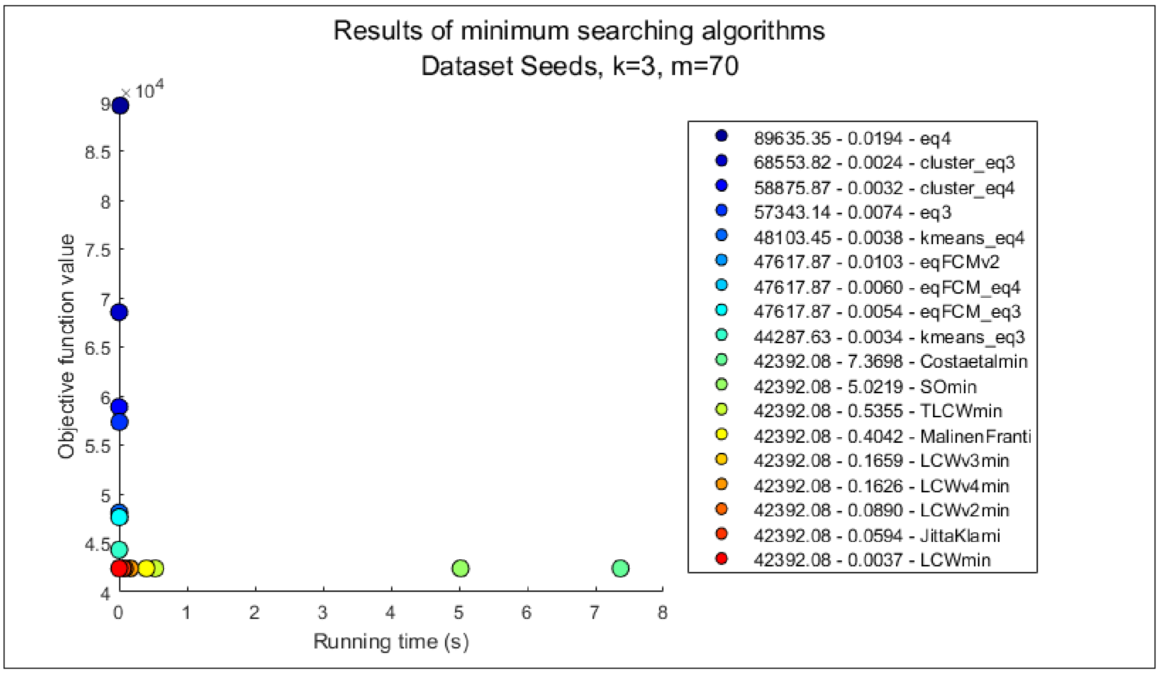

Figure 3.5: Results of minimum searching algorithms for dataset Seeds. $k = 3, m = 70$. Description of labels: objective function value - running time (s) - name of the algorithm.

Most of the methods are able to find the optimum, however there are significant differences in running times coming from the construction of the methods. More sophisticated approaches require more time before they halt, but probably perform better in general. Algorithms related to traditional cluster analysis are amongst the fastest heuristics which might be good in case of larger problem instances but the poor performance is not promising.

3.5 Results of simulations with small k and m

In this step of the analysis, we compare the heuristics on simulated datasets for small values of k and m. In this case we can generate all of the possible roommate assignments, we can find the minimum and the maximum values, and we can evaluate the algorithms against these optimums. We generate random samples, where the dimension is $d = 3$, that is students have 3 attributes.

