E-Companion: Small-size KEP example and computational experiments on the Dutch KEP objectives

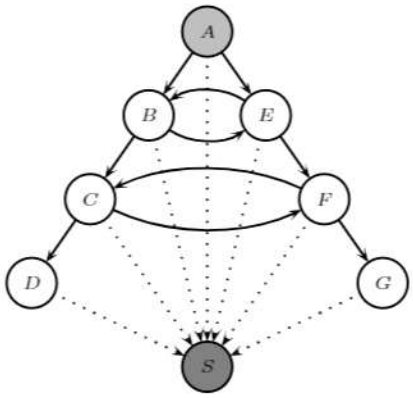
In Section EC.1 of this e-companion, we present a small-size KEP instance that is used several times to describe the behaviours of each of our algorithms. We also provide in Section EC.2 the outcomes of our best approach when used to solve the Dutch KEP objectives.

EC.1. KEP example

In the following, we introduce an example that will be used throughout the e-companion to describe the behaviours of the algorithms introduced in the paper.

EXAMPLE EC.1. Let us consider the following instance with one non-directed donor A, 6 recipients B, C, D, E, F, and G, each of whom is paired with one donor, and the compatibility graph described in Figure EC.1. Where relevant, all arcs have unitary score. For conciseness, where it is obvious we are referring to a donor, we will use, for example, donor B to refer to the donor paired with recipient B.

Figure EC.1 Compatibility graph for Example EC.1



Non-directed donor A is compatible with the recipients B and E. Donor E is compatible with recipient B and donor B is compatible with recipient E. As a result, vertices B and E can form the cycle of size two $[B, E]$. Another cycle of size two is $[C, F]$. As donor E is compatible with recipient F, and donor F is compatible with recipient G, non-directed donor A and vertices E, F, and G can form the chain of length four $A \to E \to F \to G \to S$. Other chains of length four are, for example, $A \to E \to B \to C \to S$ (which includes cross arc $B \to E$) or $A \to B \to C \to D \to S$. The

latter chain can also be shortened to form, for example, the chain of length three $A \to B \to C \to S$ or the chain of length two $A \to B \to S$.

Note that for the sake of clarity, we omit the last edge "$\to S$" when identifying a chain in the cycle formulation in the rest of this e-companion, but we do leave it in PICEF as this last edge corresponds to a variable in the model. In other words, chain $A \to B \to S$ is denoted by $A \to B$ in the cycle formulation, while it is created by edges $A \to B$ and $B \to S$ in PICEF.

EC.1.1. Cycle formulation applied to Example EC.1

In the following, we report the outputs obtained by the cycle formulation with an ILP solver when applied to Example EC.1. The first four columns give an identifier for each cycle/chain c, its length, the number of cross arcs relative to c, and the set of vertices that compose the cycle/chain. The five following columns contain the value taken by x_c after solving the cycle formulation with each of the five objective functions.

