Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Analytics and Operations Research in Logistics

A MTZ-like Formulation for the Kidney Exchange Problem.

Patrick Huizinga (613051)

Supervisor:	Dollevoet, TAB
Second assessor:	Cremers, R
Date final version:	1st July 2024

Abstract

The Kidney Exchange Problem (KEP) involves finding a packing of cardinality constrained cycles in a graph. Previous MIP formulation have a quadratic or exponential number of variables or constraints. In this work we propose a formulation with a quadratic number of variables and constraints, inspired by the Miller–Tucker–Zemlin formulation for the traveling salesman problem. We find this MTZ-like formulation has advantages over the previous ones when the problem size is large in either the number of nodes or the maximum cardinality.

1 Introduction

When someone needs a kidney transplant, they may be able to find a live donor who is willing to donate one of theirs to help the patient. A donor and patient need to match on blood type and tissue type (HLA). Unless the patient and donor are identical twins, there is a chance they do not match.

In some countries, like the Netherlands and the USA, so called kidney exchange programs are set up. When a patient does not have a compatible donor, they may receive a kidney from another patient's donor, if that patient in turn is guaranteed a kidney. The simplest case is when patient A is compatible with donor B and patient B is compatible with donor A. In that case they could exchange kidneys to help both patients.

The Kidney Exchange Problem (KEP), as described in Abraham, Blum and Sandholm (2007), is a problem dealing with "the double coincidence of wants": how to guarantee every patient whose donor is 'used' will also receive a kidney at the same time. Assuming there are no altruistic donors, donors who are not trying to help a specific patient, eventually you will end up with a cycle of patient-donor pairs.

If a patient receives a kidney before their donor donates one, there is a risk the donor backs out, as they are not legally bound to donate. As hospitals have only limited room to perform simultaneous transplants in, these cycles must be limited to a size k. If $k = 2$, the problem is a pairwise compatibility matching and the maximum cardinality matching algorithm can create the greatest number of exchanges in polynomial time (Edmonds, 1965). However when $k \geq 3$, the problem is known to be NP-hard (Abraham et al., 2007).

With altruistic donors the consequences of a donor backing out are greatly reduced. An altruistic donor does not have an associated patient and therefor there is no need to create cycles. When an altruistic donor donates to patient A, later their associated donor A can donate to patient B and donor B to patient C, etc. Should some donor back out after the donation to their associated patient, there will be no patient left wanting.

This paper introduces a MTZ-like formulation for the KEP. In comparison to earlier formulations, it has fewer variables and constraints; both are $\mathcal{O}(n^2)$. This new formulation is then compared to the Cycle formulation and the Extended Edge formulation, as those were found by Constantino, Klimentova, Viana and Rais (2013) to be dominant for at least one set of simulations. Since the Arc formulation also has $\mathcal{O}(n^2)$ number of variables, it will also be compared against the MTZ-like formulation.

The rest of this paper is organized as follows. After this introduction, Section 2 will contain a literature review. Then Section 3 will describe the problem in more detail followed by Section 4 which will describe the various formulations. In Section 5 the settings of the computational analysis will be described, followed by the results of each formulation and a comparison between the various formulations. Finally Section 6 provides some conclusions and suggestions for future research.

2 Literature Review

Roth, Sönmez and Ünver (2007) proposed two MIP formulations to solve the KEP. The Cycle formulation creates a set of all possible cycles and then selects the best combination of cycles using a set packing formulation. The Arc formulation selects arcs with constraints guaranteeing every pair that donates also receives a kidney and that there exists no path of arcs of length k or greater. The Cycle formulation has an exponential number of variables and the Arc formulation an exponential number of constraints.

Constantino et al. (2013) introduces two polynomial sized formulations (Edge Assignment and Extended Edge) for the KEP and compares those against the two exponential formulations. This paper will reproduce their results and compare them with a new formulation.

The Cycle formulation is a set packing formulation which has great performance, if you can get all the sets (cycles) in memory. For larger problems this is not possible and column generation schemes have been developed to get around this. Klimentova, Alvelos and Viana (2014) proposed one example of such a column generation approach and found it to solve the problem with much less computational effort.

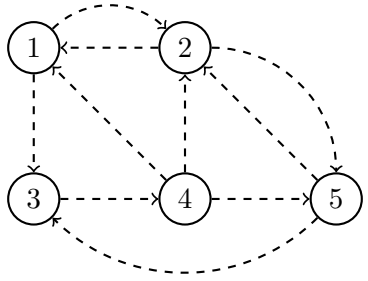
A different research direction is the inclusion of altruistic donors. For example Mak-Hau (2017) proposed extensions to the formulations of Roth et al. (2007) and Constantino et al. (2013) to find exchanges that include so called altruistic chains; chains starting with altruistic donors.

3 Problem Description

Graph theory can be used to represent the KEP. Let there be n patient-donor pairs and let $G(V, A)$ be a directed graph with $V = \{1, \ldots, n\}$ representing each pair and set $A \subseteq \{1, \ldots, n\}^2$ representing the compatibilities between pairs. Let $(i, j) \in A$ if patient i is compatible with donor j and if not, $(i, j) \notin A$. Lastly let w_{ij} represent the weight of arc (i, j), with $w_{ij} = 1$ if the goal is to maximize the number of transplants.

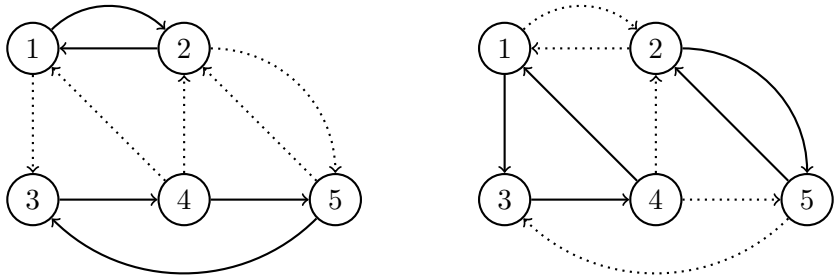
Figure 1 Shows an example of a graph G with $|V| = 5$ pairs and $|A| = 10$. In this example donor 1 is compatible with patient 3, but donor 3 not with patient 1, while pairs 1 and 2 are mutually compatible.

Figure 1: Example of a graph G.



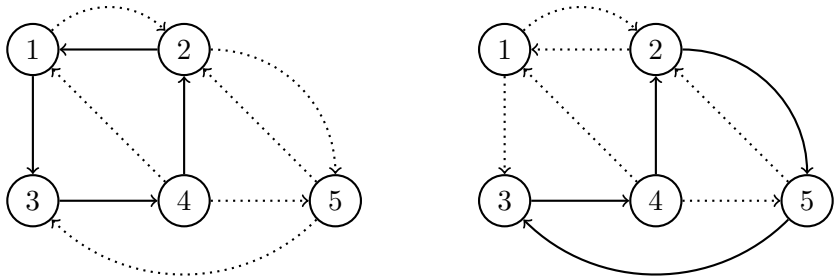
If the goal is to maximum the number of transplants, Figure 2 shows two possible solutions for $k = 3$ where all patients receive a kidney. For $k = 4$ the same optimal solutions remain, but for $k = 5$ several 5-cycle alternative solutions exist.

