BACHELOR

Maximizing the expected number of transplants in a Kidney Exchange Program

Nabben, Zoy

Award date:
2022

Maximizing the expected number of transplants in a Kidney Exchange Program.

Zoy Nabben (student id: 1253646)

Abstract

In this report, we study the kidney exchange program (KEP). This is a program that aids people who need a new kidney and have found a willing but incompatible donor, to find a compatible donor.However, in practice, planned transplants are regularly cancelled.
In this report, several objectives used in existing KEPs are studied and their performance is evaluated with respect to the expected number of successful donations. The objectives which are studied are: maximizing the number of planned donations and maximizing the number of planned donation-cycles. These objectives are studied separately, as well as in a hierarchy, with the assumption that the percentage of success is 56%.
The result is that, with this fixed success rate, maximizing the number of cycles performs very well with respect to the expected number of successful donations. However, maximizing the number of planned donations performs relatively poorly.

Contents

1 Introduction

This section will first share a bit of history about live kidney donation, then explain the need for more donations and describes what a kidney exchange program (KEP) is. Afterwards it will give a short description of the problem and solution, and the method used to find this solution.

There are many conditions that can result in the need for a new kidney. This can be the case due to trauma obtained from an accident or due to severe illness. One can obtain a kidney through either deceased or living kidney donation. Deceased kidney donation is when a kidney is donated by a deceased person, while with living kidney donation, someone who is still alive agrees to donate one of their kidneys. The advantage of living kidney donation is that immediately after a compatible living donor has been found, the transplant can be planned, and there is only a short time span in between. In the case of deceased kidney donations, the transplant may need to be transported for many hours before it can be transplanted. Hence for live kidney donations, the organ will be of better quality, and therefore more likely will reduce the risk of kidney transplant failure. Table 1 shows the number of donations performed in 2021 and 2022 [1].

	2021	2022
Live kidney donation	199	255
Deceased kidney donation	208	216

Table 1: Number of live and deceased transplants performed in 2021 and 2022 in the Netherlands.

The first successful case of a long term live kidney donation was at "The Peter Brent Brigham Hospital, 1954", with identical twins. At that time there was no medication to minimize the rejection of a donated organ so the match had to be 100% perfect. The first donation between genetically non-related patients was done in 1962, using immunosuppression [2]. The number of living kidney donations has since then increased, as has the survival rate of both the recipients and the donated organs due to the improvement of medical care. Even though the quality of medical care has increased, there is still a shortage when it comes to kidney donors.

When someone is willing to donate a kidney, for example a friend or family member, but is found to be incompatible with the recipient, then the pair can choose to participate in the Kidney Exchange Program (KEP). The participants in this program are either non-directed donors (NDDs) or recipients with an incompatible partner donor. This program will then look for a set of transplants between recipients and compatible donors using either *cycles* or *chains*. A chain of transplants will start with an NDD donating to a recipient of a pair, the donor of this pair will then donate to the recipient of an other pair, etc. A cycle is similar to a chain, but only contains pairs, and the donor of the last pair will donate to the recipient of the first pair.

For example, consider a KEP containing two pairs, A and B. The donor of A is a friend or family member who wants to donate their kidney to the recipient of pair A but is found to be incompatible and that the same holds for pair B. On the other hand, the donor of pair A is compatible with the recipient of pair B and vice versa. The KEP can then match recipient A

with donor B and recipient B with donor A and a 2-way cycle has been found.

It is important to note that even though a transplant has been planned, there is always a possibility of cancellation, for example, because the donor changed his mind and is no longer willing to donate, or the recipient has recovered to such an extend that a transplant in no longer needed. As there is no more equal exchange, other transplants in the cycle or chain will also no longer be performed.

There are multiple ways to chose the set of transplants that will be planned. In this report, the following objectives, currently in use by European kidney exchange programs, will be considered and evaluated to see which selection will lead to good outcomes in terms of expected number of successful transplants [3]:

- Maximizing the number of planned donations, and
- Maximizing the number of selected cycles.

These objectives are currently in use by KEPs from the Netherlands, Belgium, Spain, etc. The advantage of these objectives is that they are independent on the exact, and possibly unknown, probability of failure. We look at these objectives individually, as well as in a hierarchy, with the objectives having different priorities.

Now, to clarify what is meant by a hierarchy of these objectives, suppose that there are three possible selections of planned donations. The first two selections contribute 7 planned donations, with the first selection using a 7-way cycle and the second selection uses one 3-way and one 4-way cycle. The last selection uses three 2-way cycles and hence would contribute only 6 planned donations.
When considering the objective of maximizing the number of planned donations individually, one will find that the first two selections are optimal, as they both contribute 7 planned donations. When considering the objective of maximizing the number of selected cycles as a second priority, the second selection will be seen as optimal since it contains one cycle more than the first selection. On the other hand, when considering the objective of maximizing the number of selected cycles individually, the third selection, contributing 6 planned donations using three 2-way cycles is optimal. Note that this selection will also be chosen when we look at the objectives hierarchically when giving first priority to maximizing the number of cycles in the selection.

While in practice the objectives mentioned are used by KEPs in Europe, the main goal is not to maximize the number of planned donations or transplant cycles, but to maximize the expected number of successful donations. This is why I evaluate these objectives on their performance with respect to a third objective, namely maximizing the expected number of successful donations, assuming there is a fixed success rate of 56% [4]. I will investigate the performance of the objectives of maximizing the number of planned donations and maximizing the number of cycles individually and if, in a hierarchy, improvements can be made when considering both objectives.

2 Conceptual model

The problem studied in this report is summarized is the following conceptual model.

Problem 2.1. Selecting an optimal collection of planned donations from a (very large) set of possible transplants.

Given: A set of incompatible donor-recipient pairs and a set of NDDs participating in the KEP, and information about compatibilities between donors and recipients.

Find: A selection of planned donations.

Such that:

1. A donor can only donate to a recipient if they are compatible.
2. Every participant can participate in at most one transplant.
3. A donor from a pair will only donate if their partner recipient in turn receives a kidney.

Maximizing: (one of) the following objective(s):

- The number of planned donations.
- The number of transplant-cycles.
- The expected number of successful donations.

A set of planned donations is considered feasible it it meets the requirements mentioned, and it is optimal if it maximizes the objective chosen. Hence it is optimal if it contains the maximum expected number of successful donations, contains the maximum number of planned donations, and/or contains the maximum number of transplant-cycles possible for the given data. In Sections 3.2.1 and 3.2.2 there will be an extra requirement, namely an upperbound for the cycle and chain lengths.

3 Mathematical models and solution approaches

This section will discuss three different methods to to model the kidney exchange problem. In Section 3.1, a case is considered where there is no restriction on cycle and chain lengths. This allows us to model the problem as a weighted bipartite matching problem. A **matching** in a graph $G = (V, E)$ is a set of edges $M \subseteq E$ which are pairwise disjoint. A **perfect matching** $M \subseteq E$, is a matching such that $|M| = \frac{1}{2}|V|$.

In Section 3.2.1 and 3.2.2 a method will be discussed that does allow a restriction on the cycle and chain lengths. A weighted directed graph will be used to represent the information given, and an integer programming (IP) model is used to formulate the problem, represented by an arc set containing disjoint cycles and chains in the directed graph.
For both models, the arcs or the cycles and chains, depending on the IP model, will be assigned weights depending the objective to be maximized. Next a selection of arcs or a selection of cycles and chains will be chosen that maximized the combined weight of the arcs or the cycles and chains, such that the planned donations represented by the selection meets the requirements of Problem 2.1.

3.1 Maximum weight bipartite matching.

In this section, I will discuss the following mathematical problem, which is a formalized version of the conceptual model Problem 2.1 with the maximize planned donations objective.

Problem 3.1. Selecting a maximum weight perfect matching in a compatibility graph.
Given:

- A positive integer n, and n pairs $(d_1, r_1), ..., (d_n, r_n)$,
- A nonnegative integer k, and k NDDs: $d_{n+1}, ..., d_{n+k}$,
- A bipartite graph graph $G = (V, E)$ with bipartition $V = D \cup R$ and $E = O \cup C$ with

$$D = \{d_1, ..., d_n, d_{n+1}, ..., d_{n+k}\},$$

$$R = \{r_1, ..., r_n, r_{n+1}, ..., r_{n+k}\},$$

where $r_{n+1}, ..., r_{n+k}$ are dummy recipients corresponding to $d_{n+1}, ..., d_{n+k}$ respectively,

$$O = \{\{d_i, r_j\} \mid i, j \in \{1, ..., n+k\}, i = j \text{ or } j > n\},$$

and

$$C = \{\{d_i, r_j\} \mid i, j \in \{1, ..., n+k\}, \text{ and donor } d_i \text{ is compatible with recipient } r_j\}.$$

Find: A perfect matching $M \subseteq E$ of maximum weight:

$$w(M) := \sum_{e \in M} w(e)$$

where the weight function $w : E \to \{0, 1\}$ is defined as:

$$w(e) = \begin{cases} 1 & e \in C, \\ 0 & e \in O \end{cases}$$

This model has the disadvantage that it does not include an upper limit for cycle and chain size and we cannot use this model to find a solution that maximizes the number of cycles. The advantage of this model formulation is that there exists an efficient algorithm to find a maximum weight perfect matching in a weighted bipartite graph containing a perfect matching.

Note that the bipartite graph representing the KEP will contain the perfect matching $M^* := \{\{d_i, r_i\} \mid i = 1, ..., n+k\}$ of weight $w(M^*) = 0$ and that for all matchings M in the bipartite graph G, the subgraph $(V, M \cup M^*)$ consists of a set of disjoint circuits and paths. An (edge set of a) circuit c in this subgraph represents a **chain** of transplants if there exists an $i > n$ such that $(d_i, r_i) \in c$, and it represents a **cycle** of transplants otherwise. The **length** of a cycle of chain corresponding to the circuit c is defined to be $n_c := |c \cap C|$. The path components of the subgraph $(V, M \cup M^*)$ are all of length 1 and represent the pairs and NDDs that have not been selected to participate in any chosen planned donations.

The expected number of successful donations of a cycle/chain c depends on the length n_c of the cycle/chain, as well as the probability that any given transplant is successful. In case of cycles, when all n_c transplants are successful it will contribute n_c successful transplants. If

even one transplant fails, then every other transplant in the cycles will have to be cancelled and the cycle will contribute no successful transplants. In case of chains, not all transplants have to be successful for the chain to contribute successful transplants. If the first transplant is successful, but the second is not, the chain will contribute one successful transplant. If the first and the second transplant are successful, but the third is not, the chain will contribute two successful transplants, et cetra. The following formulas give the expected number of successful donations of a cycle/chain c of length n_c if p the success rate of an individual transplant:

$$E_p(c) = \begin{cases} \sum_{k=1}^{n_c-1} kp^k(1-p) + n_c p^{n_c} & \exists i > n : (d_i, r_i) \in c \\ n_c \cdot p^{n_c} & \text{otherwise} \end{cases} \tag{1}$$

An example of a bipartite graph representing a KEP is depicted in Figure 1.

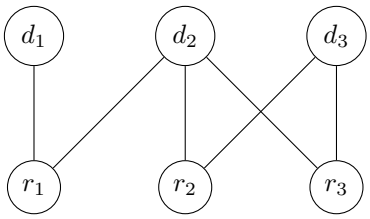


Figure 1: A bipartite graph $G = (V, E)$ representing a KEP containing 3 pairs.

A matching of maximum weight and matching all the recipients, can then be found using Algorithm 1 [5] based on the Hungarian method.

This algorithm is an iterative algorithm and in each iteration the cardinality of the previously found matching M will increase by 1. Let $G = (V, E)$, be a bipartite graph with bipartition $V = D \cup R$, containing a perfect matching, and let M be (the edge set of) a previously found matching. Then the directed graph $D^M = (V, A^M)$ is defined such that for all edges $e = \{d, r\} \in M$ there is an arc $a = (d, r) \in A^M$ with length $l_M(a) = w(e)$, and for all edges $e = \{d, r\} \in E \setminus M$, there is an arc $a = (r, d) \in A^M$ with length $l_M(a) := -w(e)$. Next a shortest path P will be found between unmatched vertices in R and D. This path determines a new matching $M' = M \triangle P = (M \cup P) \setminus (M \cap P)$ which contains one more edge. This process will be repeated until the newly found matching M' is a perfect matching. Note in a bipartite compatibility graph, there is always a perfect matching, namely M^*.

Figure 2 shows Algorithm 1 applied to the KEP represented by Figure 1 where the objective is to maximize the number of planned donations. Hence, the edges representing transplants have been assigned unit weight.

Definition 3.2. *A matching M is called **extreme** if it is of maximum weight among all the matchings of the same cardinality.*

Definition 3.3. *Let M be a matching in graph $G = (V, E)$. An M-**augmenting path** in G is a path between two vertices not covered by M, such that the edges of the path are alternatingly in M and in $E \setminus M$.*

Theorem 3.4. *The matching M found by Algorithm 1 is a maximum weight perfect matching [5].*

Proof. We will prove by induction on the iteration number that in each iteration the matching M is extreme. Before the first iteration (of the while loop), M is empty, and hence extreme. Suppose that in some iteration the matching $M \neq \emptyset$ is extreme and not yet perfect. Now let P and M' be the path and matching found in the next iteration. Let N be an extreme matching such that $|N| = |M'| = |M| + 1 > |M|$. Then $M \cup N$ contains an M-augmenting path, P'. Since P was found to be the shortest path between two unmatched vertices, one in D' and one in R', the length of P has to be smaller than or equal to the length of P': $l(P) \leq l(P')$. Since $|N \triangle P'| = |M|$, and M is extreme, the weight of $N \triangle P'$ is less than or equal to the weight of matching M: $w(N \triangle P') \leq w(M)$. Hence

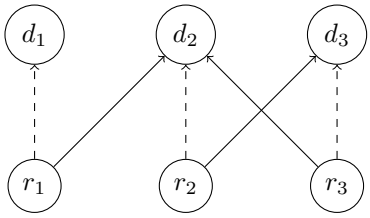
$$w(N) = w(N \triangle P') - l(P') \leq w(M) - l(P) = w(M')$$

Hence, every iteration finds an extreme matching M'.
Since it is mandatory that the bipartite graph G contains a perfect matching, we know in every iteration, there will either be an R'-D' path, or the matching will be perfect. Hence, since Algorithm 1 finds a perfect matching, and every iteration finds an extreme matching, it finds a maximum weight perfect matching. □

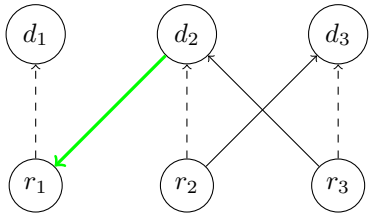
Algorithm 1 Using a weighted graph to find a matching of maximum weight [5].
Input: A bipartite graph $G = (V, E)$ with bipartition $V = D \cup R$ and containing a perfect matching, and a weight function $w : E \to \mathbb{R}_+$.
Output: An maximum weight perfect matching M.

1: $M := \emptyset$
2: **while** M is not a perfect matching **do**
3: Define $D^M = (V, A^M)$ with length function $l_M : A^M \to \mathbb{R}$ such that
4: **if** $e = \{d, r\} \in M$ **then**
5: $a = (d, r) \in A^M$.
6: $l_M(a) = w(e)$.
7: **else**
8: $a = (r, d) \in A^M$.
9: $l_M(a) = -w(e)$.
10: **end if**
11: Define $D' \subseteq D$ and $R' \subseteq R$ to be the vertices not in the matching.
12: Let P be the set of edges of the shortest $R' - D'$ path in D^M found by using the Bellman-Ford algorithm.
13: Update the matching M to $M = M \triangle P$.
14: **end while**
15: Return M.

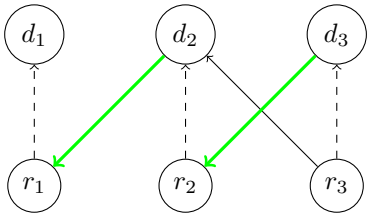
Graph D^1 depicted in Figure 2a is the result of directing the edges of the graph, according to Algorithm 1. Since we start with an empty matching, all arcs in Figure 2a, have been directed from the recipient to the donor with length $-w(e)$. Hence, the arcs $(r_1, d_2), (r_2, d_3)$, and (r_3, d_2) have length -1, and the remaining arcs have length 0.
The vertex set D' and R', containing the vertices that have yet to be matched, are the original vertex sets D and R since no vertex has been matched yet. It can be seen that there are three options for the shortest R'-D', namely, $(r_1, d_2), (r_2, d_3)$, and (r_3, d_2). Working in ascending order by the index of the vertices in R, P is defined to be $P := \{\{r_1, d_2\}\}$. We find an extreme matching $M := \{\{r_1, d_2\}\}$ of size 1, and weight 1. Since $|M| < 3$, the matching is not complete



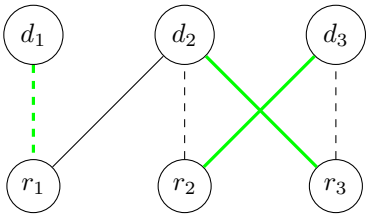
(a) The resulting directed graph $D^1 = (V, A^1)$ obtained during the first iteration of the algorithm on the bipartite graph G depicted in Figure 1. The matching is empty.