Table EC.1 Results of the cycle formulation with an ILP solver for Example EC.1

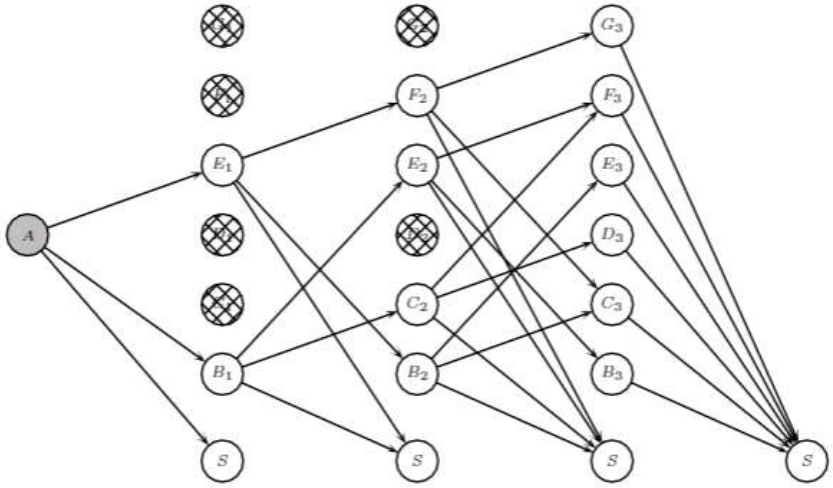
id	len.	#c.a.	comp.	$\bar{z}_1 = 5$	$\bar{z}_2 = 0$	$\bar{z}_3 = 0$	$\bar{z}_4 = 0$	$\bar{z}_5 = 4$
0	2	0	$[B, E]$	1	1	1	1	1
1	2	0	$[C, F]$	1	1	1	1	1
2	1	0	A	1	1	1	1	1
3	2	0	$A \to B$	0	0	0	0	0
4	3	2	$A \to B \to E$	0	0	0	0	0
5	4	2	$A \to B \to E \to F$	0	0	0	0	0
6	3	0	$A \to B \to C$	0	0	0	0	0
7	4	1	$A \to B \to C \to F$	0	0	0	0	0
8	4	0	$A \to B \to C \to D$	0	0	0	0	0
9	2	0	$A \to E$	0	0	0	0	0
10	3	2	$A \to E \to B$	0	0	0	0	0
11	4	2	$A \to E \to B \to C$	0	0	0	0	0
12	3	0	$A \to E \to F$	0	0	0	0	0
13	4	1	$A \to E \to F \to C$	0	0	0	0	0
14	4	0	$A \to E \to F \to G$	0	0	0	0	0

The optimal objective value for KEP_{f_1} is 5 and can be obtained by selecting cycles $[C, F]$ and $[B, E]$, and the chain A. The optimal objective values for KEP_{f_2},...,KEP_{f_5} are 0, 0, 0, and 4, respectively. These values can be obtained by selecting the same set of cycles and chains.

EC.1.2. PICEF applied to Example EC.1

In the following, we provide the graph required to model the chain structure in PICEF for Example EC.1.

Figure EC.2 Chain structure for Example EC.1 in PICEF, non-activated nodes are crosshatched



As A is compatible with the recipients B and E, nodes B_1 and E_1 are activated in the second subgraph. Since donor B is compatible with recipients C and E, and since donor E is compatible with recipients B and F, nodes B_2, C_2, E_2, and F_2 are activated in the third subgraph. An interesting observation is that every feasible chain listed in Table EC.1 can be reconstructed through a path in Figure EC.2. The opposite is not true since some paths in Figure EC.2 are infeasible (e.g., $A \to B_1 \to E_2 \to B_3 \to S$). Note that sink node S is duplicated in the figure for the sake of clarity.

EC.1.3. Cycle/chain deactivation algorithm applied to Example EC.1

We detail in Table EC.2 the outputs obtained by the cycle/chain deactivation algorithm with an ILP solver when applied to Example EC.1. The first four columns still give an identifier for each cycle/chain c, its length, the number of cross arcs relative to c, and the set of vertices that compose the cycle/chain. Columns $L(F_k^C)$ contain the value (in sub-column "V") and the reduced cost (in sub-column "RC") taken by x_c after solving the continuous relaxation of the cycle model for objective function k where $k = 1, 2, 3, 4$. Columns T_k contain the value (in sub-column "V") taken by x_c after solving the cycle model for objective function k where $k = 1, 2, 3, 4, 5$. An "x" indicates that the cycle/chain was deactivated when the model was solved.

Algorithm 1 determines that $T_1 = \lfloor L(F_1^C) \rfloor = 6$ and thus, puts chains 2-7 and 9-13 in $\mathcal{C}_1$ as they all have a reduced cost equal to -1. In other words, every feasible continuous solution containing one of any cycle/chain in $\mathcal{C}_1$ must have $\hat{z}_1 \leq 5$. The same holds for any feasible integer solution.