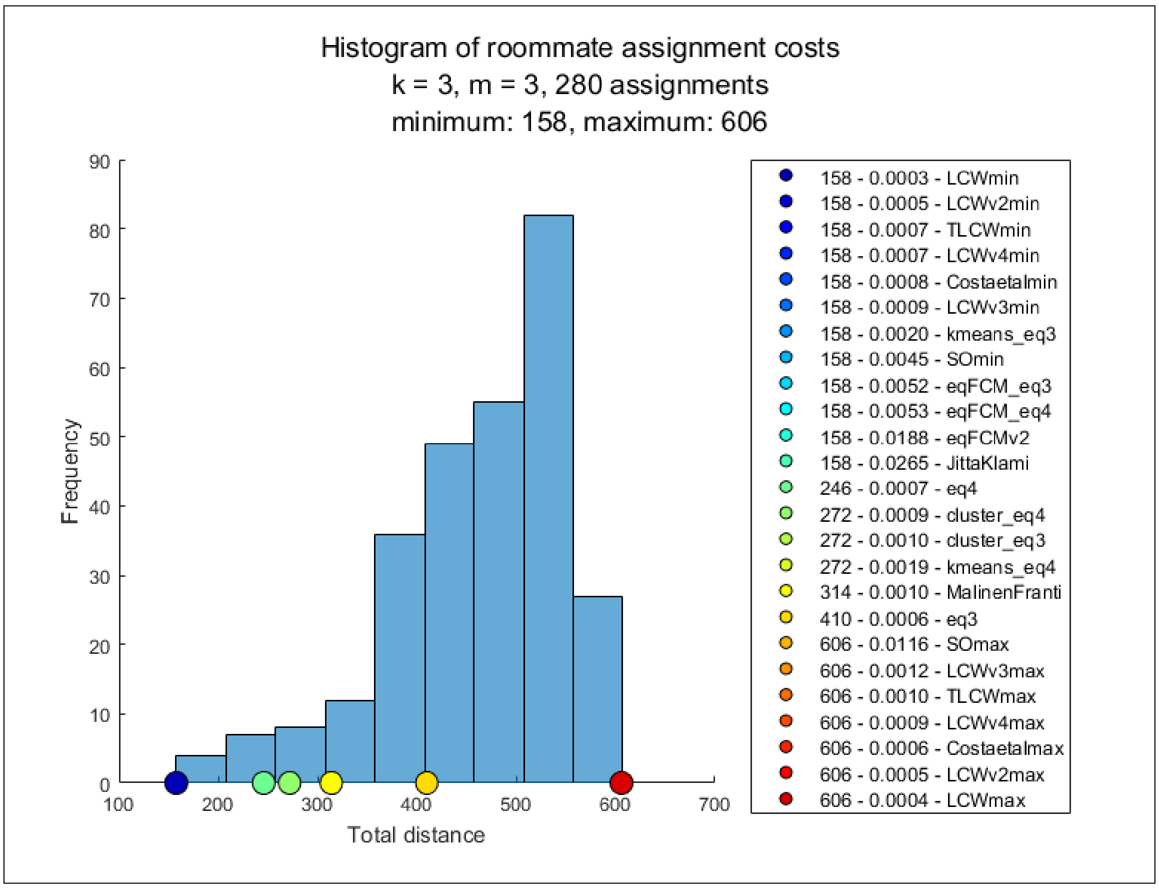

Figure 3.6: Histogram of roommate assignment costs, and the results of minimum and maximum searching algorithms. Simulated instance, $k = 3, m = 3$, 280 possible roommate assignments, the running time of generating assignments and searching of optimums was 0.1541 s. Skewness: -0.5339. Description of labels: objective function value - running time - name of the algorithm.

Figure 3.6 illustrates the histogram of the possible roommate assignment costs for a generated sample of students with $k = 3$ and $m = 3$. Also, it shows the minimum

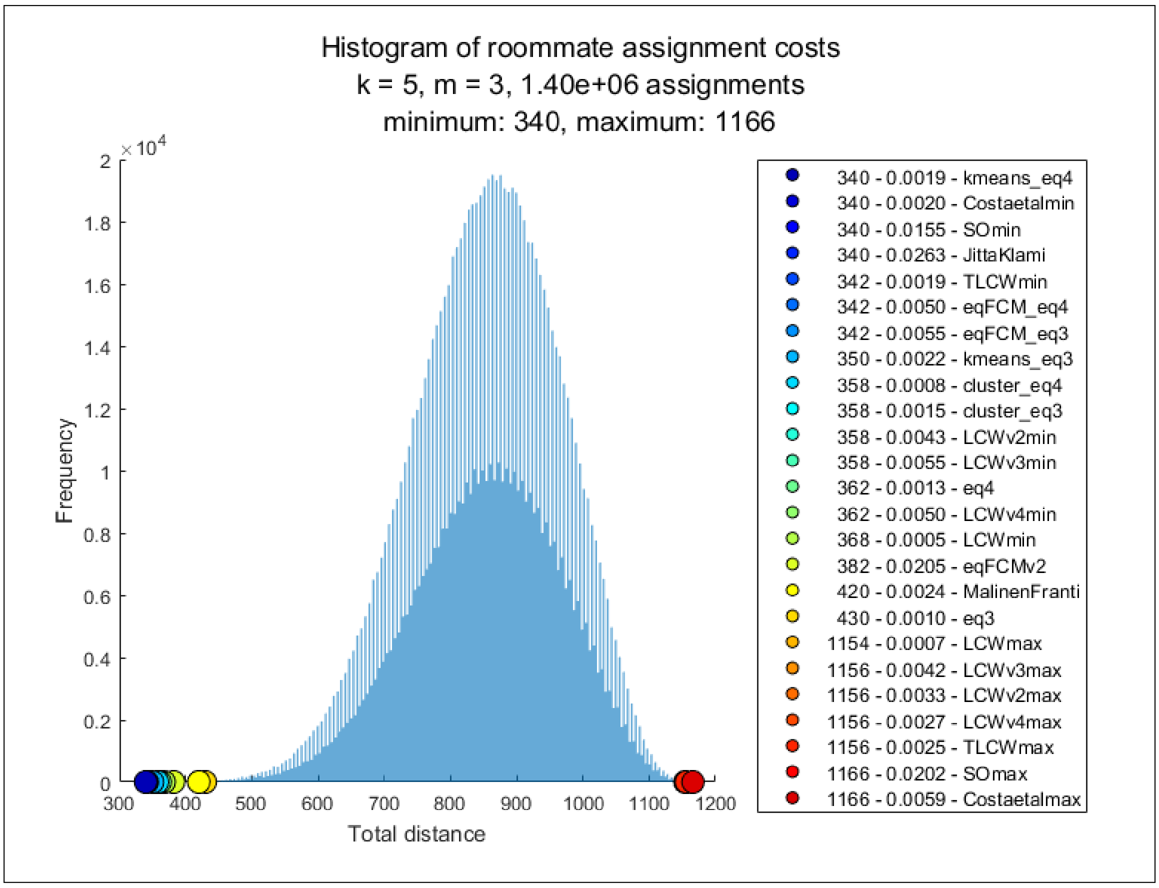

Figure 3.7: Histogram of roommate assignment costs, and the results of minimum and maximum searching algorithms. Simulated instance, $k = 5, m = 3$, 1.40e+06 possible roommate assignments, the running time of generating assignments and searching of optimums was 62.06 s. Skewness: -0.2814. Description of labels: objective function value - running time - name of the algorithm.

and maximum of the costs, and the objective function values and running times of the algorithms. Similarly, Figure 3.7 shows the case of $k = 5$ and $m = 3$ (with 1.40e+06 possible roommate assignments).