(b) The resulting directed graph $D^2 = (V, A^2)$ obtained during the second iteration. The edge $\{d_2, r_1\}$ has been added to the matching in the first iteration.



(c) The resulting directed graph $D^3 = (V, A^3)$ obtained during the third iteration of the algorithm. The edge $\{d_3, r_2\}$ has been added in the second iteration to the matching.



(d) The resulting extreme perfect matching $M := \{\{d_1, r_1\}, \{d_2, r_3\}, \{d_3, r_2\}\}$ found using Algorithm 1 on the Graph G depicted in Figure 1. The edges $\{d_2, r_2\}$ and $\{d_1, r_1\}$ have replaced the edge $\{d_2, r_1\}$ in the third iteration.

Figure 2: Applying Algorithm 1 to the bipartite graph G shown in Figure 1, where the edges from matching M are depicted by thick green arcs or edges. The dotted arcs and edges have been assigned length or weight 0, whereas the other arcs edges have been assigned weight 1 and the other arcs have been assigned length ± 1, depending on their direction.

and hence a matching of size 2 will be found in the next iteration.

Since the matching already contains edge $\{d_2, r_1\}$, the edge $\{d_2, r_1\}$ will be directed from donor to recipient with length $l((d_2, r_1)) = w(\{d_2, r_1\}) = 1$. Every other edge will be directed from recipient to donor and have weight $-w(e)$. The resulting graph D^2 is shown in Figure 2b. Since the vertices r_1 and d_2 have been matched, the vertex sets D' and R' become $\{d_1, d_3\}$ and $\{r_2, r_3\}$ respectively. Now the set of edges of shortest $R' - D'$ path, found using the Bellman-Ford algorithm, is $P := \{\{r_2, d_3\}\}$. So the extreme matching M size 2 is $M := \{\{r_1, d_2\}, \{r_2, d_3\}\}$, of weight 2. Since M is not perfect, another iteration follows.

Again, the edges in the matching M, $\{d_2, r_1\}$ and $\{d_3, r_2\}$, are directed from donor to recipient with length 1, and every other edge are directed from recipient to donor, with the arc $\{r_3, d_2\}$ having length -1, and every other arc having length 0. The obtained directed graph is depicted in Figure 2c.
Now $D' = \{d_1\}$ and $R' = \{r_3\}$. Using the Bellman-Ford algorithm, a set of edges of a shortest R'-D' path is $P := \{\{r_3, d_2\}, \{d_2, r_1\}, \{r_1, d_1\}\}$. Note that the edge $\{d_2, r_1\}$ is not only represented in the newly found path P, but it is also already an element of the earlier found matching M. Hence, the new matching becomes $M \triangle P = \{\{d_3, r_2\}, \{r_3, d_2\}, \{r_1, d_1\}\}$. Now

the extreme perfect matching $M := \{\{d_3, r_2\}, \{r_3, d_2\}, \{r_1, d_1\}\}$ of size 3 and weight 2 has been found. Hence, since the weight of the matching is defined to be the number of planned donations, we know that for the KEP depicted by Figure 1, the maximum number of transplants that can be planned is 2, with the edges $M \setminus \{\{d_1, r_1\}, \{d_2, r_2\}, \{d_3, r_3\}\} = \{\{d_3, r_2\}, \{r_3, d_2\}\}$ corresponding to the planned donations.

Since every iteration increases the cardinality of the matching M by one and we start from an empty matching, we find that the algorithm takes $\frac{1}{2}|V|$ iterations. The Bellman-Ford algorithm runs in $O(|V| \cdot |A|)$ time, and since $|A| = |E|$, we find that Algorithm (1) can be performed in time $O(|V|^2 \cdot |E|)$. This shows that using a maximum weight bipartite matching model, we can compute a maximum weight perfect matching in polynomial time.
After such a perfect matching, i.e. a maximum set of planned donations has been found, we can compute the expected number of successful donations from this set by applying formula (1).

3.2 Integer programming formulations

The next methods use directed graphs to represent KEPs, and an IP model is used to formulate Problem 2.1 with the two objectives: maximizing the number of planned donations and maximizing the number of transplant-cycles. A big advantage over the former method is that with these formulations, an upperbound can be established for the cycle and chain lengths and the objectice of maximizing the number of selected transplant cycles can be applied.
The **Arc formulation** uses variables representing the planned donations, whereas the **Cycle/chain formulation** uses variables directly representing cycles and chains.

3.2.1 Arc formulation

This section will discuss the arc formulation introduced in [6, 7]. It uses IP to select a set of arcs from a directed graph representing the KEP. This set of arcs will represent the planned donations and hence will be seen as the solution.

We will refer to the following mathematical problem, as the arc formulation of Problem 2.1:

Problem 3.5. Selecting an optimal arc set in a directed compatibility graph.
Given:

- A set $P = \{1, ..., n\}$ of $n \in \mathbb{N}$ donor-recipient pairs,
- A set $N = \{n+1, ..., n+k\}$ of $k \in \mathbb{N} \cup \{0\}$ NDDs,
- A directed graph $D = (V, A)$, where $V = P \cup N$ and

  $A = \{(i, j) \mid i \in V, j \in P,$ donor (of pair) i is compatible with the recipient of pair $j\}$,

- An upper bound for the circuit and path length, $l_{cy}, l_{ch} \in \mathbb{N}$, and
- A weight function $w : A \to \mathbb{R}$.

Find: A subset $A^* \subseteq A$ with

$$C_1^* = \{c \mid c \text{ is (an arc set of) a circuit in } A^*\}, \text{ and}$$

$$C_2^* = \{p \mid p \text{ is (an arc set of) a path in } A^*\}$$

Such that:

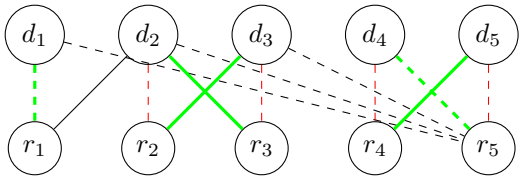
1. In $D(V, A^*) : \forall v \in N : d^+(v) \leq 1$
2. In $D(V, A^*) : \forall v \in P : d^+(v) \leq d^-(v) \leq 1$
3. $\forall c \subseteq C_1^* : |c| \leq l_{cy}$
4. $\forall p \subseteq C_2^* : |p| \leq l_{ch}$

Maximizing the weight function:

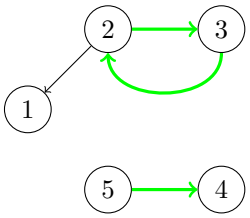
$$w(A^*) = \sum_{a \in A^*} w(a)$$

In this section, the term **chain** in the graph D will refer to an arc set of a directed path, starting with a vertex in N and then only containing vertices in P. The term **cycle** is defined to be an arc set of a directed circuit, only containing vertices in P. The **length** of a cycle or chain c is defined to be the number of arcs the path or circuit contains, $n_c := |c|$.

The definitions of the cycles and chains in this weighted directed graph D, correspond to their definitions in a weighted bipartite graph G defined in Section 3.1.
Let $c = ((i_1, i_2), (i_2, i_3), ..., (i_m, i_1)) \in C_1^*$. Now split up the vertices $i_1, ..., i_m \in P$ into two vertices, $d_{i_1}, ..., d_{i_m} \in D$ and $r_{i_1}, ..., r_{i_m} \in R$, and define the edge set $M := \{\{d_{i_k}, r_{i_l}\} \mid (i_k, i_l) \in c\}$. This results in a circuit, defining a cycle, in the bipartite graph $G = (D \cup R, M^* \cup M)$ as defined in Section 3.1.
Now let $c = ((i_1, i_2), (i_2, i_3), ..., (i_{m-1}, i_m)) \in C_2^*$. Once again split the vertices into two vertices, but now define the edge set $M := \{\{d_{i_k}, r_{i_l}\} \mid (i_k, i_l) \in c\} \cup \{d_{i_1}, r_{i_m}\}$ (since $i_1 \in N$). This again results in a circuit, defining a chain, in the bipartite graph $G = (D \cup R, M^* \cup M)$ as defined in Section 3.1.
This is shown in Figure 3 with Figure 3a depicting Graph G with a possible maximum weight perfect matching in green, and the graph D in Figure 3b depicting the arcs corresponding to the same selection of planned donations in green.



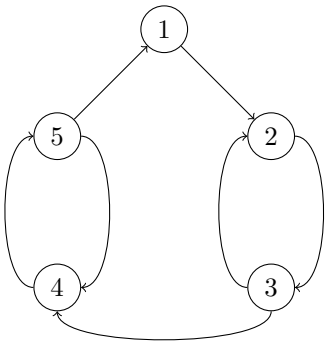
(a) A maximum weight perfect matching M in green found in graph $G = (V, E)$, meeting the requirements mentioned in Section 3.1.



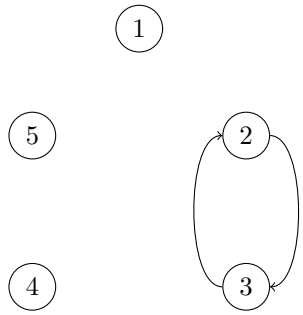
(b) The solution found in the directed graph $D = (V, A)$ representing the same collection of planned transplants as the matching shown in Subfigure 3a.

Figure 3: The relations between the solution of the bipartite graph and directed graph of a KEP shown containing 4 pairs and 1 NDD.

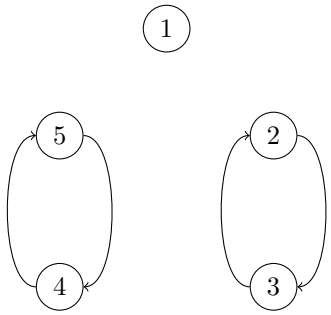
Figure 4 shows an example of a compatibility graph for a KEP containing 5 pairs with a possible corresponding feasible and possible corresponding optimal solutions. The compatibility graph $D_1 = (V, A_1)$ is depicted in Figure 4a. Figure 4b depicts an arc set A'_1 representing a feasible solution, Figure 4c depicts an arc set A''_1 representing an optimal solution with respect to maximizing the number of transplant cycles, and Figure 4d depicts an arc set A'''_1 representing an optimal solution with respect to maximizing the number of planned donations.



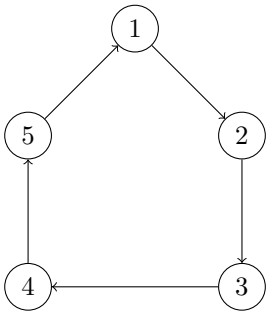
(a) An example of a compatibility graph D_1 for a KEP containing 5 pairs, with the nodes representing the pairs and the arcs representing compatibility.



(b) A feasible solution, represented by the arc set A'_1, only selecting the cycle $(2, 3)$.



(c) An optimal solution, represented by the arc set A''_1 maximizing the number of cycles.



(d) An optimal solution, represented by the arc set A'''_1 maximizing the number of arcs.

Figure 4: Three possible solutions, represented by the arc sets A'_1, A''_1 and A'''_1, for the compatibility graph D_1.

Now we are going to discuss the **position index formulation.** Every arc will be assigned multiple variables, depending on their possible placements in cycles and chains. Hence, limits on the cycle length will be enforced through the use of position-indices, denoting the position of a transplant in a cycle or chain. Furthermore, to reduce symmetry, the formulation makes use of sub-graphs [7]. An example is shown by Graph $D_2 = (V, A)$ in Figure 5.

In Figure 5, the cycle $(1, 2, 3)$ can be found. If this cycle is rejected, then we want the cycles $(1, 3, 2), (2, 1, 3)$, $(2, 3, 1), (3, 1, 2)$, and $(3, 2, 1)$ to also be rejected since they contain the exact same pairs, but in different orders and/or with a different start- and endpoint. By using sub-graphs and using subgraphs-indices, denoting from which graph the arcs are chosen, symmetry will be reduced.

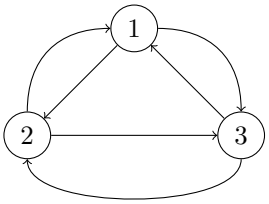
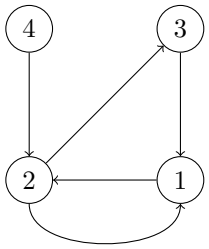
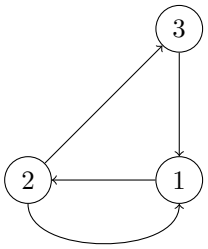


Figure 5: A directed graph D_2 representing a KEP containing three pairs with arcs representing compatibility.

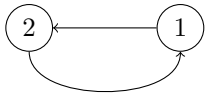
Let $D = (V, A)$ with $V = P \cup R$ be a directed graph representing a KEP containing n pairs, $P = \{1, ..., n\}$, and k NDDs, $N = \{n+1, ..., n+k\}$. Next, define for each vertex $l \in V$, a subgraph $D^l = (V^l, A^l)$ is with $V^l = \{i \in V \mid i \leq l\} = \{1, 2, ..., l\}$ and $A^l = \{(i, j) \in A \mid i, j \leq l\}$. These subgraphs, together with the position indexing, allows us to limit the cycle and chain lengths and reduce symmetries in the problem. From every subgraph D^l with $l > n$, at most one chain will be selected with start point l, and in every subgraphs D^l with $l \leq n$, at most one cycle will be selected, containing vertex l as the start- and as endpoint. An example of the construction of subgraphs is shown in Figure 6.



(a) The directed graph $D_3^4 = (V^4, A^4)$ representing a KEP containing 3 pairs and 1 NDD.



(b) The directed sub-graph $D_3^3 = (V^3, A^3)$ with $V^3 := V^4 \setminus \{4\}$.



(c) The directed sub-graph $D_3^2 = (V^2, A^2)$ with $V^2 := V^3 \setminus \{3\}$.

(d) The directed sub-graph $D_3^1 = (V^1, A^1)$ with $V^1 := V^2 \setminus \{2\} = \{1\}$.

Figure 6: The directed graph $D_3 = D_3^4$ and its sub-graphs D_3^3, D_3^2 and D_3^1, representing a KEP containing 1 NDD and 3 pairs.

Now let, $l_{cy}, l_{ch} \in \mathbb{N}$ be defined as the maximum length for cycles and chains respectively. Let $\mathcal{K}(i, j, l)$ be defined as the set of positions at which the arc (i, j) can be placed in a cycle in the subgraph D^l with start- and endpoint $l \in P$, and $i, j \in V^l$, and $(i, j) \in A^l$. Let $\mathcal{K}'(i, j, l)$ be defined as the set of positions at which the arc (i, j) can be placed in a chain in graph D^l with startpoint $l \in N$, and $i, j \in V^l$ and $(i, j) \in A^l$.

$$\mathcal{K}(i,j,l) = \begin{cases} \{1\} & i = l \\ \{2,3,...,l_{cy}-1\} & i,j < l \\ \{l_{cy}\} & j = l \end{cases}$$

$$\mathcal{K}'(i,j,l) = \begin{cases} \{1\} & i = l \\ \{2,3,...,l_{ch}\} & i < l \end{cases}$$

Now a binary variable x^l_{ijk} is defined such that $x^l_{ijk} = 1$ if and only if the arc (i,j) is selected in graph D^l at position $k \in \mathcal{K}(i,j,l)$ in a cycle. Similarly, the variable y^l_{ijk} is defined such that $y^l_{ijk} = 1$ if and only if the arc (i,j) is selected in graph D^l at position $k \in \mathcal{K}'(i,j,l)$ in a chain. Note that this limits the cycle and chain length to l_{cy} and l_{ch} respectively.

A weight function will be defined based on which objective is to be achieved. Let $l \in V$, then, when the objective is to maximize the number of planned donations, the weight function $w^l_1 : A^l \to \mathbb{R}_+$ will be defined as follows:

$$\forall a \in A^l : w^l_1(a) = 1.$$

Now to discuss how the arcs will be assigned their weight when the objective is to maximize the number of transplant-cycles. This will be done by letting the weight of an arc depend on their placement in the cycle or chain. Let $l \in V$ as defined, then the weight function $w^l_2 : A^l \to \mathbb{R}_+$ will be defined as follows:

$$w^l_2((i,j)) = \begin{cases} 1 & i = l \\ 0 & \text{Otherwise} \end{cases},$$

We choose this because if a cycle or chain c in graph D^l is selected, then there will always exists a vertex $j \in V^l$ such that $(l,j) \in c$. Hence, maximizing the number arcs starting a cycles or chains maximizes the number of cycles and chains selected.