Figure 2: 2 optimal solution, for $k \geq 3$, each with 2 cycles.



Since there is no requirement to help all patients, Figure 3 shows two possible valid solutions when $k \geq 4$. Should the arcs be weighted, it is possible one of these solutions provides a better objective value than either of the solutions in Figure 2.

Figure 3: 2 valid solution for $k \geq 4$.



4 KEP Formulations

First the exponential Arc and Cycle formulations from Roth et al. (2007) are described, followed by the Extended Edge formulation from Constantino et al. (2013). Lastly the MTZ-like formulation is introduced.

4.1 Arc Formulation

The idea of the Arc formulation is that any cycle with a length of at most k does not contain a path with $k+1$ distinct nodes and therefor to disallow all such paths.

The way this is realized is by only allowing $k-1$ arcs from any such path to be selected at any one time. Given that such a path contains $k+1$ distinct nodes, when selecting $k-1$ arcs one has 'selected' no less than k distinct nodes, the maximum allowed for a cycle.

Introduce variable $x_{ij} \in \mathbb{B}, \forall (i,j) \in A$, which is 1 if arc (i,j) is selected and 0 otherwise.

The Arc formulation can then be written as:

$$\text{Maximize} \sum_{(i,j)\in A} w_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j:(i,j)\in A} x_{ij} = \sum_{j:(j,i)\in A} x_{ji}, \qquad \forall i \in V \tag{2}$$

$$\sum_{j:(i,j)\in A} x_{ij} \leq 1, \qquad \forall i \in V \tag{3}$$

$$\sum_{1\leq p\leq k} x_{i_p i_{p+1}} \leq k-1 \qquad \forall paths(i_1,\ldots,i_{k+1}) \tag{4}$$

$$x_{ij} \in \mathbb{B}. \qquad \forall (i,j) \in A \tag{5}$$

The objective function represented by Equation (1) is the weighted sum of all selected arcs. Equation (2) ensures that if a donor from a pair donates, then the corresponding patient of that pair gets donated to. Equation (3) restricts the number of donations to 1 kidney per donor. Lastly, Equation (4) represents the main idea of this formulation.

All paths are generated as follows: $\forall v \in V$, call `GeneratePaths(A, [v])` defined in Algorithm 1 and combine the results.

Algorithm 1: Arc formulation path generation

```
GeneratePaths(arcs, path) begin
    if |path| = k + 1 then
        yield path
    else
        i ← path.last
        for j ∈ V \ path do
            if (i, j) ∈ arcs then
                yield all GeneratePaths(arcs, [path, j])
            end
        end
    end
end
```

For the row generation, the same function is used, except the realized arcs x_{ij} are used instead of all arcs A.

4.2 Cycle Formulation

The idea of the Cycle formulation is to transform the KEP into a set packing problem. Let $C(k)$ be the set of all cycles in G with length at most k. Furthermore let $V(c) \subseteq V$ be the set of nodes in cycle $c \in C(k)$ and $w(c) = \sum_{(i,j)\in c} w_{ij}$.

Introduce variable $z_c \in \mathbb{B}$, which is 1 if cycle c is selected and 0 otherwise. The Cycle

formulation can then be written as:

$$\text{Maximize} \sum_{c \in C(k)} w(c) z_c \tag{6}$$

$$\text{s.t.} \sum_{c: i \in V(c)} z_c \leq 1, \qquad \forall i \in V \tag{7}$$

$$z_c \in \mathbb{B}. \qquad \forall c \in C(k) \tag{8}$$

The objective function represented by Equation (6) is the weighted sum of all selected cycles. Equation (7) ensures that a node i can occur at most once between all selected cycles.

4.3 Extended Edge Formulation

The basic idea of the Extended Edge formulation is to duplicate the graph multiple times, let each of those duplicates contain their own independent cycles and then limit the number of selected edges per graph to k.

Constantino et al. (2013) calls each of those duplicates a layer $l \in \mathcal{L} \subseteq \{1, \ldots, |V|\}$ and defines $V^l \subseteq V$, and $A^l \subseteq A$. The reason for the definitions of $\mathcal{L}$, V^l and A^l will be described later, for now consider them equal to their super sets. Furthermore introduce $x_{ij}^l \in \mathbb{B} \forall (i,j) \in A^l$, which is a variable that describes whether an arc (i,j) is selected on layer l.

The Extended Edge formulation can then be written as:

$$\text{Maximize} \sum_{l \in \mathcal{L}} \sum_{(i,j) \in A^l} w_{ij} x_{ij}^l \tag{9}$$

$$\text{s.t.} \sum_{j:(i,j) \in A^l} x_{ij}^l = \sum_{j:(j,i) \in A^l} x_{ji}^l, \qquad \forall i \in V^l, \forall l \in \mathcal{L} \tag{10}$$

$$\sum_{l \in \mathcal{L}} \sum_{j:(i,j) \in A^l} x_{ij}^l \leq 1, \qquad \forall i \in V \tag{11}$$

$$\sum_{(i,j) \in A^l} x_{ij}^l \leq k, \qquad \forall l \in \mathcal{L} \tag{12}$$

$$\sum_{j:(i,j) \in A^l} x_{ij}^l \leq \sum_{j:(l,j) \in A^l} x_{lj}^l, \qquad \forall i \in V^l, \forall l \in \mathcal{L} \tag{13}$$

$$x_{ij}^l \in \mathbb{B}. \qquad \forall (i,j) \in A^l, \forall l \in \mathcal{L} \tag{14}$$

The objective function of Equation (9) is the weighted sum of all selected edges, over all layers. Equation (10) ensures that if a donor from a pair donates, the the corresponding patient of that pair gets donated to on the same layer. Equation (11) restricts the number of donations to 1 kidney per donor. Equation (12) restricts the number of selected edges per layer to the maximum cycle size k. Note how Equation (10) implies a selected edge will be part of a cycle on the same layer, but there are no restrictions on the number of cycles per layer.

A naive formulation has a large amount of symmetry and Equation (13) is the first part of limiting this symmetry. It makes sure layer l can only be used to select edges if at least some edge (i,j) is selected for which $i = l$.

Further elimination of symmetry is accomplished by the exact definition of $\mathcal{L}$, V^l and A^l.

The idea is to introduce the rule that some cycle c can be selected on layer l if and only if $l = \min\{i \in c\}$. The first implication is that no node $i < l$ can be part of a cycle on layer l and therefor all such nodes and their edges can be eliminated for that layer: $V^l = \{l, \ldots, |V|\}$ and $A^l = \{(i,j) \in A | i,j \in V^l\}$.

Further reduction of the formulation is accomplished by eliminating unreachable edges (i,j). If no cycle of at most k nodes exists that contains both node l and some edge (i,j) $i > l, j > l$, then such edge can be removed from A^l. Let d_{ij} be the shortest distance from node i to j in terms of the number of edges. Then for any edge (i,j), if $d_{li} + 1 + d_{jl} > k$ then there exists no cycle that contains both node l and edge (i,j) and thus can be removed from A^l. The final definition of $A^l = \{(i,j) \in A \,|\, i,j \in V^l \wedge d_{li} + 1 + d_{jl} \leq k\}$.