Table 3.3 shows the results of a simulation made on a sample of 200 problem instances (for the minimizing problem it shows only the best performing heuristics), with $k = 5$ and $m = 3$. Columns 'Distance' show the average distance from the optimum, columns 'Running time' show the average running time, and columns '# best result' show the number of cases where the algorithm found the optimum. Standard deviations are shown in brackets under the values. The rows of the table are sorted by the last column.

In terms of running times we can see that there are differences in the orders of magnitude. `LCW` algorithms (both minimizing and maximizing) are worth mentioning because with very low running times they can find the optimum in a significant number of cases.

	Distance	Running time	# best results
`LCWmax`	6,96	**0,0004**	78
	(9,03)	***(0,0001)***	
`LCWv2max`	5,64	0,0010	85
	(7,71)	*(0,0002)*	
`LCWv3max`	5,31	0,0017	90
	(7,65)	*(0,0004)*	
`LCWv4max`	4,91	0,0015	97
	(7,52)	*(0,0003)*	
`TLCWmax`	3,76	0,0014	109
	(5,73)	*(0,0002)*	
`Costaetalmax`	0,53	0,0023	180
	(1,85)	*(0,0006)*	
`SOmax`	**0,20**	0,0143	**191**
	(0,92)	*(0,0022)*	

	Distance	Running time	# best results
`LCWmin`	21,89	**0,0004**	106
	(32,92)	***(0,0001)***	
`TLCWmin`	15,54	0,0014	112
	(25,38)	*(0,0003)*	
`LCWv2min`	7,95	0,0020	141
	(16,88)	*(0,0006)*	
`LCWv3min`	4,62	0,0034	161
	(11,85)	*(0,0009)*	
`LCWv4min`	5,06	0,0029	162
	(13,98)	*(0,0005)*	
`SOmin`	4,47	0,0126	168
	(13,60)	*(0,0035)*	
`Costaetalmin`	**0,63**	0,0029	**190**
	(3,02)	*(0,0009)*	

Table 3.3: The results of maximum (top) and minimum (bottom) searching algorithms: average distances from the optimum (and standard deviations), average running times (and standard deviations), and the number of cases where the algorithm found the optimum. $k = 5, m = 3$, 200 simulated instances.

From the set of maximum searching algorithms the more sophisticated methods, `SOmax` and `Costaetalmax`, perform better in finding the optimum. However, note that the running time of `SOmax` is one order of magnitude larger than the second best `Costaetalmax`'s running time. Amongst the heuristics with swaps of size three `LCWv4max` has the best results.

Regarding the minimum searching algorithms, again, the more sophisticated methods perform better than the others. The constructive methods, heuristics related to cluster analysis, and algorithms `JittaKlami` and `MalinenFranti` did not show good results (they found the optimum in less than 30% of the cases), and thus, not displayed in the table. From the algorithms which apply swaps of size three `LCWv3min` performs better in terms of average distances, but according to the number of best results `LCWv4min` is slightly ahead. In terms of running times `LCWv4min` is the definite winner.

In the next step of the analysis we will consider only the local search methods, and from the three versions of methods that apply swaps of size three we will only keep `LCWv4`, which overall seemed to be better than the other two versions.

3.5.1 Asymmetry of minimizing and maximizing problems

We make a note about an interesting feature about the problem. For the illustration we use Figure 3.8, which shows the histogram of possible roommate assignment costs in case of parameters $k = 2, m = 15$. Note that the left tail of the distribution is very thin compared to the right tail. In such a case we say that the distribution is left-skewed, which is also proved by the calculated skewness value -0.9044. This emphasizes the possible asymmetry of the minimizing and maximizing problems.

We also note that, for all of the cases we investigated (with various pairs of parameters $k = 3, m = 3$; $k = 3, m = 4$; $k = 4, m = 3$; $k = 5, m = 3$) for all of the 200 instances we generated the skewness of the roommate assignments' costs were negative.