We also define a third weight function $w^l_3 : A^l \to \mathbb{R}_+$:

$$w^l_3((i,j)) = \begin{cases} w & j \in V^l \setminus \{l\} \text{ and } i = l, \\ 1 & \text{Otherwise.} \end{cases}$$

Claim 3.1. *The weight function w^l_3 can be used to maximize both the number of planned donations and the number of donation cycles, with priority to maximizing the number of planned donations if $w = 1 + \frac{1}{|V|}$, and with priority to maximizing the number of donation cycles if $w = 1 + |V|$.*

Proof. Let C be a cycle/chain set containing c cycles and n planned donations with a_1 2-way cycles, a_2 3-way cycles, ..., a_k $(k+1)$-way cycles with $k+1 \leq l_{cy}$. So

$$c = \sum_{i=1}^{k} a_i$$

and

$$n = \sum_{i=1}^{k} (i+1) \cdot a_i.$$

Then the set C has the following weight:

$$\begin{aligned} w_3(C) &= a_1 \cdot (1+w) + a_2 \cdot (2+w) + ... + a_k \cdot (k+w) \\ &= \sum_{d=1}^{k} a_d \cdot (d+w) \\ &= \sum_{d=1}^{k} a_d \cdot ((d+1)+(w-1)) \\ &= \sum_{d=1}^{k} a_d \cdot (d+1) + (w-1) \cdot \sum_{d=1}^{k} a_d \\ &= n + (w-1) \cdot c \end{aligned}$$

Now we have two cycle/chain sets, C_1 and C_2, with c_1 and c_2 cycles and n_1 and n_2 planned donations respectively. Let C_1 contain more planned donations and C_2 contain more donation cycles, hence $|V| > c_2 > c_1 > 0$ and $|V| \geq n_1 > n_2 > 0$. Then we find the following for $w = 1 + |V|$:

$$\begin{aligned} w_3(C_2) &= n_2 + (w-1) \cdot c_2 \\ &= n_2 + |V| \cdot c_2 \\ &= n_2 + (c_2 - 1)|V| + |V| \\ &\geq n_2 + (c_2 - 1)|V| + n_1 \\ &> (c_2 - 1)|V| + n_1 \\ &\geq c_1|V| + n_1 \\ &= w_3(C_1) \end{aligned}$$

Hence we find the the set C_2, which contributes more planned donations but fewer donation cycles, has a higher weight. So if $w = 1 + |V|$, then maximizing the number planned donations has priority.
Now we find the following for $w = 1 + \frac{1}{|V|}$:

$$\begin{aligned} w_3(C_1) &= n_1 + (w-1) \cdot c_1 \\ &= n_1 + \frac{c_1}{|V|} \\ &= (n_1 - 1) + 1 + \frac{c_1}{|V|} \\ &> (n_1 - 1) + \frac{c_2}{|V|} + \frac{c_1}{|V|} \\ &> (n_1 - 1) + \frac{c_2}{|V|} \\ &\geq n_2 + \frac{c_2}{|V|} \\ &= w_3(C_2) \end{aligned}$$

Hence in this case, we find that set C_1, which contains more donation cycles but fewer planned donations, has a higher weight. So if $w = 1 + \frac{1}{|V|}$, then the number of cycles has

priority. So if $w = 1 + \frac{1}{|V|}$, then maximizing the number cycles has priority.

□

Let $C := C_1^* \cup C_2^*$ be the set of cycles and chains of an optimal solution of Problem 3.6. Then expected number of successful donations of cycle/chain $c \in C$ of length n_c when the success rate of a single transplant is p can be calculated, following the same reasoning as in Section 3.1 of fomula (1):

$$\mathbb{E}_p(c) = \begin{cases} \sum_{k=1}^{n_c-1} kp^k(1-p) + n_c p^{n_c} & \exists i \in N, j \in P : (i,j) \in c \\ n_c \cdot p^{n_c} & \text{Otherwise.} \end{cases} \tag{2}$$

Hence we find the following **position index edge formulation** [7] (3) - (9).
Position index edge formulation:

$$\text{Maximize: } \sum_{l \in V} \sum_{(i,j) \in A^l} \left(\sum_{k \in \mathcal{K}(i,j,l)} w_{ij}^l x_{ijk}^l + \sum_{k \in \mathcal{K}'(i,j,l)} w_{ij}^l y_{ijk}^l \right) \tag{3}$$

$$\text{Subject to: } \sum_{l \in V} \sum_{j:(j,i) \in A^l} \left(\sum_{k \in \mathcal{K}(j,i,l)} x_{jik}^l + \sum_{k \in \mathcal{K}'(j,i,l)} y_{jik}^l \right) \leq 1 \qquad i \in V \tag{4}$$

$$\sum_{\substack{j:(j,i) \in A^l \wedge \\ k \in \mathcal{K}(j,i,l)}} x_{jik}^l = \sum_{\substack{j:(i,j) \in A^l \wedge \\ k+1 \in \mathcal{K}(i,j,l)}} x_{i,j,k+1}^l \qquad \begin{matrix} l \in V, \\ i \in V^l \setminus \{l\}, \\ k \in \{1, ..., l_{cy} - 1\} \end{matrix} \tag{5}$$

$$\sum_{j:(l,j) \in A^l} y_{lj1}^l \leq 1 \qquad l \in N \tag{6}$$

$$\sum_{\substack{j:(j,i) \in A^l \wedge \\ k \in \mathcal{K}'(j,i,l)}} y_{jik}^l \geq \sum_{\substack{j:(i,j) \in A^l \wedge \\ k+1 \in \mathcal{K}'(i,j,l)}} y_{i,j,k+1}^l \qquad \begin{matrix} i \in V^l, \\ k \in \{1, ..., l_{ch} - 1\} \end{matrix} \tag{7}$$

$$y_{ijk}^l \in \{0,1\} \qquad \begin{matrix} l \in N, \\ (i,j) \in A^l, \\ k \in \mathcal{K}'(i,j,l) \end{matrix} \tag{8}$$

$$x_{ijk}^l \in \{0,1\} \qquad \begin{matrix} l \in P, \\ (i,j) \in A^l, \\ k \in \mathcal{K}(i,j,l) \end{matrix} \tag{9}$$

Here the objective function (3) maximizes the weighted sum of the selected arcs, where the first sum corresponds to the arcs selected to participate in a cycle, and the second sum corresponds to the arcs selected to participate in a chain.
Constraint (4) makes sure that for all $i \in V$, $d^-(i) \leq 1$. This can be in either a circuit or a path. Combined with Constraint (5), we make sure that if the vertex $i \in V$ is selected in a cycle, then it exactly one outgoing and one incomming arc of vertex i will be selected. Combining Constraint (4) with (6), we make sure that, for a vertex $i \in N$, at most one outgoing arc will of vertex i will be selected. Lastly, combining Constraint (4) with (7), we make sure that if vertex $i \in P$ is selected in a chain, then there will be at most one outgoing arc of i that will be selected.

Constraint (8) and (9) make sure that the variables x^l_{ijk} and y^l_{ijk} are binary. Hence the arc (i, j) is either placed in a cycle, at position k from subgraph l, or it is not. And, the arc (i, j) is either placed in a chain, at position k from subgraph l, or it is not.

At the end, the number of expected successful donations the arc set A^* contributes can be calulated using formula (2). As mentioned before, this model has the advantage that now a limit can be put on the length of the cycles and chains due to limiting l_{cy} and l_{ch}. This formulaiton has not been used in my research.

3.2.2 Cycle/chain formulation

This section will discuss the cycle/chain formulation, the solution will be defined to be the subset of cycles and chains from a pre-calculated set containing all possible cycles and chains up to a maximum length. These cycles and chains will be assigned a weight depending on the objective. The solution will be the subset of maximum weight with respect to the objective chosen.

Now we will discuss a mathematical model for the **Cycle/chain formulation**.

Problem 3.6. Selecting an optimal cycle/chain set in a directed compatibility graph.
Given:

- A set $P = \{1, ..., n\}$ of $n \in \mathbb{N}$ donor-recipient pairs,
- A set $N = \{n + 1, ..., n + k\}$ of $k \in \mathbb{N} \cup \{0\}$ NDDs,
- A directed graph $D = (V, A)$, where $V = P \cup N$ and

  $A = \{(i, j) \mid i \in V, j \in P,$ donor (of pair) i is compatible with the recipient of pair $j\}$,

- An upper bound for the circuit lengths $l_{cy} \in \mathbb{N}$ and path lengths, and $l_{ch} \in \mathbb{N}$
- A weight function $f : A \to \mathbb{R}_+$

Find: A set $C^* := C^*_1 \cup C^*_2$ with $C^*_1 \subseteq C_1$ $C^*_2 \subseteq C_2$ with

$$C_1 := \{c \subseteq A \mid c \text{ is (an arc set of) a circuit in D and } |c| \leq l_{cy}\},$$

and

$$C_2 := \{p \subseteq A \mid p \text{ is (an arc set of) a path in D and } |p| \leq l_{ch}\}.$$

Such that: In $D(V, C^*)$:

1. $\forall c_1, c_2 \in C^* : c_1 \cap c_2 = \emptyset$
2. $\forall p \in C^*, i \in P : (\exists j \in P : (i, j) \in p) \implies (\exists k \in V : (k, i) \in p)$

Maximizing the weight function:

$$f(C^*) = \sum_{c \in C^*} f(c)$$

Let C be the set of all possible cycles and chains. A weight function will be defined based on which objective is to be achieved. When the objective is to maximize the number of planned donations, then the weight function $f_1 : C \to \mathbb{R}_+$ for the **cycles** and **chains**, as in Section 3.2.1, will be defined as follows:

$$\forall c \in C : f_1(c) = n_c.$$

With $n_c := |c|$ the number of arcs the cycle or chain c contains. The weight function $f_2 : C \to \mathbb{R}_+$, used to maximize the number of transplant-cycles and -chains, will be defined as follows:

$$\forall c \in C : f_2(c) = 1.$$

The function used to maximize the expected number of donations when the probability of success is p will be $E_p : C \to \mathbb{R}$ (2) as defined in Section 3.2.1.

For this method, all the cycles and chains up to length l_{cy} and l_{ch} respectively, have to be calculated first. The following **Cycle/chain formulation** will then select a subset C^* to be the solution. Next, the following formulation is defined as the **Cycle/chain formulation**:

$$\text{Maximize: } \sum_{c \in \mathcal{C}} w_c \cdot x_c \tag{10}$$

$$\text{Subject to: } \sum_{c \in \mathcal{C}} t_{v,c} \cdot x_c \leq 1 \qquad \forall v \in V \tag{11}$$

$$v_c \in \{0, 1\} \qquad \forall v \in V \tag{12}$$

$$x_c \in \{0, 1\} \qquad \forall c \in C \tag{13}$$

The variable x_c in the IP is defined as follows:

$$x_c = \begin{cases} 1 & \text{The cycle } c \text{ is chosen in the solution} \\ 0 & \text{Otherwise.} \end{cases}$$

The parameter w_c is a representation of the given weight of each cycle. This will depend on the objective to be maximized, hence we can define the parameter w_c as follows:

$$w_c = \begin{cases} n_c & \text{When maximizing } f_1, \\ 1 & \text{When maximizing } f_2, \\ n_c + w & \text{When maximizing both } f_1 \text{ and } f_2, \\ E_p(c) & \text{When maximizing the number of successful donations.} \end{cases}.$$

with $w = |V|$ when there is priority on maximizing the number of cycles, and $w = \frac{1}{|V|}$ when there is priority on maximizing the number of planned donations by Claim 3.1.

$E_p(c)$ is the number of expected successful donations which depends on the cycle c and the probability of success of a transplant p calculated using formula (2). The last parameter to discuss is $t_{v,c}$ with $v \in V$ and $c \in C$, which is defined as follows:

$$t_{v,c} = \begin{cases} 1 & \text{The vertex } v \text{ is in cycle } c, \\ 0 & \text{Otherwise.} \end{cases}$$

Hence, depending on how w_c is defined, (10) maximizes the number of expected successful donations, the number of planned donations, or the number of cycles. The constraint 11 makes sure that every pair or NDD can only participate in at most one cycle or one chain.

Of course, before this solution can be found, all the possible cycles and chains up to a chosen length have to been found. How these cycles and chains have been calculated, is discussed in Section 4. For every directed graph D there is an upper bound for the number of cycles and valid chains that can be found, which will be discussed next.

Upperbound on the number of cycles and chains when $l_{cy} = l_{ch} = \infty$.
Let $|P|$ denote the number of pairs that participate. Suppose that there is no maximum limit to the length of cycles and no limit on chains and that the for all $i, j \in P$, donor i is compatible with recipient j, except if $i = j$. Since the order does not matter we find that if we want to know how many possible cycles there are of length l, the answer will be the number of ways to select a group of l pairs from a collection of $|P|$ pairs:

$$\binom{|P|}{l}.$$

Note that, since it is assumed that only incompatible donor-recipient pairs participate in the KEP, the graph representing the KEP does not contain any loops. Hence, at a minimum two arcs have to be traversed to obtain a circuit. Also note that, since the graph contains $|P|$ vertices with both incoming and outgoing arcs, the maximum length of a circuit is $|P|$. Combining everything we can conclude that the upperbound of cycles in a directed graph with $|P|$ pairs will be:

$$\sum_{l=2}^{|P|} \binom{|P|}{l}.$$

Next, we look at an upperbound on the number of chains. Let $|N|$ the number of NDDs that participate and for all $i \in N$, $j \in P$, NDD i is compatible with the recipient of pair j. It is evident that, by simply adding an NDD to a group of n pairs, you can create a chain of length $n + 1$ due to the absence of compatibility restrictions. Note that every new group will contain exactly one NDD, but this time the order does matter (the NDD starts the chain). Hence a chain can, at maximum, include all $|P|$ pairs and one NDD. Hence, the upperbound on the chain length is $|P|+1$. Combining all this, we find that if there are $|N|$ NDDs who have been found compatible with every participating recipient, then the number of distinct groups containing exactly one NDD and at least one pair is:

$$|N| \cdot \sum_{l=1}^{|P|} \binom{|P|}{l}.$$

Hence the upper bound of the total possible number of cycles and chains is:

$$\sum_{l=2}^{|P|}\binom{|P|}{l}+|N|\cdot\sum_{l=1}^{|P|}\binom{|P|}{l}=\sum_{l=2}^{|P|}\binom{|P|}{l}+|N|\cdot|P|+|N|\cdot\sum_{l=2}^{|P|}\binom{|P|}{l}$$
$$=(|N|+1)\sum_{l=2}^{|P|}\binom{|P|}{l}+|N|\cdot|P|$$
$$<|N|\cdot|P|+(|N|+1)\sum_{l=2}^{|P|}\frac{|P|^l}{l!}.$$

In conclusion, when the size of the cycles and chains is not limited, then a upperbound of order $O\left(|N|\cdot|P|^{|P|}\right)$ is found. Hence the upperbound is exponential with respect to the number of participating pairs. Now we how it will improve when an upperbound is established on the cycle and chain lengths.

Maximum size of the model with $l_{cy}, l_{ch} < \infty$.
Now to calculate the upper bound when the maximum length of a cycle and chain is $l_{cy} < \infty$ and $l_{ch} < \infty$ respectively. Note that the following still holds for the number of possible cycles of length $l \in \{2, ..., l_{cy}\}$, when in a complete graph:

$$\binom{|P|}{l}.$$

Hence we find the following formula to calculate the total number of cycles with length l_{cy}:

$$\sum_{l=2}^{l_{cy}}\binom{|P|}{l}.$$

Now for the number of chains. Let $L = \min\{l_{cy}+1, l_{ch}\}$. Once again use the fact that by adding an NDD to a group of $l < L$ pairs, a chain of length $l+1 \leq L$ is obtained, hence:

$$|N|\cdot\sum_{l=1}^{L-1}\binom{|P|}{l}.$$

The upper bound of the total possible number of cycles and chains where cycles have maximum length l_{cy} and chains have maximum length L will be the following:

$$\sum_{l=2}^{l_{cy}}\binom{|P|}{l}+|N|\cdot\sum_{l=1}^{L-1}\binom{|P|}{l}$$

Hence if $l_{cy}+1 \leq l_{ch}$, then an upperbound for the number of cycles and chains is of order $O\left(|N|\cdot|P|^{l_{ch}}\right)$. If $l_{cy}+1 > l_{ch}$, then an upperbound of order $O\left(|N|\cdot|P|^{l_{cy}}\right)$ is found. Hence, in both cases, the upperbound has now become polynomial with respect to the number of participating pairs.

As we have seen, when there is no restriction on the cycle and chain size, l_{cy} and l_{ch}, then the number of variables might increases exponentially when there are more pair in the KEP. However by putting a restriction on the size, it is found that the number of variables increases polynomially in the cycle/chain lengths. In this report, we considered pairs and chains up to size 4, hence the upperbound has order $O\left(|N|\cdot|P|^4\right)$.