Table EC.2 Results of the cycle/chain deactivation obtained by an ILP solver for Example EC.1

id	len.	# c.a.	comp.	f_1				f_2			f_3			f_4			f_5
				$L(F_1^C)=6$		$T_1=6$	$T_1=5$	$L(F_2^C)=0$		$T_2=0$	$L(F_3^C)=0$		$T_3=0$	$L(F_4^C)=0$		$T_4=0$	
				V	RC	V	V	V	RC	V	V	RC	V	V	RC	V	V
0	2	0	$[B,E]$	0.5	0	1	0	0.52	0	1	0.69	0	1	0.68	0	1	1
1	2	0	$[C,F]$	0.5	0	1	1	0.83	0	1	1	0	1	1	0.14	1	1
2	1	0	A	0	-1	x	0	0.19	0	1	0.37	0	1	0.36	0	1	1
3	2	0	$A\to B$	0	-1	x	0	0.16	0	0	0.31	0	0	0.32	0	0	0
4	3	2	$A\to B\to E$	0	-1	x	1	0.15	0	0	0	1	x	x	x	x	x
5	4	2	$A\to B\to E\to F$	0	-1	x	0	0	1	x	x	x	x	x	x	x	x
6	3	0	$A\to B\to C$	0	-1	x	0	0.17	0	0	0	1	x	x	x	x	x
7	4	1	$A\to B\to C\to F$	0	-1	x	0	0	1	x	x	x	x	x	x	x	x
8	4	0	$A\to B\to C\to D$	0.5	0	0	0	0	0.58	x	x	x	x	x	x	x	x
9	2	0	$A\to E$	0	-1	x	0	0.16	0	0	0.31	0	0	0.32	0	0	0
10	3	2	$A\to E\to B$	0	-1	x	0	0	0	0	0	1	x	x	x	x	x
11	4	2	$A\to E\to B\to C$	0	-1	x	0	0	1	x	x	x	x	x	x	x	x
12	3	0	$A\to E\to F$	0	-1	x	0	0.17	0	0	0	1	x	x	x	x	x
13	4	1	$A\to E\to F\to C$	0	-1	x	0	0	1	x	x	x	x	x	x	x	x
14	4	0	$A\to E\to F\to G$	0.5	0	0	0	0	0.58	x	x	x	x	x	x	x	x

With integrality constraints, the best solution obtained is $\ddot{z}_1 = 4$, which is strictly smaller than T_1. Thus, the algorithm decreases T_1 to 5, reactivates all the chains, and finds an integer solution of objective value 5 by selecting cycle 1 and chain 4.

When it switches to f_2, the algorithm finds that $T_2 = \lceil L(F_2^C) \rceil = 0$ and puts chains 5, 7, 8, 11, 13, and 14 in $\mathcal{C}_2$. With integrality constraints, the best solution obtained is $\ddot{z}_2 = 0$, which matches the bound. The chains in $\mathcal{C}_2$ are deactivated for the rest of the algorithm.

When it arrives at f_3, the algorithm finds that $T_3 = \lceil L(F_3^C) \rceil = 0$ and puts chains 4, 6, 10, and 12 in $\mathcal{C}_3$. With integrality constraints, the best solution obtained is $\ddot{z}_3 = 0$, which matches the bound. The chains in $\mathcal{C}_3$ are deactivated for the rest of the algorithm.

When processing the fourth objective f_4, the algorithm finds that $T_4 = \lfloor L(F_1^C) \rfloor = 0$, but does not find any chain or cycle to put in $\mathcal{C}_4$ as all the reduced costs are equal to 0. With integrality constraints, the best solution obtained is $\ddot{z}_4 = 0$, which matches the bound.

When it reaches f_5, only 5 cycles/chains out of the 15 are still activated. The algorithm finds a solution with score 4 that selects cycles $[C,F]$, $[B,E]$, and the chain that contains the non-directed donor A.

EC.1.4. Diving algorithm applied to Example EC.1

In the following, we detail the outputs obtained by the diving algorithm with an ILP solver when applied to Example EC.1. We remark that $\bar{\Lambda}_2 = \bar{\Lambda}_3 = 1$ in this example.