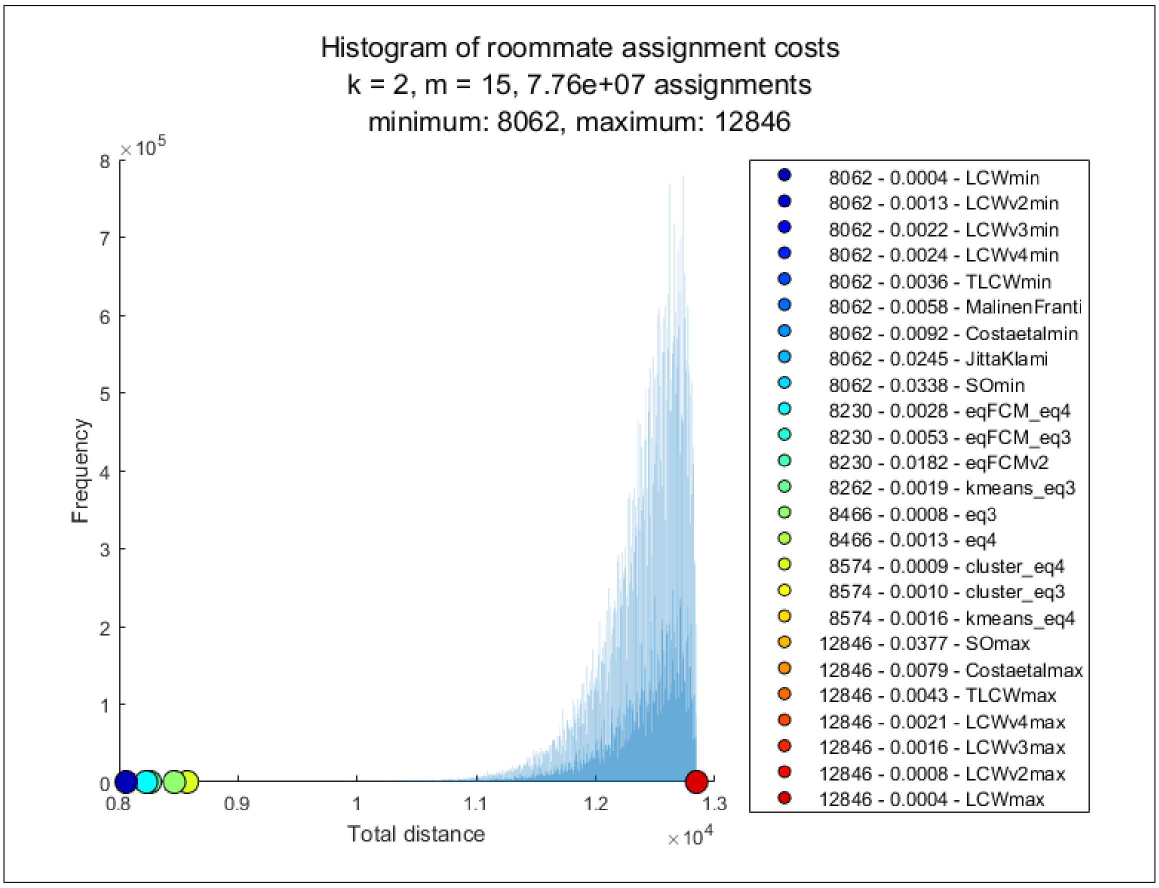

Figure 3.8: Histogram of roommate assignment costs, and the results of minimum and maximum searching algorithms. Simulated instance, $k = 2, m = 15$, 7.76e+07 possible roommate assignments, the running time of generating assignments and searching of optimums was 54 m 6 s. Skewness: -0.9044. Description of labels: objective function value - running time - name of the algorithm.