4 Implementation of the arc/chain formulation

In Section 3.2.1, I defined a cycle c as a selection of arcs. In this section, we use the **vertex sequence** to represent a cycle:

Definition 4.1. *Let $D = (V, A)$ be a directed graph containing the finite path $p = ((i_0, i_1), (i_1, i_2), ..., (i_{n-1}, i_n))$. Then the sequence of vertices $(i_1, i_2, \ldots, i_n)$ is defined to be the **vertex sequence** of p.*

(Note that path p will be closed if $i_0 = i_n$.) Now we show Algorithm 2 representing the code used to find all possible chains up to and including the chains of length l_{ch}. In Algorithm 2 and 3, an arc $a = (i_k, i_{k+1}) \in A$ can **extend** path $p = (i_0, ..., i_n)$ if and only if $k = n$.

Algorithm 2 Finding all chains of maximum length l_{ch}.
Input: A set of pairs $P = \{1, ..., n\}$, a set of NDDs $N = \{n + 1, ..., n + k\}$, an arc list A and an upperbound $l_{ch} \geq 1$.
Output: The collection C of chains of length $l \leq l_{ch}$.

```
C := ∅
for l = 1 to l = l_ch do
    if l = 1 then
        S_0 := ∅
        for all i ∈ N do
            Find all arcs in (i, j) ∈ A with j ∈ P and add them to C and S_0.
        end for
        Update A := A \ S_0.
          ▷ These arcs can only be placed at the first position of a chain (see Section 3.2.1).
    else
        S_1 := ∅
        for every path p ∈ S_0 do
            for every a ∈ A that can extent p do
                if the extended path p' = p ∪ a does not traverse a vertex multiple times then
                                                          ▷ To prevent sub-cycles.
                    if there no path in S_1 containing the same vertices then
                                                          ▷ To prevent symmetry.
                        Add the vertex sequence of p' to C.
                    end if
                    Add the vertex sequence of the path p' to S_1.
                end if
            end for
        end for
        S_0 := S_1
    end if

    if S_0 = ∅ then
            ▷ There are no more paths to be extended, all possible chains have been found.
        Break
    end if
end for
Return C
```

Let $D(V, A)$ be a directed compatibility graph, with $V = P \cup N$ with $P := \{1, ..., n\}$ and $N := \{n + 1, ..., n + k\}$. First we define an empty set C, which will contain all found vertex sequences representing found chains. Next we create a set S_0 containing the vertex sequence of all the paths of length $l < l_{ch}$ that do not contain subcycles, and a set S_1 to collect all the vertex sequences of the new paths that do not contain any subcycles and are of length $l + 1 \leq l_{ch}$, found by extending the paths in S_0. First we look for all arcs $a = (i_0, i_1) \in A$ with $i_0 \in N$. In every iteration l we look for all possible paths p of length l. For every newly found path, we check if (1) it does not traverse a vertex multiple times and (2) there has not already been another path $p' \in S_1$ containing the same vertices (in a different order). If (1) holds, then the vertex sequence of path p will be added to S_1. If (2) also holds, then the vertex sequence of path p' will also be considered a chain and will be added to C.

Next we look for all possible cycles in D using Algorithm 3.

Algorithm 3 Finding all cycles of maximum length l_{cy}.
Input: A set of pairs $P = \{1, ..., n\}$, an arc list A and an upperbound $l_{cy} \geq 2$.
Output: The collection C of cycles of length $l \leq l_{cy}$.

$C := \emptyset$
for $i \in P$ **do**
  for $l = 1$ to l_{cy} **do**
    if $l = 1$ **then**
      $S_0 := \emptyset$
      Find all arcs $(i, j) \in A$ with and add them to S_0.
      Update $A = A \setminus S_0$.

    else
      $S_1 := \emptyset$
      $S_2 := \emptyset$
      for every $p \in S_0$ **do**
        for every arc $a \in A$ that can extent p **do**
          if path $p' = p \cup a$ is closed and there is no $p^* \in S_2$ traversing the same vertices **then**
            Add the vertex set of p' to S_2 and C.

          else if path p' does not traverse a vertex multiple times **then**
            Add the vertex sequence of p' to S_1.

          end if
        end for
      end for
      $S_0 := S_1$
    end if
  end for

  Update A such that it no longer contains any incoming arcs of vertex i
end for
Return C

In contrast to the method for finding chains, we are now focusing on iteratively finding all circuits containing vertex $i \in P$, and then alter the arc list A such that it no longer contains the incoming and outgoing arcs of vertex i to prevent symmetry. (For example, if we find the vertex sequence $(1, 2, 1)$, we don't also find the vertex sequence $(2, 1, 2)$ representing the same circuit in the next iteration.) A found path p will be seen as a circuit if it is closed, it does not contain a subcycle and no other circuit has already been found traversing the same pairs.

5 Results

In this section, I will share the results that have been found. We will start by discussing the results found when considering complete graphs, representing KEPs without compatibility restrictions. Next we discuss the results on realistic graphs representing KEPs with compatibility restrictions.
In the complete graphs representing KEPs without compatibility restrictions, maximizing the number of planned donations and maximizing transplant-cycles performs well with respect to the expected number of successful donations. However when considering compatibility restrictions, the realistic graphs show that only maximizing cycles performs well. Maximizing the number of planned donations performs poorly with respect to the number of expected successful donations.

5.1 Complete graphs

In this subsection, the relationship between the number of donation cycles and the the number of planned donations, the relation between the expected number of successful donations and the number of donation cycles, and the relation between the number of planned donations and the expected number of successful transplants on complete graphs is studied. For this purpose we look at Problem 5.1:

Problem 5.1. Finding a cycle set in a complete graph.
Given:

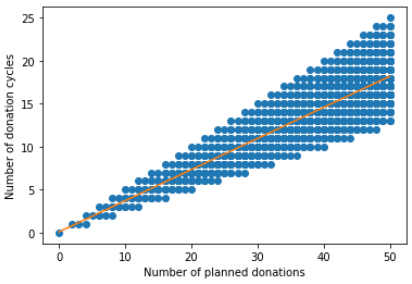
- A set $V = \{1, ..., n\}$ of $n \in \mathbb{N}$ donor-recipient pairs,
- A complete graph $D = (V, A)$, and
- The weight function E_p from section 3.2.1, see formula (2).

Find: A set of cycles in D, C^*, with $A^* = \{a \in A \mid \exists c \in C^* : a \in c\}$.

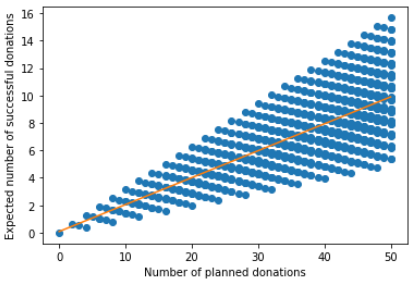
Such that: In $G(V, A^*)$:

$$\forall v \in V : d^+(v) = d^-(v) = 1$$

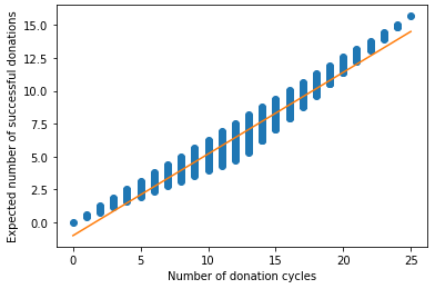
We looked at solutions C^* with $|A^*| = 1, ..., 50$ planned donations. For each of these solutions, all combinations of cycles of size 2, 3, and 4 that could produce this number of planned donations was determined, combined with the resulting expected number of successful donations. The results are shown in Figure 7.
Figure 7 shows three figures depicting the relation between the number of planned donations, the number of cycles, and the expected number of successful donations associated with the set C^*.



(a) Showing the relationship between the number of cycles and the number of planned donations. There is a dot at point (x, y) if there is a cycle set C^* with x planned donations and y donation cycles.



(b) Showing the relationship between the expected number of successful donations and the number of planned donations. There is a dot at point (x, y) if there is a cycle set C^* with x planned donations and y expected successful donations.



(c) Showing the relationship between the expected number of successful donations and the number of cycles in the solution. There is a dot at point (x, y) if there is a cycle set C^* with x cycles and y expected successful donations.

Figure 7: Showing the relationships the number of planned donations, cycles and expected donations, when we are working in a complete directed graph.

The Figures 7a, 7b and 7c show that the expected number of successful donations increases when the number of cycles and the number of planned donations associated with the cycle set C^* increases.
However Figure 7a and 7b also shows that when the number of planned donations increases, the spread of the possible resulting expected number of successful donations, as well as the resulting number of cycles, becomes wider.

This has as result that maximizing the number of planned donations will not always have the desired result when it comes to maximizing the expected number of successful donations. For example, consider a KEP with two potential cycle sets: cycle set C_1 containing one 2-way cycle and cycle set C_2 containing one 4-way cycle (so both C_1 and C_2 contain one cycle, but C_1 contributes only two planned donations while cycle set C_2 contributes four planned donations). Then the expected number of successful donations from cycle set C_1 would be 0.63, while the expected number of successful donations cycle set C_2 would present is 0.39. So in this case, maximizing the number of planned donations would give a poor result with respect to the expected number of successful donations.
Figure 7c shows that there are similar cases when it comes to maximizing the number of cycle. However this spread is more limited.

So we found that, even though there are exceptions, the number of expected successful donations increases when the number of cycles and planned donations included in the solutions increases. Hence when there are no compatibility restrictions, maximizing the number of successful donations and maximizing the number of transplant cycles performs well with respect to maximizing the expected number of successful donations.

5.2 Realistic graphs representing KEPs

First a KEP will be modelled using code by Saidman, Susan L. and Roth, Alvin E. and etc [8]. This code needs the knowledge of how many pairs, NDDs, and countries will be participating in this KEP. Then it assignes bloodtypes to the donor and recipients and a panel-reactive antibody (pra). The pra has an influence on the possibility percentage that two people of the same bloodtype will be found compatible. Next, all possible cycles and chains will be found and the solver CPLEX 20.1, provided by the modelling language AIMMS, will be consulted to find a solution based on the following objectives:

- Maximizing the number of planned donations,
- Maximizing the number of cycles,
- First maximizing the number of planned donations and then maximizing the number of cycles,
- First maximizing the number of cycles and then maximizing the number of planned donations, and
- Maximizing the number of expected successful donations.

The first and the second objective were chosen to see how much the number of successful donations would improve when the hierarchical objective functions would be used. The third and fourth objective is to compare which hierarchy would result in the highest number of successful donations. The last objective, to maximize the expected number of successful donations, is the true objective. The other objectives are evaluated by comparing them with this objective.

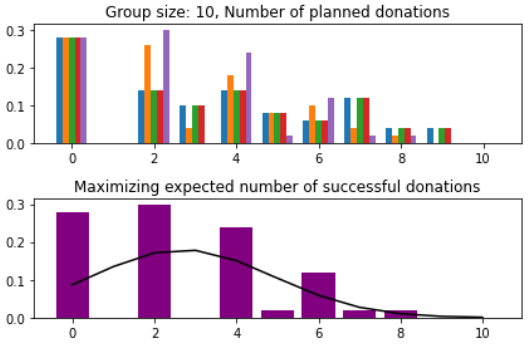
The Figures 8, 9, and 10 show the number of planned donations, number of cycles and number of successful donations (rounded upwards) on the x-axis, and the percentage of the sample size on the y-axis respectively. The first component of 8a, 9a, and 10a shows the contrast between the objectives for which the maximum solution was found:

purple:	Maximizing the number of expected successful donations
blue:	Maximizing the number of planned donations
yellow:	Maximizing the number of cycles
green:	The hierarchy that first maximizes the number of planned donations and then maximizes the number of cycles
red:	The hierarchy that first maximizes the number of cycles and then maximizes the number of planned donations

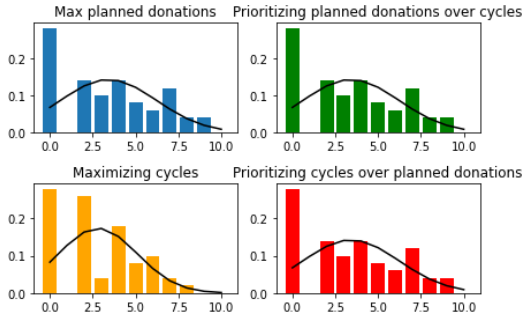
Table 2: The color scheme used for the Figures 8, 9, 10, and all the figures in Section 8.1.

We have modelled a KEP containing 10, 20, 30, 40 and 50 pairs all 50 times and created normalized bar charts to compare the resulting number of planned donations, cycles and the number of expected donations with respect to the objectives.

The black curves in the graphs that show the results individually represent the normal distribution whose mean and variance was calculated using the results found of 50 simulations. The other tables are found in Section 8.1. The means, and the 95% intervals, of these objectives combined with the number of pairs and the number of planned, cycles, and successful donations achieved by the objectives are shown in Table 4.

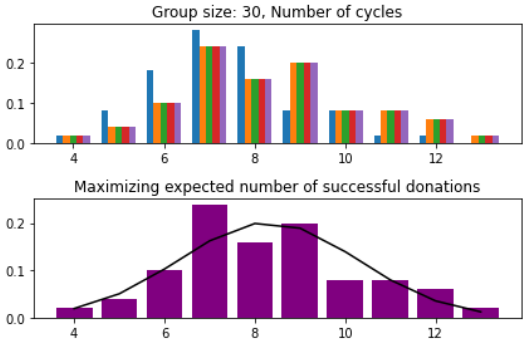


(a) The first figure shows the results of all the objectives in the same bar chart. The second figure has isolated the objective to maximize the number of expected successful donations.

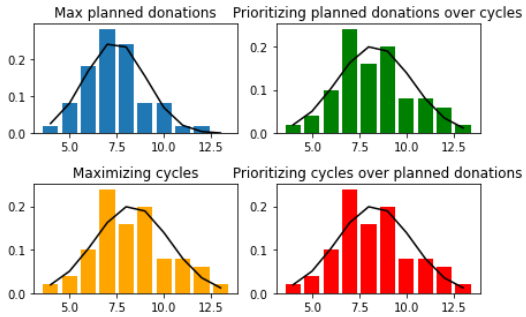


(b) Every component shows the results of the objectives individually, as well as in the hierarchy, against the normal distribution with the sample mean and sample variance.

Figure 8: The result of simulating a KEP containing 10 pairs with a sample size of 50 when maximizing the number of planned donations in the solution.

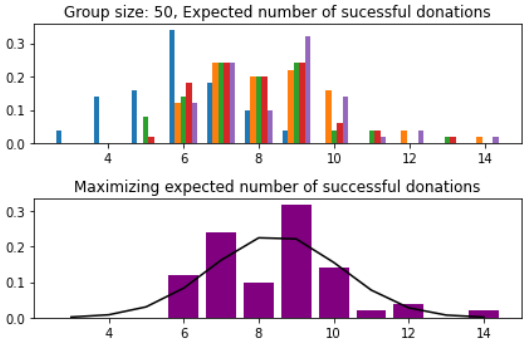


(a) The first component compares the objectives whereas the second component has isolated the objective to maximize the number of expected successful donations.

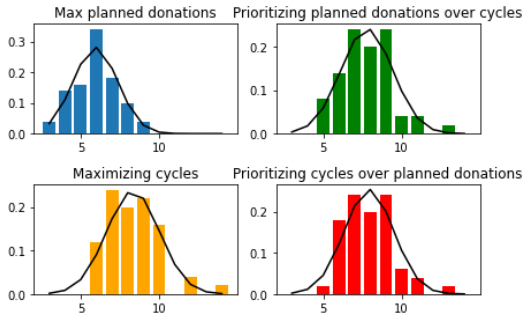


(b) Every component shows the results of the objectives individually, as well as in the hierarchy, against the normal distribution with the sample mean and sample variance.

Figure 9: The result of simulating a KEP containing 30 pairs with a sample size of 50 when maximizing the number of cycles in the solution.



(a) The first component compares the objectives whereas the second component has isolated the objective to maximize the number of expected successful donations.



(b) Every component shows the results of the objectives individually, as well as in the hierarchy, against the normal distribution with the sample mean and sample variance.

Figure 10: The result of simulating a KEP containing 50 pairs 50 times when maximizing the number of expected successful donations in the solution.

Table 3 shows the relation between the the size of a cycle, and the expected number of successful transplants it would contribute. This table shows that when the probability of a successful donation is 56%, then the number of successful donations is higher for smaller cycles.

Cycle size	Probability of success of the cycle	Expected successful donations of the cycle
2	$0.56^2 = 0.3136$	$2 \cdot 0.3136 = 0.6272$
3	$0.56^3 = 0.1756$	$3 \cdot 0.1756 = 0.5268$
4	$0.56^4 = 0.0983$	$4 \cdot 0.0983 = 0.3934$

Table 3: This shows for larger cycles, the expected number of successful donations is smaller, even though the number of planned donations increases.