If for any node i all edges $(i,j), j \geq l$ or all edges $(j,i), j \geq l$ can be eliminated from A^l in this manner, then i can be removed from V^l. If $V^l = \varnothing$, which also implies $A^l = \varnothing$, then layer l can be removed from $\mathcal{L}$.

In the end the rule that some cycle c can be selected on layer l if and only if $l = \min\{i \in c\}$ can't be completely guaranteed. As k increases, it becomes more likely that multiple edges are allowed on layer l, such that they each could be part of a cycle with node l and together form an 'impostor' cycle that does not contain node l. Then if some cycle exists that satisfies Equation (13), this 'impostor' could be selected on layer l as well. For example when $k = 5$, all edges of Figure 1 from the problem description can be part of a 5-cycle that contains node 5. Therefor layer 1 will contain all edges and either solution from Figure 2 can be selected in layer 1 as that would violate none of the constraints.

The definition in this section contains some minor simplifications from the definition in Constantino et al. (2013). First, Equation (11) is defined over $i \in \bigcup_{l \in \mathcal{L}} V^l$, instead of $i \in V$, which would skip the rare node that can not be part of any k-cycle.

Second, whereas this thesis uses d_{ij} to calculate the distance between two nodes, Constantino et al. (2013) uses the more precise d^l_{ij}, which only considers paths whose edges are all in A^l. The more coarse grained d_{ij} could allow unreachable nodes if the shortest distance between i and j cross some node excluded from some layer. However, as the density of the edges increases this problem becomes less likely and as the total number of nodes increases this problem becomes less pronounced.

4.4 MTZ-like Formulation

Start with the variables $x_{ij} \in \mathbb{B}$ which identifies whether arc (i,j) is used. The basic idea behind this MTZ-like formulation is to assign each node in a cycle a unique natural number u and then constrain $u \leq k$. As with the Miller–Tucker–Zemlin (MTZ) formulation, a simple way to enforce uniqueness is to make u ordered within a cycle; $x_{ij} = 1 \implies u_i < u_j$, except for some node j which is designated as a 'depot'.

To be able to identify this depot, two additional variables per node are introduced; $y_i = 1, \ldots, n$, that identifies which cycle a node i is a part of; and $z_i \in \mathbb{B}$, that indicates whether node i is the 'depot'.

The pseudo-MIP formulation is then given by:

$$\text{Maximize} \sum_{(i,j)\in A} w_{ij}x_{ij} \tag{15}$$

$$\text{s.t.} \sum_{j:(i,j)\in A} x_{ij} = \sum_{j:(j,i)\in A} x_{ji}, \qquad \forall i \in V \tag{16}$$

$$\sum_{j:(i,j)\in A} x_{ij} \leq 1, \qquad \forall i \in V \tag{17}$$

$$x_{ij} = 1 \implies y_i = y_j, \qquad \forall (i,j) \in A \tag{18}$$

$$y_i \neq i \implies z_i = 0, \qquad \forall i \in V \tag{19}$$

$$x_{ij} = 1 \implies u_i + 1 \leq u_j \vee z_i = 1. \qquad \forall (i,j) \in A \tag{20}$$

The objective function and Equations (16) and (17) are the standard KEP constraints, as also seen in the Arc formulation. Equation (18) ensures that if donor i donates to patient j, then both must be in the same cycle. Equation (19) makes sure only one node per cycle is considered the depot, as at most one node can have the same index as the cycle. Finally, Equation (20) ensures that for each connected arc (i, j) either node i has a greater order than node j, or that node i is the depot node in the cycle.

This formulation contains up-to-k-fold symmetry per cycle; each of the nodes in a cycle could be assigned the 'depot'. To prevent such symmetry, the insight from Constantino et al. (2013) can be adapted; only the node with he greatest index in a cycle can be the depot. For this, let $y_i \geq i$. Equation (19) can then also be simplified to $y_i > i \implies z_i = 0$.

Equation (18) can be transformed into the linear inequalities $y_i - y_j \leq (n-j)(1-x_{ij})$ and $y_j - y_i \leq (n-i)(1-x_{ij})$. Equation (19) can become the linear inequality $y_i \leq n-(n-i)z_i$ and Equation (20) can be transformed into $u_i + 1 - u_j \leq k(1 - x_{ij} + z_i)$.

The final MIP formulation then becomes:

$$\text{Maximize} \sum_{(i,j)\in A} w_{ij}x_{ij}$$

$$\text{s.t.} \sum_{j:(i,j)\in A} x_{ij} = \sum_{j:(j,i)\in A} x_{ji}, \qquad \forall i \in V$$

$$\sum_{j:(i,j)\in A} x_{ij} \leq 1, \qquad \forall i \in V$$

$$y_i - y_j \leq (n-j)(1-x_{ij}), \qquad \forall (i,j) \in A$$

$$y_j - y_i \leq (n-i)(1-x_{ij}), \qquad \forall (i,j) \in A$$

$$y_i \leq n-(n-i)z_i, \qquad \forall i \in V$$

$$u_i + 1 - u_j \leq k(1 - x_{ij} + z_i, \qquad \forall (i,j) \in A$$

$$x_{ij} \in \mathbb{B}, \qquad \forall (i,j) \in A$$

$$u_i \in \{i, \ldots, n\}, \qquad \forall i \in V$$

$$z_i \in \mathbb{B}. \qquad \forall i \in V$$

As $|A| \leq n^2$, both the number of variables and the number of constraints is $\mathcal{O}(n^2)$. Tech-

nically u_i requires size $\mathcal{O}(\log n)$ to be represented. Practically MIP solvers will use IEEE 754 floating point numbers to represent variables, which in the 32-bit variant can store integers up to $2^{24} = 16,777,216$ without loss of fidelity.

5 Computational Analysis

To empirically test the different formulations, they have been implemented and run for a variety of configurations. First the environment and tested configurations will be described, followed by the results per formulation and this section will close with a comparison of all formulations.

5.1 Environment and Configurations

All formulations were implemented and run using C#/.Net 8.0 and Gurobi 11.02. The computer had an AMD Ryzen™ 9 7950X3D CPU containing 16 cores with a base clock speed of 4.2 GHz and 128 GB of DDR5 RAM.

Every combination of the following settings was run: formulation is Arc (with row generation), Cycle, Extended Edge or MTZ-like; number of patient-donor pairs $n = 50, 100, 200$; compatibility density $d = 0.1, 0.2, 0.5, 0.8$; max cycle size $k = 3, 4, 5, 6, 10, 15, 20$; with $w = 1$ or $w \sim Unif(0, 1)$ (weighted arcs). When a certain configuration could not be run due to an out of memory error, a similar configuration with a greater n or greater d was not attempted. For the Cycle and and MTZ-like formulations, these tests were extended with $n = 500, 1000$.

For each configuration, a new program was launched to prevent any influence from previous runs, like JIT optimizations or memory leaks. Each run spawned 10 threads that each ran one single-threaded instance simultaneously. Appendix B.1 provides more details on this program.