3.6 Results for larger instances of the m-roommates problem

3.6.1 Edmonds' algorithm as a benchmark

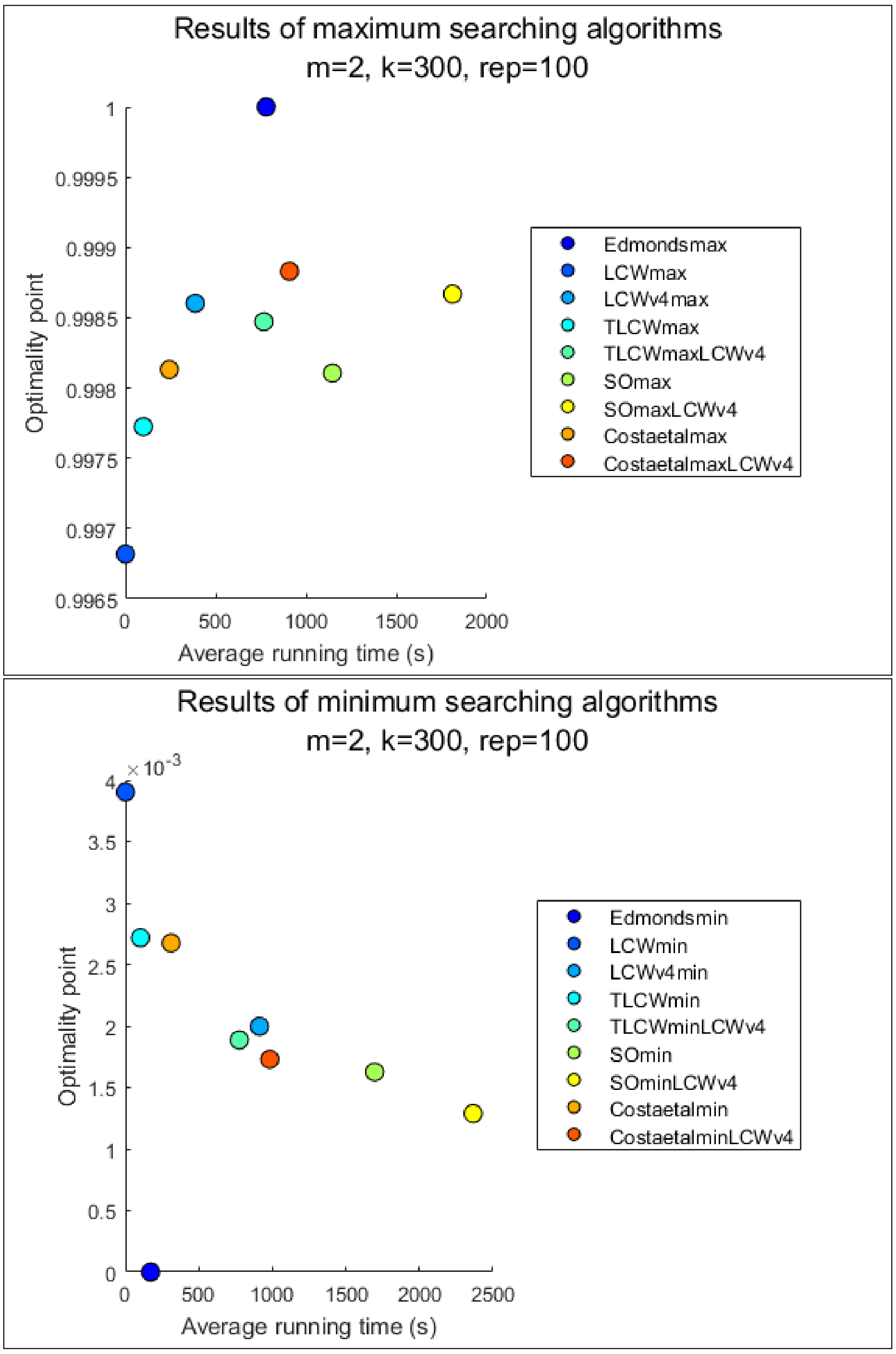

Figure 3.9: Results of maximum and minimum searching heuristics, including Edmonds' algorithm. $m = 2, k = 300$, 100 simulated instances.

In case of pairs we can determine the optimal solution in polynomial time. Hence, in this case we can compare the results of the heuristic approaches to the actual optimum. Figure 3.9 shows the results for 100 simulated instances for groups of size $m = 2$ and $k = 300$ rooms. In the set of algorithms we also include a minimum and a maximum searching method which are based on Edmonds' algorithm[2]. These are called `Edmondsmin` and `Edmondsmax`, respectively.

Based on the results it seems the heuristic approaches can get relatively close to the optimum. However, not even the more sophisticated methods can entirely decrease the distance between the result of the `LCW` methods and the optimums. We note that in case of the maximum searching methods `LCWv4` seems to be relevant on its own too, however, in case of the minimum searching methods it performs worse than heuristics `TLCminLCWv4` and `CostaetalminLCWv4`. Also, we highlight that applying `LCWv4` to the result of another approach it significantly improves the solution in the most cases.

3.6.2 Results for groups of size at least three

Figure 3.10 shows the results for 100 simulated instances for groups of size $m = 5$ and $k = 120$ rooms. Regarding the optimality point, we see results close to each other on both sub-figures. Note that `LCWv4max` is redundant as method `TLCWmax` performs better in terms of both optimality point and runnin time. However, in the minimizing case `LCWv4min` is relevant on its own based on it has better result than `Costaetalmin` and `SOmin`. For both minimizing and maximizing cases we can say that `LCWv4` improves significantly the objective function values of other heuristics' results.

Finally, note that in case of minimization running the algorithms take significantly longer than in case of maximization. For `Costaetal`, we may reject at any significance level the hypothesis H0 that the average of differences of minimizing and maximizing running times is zero, according to one-sample t-test. This property holds for the other methods too, so this emphasizes the asymmetry of minimization and maximization problems.

[2]The source of the MATLAB codes are available online, see Saunders (2022).