Now to discuss Table 4. First, note that the average of the number of cycles is the same for the hierarchical objective of maximizing the number of cycles, then maximizing the number of planned donations, and the objectives of maximizing expected successful transplants. This also holds for their 95% confidence intervals. The performance of the objectives with the corresponding KEP sizes is shown in Table 5. This shows that maximizing the number of cycles gives positive results with respect to maximizing the expected number of successful donations is correct.
The same however can't be said for the objective of maximizing the number of planned donations in the solution. Comparing the hierarchies to only focus on maximizing the number of cycles, it is visible that the average number of successful donations has decreased. This can be explained by the fact that, as has already been mentioned, the expected number of successful donations is higher for a smaller cycles, when the probability of a successful donation is 56%. So, even though in a complete graph, maximizing the number of planned donations also maximizes the expected number of successful donations, the same does not hold for graphs representing realistic KEPs. Hence, maximizing the number of planned donations does not yields the best possible number of donations with respect to realistic KEPs. Table 4 shows that the objective that achieves the best expected number of successful donations has the smallest average of planned donations compared to the other objectives.

50 pairs	**Planned donations**	**Cycles**	**Successful donations**
Maximizing planned donations	31.2 ± 1.65	10.54 ± 0.63	5.48 ± 0.39
Maximizing cycles	26.7 ± 1.54	12.74 ± 0.73	7.87 ± 0.45
Max. planned donations then cycles	31.2 ± 1.65	12.62 ± 0.76	7.26 ± 0.48
Max. cycles then planned donations	31.08 ± 1.67	12.74 ± 0.73	7.39 ± 0.45
Maximizing successful donations	25.9 ± 1.49	12.74 ± 0.73	7.95 ± 0.45
40 pairs			
Maximizing planned donations	24.74 ± 1.19	8.32 ± 0.54	4.24 ± 0.75
Maximizing cycles	20.34 ± 1.26	9.64 ± 0.60	5.79 ± 0.90
Max. planned donations then cycles	24.74 ± 1.19	9.52 ± 0.58	5.19 ± 0.89
Max. cycles then planned donations	24.6 ± 1.16	9.64 ± 0.60	5.29 ± 0.94
Maximizing successful donations	19.62 ± 1.24	9.64 ± 0.60	5.85 ± 0.89
30 pairs			
Maximizing planned donations	16.28 ± 1.23	5.42 ± 0.45	2.78 ± 0.27
Maximizing cycles	13.16 ± 1.18	6.3 ± 0.56	3.89 ± 0.34
Max. planned donations then cycles	16.28 ± 1.23	6.3 ± 0.56	3.54 ± 0.35
Max. cycles then planned donations	16.28 ± 1.23	6.3 ± 0.56	3.55 ± 0.35
Maximizing successful donations	12.84 ± 1.19	6.3 ± 0.56	3.92 ± 0.34
20 pairs			
Maximizing planned donations	10.2 ± 1.00	3.5 ± 0.38	1.84 ± 0.22
Maximizing cycles	8.22 ± 0.93	3.94 ± 0.44	2.44 ± 0.27
Max. planned donations then cycles	10.2 ± 1.00	3.9 ± 0.42	2.18 ± 0.26
Max. cycles then planned donations	10.16 ± 0.98	3.94 ± 0.44	2.22 ± 0.28
Maximizing successful donations	8 ± 0.90	3.94 ± 0.44	2.46 ± 0.27
10 pairs			
Maximizing planned donations	3.42 ± 0.78	1.22 ± 0.27	0.66 ± 0.15
Maximizing cycles	2.8 ± 0.64	1.3 ± 0.30	0.78 ± 0.18
Max. planned donations then cycles	3.42 ± 0.78	1.3 ± 0.30	0.73 ± 0.17
Max. cycles then planned donations	3.42 ± 0.78	1.3 ± 0.30	0.73 ± 0.17
Maximizing successful donations	2.68 ± 0.62	1.3 ± 0.30	0.79 ± 0.18

Table 4: Results after simulating the KEP containing x pairs 50 times. It shows the means found combined with the 95% interval.

nr. participating pairs:	50	40	30	20	10
Maximizing planned donations	69%	72%	71%	75%	84%
Maximizing cycles	99%	99%	99%	99%	99%
Max. planned donations then cycles	91%	89%	90%	89%	92%
Max. cycles then planned donations	93%	90%	91%	90%	92%

Table 5: The performance with the percentage of the maximum possible expected successful donations the objectives contribute.

5.3 Discussion of the results

In this section we will see how big the percentage of success has to be to make a n-way cycle preferable over an $(n-1)$-way cycle. It also calculates the increase in the expected number of successful donations when subcycles are taken into account.

5.3.1 Success rate needed such that maximizing the number of planned donations improves the solution.

Let $x > 0$ denote the percentage of success of a donation. Then a cycle of length n will have at most the same expected number of successful donations as a cycle of length $n + 1$ if:

$$\begin{aligned}(n+1)x^{n+1} &\geq nx^n \\ \implies 0 &\geq nx^n - (n+1)x^{n+1} \\ \implies 0 &\geq x^n(n-(n+1)x) \\ \implies 0 &\geq n-(n+1)x \\ \implies x &\geq \frac{n}{n+1}.\end{aligned}$$

Hence we find that a cycle of length $n + 1$ would result in a higher expected number of successful donations than a cycle of length n if the success rate of a donation is at least $\frac{n}{n+1}\%$. Since we consider a success rate of 56%, we find that including 2-way cycles shows better results with respect to maximizing the number of successful donations, than including 3-way (or 4-way) cycles.

5.3.2 Prioritizing cycles with subcycles

In the UK they also consider prioritizing 3-way cycles with 2-way subcycles. We are going to calculate the advantage of being able to switch to a subcyle in case the original cycle fails. An example is shown using Graph $D_5 = (V, A)$ in Figure 11. In this case, should pair 3 not be able to participate in the donations, then the 2-way subcycle can be utilized and there can still be two instead of three successful donations.

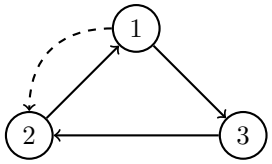


Figure 11: A directed graph D_5 representing the cycle (1, 3, 2) with sub-cycle (1, 2).

Let the probability of a successful donations be p. Let c_1 be a cycle of length n and c_2 be a subcycle on c_1 of size $n - 1$ vertices. We now find the following number of expected successful donations of cycle c_1 when we take into account that we can use subcycle c_2 in case the cycle c_1 fails:

$$E_p(c) = n \cdot p^n + (n-1)p^{n-1}(1-p^2)$$

with,

n : The number of planned donations the cycle c_1 contributes.
p^n : The probability that cycle c_1 is successful.
$(n-1)$: The number of planned donations the cycle c_2 contributes.
p^{n-1} : The probability that cycle c_2 is successful.
$1-p^2$: The probability that at least one arc $a_1, a_2 \in c_1 \setminus c_2$ fails.

Applying this formula to the cycle shown in Figure 11, we find the following for a three-way cycle with a two-way subcycle when the probability of success is 56%:

$$\begin{aligned} E_p(c) &= 3 \cdot 0.56^3 + (3-1)0.56^{3-1}(1-56^2) \\ &= 0.9574 \end{aligned}$$

So, when also taking into account the sub-cycles of length $n-1$ of cycles of length n, the expected number of successful donations of the cycle of length n increases by $(n-1)p^{n-1}(1-p^2)$.

6 Conclusion

In complete graphs, we have looked at all possible ways to cover sets of arcs with cycles and found that on average the expected number of successful donations increases when the number of planned donations or the number of donation cycles increases.

Moreover, we have studied the performance of two objectives with respect to the maximum expected number of successful donations in graphs representing realistic KEP instances. Maximizing the number of planned donations only results in 69% - 84% of the maximum expected number of successful donations. The objective maximizing the number of cycles however, results in 99% of the maximum expected number of successful donations. When considering these two objectives in a hierarchy, we find that, instead of an improvement with respect to the expected number of successful donations, it only results in 89% - 92% of the maximum expected number of successful donations when we have priority on maximizing the number of planned donations, and 90% - 93% when we have priority on maximizing the number of cycles.

7 Discussion

For future work, it can be interesting to look at the hierarchy with prioritizing maximizing the number of cycles and then minimizing the number of planned donations instead of maximizing the number of planned donations, when the assumption that the probability of a planned donation will be performed is below 66%. I believe this new hierarchy will perform very well with respect to the expected number of successful donations.

Note that there are other sources in which another success rate of a planned donation has been chosen since the exact success rate is not known. From Section 5.3.1, we know that if the success rate is greater, bigger cycles will contribute more successful donations. Hence, for higher success rates, the objective of maximizing the number of planned donations will yield better results.

It would also be interesting to consider selecting cycles which have a sub-cycle, as has been

discussed in Section 5.3.2. It would be interesting how this would effect the found results, and what impact it has when maximizing or minimizing the number of planned donations in the solutions found.

References

[1] "Organen: cijfers afgelopen maanden — Nederlandse Transplantatie Stichting." [Online]. Available: https://www.transplantatiestichting.nl/publicaties-en-naslag/cijfers-over-donatie-en-transplantatie/organen-cijfers-afgelopen-maanden

[2] M. Hatzinger, M. Stastny, P. Grützmacher, and M. Sohn, "Die Geschichte der Nierentransplantation [or The history of kidney transplantation]," *Der Urologe. Ausg. A*, vol. 55, no. 10, pp. 1353–1359, 10 2016. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/27518791/

[3] P. Biró, J. van de Klundert, D. Manlove, W. Pettersson, T. Andersson, L. Burnapp, P. Chromy, P. Delgado, P. Dworczak, B. Haase, A. Hemke, R. Johnson, X. Klimentova, D. Kuypers, A. Nanni Costa, B. Smeulders, F. Spieksma, M. O. Valentín, and A. Viana, "Modelling and optimisation in European Kidney Exchange Programmes," *European Journal of Operational Research*, vol. 291, no. 2, pp. 447–456, 6 2021.

[4] J. P. Dickerson, A. D. Procaccia, and T. Sandholm, "Failure-Aware Kidney Exchange," *MANAGEMENT SCIENCE*, vol. 65, no. 4, pp. 1768–1791, 2019. [Online]. Available: http://pubsonline.informs.org/journal/mnsc/http://orcid.org/0000-0003-2231-680X

[5] A. Schrijver, "17.2 - The Hungarian method," in *Combinatoral Optimization - Polyhedra and Efficiency*, 1st ed. Springer-Verlag Berlin Heidelberg, 2003, ch. 17.2., pp. 286–288.

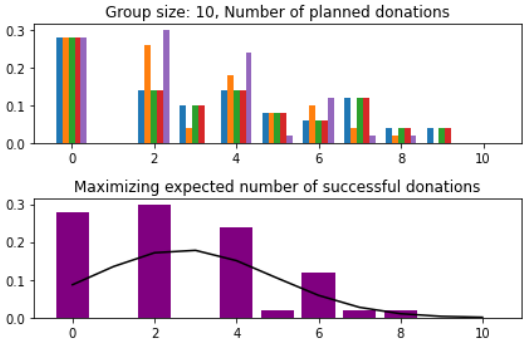
[6] M. Constantino, X. Klimentova, A. Viana, and A. Rais, "New insights on integer-programming models for the kidney exchange problem," *European Journal of Operational Research*, vol. 231, no. 1, pp. 57–68, 11 2013.

[7] J. P. Dickerson, D. F. Manlove, J. Trimble, B. Plaut, T. Sandholm, J. P. Dickerson, B. Plaut, T. Sandholm, . D. F. Manlove, and J. Trimble, "Position-Indexed Formulations for Kidney Exchange," pp. 25–42, 7 2016. [Online]. Available: http://doi.acm.org/10.1145/2940716.2940759

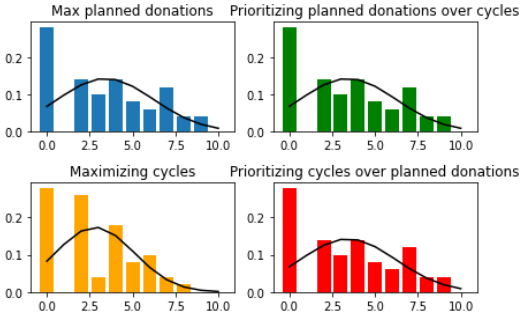
[8] S. L. Saidman, A. E. Roth, T. Sönmez, M. U. Ünver, and F. L. Delmonico, "Increasing the Opportunity of Live Kidney Donation by Matching for Two- and Three-Way Exchanges," *Transplantation*, vol. 81, no. 5, pp. 773–782, 3 2006.

8 Appendix

8.1 Bar charts showing the results

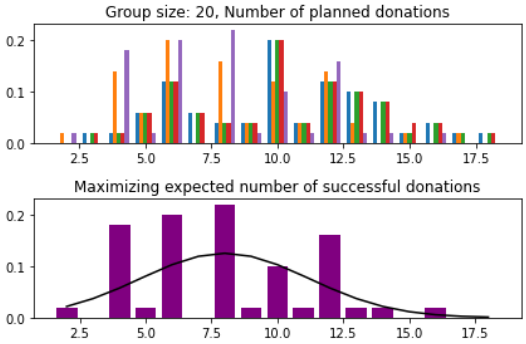


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

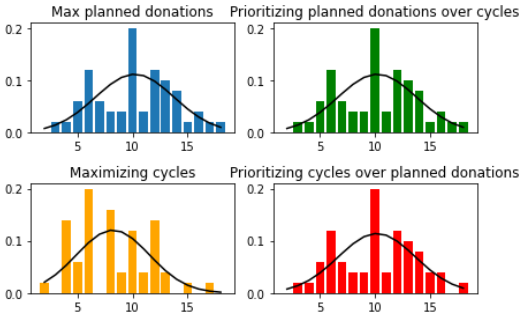


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 12: The result of simulating a KEP containing 10 pairs with a sample size of 50 when maximizing the number of planned donations in the solution.

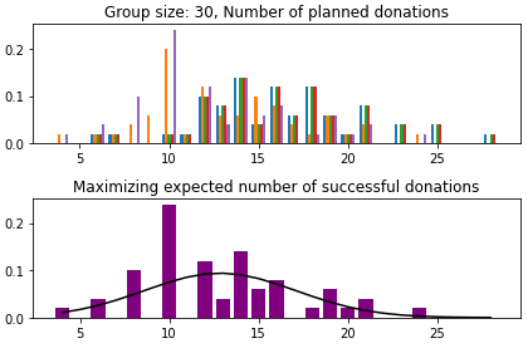


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

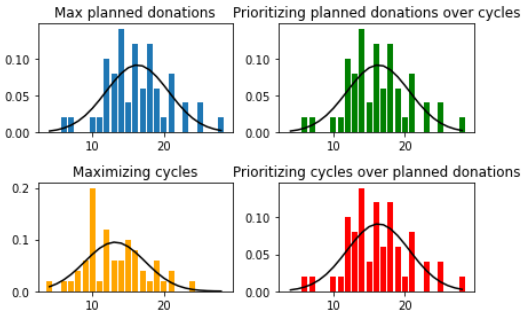


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 13: The result of simulating a KEP containing 20 pairs with a sample size of 50 when maximizing the number of planned donations in the solution.

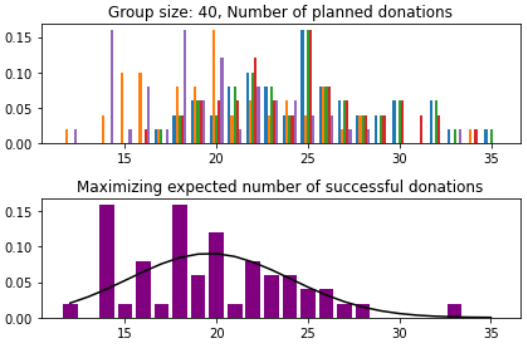


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

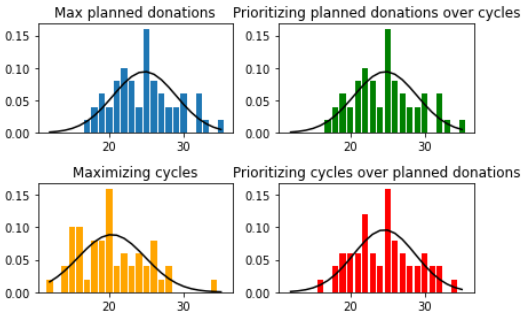


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 14: The result of simulating a KEP containing 30 pairs with a sample size of 50 when maximizing the number of planned donations in the solution.

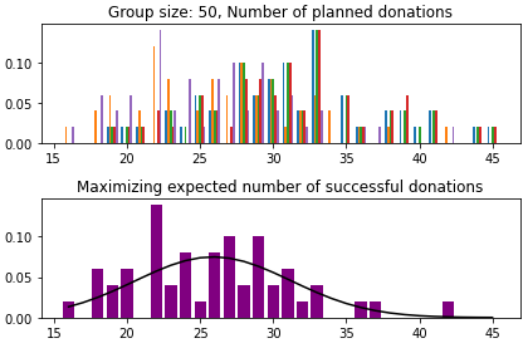


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

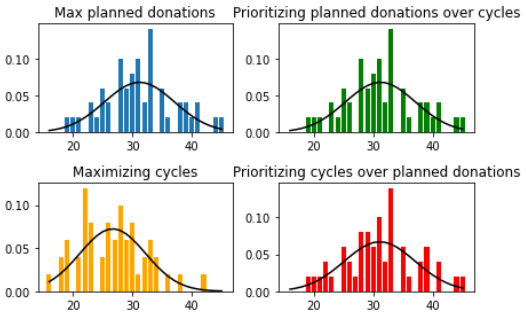


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 15: The result of simulating a KEP containing 40 pairs with a sample size of 50 when maximizing the number of planned donations in the solution.