Table EC.3 Results of the diving algorithm obtained by an ILP solver for Example EC.1, $\bar{\Lambda}_2 = \bar{\Lambda}_3 = 1$

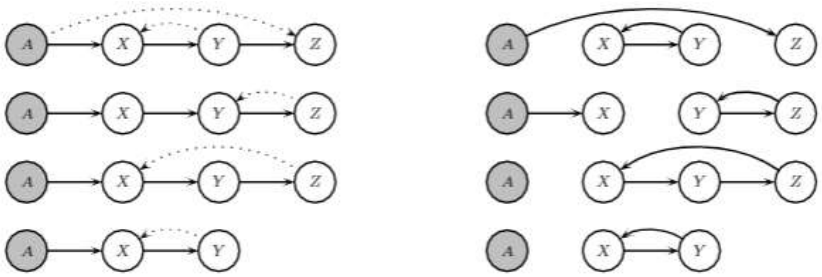
id	len.	# c.a.	comp.	$L(F_1^C)$		$L(F_2^C)$		$L(F_3^C)$		F_3^C	F_3^C	F_2^C	$L(F_2^C)$		$L(F_3^C)$		F_3^C	$L(F_4^C)$		F_4^C	F_5^C
				V	RC	V	RC	V	RC	V	V	V	V	RC	V	RC	V	V	RC	V	V
0	2	0	$\{B, E\}$	0.5	0	0.5	0	0.5	0				0.52	0	0.69	0	1	0.68	0	1	1
1	2	0	$\{C, F\}$	0.5	0	0.5	0	0.5	0				0.83	0	1	0	1	1	0.14	1	1
2	1	0	A	0	-1	x	x	x	x	I	I	I	0.19	0	0.37	0	1	0.36	0	1	1
3	2	0	$A \to B$	0	-1	x	x	x	x	N	N	N	0.16	0	0.31	0	0	0.32	0	0	0
4	3	2	$A \to B \to E$	0	-1	x	x	x	x	F	F	F	0.15	0	0	1	x	x	x	x	x
5	4	2	$A \to B \to E \to F$	0	-1	x	x	x	x	E	E	E	0	1	x	x	x	x	x	x	x
6	3	0	$A \to B \to C$	0	-1	x	x	x	x	A	A	A	0.17	0	0	1	x	x	x	x	x
7	4	1	$A \to B \to C \to F$	0	-1	x	x	x	x	S	S	S	0	1	x	x	x	x	x	x	x
8	4	0	$A \to B \to C \to D$	0.5	0	0.5	0	0.5	0	I	I	I	0	0.58	x	x	x	x	x	x	x
9	2	0	$A \to E$	0	-1	x	x	x	x	B	B	B	0.16	0	0.31	0	0	0.32	0	0	0
10	3	2	$A \to E \to B$	0	-1	x	x	x	x	L	L	L	0	0	0	1	x	x	x	x	x
11	4	2	$A \to E \to B \to C$	0	-1	x	x	x	x	E	E	E	0	1	x	x	x	x	x	x	x
12	3	0	$A \to E \to F$	0	-1	x	x	x	x				0.17	0	0	1	x	x	x	x	x
13	4	1	$A \to E \to F \to C$	0	-1	x	x	x	x				0	1	x	x	x	x	x	x	x
14	4	0	$A \to E \to F \to G$	0.5	0	0.5	0	0.5	0				0	0.58	x	x	x	x	x	x	x
Variables			Λ_2	0		0		0		0	1	0	0		0		0	0		0	0
			Λ_3	0		0		0		1	0	0	0		0		0	0		0	0
			T_1	6		6		6		6	6	5	5		5		5	5		5	5
			T_2	–		1		1		1	–	–	0		0		0	0		0	0
			T_3	–		–		0		–	–	–	–		0		0	0		0	0
			T_4	–		–		–		–	–	–	–		–		–	0		0	0