Every instance is generated in a reproducible way such that different formulations are tested using the exact same instance. Furthermore for a given seed, a smaller instance is a strict subproblem of a larger instance. For example a graph $A_{0.2}$ for a problem with density 0.2 is a subgraph of $A_{0.5}$ for density 0.5. Likewise a graph A_{50} for $N = 50$ is a subgraph of A_{100} for $N = 100$. More details can be found in Appendix A.

Measurement of the setup time starts from the moment an instance is generated until hand-off to the Gurobi solver occurs. The running time is taken as reported by the solver.

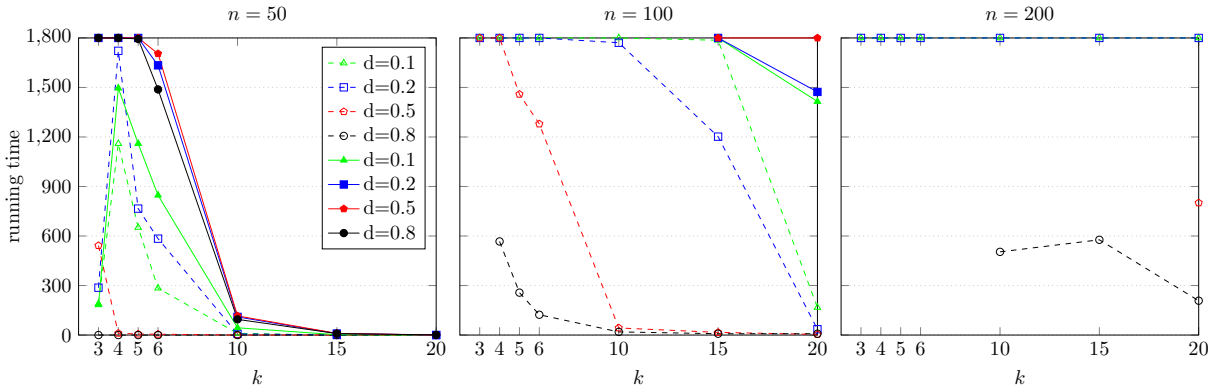
In total 7500 instances ran without failure, of which 1680 were the Arc formulation (with row generation), 780 the Cycle, 1680 the Extended Edge, and 3360 the MTZ-like formulation.

The next four subsections provide individual results for each formulation, then section 5.6 provides comparisons between formulations. For each formulation the running times in seconds and remaining LP gap in percentage are reported. The running time is the average of all 10 runs. The LP gap is the average of the results with a gap > 0. The LP gap is the result of the formula $(UB - Opt)/Opt$, where UB is the upper bound of the LP relaxation and Opt is the objective value of the best found solution. Should at least 9 instances have a running time of 1800 seconds the running time is omitted and the mean LP gap of all ten instances is reported.

Tip: the sections for formulation results are positioned in the same location on the page to make it easy to compare them. Those sections also link to each other in such a way that any two formulations can be compared by repeatedly following a reference without moving your cursor.

5.2 Arc

See also Section 5.3 for Cycle, Section 5.4 for Edge and Section 5.5 for MTZ results.



Solid markers and lines for $w_{ij} \sim Unif(0,1)$, open markers and dashed lines for $w_{ij} = 1$.

Table 1: Running times (seconds)/LP gap (%) of Arc formulation (row generation)

		$w=1$				$w \sim Unif(0,1)$			
n	k	$d=0.1$	0.2	0.5	0.8	0.1	0.2	0.5	0.8
	3	93	191	153	0	47	/10.5	/6.0	/4.1
	4	887	447	3	0	656/0.7	/6.5	/4.4	/2, 8
	5	410	197	0	0	579/1.2	/4.4	/2.8	/3.3
50	6	50	56	0	0	327	1498/1.3	1597/0.8	1398/0.3
	10	1	0	0	0	4	16	16	12
	15	0	0	0	0	0	0	0	0
	20	0	0	0	0	0	0	0	0
	3	/74.3	/20.6	39.4	–	/33.5	/27.5	/12.9	/12.1
	4	–	/20.6	41.0	–	/19.2	/12.3	/9.2	/0.9
	5	/86.1	/20.6	132410.3	862	/13.5	/9.4	/19.0	–
100	6	/99.2	/20.6	386	226	/20.0	/18.6	/65.0	–
	10	/45.7	/9.6	2	8	/92.0	/80.6	/59.2	/50.2[a]
	15	515/0.7	12	0	0	1625/15.5	/12.1	1668/2.0	1559/0.7
	20	1	1	0	0	57	291	335	325
	3	/45.7	/10.5	–	–	/52.5	/42.0	–	–
	4	/45.7	/10.5	–	–	/45.5	/48.6	–	–
	5	/45.7	/10.5	–	–	/48.8	/28.8	–	–
200	6	/45.7	/10.5	–	–	/49.5	/64.8	–	–
	10	/45.7	/10.5	–	–	–	–	–	–
	15	/45.7	/10.5	892	120	–	–	–	–
	20	/41.3	/2.3	94	70	–	/72.5	–	–

– average LP gap $\geq 100\%$. [a] 1 instances had no result.
Running time omitted if =1800 for all instances.

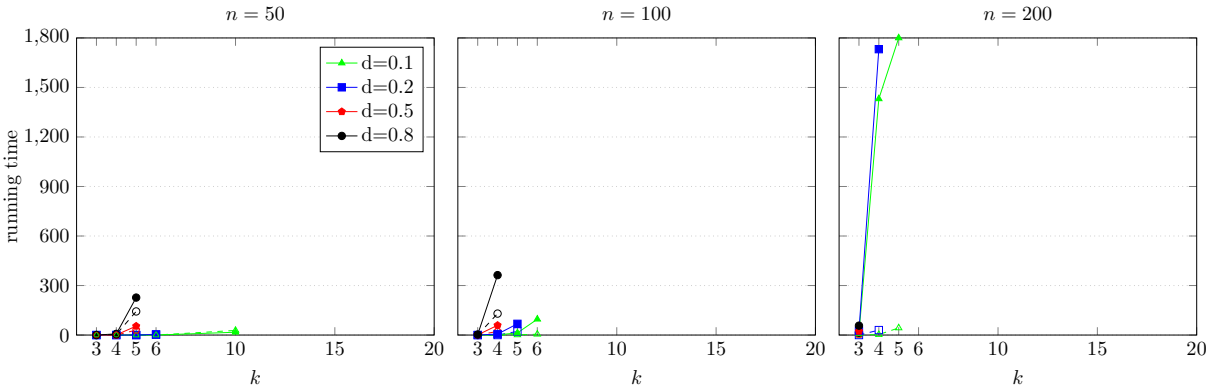
Solutions are found quicker as k increases. One explanation is that those solutions are closer to the optimal solution with an unrestricted cycle length and therefor would require fewer constraints to be generated. The only exception is $k = 3$ which is sometimes faster than $k \geq 4$.

For higher densities, there appears to be a bifurcation of results. Either the solution is found much faster than with lower densities, or no solution is found at all.