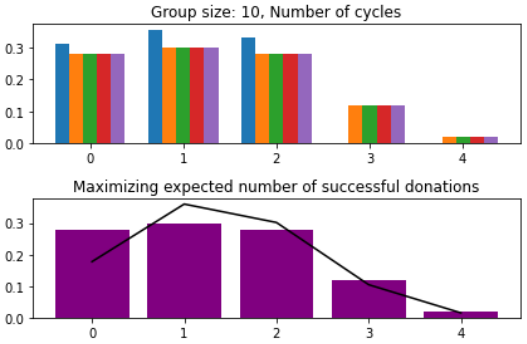


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

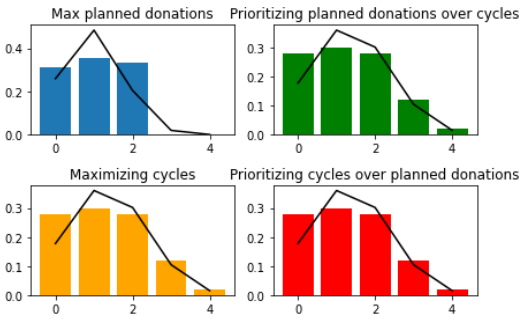


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 16: The result of simulating a KEP containing 50 pairs with a sample size of 50 when maximizing the number of planned donations in the solution.

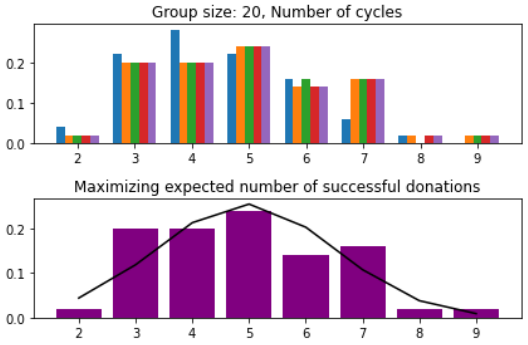


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

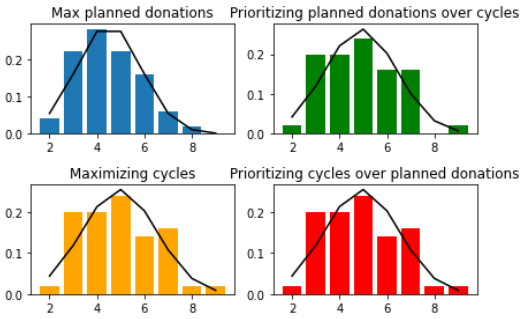


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 17: The result of simulating a KEP containing 10 pairs with a sample size of 50 when maximizing the number of cycles in the solution.

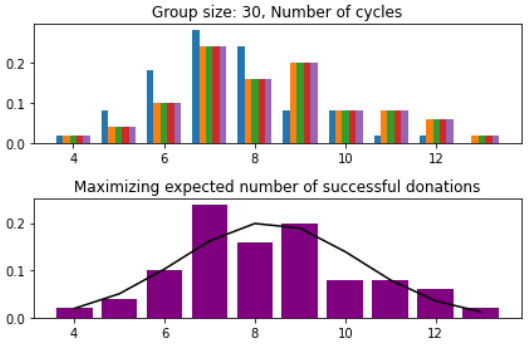


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

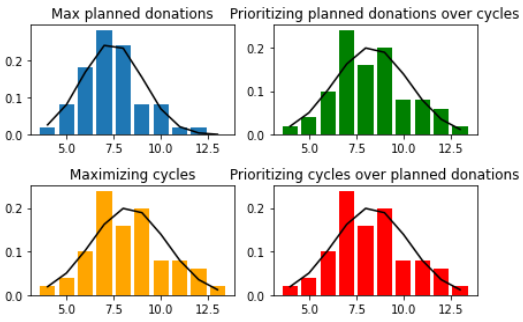


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 18: The result of simulating a KEP containing 20 pairs with a sample size of 50 when maximizing the number of cycles in the solution.

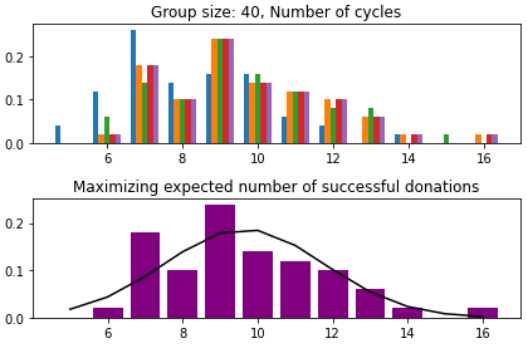


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

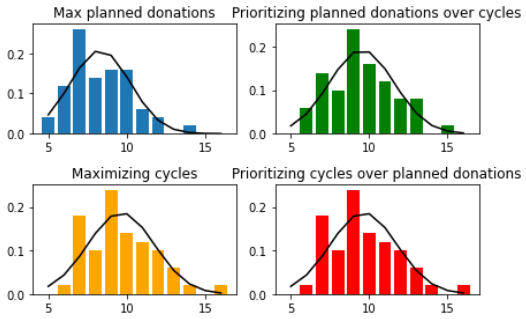


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 19: The result of simulating a KEP containing 30 pairs with a sample size of 50 when maximizing the number of cycles in the solution.

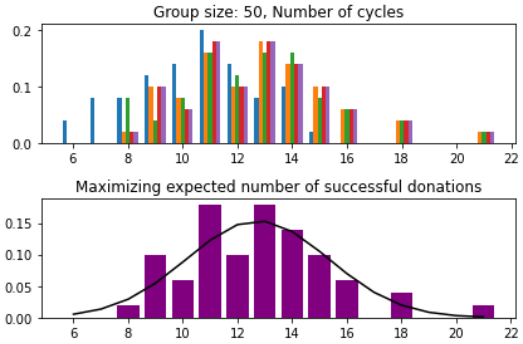


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

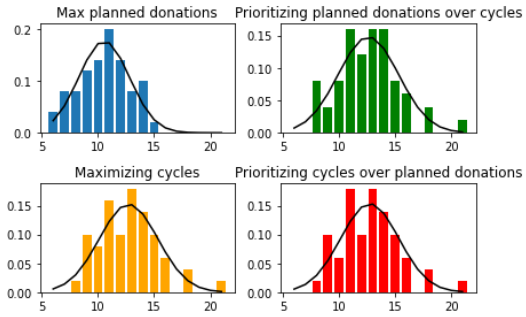


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 20: The result of simulating a KEP containing 40 pairs with a sample size of 50 when maximizing the number of cycles in the solution.

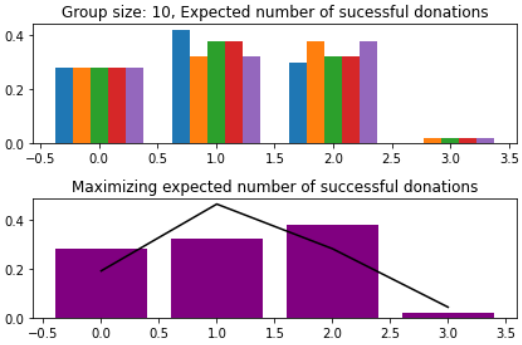


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

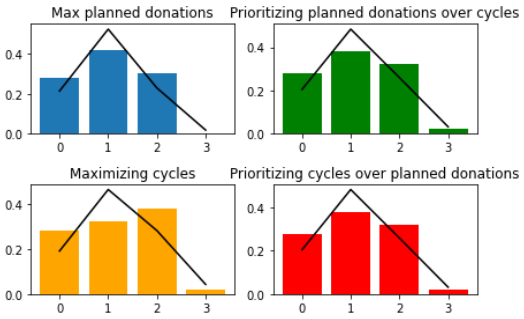


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 21: The result of simulating a KEP containing 50 pairs with a sample size of 50 when maximizing the number of cycles in the solution.

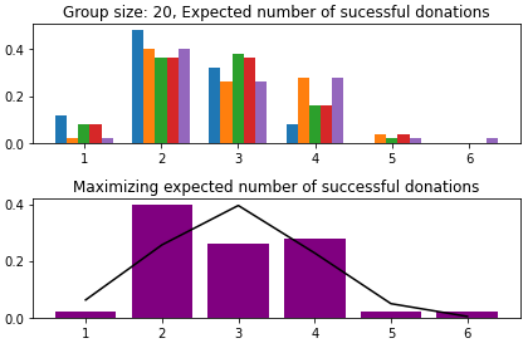


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

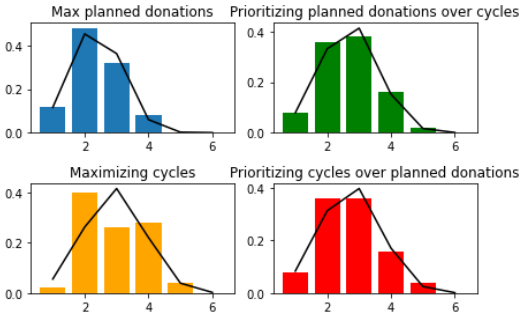


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 22: The result of simulating a KEP containing 10 pairs with a sample size of 50 when maximizing the number of expected successful donations in the solution.

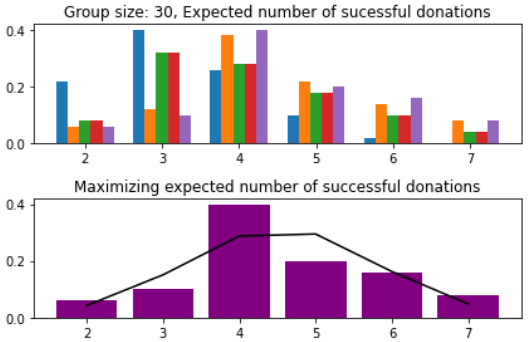


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

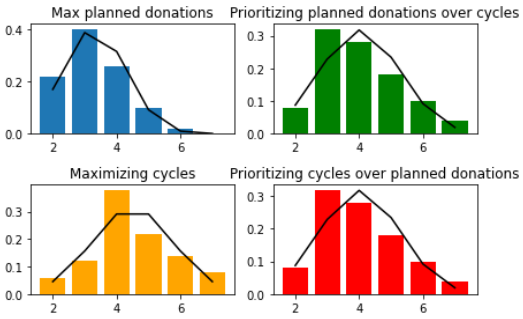


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 23: The result of simulating a KEP containing 20 pairs with a sample size of 50 when maximizing the number of expected successful donations in the solution.

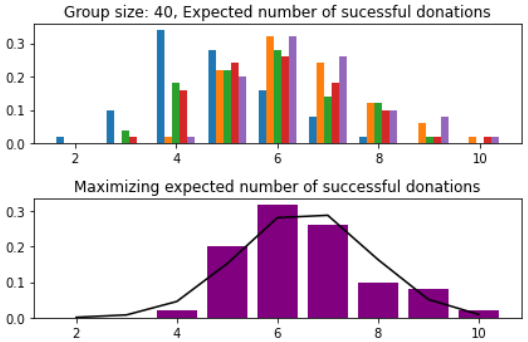


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

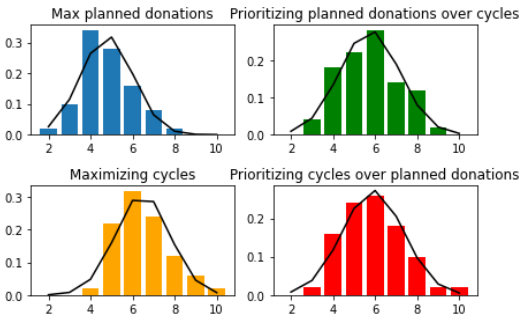


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 24: The result of simulating a KEP containing 30 pairs with a sample size of 50 when maximizing the number of expected successful donations in the solution.

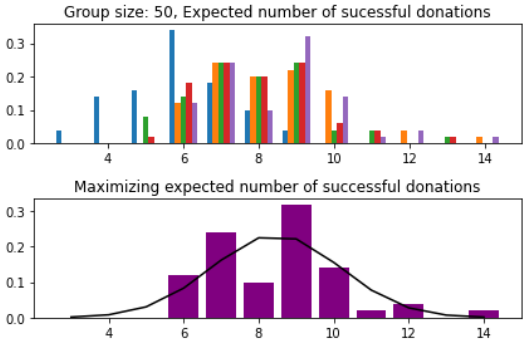


(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.

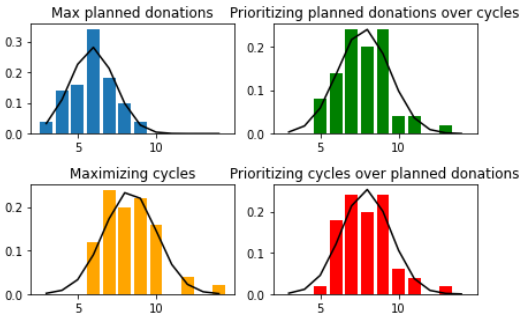


(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 25: The result of simulating a KEP containing 40 pairs with a sample size of 50 when maximizing the number of expected successful donations in the solution.



(a) The first component compares the objectives where as the second component has isolated the objective to maximize the number of expected successful donations.



(b) very component shows an objective individually, with the color scheme of Table 2, against the normal distribution with the sample mean and sample variance.

Figure 26: The result of simulating a KEP containing 50 pairs with a sample size of 50 when maximizing the number of expected successful donations in the solution.