Algorithm 2 determines that $T_1 = \lfloor L(F_1^C) \rfloor = 6$ and thus, puts chains 2-7 and 9-13 in $\mathcal{C}_1$ as they all have a reduced cost equal to -1. It then calculates $T_2 = \lceil L(F_2^C) \rceil = 1$, but does not find any chain or cycle to put in $\mathcal{C}_2$ as all the reduced costs are equal to 0. It continues by calculating $T_3 = \lceil L(F_3^C) \rceil = 0$ and does not find any chain or cycle to put in $\mathcal{C}_3$ either. With integrality constraints, the algorithm does not find any feasible solution to F_3^C, and thus, increments variable Λ_3 which is now equal to 1. As 1 is also the limit set by $\bar{\Lambda}_3$, the algorithm removes any assumption it has on T_3, empties $\mathcal{C}_3$, and solves again F_3^C. Since it was not able to find any feasible solution, the algorithm backtracks to the second objective function, increments Λ_2, and resets Λ_3. As Λ_2 is now equal to the limit $\bar{\Lambda}_2$, it removes any assumption it has on T_2, empties $\mathcal{C}_2$, and solves again F_2^C. Once more, no feasible solution could be found, so the algorithm backtracks to the first objective function, sets T_1 to 5, empties $\mathcal{C}_1$, resets Λ_2, and solves again $T_2 = \lceil L(F_2^C) \rceil$, which is now equal to 0. Because of their reduced costs, chains 5, 7, 8, 11, 13, and 14 are put in $\mathcal{C}_2$. The algorithm continues with $T_3 = \lceil L(F_3^C) \rceil = 0$ and puts chains 4, 6, 10, and 12 in $\mathcal{C}_3$. With integrality constraints, the best solution obtained is $\ddot{z}_3 = 0$, which matches the bound. The chains in $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ are deactivated for the rest of the algorithm. Algorithm 2 behaves similarly to Algorithm 1 for the two last objective functions f_4 and f_5, and it finds the same optimal solution containing cycles 0 and 1, and chain 2.

EC.1.5. Dominated chains for the UKLKSS and their application to Example EC.1

In the following, we detail four generic cases of dominated chains (on the left side) and their respective dominating set (on the right side).

Figure EC.3 Dominated chains for the UKLKSS and their dominating sets



For example, the chain of length three $A \to X \to Y$ with a cross arc between Y and X contributes to 3 units in f_1, 0 units in f_2, and 1 unit in f_3. Leaving the altruistic donor A alone and selecting the cycle of size two $[X, Y]$ instead still contributes to 3 units in f_1 and 0 units in f_2, but now counts as 0 units in f_3, which is an improvement with respect to selecting the chain $A \to X \to Y$. Similarly, the chain of length four $A \to X \to Y \to Z$ with a cross arc between Z and Y contributes to 4 units in f_1, 1 unit in f_2, and 0 units in f_3. Using instead chain $A \to X$ and cycle $[Y, Z]$ still contributes to 4 units in f_1, but now counts as 0 units in f_2 and 0 units in f_3, which is an improvement with respect to selecting the chain $A \to X \to Y \to Z$. The dominated chains of Example EC.1 are outlined in the following table.

Table EC.4 Dominated chains identification for Example EC.1

id	len.	# c.a.	cycle members	is dominated	dominating set
0	2	0	$[B, E]$	n/a	n/a
1	2	0	$[C, F]$	n/a	n/a
2	1	0	A	no	-
3	2	0	$A \to B$	no	-
4	3	2	$A \to B \to E$	yes	{0,2}
5	4	2	$A \to B \to E \to F$	no	-
6	3	0	$A \to B \to C$	no	-
7	4	1	$A \to B \to C \to F$	yes	{1,3}
8	4	0	$A \to B \to C \to D$	no	-
9	2	0	$A \to E$	no	-
10	3	2	$A \to E \to B$	yes	{0,2}
11	4	2	$A \to E \to B \to C$	no	-
12	3	0	$A \to E \to F$	no	-
13	4	1	$A \to E \to F \to C$	yes	{1,9}
14	4	0	$A \to E \to F \to G$	no	-

EC.1.6. Hybrid algorithm applied to Example EC.1

In the following, we detail the outputs obtained by the hybrid algorithm with an ILP solver when applied to Example EC.1.

Table EC.5 Results of the extension of the diving algorithm to PICEF obtained by an ILP solver for Example EC.1, $\bar{\Lambda}_2 = \bar{\Lambda}_3 = 1$