Out of 840 results for $w = 1$, 84 found no solution and 289 did not find the optimal solution within 1800 seconds. Of those, 31 had a LP gap > 2, 53 a gap > 1 and 89 a gap > 0.5. Curiously, the largest gap for density 0.2 is 0.28, while the other densities each have multiple gaps > 1.

5.3 Cycle

See also Section 5.2 for Arc, Section 5.5 for MTZ and Section 5.4 for Edge results.



Solid markers and lines for $w_{ij} \sim Unif(0,1)$, open markers and dashed lines for $w_{ij} = 1$.

Table 2: Running times (seconds)/LP gap (%) of Cycle formulation

		$w = 1$				$w \sim Unif(0,1)$			
n	k	$d = 0.1$	0.2	0.5	0.8	0.1	0.2	0.5	0.8
50	3	0	0	0	0	0	0	0	0
	4	0	0	0	4	0	0	1	7
	5	0	0	30	143	0	0	54	227
	6	0	2	–	–	0	4	–	–
	10	28	–	–	–	17	–	–	–
	15	–	–	–	–	–	–	–	–
	20	–	–	–	–	–	–	–	–
100	3	0	0	0	0	0	0	1	3
	4	0	0	22	130	1	6	59	362
	5	1	22	–	–	13	67	–	–
	6	5	–	–	–	96	–	–	–
	10	–	–	–	–	–	–	–	–
	15	–	–	–	–	–	–	–	–
	20	–	–	–	–	–	–	–	–
200	3	0	0	5	14	10	26	23	57
	4	3	30	–	–	1431	1733/0.4	–	–
	5	43	–	–	–	/2.2	–	–	–
	6	–	–	–	–	–	–	–	–
	10	–	–	–	–	–	–	–	–
	15	–	–	–	–	–	–	–	–
	20	–	–	–	–	–	–	–	–
500	3	4	15	166	–	/0.5	/0.4	/0.3	–
	4	144	–	–	–	/2.7	–	–	–
1000	3	65	221	–	–	/1.2	/0.9	–	–

– run did not complete due to running out of memory.
Running time omitted if = 1800 for all instances.

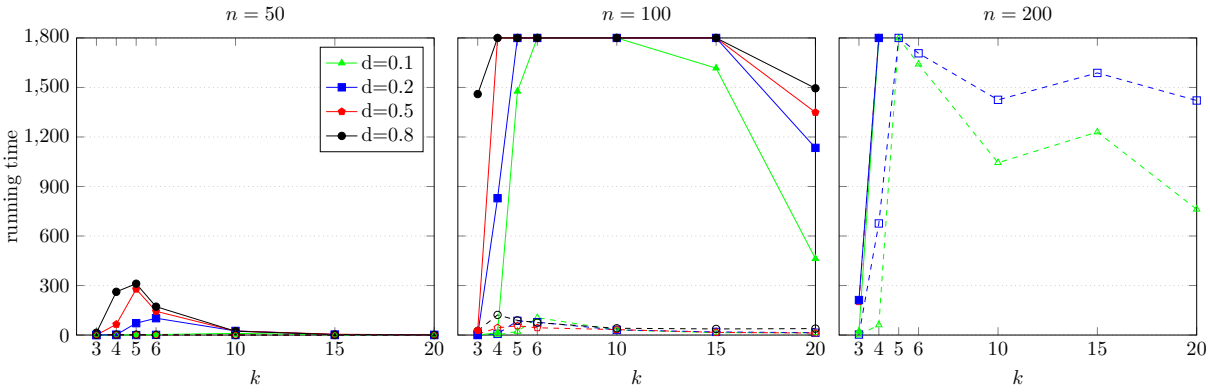
As expected due to the exponential number of variables, with an average RAM of 12-13 GB per instance, the Cycle formulation has a limited number of supported configurations.

For $w = 1$ and configurations that don't run out of memory, the Cycle formulation finds a solution in minutes.

Out of 780 runs without an error, 85 were cut off after 1800 seconds. Of those, 11 had a LP gap > 0.025, with the largest being 0.033.

5.4 Extended Edge

See also Section 5.5 for MTZ, Section 5.2 for Arc and Section 5.3 for Cycle results.



Solid markers and lines for $w_{ij} \sim Unif(0,1)$, open markers and dashed lines for $w_{ij} = 1$.

Table 3: Running times (seconds)/LP gap (%) of Extended Edge formulation

		$w=1$				$w \sim Unif(0,1)$			
n	k	$d=0.1$	0.2	0.5	0.8	0.1	0.2	0.5	0.8
50	3	0	0	0	1	0	0	0	14
	4	0	0	1	2	0	2	65	262
	5	0	0	1	2	0	72	278	311
	6	1	1	1	1	3	102	144	172
	10	1	0	0	0	10	25	24	23
	15	0	0	0	0	1	3	5	3
	20	0	0	0	0	1	1	1	1
100	3	0	0	5	25	0	1	27	1460/0.3
	4	1	7	44	131	14	829/0.5	/3.4	/4.1
	5	22	86	56	88	1477/1.3	/7.3	/5.9	/3.7
	6	106	78	44	75	/4.2	/7.4	/4.1	/4.5
	10	31	31	33	41	/2.4	/2.7	/1.4	/1.0
	15	14	19	16	37	1617/0.3	/0.7	/0.2	/0.2
	20	14	14	12	39	463	1134	1349	1495
200	3	0	3	202	–	23	212	/1.1	–
	4	61	675	–	–	/2.2	/4.4	/49.5	–
	5	1798/1.8	/9.2	–	–	/17.4	/32.9	–	–
	6	1641/2.0	1707/4.9	–	–	/39.4	/41.5	–	–
	10	1044/0.5	1425/2.7	–	–	/34.5	/23.2	–	–
	15	1230/0.9	1588/1.8	–	–	/22.6	/9.0	–	–
	20	762	1421/2.0	–	–	/7.7	/5.8	–	–

– average LP gap $\geq 100\%$.
Running time omitted if =1800 for all instances.

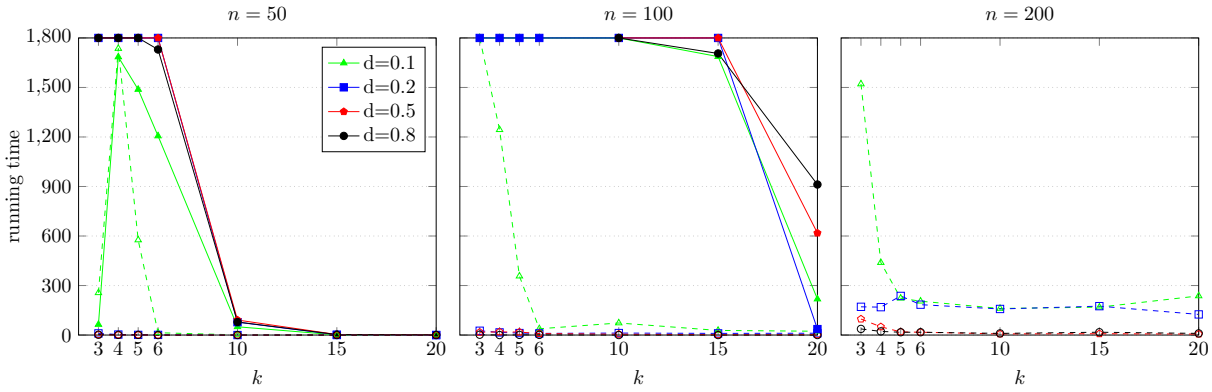
The results form a sort of triangle. The highest running times are around $k = 6, 10$ and higher densities. The lower running times for small k are probably due to the filtering out of unreachable nodes leaving a smaller problem space. For large k the explanation is probably the solution approaching the best solution for unrestricted k, thereby making the problem 'easier'.