8.2 Python code

```
import time
import pandas as pd
import csv
from numpy import random

from scipy.stats import binom


def CreatCyclesAndChains(doc, max_length_chain, max_length_cycle):
# To Simulate a KEP with a document that contains all information about the
# number of NDDs, Pairs, Arcs, and participating countries obtained by running
# the code from source [8]
    with open(doc) as f: # modifies the uploaded data such that the program
                         # can use it, Nr_Pairs, Nr_NDD and Nr_Arcs will hold
                         # the number of pairs, NDDs and arcs in integer form
                         # respectively,

        Nr_Pairs = f.readline()[:]
        Nr_Pairs = int(Nr_Pairs[11:len(Nr_Pairs)])

        Nr_NDD = f.readline()[:]
        Nr_NDD = int(Nr_NDD[9: len(Nr_NDD)])

        Nr_Arcs = f.readline()[:]
        Nr_Arcs = int(Nr_Arcs[10: len(Nr_Arcs)])

        Arcs = f.readlines()[1+Nr_Pairs+Nr_NDD:1+Nr_Pairs+Nr_NDD+ Nr_Arcs]
```

```
Info = []
with open(doc) as f:
    for line in f:
        arc = line.split()
        Info.append(arc)
del(Info[0: 4 + Nr_Pairs + Nr_NDD])

for info in range(len(Info)):
    Info[info] = Info[info][0].split(",")[0:2]

for info in range(len(Info)):
    x_coor = Info[info][0].split("(")
    y_coor = Info[info][1].split(")")
    Info[info][0] = x_coor[1]
    Info[info][1] = y_coor[0]

for arcs in range(Nr_Arcs):
    for info in range(2):
        Info[arcs][info] = int(Info[arcs][info])

##############################################################################
# Will put every pair whose donor is incompatible with every participating
# recipient in the list "endings"

endings = []
for end_1 in range(len(Info)-1):
    if Info[end_1 + 1][0] - Info[end_1][0] > 1:
        for end_2 in range(Info[end_1 + 1][0] - Info[end_1][0] - 1):
            endings.append(Info[end_1][0] + 1 + end_2)

##############################################################################
# The list "all_in" will end up containing all found valid chains and cycles
# Here all the valid chains will be calculated
all_in = []

if max_length_chain > 0:
    chains_cycles = []
    starts = []
    for step in range(max_length_chain + 1):
    # will go over all chains, see if there is a next step

        if step == 0:
            chains_cycles = []
            for NDD in range(Nr_NDD):
                chains_cycles.append([Nr_Pairs + NDD])
                starts.append(Nr_Pairs + NDD)

        else:
            update = chains_cycles.copy()
```

```
                # will not effect the original data
            chains_cycles = []
                # will become one step larger
            check = []
                # check if a chain representing the same group of pairs has
                # already been found, f.e. 1-2-3 vs 1-3-2

            for pair_chain in range(len(update)):

                for pair in range(len(Info)):
                    if update[pair_chain][-1] == Info[pair][0] and
                    Info[pair][1] not in update[pair_chain]:
                        # new link found

                        chain = update[pair_chain].copy()
                        chain.append(Info[pair][1])

                        if len(update[pair_chain]) == 1 or Info[pair][-1]
                        in endings:
                            all_in.append(chain)

                        else:
                            if len(check) == 0:
                                check.append(sorted(chain))
                                all_in.append(chain)

                            elif sorted(chain) not in check:
                                check.append(sorted(chain))
                                all_in.append(chain)

                        if Info[pair][1] not in endings:
                            chains_cycles.append(chain)

        if step == 1:
            # NDDs can only be placed at the start of a chain, hence they
            # can now be taken out
            for info in range(len(Info)):
                if Info[- 1][0] >= Nr_Pairs:
                    del(Info[-1])
                else:
                    break

##############################################################################
# Here all the valid chains will be found

if max_length_cycle - 1 > 0:
    chains_cycles = []
    info = 0
    while info in range(len(Info)):
```

```
            if Info[info][-1] not in endings:
                info += 1
            else:
                del(Info[info])

        for pair in range(Nr_Pairs - 1):
            # the last one will never go to itself, and every other ones are
            # less then so will already have been deleted
            if pair not in endings:
                if pair > 0:
                    info = 0
                    while info in range(len(Info)):
                        if Info[info][0] >= pair and Info[info][1] >= pair:
                            info += 1
                        else:
                            del(Info[info])

                if len(Info) == 0:
                    # all possible links have already been used
                    break

                else:
                    for step in range(max_length_cycle):
                        # amount of steps, plus going back to original
                        if step == 0:
                            # will contain all chains which might become cycles
                            check = 0
                            for info in range(len(Info)):
                                if Info[info][-1] == pair:
                                    check = 1
                                    break

                            if check == 0:
                                break

                            update = []
                            info = 0
                            while info in range(len(Info)):
                                if Info[info][0] == pair:
                                    update.append(Info[info].copy())
                                    del(Info[info])

                                elif Info[info][0] > pair:
                                    break

                            if len(update) == 0:
                                break

                        else:
```

```
                    chains_cycles = update.copy()
                    update = []
                    check = []


                    for cyc in range(len(chains_cycles)):
                        for info in range(len(Info)):
                            if chains_cycles[cyc][-1] == Info[info][0]
                            and chains_cycles[cyc][0] == Info[info][1]:
                                cycle = chains_cycles[cyc].copy()

                                if len(cycle) == 2:
                                    all_in.append(cycle)

                                elif sorted(cycle) not in check:
                                    all_in.append(cycle)
                                    check.append(sorted(cycle))

                            elif chains_cycles[cyc][-1] == Info[info][0]
                            and Info[info][-1] not in chains_cycles[cyc]:
                                                # check for in loops
                                cycle = chains_cycles[cyc].copy()
                                cycle.append(Info[info][1])
                                update.append(cycle)

    ##########################################################################
    # Calculating the expected number of successful donations of the chains and cyc
    # linking the chains and cycles with the pairs and NDDs they contain and
    # putting everything in a table in a excel file.

    pairs = [['cycles']]
    for pair in range(Nr_Pairs + Nr_NDD):
        pairs[0].append(pair)

    pairs[0].append("Expected people helped")

#   Links the pairs to the cycles and chains they are in
    for cyc in range(len(all_in)):
        data = []
        data.append(all_in[cyc])
        for pair in range(Nr_Pairs + Nr_NDD):
            if pairs[0][pair + 1] in all_in[cyc]:
                data.append(1)
            else:
                data.append(0)

#   If we are in a chain, then you calculate the expected people helped as
#   follows:
        # chain
```

```
        if all_in[cyc][0] >= Nr_Pairs:
            exp = 0
            for step in range(len(data[0]) - 1):
                exp += (step + 1) *  0.56**(step + 1)
            data.append(exp)

        else:
            data.append(len(all_in[cyc]) * 0.56**len(all_in[cyc]))

        pairs.append(data)

    df = pd.DataFrame(pairs)
    return(df)

doc = 'Graph_40_0_4equal_0.txt'
df = CreatCyclesAndChains(doc, 0, 4)
df.to_excel('Graph_40_0_4equal_0_44.xlsx', index = False)

##############################################################################
##############################################################################
# The definitions to create the graphs to show the results
# We use two graphs with 2 and 4 components respectively.
# The first component of the first graph contains all the information
# and a second component depicting the information for the expected successful
# donations.
# The components of the second graph isolate each objective:
# maximizing the planned donations, the cycles, and the two possible
# hierarchies.
# "subplots" refers to the number components in the graph,
# "number" refers to the number of pairs in the simulation, and
# "objective" is 1 (for planned donations), 2 (for cycles) or
# 3 (for the expected number of successful donations).

# To create the figure containing the bar chart representing all objectives,
# with the bar chart isolating the objective of the expected number of
# successful donations

def Graph(subplots, number, objective):
    if subplots == 2:
        fig, ax = plt.subplots(2)
    elif subplots == 4:
        fig, ax = plt.subplots(2, 2)
    else:
        print("Invalid number of subplots, give 2 or 4")

    if objective == 1:

        if number == 10:
            if subplots == 2:
```

```
            ax[0].set_title('Group size: 10, Number of planned donations')

        a_1 = [14, 0,  7, 5,  7, 4, 3, 6, 2, 2, 0]
        a_2 = [14, 0, 13, 2,  9, 4, 5, 2, 1, 0, 0]
        a_3 = [14, 0,  7, 5,  7, 4, 3, 6, 2, 2, 0]
        a_4 = [14, 0,  7, 5,  7, 4, 3, 6, 2, 2, 0]
        a_5 = [14, 0, 15, 0, 12, 1, 6, 1, 1, 0, 0]

        mn = 0

    elif number == 20:
        if subplots == 2:
            ax[0].set_title('Group size: 20, Number of planned donations')

        a_1 = [0, 1, 1, 3,  6, 3,  2, 2, 10, 2,
               6, 5, 4, 1,  2, 1,  1]
        a_2 = [1, 0, 7, 3, 10, 0,  8, 2,  6, 2,
               7, 2, 0, 1,  0, 1,  0]
        a_3 = [0, 1, 1, 3,  6, 3,  2, 2, 10, 2,
               6, 5, 4, 1,  2, 1,  1]
        a_4 = [0, 1, 1, 3,  6, 3,  2, 2, 10, 2,
               6, 5, 4, 2,  2, 0,  1]
        a_5 = [1, 0, 9, 1, 10, 0, 11, 1,  5, 1,
               8, 1, 1, 0,  1, 0,  0]

        mn = 2

    elif number == 30:
        if subplots == 2:
            ax[0].set_title('Group size: 30, Number of planned donations')

        a_1 = [0, 0, 1, 1, 0, 0,  1, 1, 5, 4,
               7, 2, 6, 3, 6, 3,  1, 4, 0, 2,
               0, 2, 0, 0, 1]
        a_2 = [1, 0, 1, 1, 2, 3, 10, 1, 6, 3,
               3, 5, 4, 2, 1, 3,  1, 2, 0, 0,
               1, 0, 0, 0, 0]
        a_3 = [0, 0, 1, 1, 0, 0,  1, 1, 5, 4,
               7, 2, 6, 3, 6, 3,  1, 4, 0, 2,
               0, 2, 0, 0, 1]
        a_4 = [0, 0, 1, 1, 0, 0,  1, 1, 5, 4,
               7, 2, 6, 3, 6, 3,  1, 4, 0, 2,
               0, 2, 0, 0, 1]
        a_5 = [1, 0, 2, 0, 5, 0, 12, 0, 6, 2,
               7, 3, 4, 0, 1, 3,  1, 2, 0, 0,
               1, 0, 0, 0, 0]

        mn = 4
```

```
elif number == 40:
    if subplots == 2:
        ax[0].set_title('Group size: 40, Number of planned donations')

    a_1 = [0, 0, 0, 0, 0, 1, 2, 3, 2, 4,
           5, 4, 2, 8, 4, 3, 2, 2, 3, 0,
           3, 1, 0, 1]
    a_2 = [1, 0, 2, 5, 5, 1, 4, 4, 8, 2,
           3, 2, 3, 2, 4, 1, 2, 0, 0, 0,
           0, 0, 1, 0]
    a_3 = [0, 0, 0, 0, 0, 1, 2, 3, 2, 4,
           5, 4, 2, 8, 4, 3, 2, 2, 3, 0,
           3, 1, 0, 1]
    a_4 = [0, 0, 0, 0, 1, 0, 2, 3, 3, 3,
           6, 3, 2, 8, 4, 3, 2, 2, 3, 2,
           2, 0, 1, 0]
    a_5 = [1, 0, 8, 1, 4, 1, 8, 3, 6, 1,
           4, 3, 3, 2, 2, 1, 1, 0, 0, 0,
           0, 1, 0, 0]

    mn = 12

elif number == 50:
    if subplots == 2:
        ax[0].set_title('Group size: 50, Number of planned donations')

    a_1 = [0, 0, 0, 1,
           1, 1, 0, 2, 1, 3, 2, 0, 5, 3,
           4, 5, 2, 7, 0, 3, 1, 0, 2, 2,
           1, 2, 0, 0, 1, 1]
    a_2 = [1, 0, 2, 3,
           0, 2, 6, 4, 0, 2, 4, 3, 5, 3,
           4, 1, 2, 3, 2, 0, 1, 0, 1, 0,
           0, 0, 1, 0, 0, 0]
    a_3 = [0, 0, 0, 1,
           1, 1, 0, 2, 1, 3, 2, 0, 5, 3,
           4, 5, 2, 7, 0, 3, 1, 0, 2, 2,
           1, 2, 0, 0, 1, 1]
    a_4 = [0, 0, 0, 1,
           1, 1, 2, 1, 0, 3, 2, 1, 4, 4,
           3, 5, 2, 7, 0, 3, 1, 0, 2, 3,
           0, 2, 0, 0, 1, 1]
    a_5 = [1, 0, 3, 2,
           3, 0, 7, 2, 4, 1, 4, 5, 2, 5,
           2, 3, 1, 2, 0, 0, 1, 1, 0, 0,
           0, 0, 1, 0, 0, 0]

    mn = 16
```

```
    else:
        print("An invalid number has been put in for the number of planned
        donations, choose between 10, 20, 30, 40, and 50")

elif objective == 2:
    if number == 10:
        if subplots == 2:
            ax[0].set_title('Group size: 10, Number of cycles')

        a_1 = [14, 16, 15, 0, 0]
        a_2 = [14, 15, 14, 6, 1]
        a_3 = [14, 15, 14, 6, 1]
        a_4 = [14, 15, 14, 6, 1]
        a_5 = [14, 15, 14, 6, 1]

        mn = 0

    elif number == 20:
        if subplots == 2:
            ax[0].set_title('Group size: 20, Number of cycles')

        a_1 = [2, 11, 14, 11, 8, 3, 1, 0]
        a_2 = [1, 10, 10, 12, 7, 8, 1, 1]
        a_3 = [1, 10, 10, 12, 8, 8, 0, 1]
        a_4 = [1, 10, 10, 12, 7, 8, 1, 1]
        a_5 = [1, 10, 10, 12, 7, 8, 1, 1]

        mn = 2

    elif number == 30:
        if subplots == 2:
            ax[0].set_title('Group size: 30, Number of cycles')

        a_1 = [1, 4, 9, 14, 12, 4, 4, 1, 1, 0]
        a_2 = [1, 2, 5, 12, 8, 10, 4, 4, 3, 1]
        a_3 = [1, 2, 5, 12, 8, 10, 4, 4, 3, 1]
        a_4 = [1, 2, 5, 12, 8, 10, 4, 4, 3, 1]
        a_5 = [1, 2, 5, 12, 8, 10, 4, 4, 3, 1]

        mn = 4

    elif number == 40:
        if subplots == 2:
            ax[0].set_title('Group size: 40, Number of cycles')

        a_1 = [2, 6, 13, 7,  8, 8,
               3, 2,  0, 1,  0, 0]
        a_2 = [0, 1,  9, 5, 12, 7,
               6, 5,  3, 1,  0, 1]
```

```
            a_3 = [0, 3,  7, 5, 12, 8,
                   6, 4,  4, 0,  1, 0]
            a_4 = [0, 1,  9, 5, 12, 7,
                   6, 5,  3, 1,  0, 1]
            a_5 = [0, 1,  9, 5, 12, 7,
                   6, 5,  3, 1,  0, 1]

            mn = 5

        elif number == 50:
            if subplots == 2:
                ax[0].set_title('Group size: 50, Number of cycles')

            a_1 = [ 2, 4, 4, 6, 7,
                   10, 7, 4, 5, 1,
                    0, 0, 0, 0, 0, 0]
            a_2 = [ 0, 0, 1, 5, 4, 8,
                    5, 9, 7, 5, 3, 0,
                    2, 0, 0, 1]
            a_3 = [ 0, 0, 4, 2, 4, 8,
                    6, 8, 8, 4, 3, 0,
                    2, 0, 0, 1]
            a_4 = [ 0, 0, 1, 5, 3, 9,
                    5, 9, 7, 5, 3, 0,
                    2, 0, 0, 1]
            a_5 = [ 0, 0, 1, 5, 3, 9,
                    5, 9, 7, 5, 3, 0,
                    2, 0, 0, 1]

            mn = 6

        else:
            print("An invalid number has been put in for the number of planned
            donations, choose between 10, 20, 30, 40, and 50")

    elif objective == 3:
        if number == 10:
            if subplots == 2:
                ax[0].set_title('Group size: 10, Expected number of successful
                donations')

            a_1 = [14, 21, 15, 0]
            a_2 = [14, 16, 19, 1]
            a_3 = [14, 19, 16, 1]
            a_4 = [14, 19, 16, 1]
            a_5 = [14, 16, 19, 1]

            mn = 0
```

```
elif number == 20:
    if subplots == 2:
        ax[0].set_title('Group size: 20, Expected number of successful
        donations')

    a_1 = [6, 24, 16,  4, 0, 0]
    a_2 = [1, 20, 13, 14, 2, 0]
    a_3 = [4, 18, 19,  8, 1, 0]
    a_4 = [4, 18, 18,  8, 2, 0]
    a_5 = [1, 20, 13, 14, 1, 1]

    mn = 1

elif number == 30:
    if subplots == 2:
        ax[0].set_title('Group size: 30, Expected number of successful
        donations')

    a_1 = [11, 20, 13,  5, 1, 0]
    a_2 = [ 3,  6, 19, 11, 7, 4]
    a_3 = [ 4, 16, 14,  9, 5, 2]
    a_4 = [ 4, 16, 14,  9, 5, 2]
    a_5 = [ 3,  5, 20, 10, 8, 4]

    mn = 2

elif number == 40:
    if subplots == 2:
        ax[0].set_title('Group size: 40, Expected number of successful
        donations')

    a_1 = [1, 5, 17, 14,  8,  4, 1, 0, 0]
    a_2 = [0, 0,  1, 11, 16, 12, 6, 3, 1]
    a_3 = [0, 2,  9, 11, 14,  7, 6, 1, 0]
    a_4 = [0, 1,  8, 12, 13,  9, 5, 1, 1]
    a_5 = [0, 0,  1, 10, 16, 13, 5, 4, 1]

    mn = 2

elif number == 50:
    if subplots == 2:
        ax[0].set_title('Group size: 50, Expected number of successful
        donations')

    a_1 = [2, 7, 8, 17,  9,  5,  2, 0, 0, 0,
           0, 0]
    a_2 = [0, 0, 0,  6, 12, 10, 11, 8, 0, 2,
           0, 1]
    a_3 = [0, 0, 4,  7, 12, 10, 12, 2, 2, 0,
```

```
                1, 0]
        a_4 = [0, 0, 1,  9, 12, 10, 12, 3, 2, 0,
                1, 0]
        a_5 = [0, 0, 0,  6, 12,  5, 16, 7, 1, 2,
                0, 1]

        mn = 3

    else:
        print("An invalid number has been put in for the number of planned
        donations, choose between 10, 20, 30, 40, and 50")

else:
    print("An invalid number has been put in for the objective")

x = np.arange(mn, len(a_1) + mn, 1, dtype = int)  # the label locations

a_1d = []
a_2d = []
a_3d = []
a_4d = []
a_5d = []

for i in range(len(a_1)):
    a_1d.append(a_1[i]/sum(a_1))
    a_2d.append(a_2[i]/sum(a_2))
    a_3d.append(a_3[i]/sum(a_3))
    a_4d.append(a_4[i]/sum(a_4))
    a_5d.append(a_5[i]/sum(a_5))

a_1t = []
a_2t = []
a_3t = []
a_4t = []

a2_1t = []
a2_2t = []
a2_3t = []
a2_4t = []

for i in range(len(a_1)):
    a_1t.append(a_1[i] * (i + mn))
    a_2t.append(a_2[i] * (i + mn))
    a_3t.append(a_3[i] * (i + mn))
    a_4t.append(a_4[i] * (i + mn))

    a2_1t.append(a_1[i] * ((i + mn)**2))
    a2_2t.append(a_2[i] * ((i + mn)**2))
    a2_3t.append(a_3[i] * ((i + mn)**2))
```

```
        a2_4t.append(a_4[i] * ((i + mn)**2))


    if subplots == 2:
        width = 0.15  # the width of the bars

        ax[1].set_title('Maximizing expected expected number of successful
        donations')
        ax[0].bar(x - width * 2, a_1d, width, label='Max planned donations')
        ax[0].bar(x - width, a_2d, width, label='Max cycles')
        ax[0].bar(x, a_3d, width, label='Prioritize planned donations')
        ax[0].bar(x + width, a_4d, width, label='Max expected successful
        donations')
        ax[0].bar(x + width * 2, a_5d, width, label='a_5d')

        a_exp = []
        a_t = []
        a_t2 = []

        for i in range(len(a_1)):
            a_exp.append(a_5[i]/sum(a_5))
            a_t.append(a_5[i] * (i + mn))
            a_t2.append(a_5[i] * ((i + mn)**2))

        ax[1].bar(x, a_exp, color = 'purple')

        mu = sum(a_t)/50

        sigma = sqrt(sum(a_t2)/50 - (sum(a_t)/50)**2)

        fitNormDist = stats.norm(mu, sigma)

        ys = fitNormDist.pdf(x)

        ax[1].plot(x, ys, color = 'black')

    elif subplots == 4:
        ax[0, 0].set_title('Max planned donations')
        ax[1, 0].set_title('Maximizing cycles')
        ax[0, 1].set_title('Prioritizing planned donations over cycles')
        ax[1, 1].set_title('Prioritizing cycles over planned donations')

        ax[0, 0].bar(x, a_1d, color = 'blue')
        ax[1, 0].bar(x, a_2d, color = 'orange')
        ax[0, 1].bar(x, a_3d, color = 'green')
        ax[1, 1].bar(x, a_4d, color = 'red')

        mu1 = sum(a_1t)/50
        mu2 = sum(a_2t)/50
```

```
            mu3 = sum(a_3t)/50
            mu4 = sum(a_4t)/50

            sigma1 = sqrt(sum(a2_1t)/50 - (sum(a_1t)/50)**2)
            sigma2 = sqrt(sum(a2_2t)/50 - (sum(a_2t)/50)**2)
            sigma3 = sqrt(sum(a2_3t)/50 - (sum(a_3t)/50)**2)
            sigma4 = sqrt(sum(a2_4t)/50 - (sum(a_4t)/50)**2)

            fitNormDist1 = stats.norm(mu1, sigma1)
            fitNormDist2 = stats.norm(mu2, sigma2)
            fitNormDist3 = stats.norm(mu3, sigma3)
            fitNormDist4 = stats.norm(mu4, sigma4)

            ys1 = fitNormDist1.pdf(x)
            ys2 = fitNormDist2.pdf(x)
            ys3 = fitNormDist3.pdf(x)
            ys4 = fitNormDist4.pdf(x)

            ax[0, 0].plot(x, ys1, color = 'black')
            ax[1, 0].plot(x, ys2, color = 'black')
            ax[0, 1].plot(x, ys3, color = 'black')
            ax[1, 1].plot(x, ys4, color = 'black')

        fig.tight_layout()
        plt.show()

    for a in range(3):
        b = a + 1
        for c in range(5):
            d = 10 * (c + 1)
            for e in range(2):
                f = 2 * (e + 1)
                plt.figure()
                Graph(f, d, b)

################################################################################
Creating the plots for the complete graphs showing all possible ways
to divide x planned donations in donation cycles and the corresponding
resulting expected successful donations
################################################################################

    planne = []
    cycles = []
    donati = []
    for x in range(51):
        for a in range(26):
            for b in range(17):
                for c in range(13):
                    if 2*a + 3*b + 4*c == x:
```

```
                    planne.append(int(x))
                    cycles.append(a + b + c)
                    donati.append(a*0.6272 + b*0.526848 + c*0.39338)

planne = np.array(planne)
cycles = np.array(cycles)
donati = np.array(donati)

plt.plot(planne, cycles, 'o')
plt.xlabel('Number of planned donations')
plt.ylabel('Number of donation cycles')

m, b = np.polyfit(planne, cycles, 1)
plt.plot(planne, m*planne+b)

plt.figure()
plt.plot(planne, donati, 'o')
plt.xlabel('Number of planned donations')
plt.ylabel('Number of expected successful donations')

m, b = np.polyfit(planne, donati, 1)
plt.plot(planne, m*planne+b)

plt.figure()
plt.plot(cycles, donati, 'o')
plt.xlabel('Number of donation cycles')
plt.ylabel('Number of expected successful donations')

m, b = np.polyfit(cycles, donati, 1)
plt.plot(cycles, m*cycles+b)
```