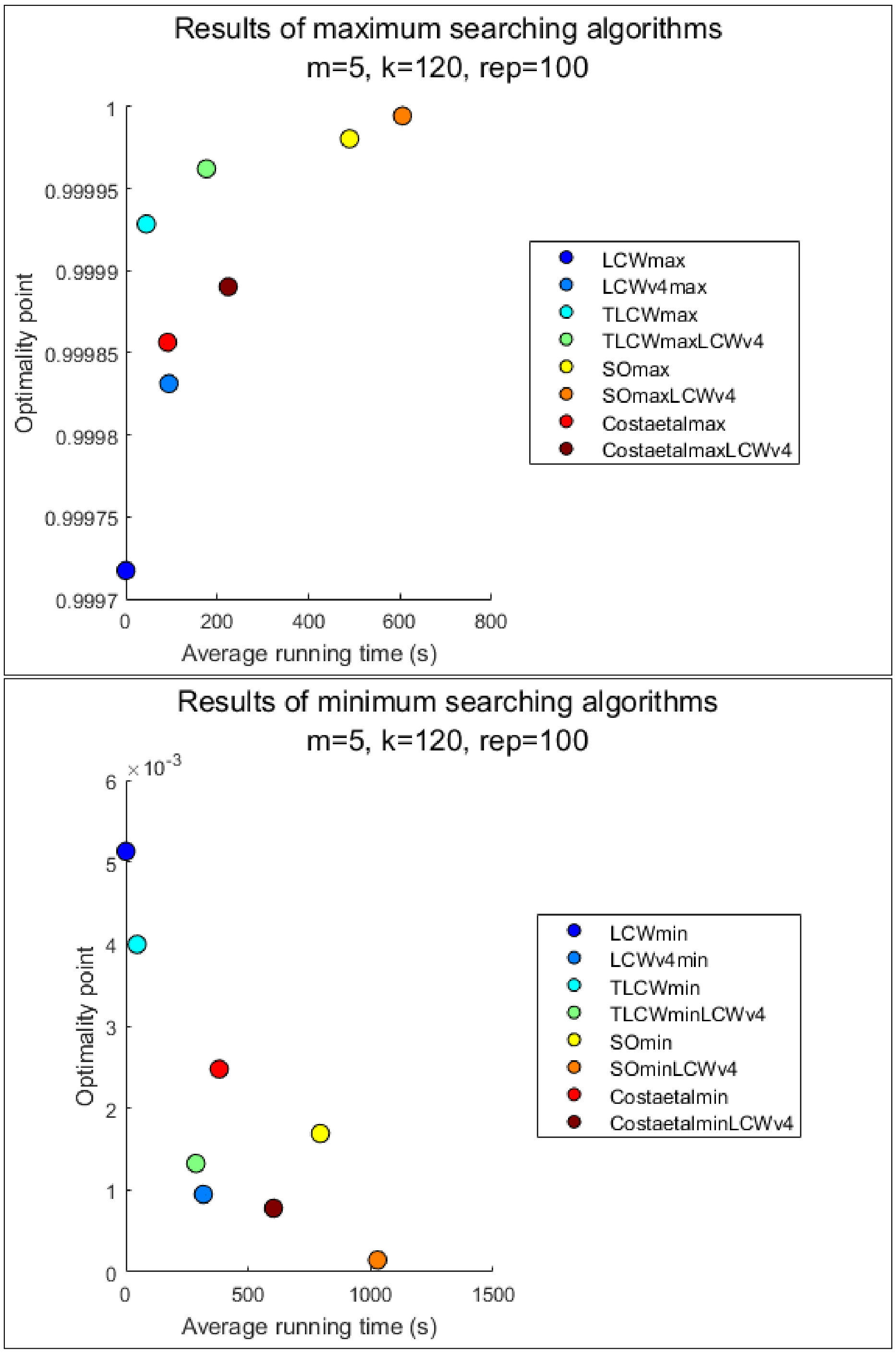

Figure 3.10: Results of maximum and minimum searching heuristics. $m = 5, k = 120$, 100 simulated instances.

Figure 3.11 shows the results for 100 simulated instances for groups of size $m = 60$ and $k = 10$ rooms. In case of the maximum searching methods practically speaking returned the same optimum in all of the cases - for `LCWv4` the difference of 1e-8 is probably caused by a numerical precision error. Thus, in such cases method `LCWmax` might be sufficient to find a feasible solution.

In case of the minimum searching methods we may notice that the range of the optimality points is larger than in case of larger number of groups. We note that `LCWv4min` is again, performs well in improving the results of other approaches. We also note that `LCWv4min` seems to have better results than `TLCWminLCWv4`, though, they relation might not be stable.