Interesting to note is the sometimes enormous LP gaps in the found solutions. For $n = 200$, 28 instances had a gap > 3 and all of those gaps were in the range $[14692, 40706]$.

The configuration $n = 200, k = 4$ is one where all instances of $w = 1$ had a higher LP gap than same seeds with $w \sim Unif(0,1)$, in all cases being more than double.

5.5 MTZ-like

See also Section 5.4 for Edge, Section 5.3 for Cycle and Section 5.2 for Arc results.



Solid markers and lines for $w_{ij} \sim Unif(0,1)$, open markers and dashed lines for $w_{ij} = 1$.

Table 4: Running times (seconds)/LP gap (%) of MTZ-like formulation

		$w=1$				$w \sim Unif(0,1)$			
n	k	$d=0.1$	0.2	0.5	0.8	0.1	0.2	0.5	0.8
50	3	256	10	2	2	64	/13.1	/6.9	/4.9
	4	1736/6.6	3	1	1	1685/8.6	/10.6	/4.7	/3.3
	5	576/5.0	1	1	1	1488/6.0	/7.1	/3.1	/1.8
	6	14	1	0	0	1206/3.2	/3.0	/1.9	1730/1.0
	10	1	1	0	0	50	78	91	79
	15	0	0	0	0	1	2	3	3
	20	0	0	0	0	0	0	1	1
100	3	/5.7	29	17	2	/34.6	/18.9	/10.3	/6.6
	4	1244/2.0	16	19	0	/22.0	/12.9	/7.1	/5.4
	5	357/1.0	12	18	1	/15.6	/9.8	/5.9	/4.0
	6	38	10	9	1	/13.7	/7.7	/4.7	/3.3
	10	73	12	5	0	/4.0	/3.5	/3.2	/2.1
	15	28	10	2	0	1687/0.9	/1.0	/0.4	1706/0.3
	20	23	11	2	0	218	360	618	912
200	3	1521/0.9	171	97	37	/32.3	/21.7	/13.0	/7.3
	4	439	169	50	22	/25.6	/18.1	/9.9	/5.9
	5	223	237	19	17	/22.5	/15.6	/8.1	/4.5
	6	205	184	17	17	/19.4	/14.2	/7.5	/4.1
	10	161	158	7	10	/12.2	/11.5	/5.4	/3.2
	15	169	175	8	7	/7.6	/10.0	/4.9	/2.8
	20	237	125	6	10	/5.3	/3.7	/1.2	/0.7
500	3	/3.8	/6.3	1707/0.5	884/0.2	/48.0	/36.8	/35.5	/34.3
	4	/3.3	/3.3	1677/0.3	810/0.6	/41.9	/28.6	/19.3	/25.7
	5	/3.8	/2.3	882	269	/38.4	/26.0	/14.1	/26.5
	6	/3.7	/1.7	1429/0.2	226	/36.4	/23.4	/15.1	/26.6
	10	/3.5	/1.4	220	48	/37.1	/21.6	/13.1	/25.2
	15	/3.3	1741/1.3	119	55	/35.9	/16.9	/13.9	/23.5
	20	/4.4	1108/0.5	63	30	/34.0	/16.9	/15.4	/20.7
	30	/4.2	123	47	30	/35.1	/15.3	/14.1	/20.0
	40	/3.5	57	56	31	/34.9	/13.9	/12.8	/18.6
	50	/3.3	41	53	30	/35.3	/13.1	/9.3	/17.8

– average LP gap $\geq 100\%$.
Running time omitted if =1800 for all instances.

Table 5: Running times (seconds)/LP gap (%) of MTZ-like formulation (continued)

		$w = 1$				$w \sim Unif(0,1)$			
n	k	$d = 0.1$	0.2	0.5	0.8	0.1	0.2	0.5	0.8
	3	–	/75.8	/86.3	/50.2	–	–	–	–
	4	–	/54.6	/70.1	/28.6	–	–	–	–
	5	/77.9	/47.5	/39.9	/28.0	–	–	–	–
1000	6	/80.0	/38.5	/20.8	/33.8	–	–	–	–
	10	/61.4	/34.2	/20.8	1448/36.0	–	–	–	–
	15	/58.8	/29.5	1634/29.3	1544/27.1	/93.9	–	–	–
	20	/56.4	/35.2	1195/76.9	1130/15.0	/82.6	/87.7	–	–
	40	/61.7	1716/36.7	1190/20.9	1144/7.0	/68.0	/65.5	–	–
	60	/62.3	1632/36.8	1343/0.3	665/0.2	/54.6	/53.9	–	–
	80	/52.9	814/20.0	1075/0.4	911/0.2	/51.0	/50.6	–	–
	100	/69.5	905/12.0	915/0.5	720/0.4	/48.6	/42.9	–	–

– average LP gap $\geq 100\%$.
Running time omitted if =1800 for all instances.

The MTZ-like formulation generally finds a solution quicker as k increases. For $w = 1$, solutions are also found quicker as the density goes up.

All configurations for $n = 50, 100, 200, 500$ and $w_{ij} = 1$ were run, for 1120 instances in total. Of those instances, 184 were cut off after 1800 seconds. Of those 184 instances, 28 results had a LP gap > 0.05 and 2 had a gap > 0.1.

All 560 configurations for $w \sim Unif(0, 1)$ and $n = 50, 100$ were run. Of those instances, 377 were cut off after 1800 seconds. Of those 377 instances, 199 had a LP gap > 0.05, 88 results had a gap > 0.1 and 20 had a gap > 0.2. All of those 20 involved $n = 100$ with either $k = 3$, $d = 0.1$ or both.

Probably the strangest result is the fact that $n = 200, d = 0.1, w = 1$ outperforms $n = 100$. A likely explanation is the limited number of optimal solutions for $n = 100$. For example for $k = 3$, only 5 instances have an optimal value of 100.

This would provide an additional observation for the idea that the MTZ-like formulation benefits from more possible optimal solutions. In practically all cases, as the density increases or when k is larger, the running time decreases.

The results for $n = 1000$ start to show the limitations of the MTZ-like formulation. Even when using $k = 1000$, no feasible solutions for higher densities can be found in 1800 seconds, even though the Arc formulation with 'row generation' can find the optimal solution in 2 seconds (results for $k = 1000$ are not included in this document).