8.3 AIMMS code

```
Model Main_KEP_09 {
        Section Model_Section {
                DeclarationSection For_AXLL {
                        StringParameter spRead;
                        StringParameter spPeople;
                        StringParameter spCycles;
                        StringParameter spExp;
                        StringParameter spPlacement;
                        StringParameter spWorkbook;
                        StringParameter spSheet;
                }
                Variable Donation {
                        IndexDomain: (c,p);
                        Range: binary;
                }
```

```
Variable Used {
        IndexDomain: c;
        Range: free;
}
Variable nr_Cycles {
        Range: free;
        Definition: sum[c, Used(c)];
}
Variable nr_Helped {
        Range: free;
        Definition: sum[(c, p), Donation(c, p)];
}
Variable TotalExpected {
        Range: free;
        Definition: sum[c, Used(c) * ExpectedHelped(c)];
}
Parameter Helped {
        IndexDomain: p;
        Range: integer;
        Definition: sum[c, Donation(c, p)];
}
Parameter Connection {
        IndexDomain: (c,p);
}
Parameter ExpectedHelped {
        IndexDomain: c;
}
Constraint Demand {
        IndexDomain: (c,p);
        Definition: Donation(c, p) <= Connection(c, p);
}
Constraint Sypply {
        IndexDomain: p;
        Definition: sum[c, Donation(c, p)] <= 1;
}
Constraint Cycle {
        IndexDomain: (c,p);
        Definition: Connection(c, p) * (Donation(c, p) - Used(c))
        = 0;
}
MathematicalProgram nr_Helped_prog {
        Objective: nr_Helped;
        Direction: maximize;
        Constraints: AllConstraints;
        Variables: AllVariables;
        Type: MIP;
}
MathematicalProgram nr_Cycles_prog {
        Objective: nr_Cycles;
```

```
                Direction: maximize;
                Constraints: AllConstraints;
                Variables: AllVariables;
                Type: MIP;
        }
        MathematicalProgram TotExp_prog {
                Objective: TotalExpected;
                Direction: maximize;
                Constraints: AllConstraints;
                Variables: AllVariables;
                Type: MIP;
        }
        Set Chosen_Cycles {
                SubsetOf: Cycles;
                Index: c1;
                Definition: {
                        {c | Used(c) = 1}
                }
        }
        Set Cycles {
                Index: c;
        }
        Set Pairs {
                Index: p;
        }
}
Section MultiObjSection {
        Procedure SolveMultiObj12 {
                Body: {
                        KEP_GMP:=gmp::Instance::Generate(nr_Helped_prog);
                        # Comment: has to end with an ";"

                        retcode := GMP::Column::SetAsMultiObjective(
                                GMP      :  KEP_GMP,
                                column   :  nr_Helped,
                                priority :  1,
                                weight   :  1,
                                abstol   :  0,
                                reltol   :  0);
                        if not retcode then raise error
                        "Unable to set TotalCost as an objective" ;
                        endif ;

                        retcode := GMP::Column::SetAsMultiObjective(
                                GMP      :  KEP_GMP,
                                column   :  nr_Cycles,
                                priority :  2,
                                weight   :  1,
                                abstol   :  0,
```

```
                        reltol   :  0);
                    if not retcode then raise error
                    "Unable to set TotalCalories as an objective" ;
                    endif ;

                    GMP::Instance::Solve( KEP_GMP );

                    display "After SolveMultiObj", nr_Helped,
                    nr_Cycles, Donation;
            }
        }
        Procedure SolveMultiObj21 {
            Body: {
                    KEP_GMP:=gmp::Instance::Generate(nr_Helped_prog);
                    # Comment: has to end with an ";"

                    retcode := GMP::Column::SetAsMultiObjective(
                        GMP      :  KEP_GMP,
                        column   :  nr_Helped,
                        priority :  2,
                        weight   :  1,
                        abstol   :  0,
                        reltol   :  0);
                    if not retcode then raise error
                    "Unable to set nr_Helped as an objective" ;
                    endif ;

                    retcode := GMP::Column::SetAsMultiObjective(
                        GMP      :  KEP_GMP,
                        column   :  nr_Cycles,
                        priority :  1,
                        weight   :  1,
                        abstol   :  0,
                        reltol   :  0);
                    if not retcode then raise error
                    "Unable to set nr_Cycles as an objective" ;
                    endif ;

                    GMP::Instance::Solve( KEP_GMP );

                    display "After SolveMultiObj", nr_Helped,
                    nr_Cycles, Donation;
            }
        }
        Parameter retcode;
        ElementParameter KEP_GMP {
            Range: AllGeneratedMathematicalPrograms;
        }
}
```

```
Procedure Read_From_Excel {
        Body: {
                Spreadsheet::SetVisibility(
                "Graph_92_8_2equal_0_4.xlsx", 'Off');
                        Spreadsheet:: SetActiveSheet(
                        "Graph_92_8_2equal_0_4.xlsx", "Sheet1");
                        Spreadsheet:: RetrieveSet(
                                Workbook: "Graph_10_8_2equal_0_4.xlsx",
                                Set: Cycles,
                                Range: "A3: A7806",
                                Sheet: "Sheet1");

                        Spreadsheet::RetrieveSet(
                                Workbook: "Graph_92_8_2equal_0_4.xlsx",
                                Set: Pairs,
                                Range: "B2: AY2",
                                Sheet: "Sheet1");

                        Spreadsheet::RetrieveTable(
                                Workbook: "Graph_92_8_2equal_0_4.xlsx",
                                Parameter: Connection,
                                DataRange: "B3: AY7806",
                                RowsRange: "A3: A7806",
                                ColumnsRange: "B2: AY2",
                                Sheet: "Sheet1");

                        Spreadsheet::RetrieveParameter(
                                Workbook: "Graph_92_8_2equal_0_4.xlsx",
                                Parameter: ExpectedHelped,
                                Range:  "AZ3:AZ7806",
                                Sheet: "Sheet1");

                        Spreadsheet:: CloseWorkBook(
                        "Graph_92_8_2equal_0_4.xlsx", 0);
        }
        Comment: {
                "Spreadsheet::SetVisibility(\"Graph_92_8_2equal_0.xlsx\", \
                        Spreadsheet:: SetActiveSheet(\"Graph_92_8_2equal_0.
                        Spreadsheet:: RetrieveSet(
                                Workbook: \"Graph_92_8_2equal_0.xlsx\",
                                Set: Cycles,
                                Range: \"A3: A210\",
                                Sheet: \"Sheet1\");

                        Spreadsheet::RetrieveSet(
                                Workbook: \"Graph_92_8_2equal_0.xlsx\",
                                Set: Pairs,
                                Range: \"B2: AY2\",
                                Sheet: \"Sheet1\");
```

```
                        Spreadsheet::RetrieveTable(
                                Workbook: \"Graph_92_8_2equal_0.xlsx\",
                                Parameter: Connection,
                                DataRange: \"B3: AY210\",
                                RowsRange: \"A3: A210\",
                                ColumnsRange: \"B2: AY2\",
                                Sheet: \"Sheet1\");

                        Spreadsheet:: CloseWorkBook(\"Graph_92_8_2equal_0.x

                Spreadsheet::SetVisibility(\"Graph_92_8_2equal_0_4.xlsx\",
                        Spreadsheet:: SetActiveSheet(\"Graph_92_8_2equal_0_
                        Spreadsheet:: RetrieveSet(
                                Workbook: \"Graph_92_8_2equal_0_4.xlsx\",
                                Set: Cycles,
                                Range: \"A3: A7806\",
                                Sheet: \"Sheet1\");

                        Spreadsheet::RetrieveSet(
                                Workbook: \"Graph_92_8_2equal_0_4.xlsx\",
                                Set: Pairs,
                                Range: \"B2: AY2\",
                                Sheet: \"Sheet1\");

                        Spreadsheet::RetrieveTable(
                                Workbook: \"Graph_92_8_2equal_0_4.xlsx\",
                                Parameter: Connection,
                                DataRange: \"B3: AY7806\",
                                RowsRange: \"A3: A7806\",
                                ColumnsRange: \"B2: AY2\",
                                Sheet: \"Sheet1\");

                        Spreadsheet::RetrieveParameter(
                                Workbook: \"Graph_92_8_2equal_0_4.xlsx\",
                                Parameter: ExpectedHelped,
                                Range:   \"AZ3:AZ7806\",
                                Sheet: \"Sheet1\");

                        Spreadsheet:: CloseWorkBook(\"Graph_92_8_2equal_0_4
        }
}
Procedure Read_From_Excel_Expection {
        Body: {
                Spreadsheet::SetVisibility("Graph_92_8_2equal_0_4.xlsx", 'C
                        Spreadsheet:: SetActiveSheet("Graph_92_8_2equal_0_4
                        Spreadsheet::RetrieveParameter(
                                Workbook: "Graph_92_8_2equal_0_4.xlsx",
                                Parameter: ExpectedHelped,
```

```
                                        Range:   "AZ3:AZ7806",
                                        Sheet:  "Sheet1");

                                Spreadsheet::  CloseWorkBook("Graph_92_8_2equal_0_4.
        }
}
Procedure Read_library {
        Body: {
                if not axll::WorkBookIsOpen(WorkbookFilename : spRead ) the
                        axll::OpenWorkBook(WorkbookFilename : spRead );
                endif;

                axll::SelectSheet(SheetName : "Sheet1" );
                axll::ReadSet(
                        SetReference          : Cycles ,
                        SetRange              : "A3: A19800" ,
                        ExtendSuperSets       :  1,
                        MergeWithExistingElements: 0,
                        SkipEmptyCells        :  0);

                axll::ReadSet(
                        SetReference          : Pairs,
                        SetRange              : "B2: Ay2",
                        ExtendSuperSets       :  1,
                        MergeWithExistingElements :  0,
                        SkipEmptyCells        :  0);

                axll::ReadTable(
                        IdentifierReference   :  Connection,
                        RowHeaderRange        : "A3: A19800" ,
                        ColumnHeaderRange     : "B2: Ay2" ,
                        DataRange             : "B3: Ay19800",
                        ModeForUnknownElements : 0,
                        MergeWithExistingData  : 0);

                axll::ReadList(
                        IdentifierReference   : ExpectedHelped,
                        RowHeaderRange        : "A3:A19800",
                        DataRange             : "Az3:Az19800",
                        ModeForUnknownElements : 0,
                        MergeWithExistingData  : 0);

                axll::CloseWorkBook(WorkbookFilename : spRead );
        }
}
Procedure Write_library {
        Body: {
                spWorkbook := "Output.xlsx";
```

```
                if not axll::WorkBookIsOpen(WorkbookFilename :
                spWorkbook ) then
                        axll::OpenWorkBook(WorkbookFilename :
                        spWorkbook );
                endif;

                spSheet := "Sheet1";

                axll::SelectSheet(SheetName : spSheet );
                axll::WriteSingleValue(
                        ScalarReference : TotalExpected ,
                        Cell            : spExp );

                axll::CloseWorkBook(WorkbookFilename : spWorkbook );

                axll::WriteSingleValue(
                        ScalarReference : nr_Helped ,
                        Cell            : spPeople );

                axll::WriteSingleValue(
                        ScalarReference : nr_Cycles ,
                        Cell            : spCycles );"
        }
}
Procedure Helped_proc {
        Body: {
                solve nr_Helped_prog;
                        if (nr_Helped_prog.ProgramStatus <> 'Optimal')
                        then
                                empty Donation, nr_Helped, nr_Cycles,
                                TotalExpected;
                        endif;

                display "After Solve nr_Helped_prog", Donation,
                nr_Helped, nr_Cycles, TotalExpected;
        }
}
Procedure Cycles_proc {
        Body: {
                solve nr_Cycles_prog;
                        if (nr_Cycles_prog.ProgramStatus <> 'Optimal')
                        then
                                empty Donation, nr_Helped, nr_Cycles,
                                TotalExpected;
                        endif;

                display "After Solve nr_Cycles_proc", Donation,
                nr_Helped, nr_Cycles, TotalExpected;
        }
```

```
        }
        Procedure TotalExp_proc {
                Body: {
                        solve TotExp_prog;
                                if (TotExp_prog.ProgramStatus <> 'Optimal') then
                                        empty Donation, nr_Helped, nr_Cycles,
                                        TotalExpected;
                                endif;

                        display "After Solve TotExp_proc", Donation, nr_Helped,
                        nr_Cycles, TotalExpected;
                }
        }
        Procedure MainInitialization {
        }
        Procedure PostMainInitialization {
        }
        Procedure MainExecution;
        Procedure PreMainTermination {
                Body: {
                        return DataManagementExit();
                }
        }
        Procedure MainTermination {
                Body: {
                        return 1;
                }
        }
}
```