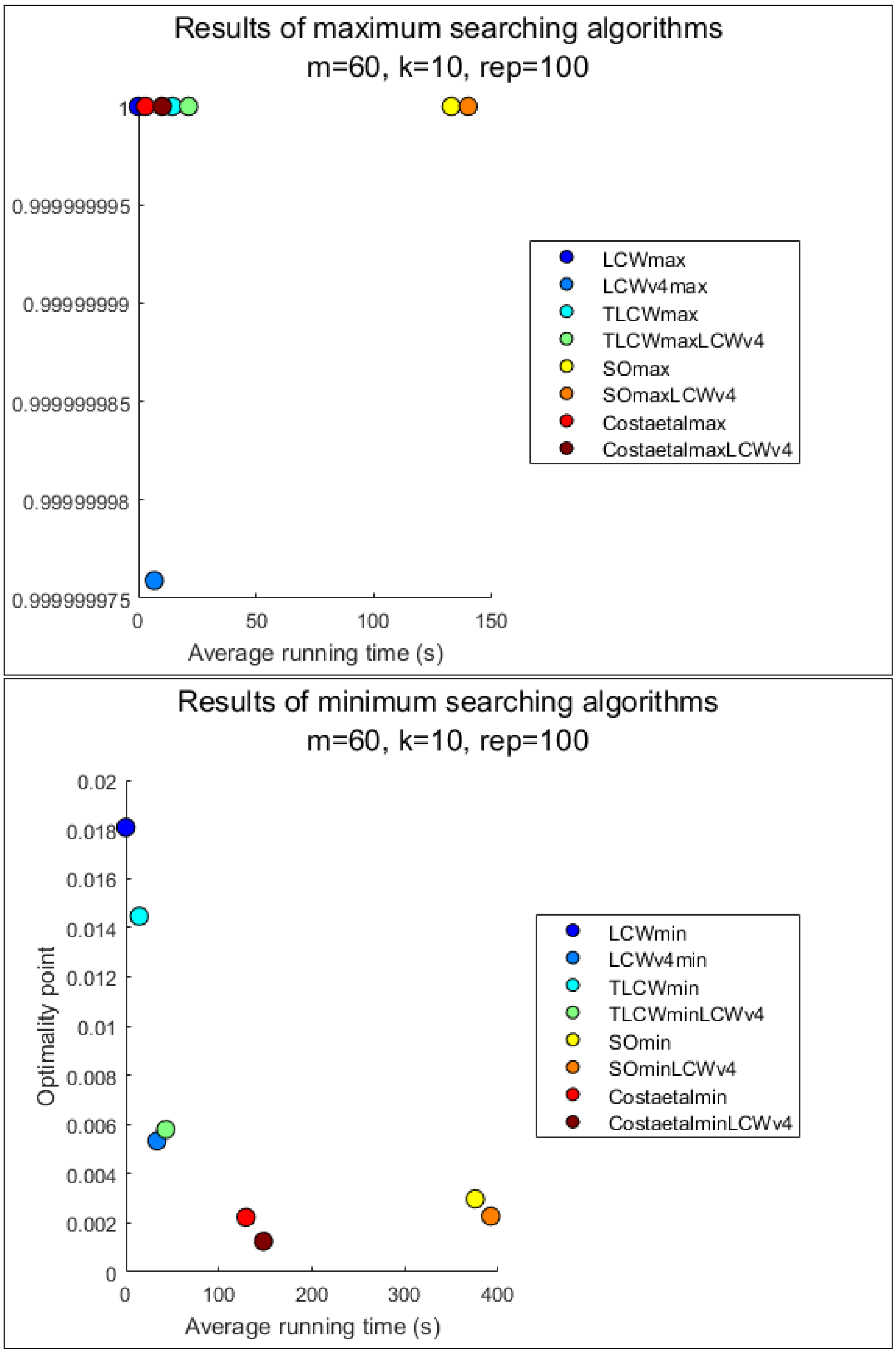

Figure 3.11: Results of maximum and minimum searching heuristics. $m = 60, k = 10$, 100 simulated instances.

Chapter 4

Bibliography

Arkin, E. M., Bae, S. W., Efrat, A., Okamoto, K., Mitchell, J. S. B. and Polishchuk, V. (2009). Geometric stable roommates. *Information Processing Letters*, 109(4):219–224. DOI: 10.1016/j.ipl.2008.10.003.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035. ISBN 978-0-898716-24-5.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A. and Protasi, M. (2003). *Complexity and approximation: Combinatorial optimization problems and their approximability properties.* Springer-Verlag Berlin Heidelberg 1999. ISBN: 978-3-642-63581-6. DOI: 10.1007/978-3-642-58412-1.

Bertoni, A., Goldwurm, M., Lin, J. and Saccà, F. (2012). Size constrained distance clustering: separation properties and some complexity results. *Fundamenta Informaticae*, 115(1):125–139. DOI: 10.3233/FI-2012-644.

Biró, P. (2006). Stabil párosítási modellek és ezeken alapuló központi párosító programok. *Szigma*, 37(3-4):153–175. `https://journals.lib.pte.hu/index.php/szigma/article/view/1096`.

Costa, L. R., Aloise, D. and Mladenović, N. (2017). Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. *Information Sciences*, 415-416:247–253. DOI: 10.1016/j.ins.2017.06.019.

Edmonds, J. (1965). Maximum matching and a polyhedron with $0, 1$-vertices. *Journal of Research of the National Bureau of Standards*, 69B(1-2):125–130. DOI: 10.6028/jres.069b.013.

Edwards, A. W. F. and Cavalli-Sforza, L. L. (1965). A method for cluster analysis. *Biometrics*, 21(2):362–375. DOI: 10.2307/2528096.

Feo, T. A. and Khellaf, M. (1990). A class of bounded approximation algorithms for graph partitioning. *Networks*, 20(2):181–195. DOI: 10.1002/net.3230200205.

Feo, T. A., Goldschmidt, O. and Khellaf, M. (1992). One-half approximation algorithms for the k-partition problem. *Operations Research*, 40:S170–S173. `https://www.jstor.org/stable/3840846`.

Gale, D. and Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15. DOI: 10.2307/2312726.

Gallego, M., Laguna, M., Marti, R. and Duarte, A. (2013). Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of the Operational Research Society*, 64(5):724–734. DOI: 10.1057/jors.2012.66.

Höppner, F. and Klawonn, F. (2008). Clustering with size constraints. In *Computational Intelligence Paradigms*, volume 137 of *Studies in Computational Intelligence*, pages 167–180. ISBN 978-3-540-79473-8. DOI: 10.1007/978-3-540-79474-5_8.

Jitta, A. and Klami, A. (2018). On controlling the size of clusters in probabilistic clustering. In *Thirty-Second AAAI Conference on Artificial Intelligence*, volume 32, pages 3350–3357. Palo Alto, CA: AAAI Press. `https://ojs.aaai.org/index.php/AAAI/article/view/11793`.