5.6 Comparison

Table 6 gives an overview of the best performing formulation for each problem configuration. Best is determined by comparing the worst performing instance for each formulation. In some cases that means even though the average time or LP gap of a formulation might be worse than another, but due to the variance in performance it could still end up to be considered as better. In case the worst performance of multiple formulations are (almost) equal, they are all included in alphabetical order. Two instances are considered almost equal if their total running times differ less than 10% or 1 second, whichever is greater, or if the LP gaps are less than 10% apart.

Table 6: Best worst-performing instance

		$w = 1$				$w \sim Unif(0,1)$			
n	k	$d = 0.1$	0.2	0.5	0.8	0.1	0.2	0.5	0.8
	3	c/ee	c/ee	c/ee	c/ee	c/ee	c/ee	c/ee	c
	4	c/ee	c/ee	c	ar	c/ee	c	c	c
	5	c/ee	c/ee	ar	ar	c/ee	c	c	c
50	6	c	m	ar/ee/m	ar/ee/m	c	c	ee	ee
	10	ee/m	ar/ee/m	ar/ee/m	ar/m	ar	ar	ar	ar/ee
	15	ar/ee/m	ar/ee/m	ar/ee/m	ar/ee/m	ar	ar	ar	ar
	20	ar/ee/m	ar/ee/m	ar/ee/m	ar/ee/m	ar/ee/m	ar/m	ar	ar
	3	c/ee	c/ee	c	c	c/ee	c/	c	c
	4	c	c	m	m	c	c	c	c
	5	c	m	m	m	c	c	m	ee/m
100	6	c	m	m	m	c	ee	ee/m	m
	10	ee	m	ar	m	ee	ee	ee	ee
	15	ee	m	ar	m	ar	ee	ee	ee/m
	20	ar	ar	ar	ar/m	ar	ar/m	ar	ar
	3	c/ee	c	c	c	c	c	c	c
	4	c	c	m	m	c	c	m	m
	5	c	m	m	m	c	m	m	m
200	6	m	m	m	m	m	m	m	m
	10	m	m	m	m	m	m	m	m
	15	m	m	m	m	m	m	m	m
	20	m	m	m	m	m	m	m	m

ar: Arc (with Row generation) *c*: Cycle
ee: Extended Edge *m*: MTZ-like

When looking at the upper bounds (UB) of the LP relaxation, it seems as thought they are generally speaking close together. For example for the configuration $n = 100, d = 0.5k = 4, w \sim Unif(0,)$, where the Cycle formulation found an exact solution, all formulations have an upper bound within 2.5% of the exact solution. This goes even for the Arc formulation, with its average gap of 92%.

The Extended Edge formulation seems to give lower UBs in general. For example for $n = 200, d = 0.2k = 6, w \sim Unif(0,)$ and the same seeds, every UB is lower than the UB of the MTZ formulation, though the difference is always smaller than 0.1%. This is despite the average LP gap of MTZ being 14.2% compared to 41.5% of Extended Edge.

There appears to be one exceptions to this observations, the gaps > 3 for the Extended Edge formulation, as mentioned in section 5.4. In those cases the UB can be more than a million.

6 Conclusion

For any problem that can fit in the memory requirements of the Cycle formulation, there is barely a contest: use the Cycle formulation. Especially for weighted arcs, it provides superior performance in terms of running time and LP gap. However, this means that for the 12-13 GB RAM used in this test, when the number of patient-donor pairs grow, problems are limited to $k = 3$ and/or densities $d = 0.1$.

When the Cycle formulation is not viable and arcs are not weighted, the MTZ-like formula-

tion is probably a best choice. Although there are cases where the Extended Edge or the Arc formulation performs better, the improvement is mostly minimal.

When arcs are weighted, the Extended Edge formulation generally performs better than the MTZ formulation for smaller models ($n = 50$), while the MTZ formulation gains the upper hand for larger models ($n = 200$). When k becomes large compared to n, when the solution starts to resemble the optimal solution without restrictions, the Arc formulation is advantageous.

For future research, altruistic donor chains could be added to the MTZ-like formulation. Mak-Hau (2017) has already proposed extensions for the Cycle and Extended Edge formulations. In both cases MTZ-style constraints were added to support altruistic chains. It is likely supporting altruistic chains would be an even better fit for the MTZ-like formulation from this paper.

For the Cycle formulation various column generation schemes have been proposed. It could be interesting to see how those perform on larger densities or larger cycle sizes, especially compared to the MTZ-like formulation.

References

Abraham, D. J., Blum, A. & Sandholm, T. (2007). Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th acm conference on electronic commerce* (pp. 295–304).

Constantino, M., Klimentova, X., Viana, A. & Rais, A. (2013). New insights on integer-programming models for the kidney exchange problem. *European Journal of Operational Research*, *231*(1), 57–68.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of mathematics*, *17*, 449–467.

Klimentova, X., Alvelos, F. & Viana, A. (2014). A new branch-and-price approach for the kidney exchange problem. In B. Murgante et al. (Eds.), *Computational science and its applications – iccsa 2014* (pp. 237–252). Cham: Springer International Publishing.

Mak-Hau, V. (2017). On the kidney exchange problem: cardinality constrained cycle and chain problems on directed graphs: a survey of integer programming approaches. *Journal of Combinatorial Optimization*, *33*, 35–59.

Roth, A. E., Sönmez, T. & Ünver, M. U. (2007). Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review*, *97*(3), 828–851.

A Instance Generation

This section describes the instance generation in more detail.

First the empty $N \times N$ matrices A and w are created. Then, given a seed s and density d, the matrices will be filled using the following algorithm:

Algorithm 2: A and w generation

```
rng ← createRNG(s)
for i ← 1 to N do
    for j ← 1 to i − 1 do
        A_{i,j} ← (rng.Next() < d)
        w_{i,j} ← rng.Next()
        A_{j,i} ← (rng.Next() < d)
        w_{j,i} ← rng.Next()
```

$rng.Next() \in [0, 1)$ and will advance the state of rng, such that $rng.next() \neq rng.next()$.

Weights are always generated, even when they are not used to make sure the state of rng stays consistent between instances with and without weights and therefor generate the exact same values for A. To simplify the implementations of the formulations, w will be replaced by a binary matrix if weights are not needed.

Since $j \neq i$, matrices A and w will have all-zero diagonals. Since $j < i$, to fill an $N \times N$ matrix, first the $n \times n$ matrix will be filled, $\forall n < N$.

B Applications

For this paper the application Kep.Runner was developed. Kep.Runner runs one or more configurations and writes the results to a file.

All code and the results of all runs can be found at https://github.com/patrickhuizinga/Kep under commit 6201120b053f718c208c563b7773f47b6313ad00.

B.1 Kep.Runner

Usage: `Kep.Runner form1 [form2..] [n n1..] [d d1..] [k k1..] [w] [t t1]`

`form1`, the first formulation to run, is the only mandatory parameter. For every other parameter the defaults are: `n 10`, `d 0.2`, `k 3`, `w False` and `t 1`

`form` options: `cycle` to run the Cycle formulation, `edge` to run the Extended Edge formulation, `mtz` to run the MTZ-like formulation, `arcPath` to run the Arc formulation with all constraints generated upfront, `arcPathRowGen` to run the Arc formulation with row generation.