Kel'manov, A. V. and Pyatkin, A. V. E. (2016). On the complexity of some quadratic Euclidean 2-clustering problems. *Computational Mathematics and Mathematical Physics*, 56:491–497. DOI: 10.1134/S096554251603009X.

Király, B. and Tóth, L. (2011). Kombinatorika jegyzet és feladatgyűjtemény. Pécsi Tudományegyetem.

Kondor, G. (2018). k-szobatárs probléma metrikus térben – klaszterezés egyenlő elemszámú és kisméretű csoportokkal. In *Tavaszi Szél 2018 Konferencia = Spring Wind 2018: Konferenciakötet II.*, pages 549–563. ISBN: 9786155586316.

Kondor, G. (2022a). NP-hardness of m-dimensional matching problems. *Műhelytanulmány.*

Kondor, G. (2022b). Egyoldali párosítási piacok nehézségi eredményei magasabb dimenzióban. *Közgazdasági Szemle. Megjelenés alatt.*

Dr. Kovács, E., Szüle, B., Fliszár, V. and Vékás, P. (2011). *Pénzügyi adatok statisztikai elemzése: Egyetemi tankönyv.* Tanszék Kft., Budapest.

Lam, C.-K. and Plaxton, C. G. (2019). On the existence of three-dimensional stable matchings with cyclic preferences. In Fotakis, D. and Markakis, E., editors, *Algorithmic Game Theory*, SAGT 2019. Lecture Notes in Computer Science, vol 11801. Springer, Cham. DOI: 10.1007/978-3-030-30473-7_22.

Lin, J., Bertoni, A. and Goldwurm, M. (2016). Exact algorithms for size constrained 2-clustering in the plane. *Theoretical Computer Science*, 629:80–95. DOI: 10.1016/j.tcs.2015.10.005.

Malinen, M. I. and Fränti, P. (2014). Balanced k-means for clustering. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, S+SSPR 2014, pages 32–41. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-662-44415-3_4.

Morrill, T. (2010). The roommates problem revisited. *Journal of Economic Theory*, 145 (5):1739–1756. DOI: 10.1016/j.jet.2010.02.003.

Nobel Prize. (2012a). Press release. NobelPrize.org. `https://www.nobelprize.org/prizes/economic-sciences/2012/press-release/`.

Nobel Prize. (2012b). Scientific background. NobelPrize.org. `https://www.nobelprize.org/uploads/2018/06/advanced-economicsciences2012.pdf`.

Novick, B. (2009). Norm statistics and the complexity of clustering problems. *Discrete Applied Mathematics*, 157(8):1831–1839. DOI: 10.1016/j.dam.2009.01.003.

Pyatkin, A., Aloise, D. and Mladenović, N. (2017). NP-hardness of balanced minimum sum-of-squares clustering. *Pattern Recognition Letters*, 97:44–45. DOI: 10.1016/j.patrec.2017.05.033.

Rujeerapaiboon, N., Schindler, K., Kuhn, D. and Wiesemann, W. (2019). Size matters: Cardinality-constrained clustering and outlier detection via conic optimization. *SIAM Journal on Optimization*, 29(2):1211–1239. DOI: 10.1137/17M1150670.

Saunders, D. (2022). Weighted maximum matching in general graphs. MATLAB Central File Exchange. Letöltve: 2022. február 22. `https://www.mathworks.com/matlabcentral/fileexchange/42827-weighted-maximum-matching-in-general-graphs`.

Segev, D. L., Gentry, S. E., Warren, D. S., Reeb, B. and Montgomery, R. A. (2005). Kidney paired donation and optimizing the use of live donor organs. *Journal of the American Medical Association*, 293(15):1883–1890. DOI: 10.1001/jama.293.15.1883.

Weitz, R. R. and Lakshminarayanan, S. (1996). On a heuristic for the final exam scheduling problem. *Journal of the Operational Research Society*, 47(4):599–600. DOI: 10.1057/jors.1996.72.

Chapter 5

Publications

Journal articles

- Bihary, Zsolt ; Csóka, Péter ; Kondor, Gábor (2018). A részvénytartás spektrális kockázata hosszú távon. Közgazdasági Szemle, 65 : 7-8 pp. 687-700. DOI: 10.18414/KSZ.2018.7-8.687
- Csóka, Péter ; Kondor, Gábor (2019). Delegációk igazságos kiválasztása társadalmi választások elméletével. Közgazdasági Szemle, 66 : 7-8 pp. 771-787. DOI: 10.18414/KSZ.2019.7-8.771
- Csóka, Péter ; Kondor, Gábor (2020). Csődszabályok pénzügyi hálózatokban. Alkalmazott Matematikai Lapok, 37 : 2 pp. 233-245. 10.37070/AML.2020.37.2.08
- Kovács-Szamosi, Rita ; Kondor, Gábor ; Varga, József (2021). Derivatív-ügyletek az iszlám bankrendszerben. Köz-Gazdaság, 16 : 4 pp. 203-221. DOI: 10.14267/RETP2021.04.12
- Kondor, Gábor (2022). Egyoldali párosítási piacok nehézségi eredményei magasabb dimenzióban. Közgazdasági Szemle. In press.

Woring papers

- Kondor, Gábor (2022). NP-hardness of m-dimensional matching problems.