`n` is an integer that configures the number of patient-donor pairs.

`d` is a real, $0.0 < d \leq 1.0$ that configures the probability of the compatibility between to pairs.

`k` is an integer that configures the k parameter.

`w` is a boolean (`true` or `false`) that configures whether weighted arcs are used. If `w` is completely omitted, it will default to `false`. If `w` is given, but not followed by any values, it will default to `true`.

`t` is an integer that configures the number of threads that are run simultaneously.

If for all parameters only a single value is given, including keeping it on its default, the application will run in single process mode. In single process mode $t1$ threads will be started and each will run the given configuration with a distinct, but deterministic seed. The results of each threads will be written to `output.dat` in fixed column format.

If for any parameter multiple values are given, the application will run in multi process mode. In multi process mode, for each combination of the configuration a single process Kep.Runner will be started. If such a child process reports an error or crashes, for example for running out of memory, that error will be written to `error.dat`. Once a child process has finished, the next one will be started, such that at most one child process is running at a time.

Example1: `Kep.Runner cycle n 50 d 0.2 k 3 t 5` will run the Cycle formulation on $n = 50$ pairs with a compatibility density of $d = 20\%$, $w = 1$ on all arcs, limiting cycles to $k = 3$, running 5 threads simultaneously.

Example2: `Kep.Runner mtz cycle n 20 d 0.1 0.2 k 3 4 5 6 w t 10` will run 16 different configurations: both MTZ-like and Cycle formulations on $n = 20$, $d = 0.1, 0.2$, $k = 3, 4, 5, 6$, $w = Unif(0, 1)$, running 10 threads simultaneously. On my test machine that gives the output:

```
2024-07-01 21:28:02Z  Start  mtz n 30 k 3 d 0.1 t 10 w True
2024-07-01 21:28:03Z  Start  cycle n 30 k 3 d 0.1 t 10 w True
2024-07-01 21:28:03Z  Start  mtz n 30 k 4 d 0.1 t 10 w True
2024-07-01 21:28:04Z  Start  cycle n 30 k 4 d 0.1 t 10 w True
2024-07-01 21:28:04Z  Start  mtz n 30 k 5 d 0.1 t 10 w True
```

```
2024-07-01 21:28:05Z  Start  cycle n 30 k 5 d 0.1 t 10 w True
2024-07-01 21:28:05Z  Start  mtz n 30 k 6 d 0.1 t 10 w True
2024-07-01 21:28:06Z  Start  cycle n 30 k 6 d 0.1 t 10 w True
2024-07-01 21:28:06Z  Start  mtz n 30 k 3 d 0.2 t 10 w True
2024-07-01 21:30:11Z  Start  cycle n 30 k 3 d 0.2 t 10 w True
2024-07-01 21:30:11Z  Start  mtz n 30 k 4 d 0.2 t 10 w True
2024-07-01 21:40:15Z  Start  cycle n 30 k 4 d 0.2 t 10 w True
2024-07-01 21:40:15Z  Start  mtz n 30 k 5 d 0.2 t 10 w True
2024-07-01 21:42:40Z  Start  cycle n 30 k 5 d 0.2 t 10 w True
2024-07-01 21:42:40Z  Start  mtz n 30 k 6 d 0.2 t 10 w True
2024-07-01 21:43:17Z  Start  cycle n 30 k 6 d 0.2 t 10 w True
```

Already in the program output you can see that (some) MTZ-like instances run up to 10 minutes, while no Cycle instance takes more than a second.

B.2 output.dat

Assuming a configuration did not crash, its results will be appended to the file `output.dat` in a fixed column format, with each column separated by one or more spaces.

The order of the columns in the file is: formulation name, n, k, $d(\%)$, $a(\%)$, L, seed, weight ? 1 : 0, setup time, running time, objective, LP gap. The columns a and L are always 0 and 99 respectively as the exist to support unused altruistic donors. The weight column is `1` if weights are used ($w \sim Unif(0,1)$) or `0` if weights are not used ($w = 1$). If a solution was not found, the objective will be `-0` and the LP gap will be `1E+100`.

Using example1 from Kep.Runner: `Kep.Runner cycle n 50 d 0.2 k 3 t 5`, `output.dat` will contain:

```
cycle            50  3 20  0 99  44 0    0    0         50          0
cycle            50  3 20  0 99  46 0    0    0         50          0
cycle            50  3 20  0 99  43 0    0    0         50          0
cycle            50  3 20  0 99  42 0    0    0         50          0
cycle            50  3 20  0 99  45 0    0    0         50          0
```

C Flawed Arc formulation row generation

An earlier version of the Arc formulation would first generate all possible cycles longer than k and then extract all paths. This created duplication in two ways: multiple cycles that differed only slightly could contain the same path and due to the generation of duplicate cycles starting at a different node, even more copies of the same path would be generated.

Unsurprisingly, creating that many duplicate paths slows down solving the problem in the vast majority of cases. Table 7 contains the running times and LP gaps of that implementation for all the configurations that were run.

Table 7: Running times (seconds)/LP gap (%) of Arc formulation (row generation)

		$w = 1$				$w \sim Unif(0,1)$			
n	k	$d = 0.1$	0.2	0.5	0.8	0.1	0.2	0.5	0.8
	3	190	287	542	0	138	/9.7	/6.6	/4.2
	4	1160/4.4	1722/5.2	11	0	1496/7.4	/8.3	/5.9	/10.1
	5	625/4.3	766/6.4	6	0	1160/6.1	/5.1	/11.2	/6.3
50	6	284	584/4.2	5	0	849/1.7	1634/2.0	1705/3.5	1488/0.7
	10	7	0	0	10	44	110	116	95
	15	0	0	0	0	2	9	10	9
	20	0	0	0	0	0	0	0	1
	3	/20.9	/19.5	/41.0	–	/36.1	/23.5	/32.4	/94.7
	4	–	/20.6	/41.0	–	/24.2	/25.7	/80.6	/56.8
	5	–	/20.6	1459/41.0	–	/45.8	/31.0	/86.1	/66.3
100	6	/94.9	/20.6	1279/28.0	123[a]	/52.3	/56.4	/70.0	/43.2
	10	/59.7	/18.7	43	19	/77.1	/71.4	/59.1	/46.5[a]
	15	/29.9	1202/5.5	16	8	/41.8	/43.7	/33.6	/13.9
	20	167	35	4	8	1415/11.7	1473/9.5	/2.3	/1.5
	3	/45.7	/10.5	–	–	?	?	?	?
	4	/45.7	/10.5	–	–	?	?	?	?
	5	/45.7	/10.5	–	–	?	?	?	?
200	6	/45.7	/10.5	–	–	?	?	?	?
	10	/45.7	/10.5	–	–	?	?	?	?
	15	/45.7	/10.5	–	–	?	?	?	?
	20	/40.9	/9.3	881[a]	208[a]	?	?	?	?

– average LP gap $\geq 100\%$. [a] 1 or more instances had no result.
Running time omitted if =1800 for all instances.

Only for $n = 100, d = 10\%, k = 3, w = 1$ is the result of the new implementation significantly worse, with a gap of 74.3% vs 20.9%.