id	len.	comp.	$L(F_1^P)$		$L(F_2^P)$		$L(F_3^P)$		F_3^P	F_3^P	F_2^P	$L(F_2^P)$		$L(F_3^P)$		F_3^P
			V	RC	V	RC	V	RC	V	V	V	V	RC	V	RC	V
0	2	$[B,E]$	0.5	0	0.5	0	0.5	0				0.53	0	0.69	0	1
1	2	$[C,F]$	0.5	0	0.5	0	0.5	0				0.81	0	1	0	1
2	1	$A \to B_1$	0.5	0	0.5	0	0.5	0				0.34	0	0.31	0	1
3	1	$A \to E_1$	0.5	0	0.5	0	0.5	0				0.47	0	0.31	0	0
4	1	$A \to S$	0	-1	x	x	x	x				0.19	0	0.37	0	0
5	1	$B_1 \to E_2$	0	0	0	0	0	0				0	0	0	1	x
6	1	$B_1 \to C_2$	0.5	0	0.5	0	0.5	0				0.19	0	0	1	x
7	1	$B_1 \to S$	0	-1	x	x	x	x				0.15	0	0.31	0	0
8	1	$E_1 \to B_2$	0	0	0	0	0	0				0.13	0	0	1	x
9	1	$E_1 \to F_2$	0.5	0	0.5	0	0.5	0	I	I	I	0.19	0	0	1	x
10	1	$E_1 \to S$	0	-1	x	x	x	x	N	N	N	0.15	0	0.31	0	0
11	1	$B_2 \to E_3$	0	-1	x	x	x	x	F	F	F	0	1	x	x	x
12	1	$B_2 \to C_3$	0	-1	x	x	x	x	E	E	E	0	1	x	x	x
13	1	$B_2 \to S$	0	-1	x	x	x	x	A	A	A	0.13	0	0	0	0
14	1	$E_2 \to B_3$	0	-1	x	x	x	x	S	S	S	0	1	x	x	x
15	1	$E_2 \to F_3$	0	-1	x	x	x	x	I	I	I	0	1	x	x	x
16	1	$E_2 \to S$	0	-1	x	x	x	x	B	B	B	0	0	0	0	0
17	1	$C_2 \to F_3$	0	-1	x	x	x	x	L	L	L	0	1	x	x	x
18	1	$C_2 \to D_3$	0.5	0	0.5	0	0.5	0	E	E	E	0	0.7	x	x	x
19	1	$C_2 \to S$	0	-1	x	x	x	x				0.19	0	0	0	0
20	1	$F_2 \to C_3$	0	-1	x	x	x	x				0	1	x	x	x
21	1	$F_2 \to G_3$	0.5	0	0.5	0	0.5	0				0	0.7	x	x	x
22	1	$F_2 \to S$	0	-1	x	x	x	x				0.19	0	0	0	0
23	1	$B_3 \to S$	0	0	0	0	0	0				0	0	0	0	0
24	1	$E_3 \to S$	0	0	0	0	0	0				0	0	0	0	0
25	1	$C_3 \to S$	0	0	0	0	0	0				0	0	0	0	0
26	1	$F_3 \to S$	0	0	0	0	0	0				0	0	0	0	0
27	1	$D_3 \to S$	0.5	0	0.5	0	0.5	0				0	0	0	0	0
28	1	$G_3 \to S$	0.5	0	0.5	0	0.5	0				0	0	0	0	0
		Λ_2	0		0		0		0	1	0	0		0		0
		Λ_3	0		0		0		1	0	0	0		0		0
Variables		T_1	6		6		6		6	6	5	5		5		5
		T_2	–		1		1		1	–	–	0		0		0
		T_3	–		–		0		–	–	–	–		0		0

We omit the full description of the algorithm outputs, since they are similar to those presented in Table EC.3. We simply observe that every cycle/chain that is not deactivated in Table EC.3 at a given step can be reconstructed by some arcs that are not deactivated at that same step in Table EC.5. Once the third objective function is solved, we do a complete graph exploration of the chain structure to transform every feasible path in PICEF into a chain in the cycle formulation. In the example, 15 arcs are still activated:

- $A \to B_1$, $A \to E_1$, $A \to S$ from subgraph 1,
- $B_1 \to S$, $E_1 \to S$ from subgraph 2,

- $B_2 \to S$, $E_2 \to S$, $C_2 \to S$, $F_2 \to S$ from subgraph 3,
- $B_3 \to S$, $E_3 \to S$, $C_3 \to S$, $F_3 \to S$, $D_3 \to S$, $G_3 \to S$ from subgraph 4.

These arcs allow us to reconstruct chains $A \to S$, $A \to B_1 \to S$, and $A \to E_1 \to S$ in the cycle formulation. These were also the chains that were activated at the fourth objective function in Table EC.3.

EC.2. Randomly generated instances with the Dutch KEP objectives

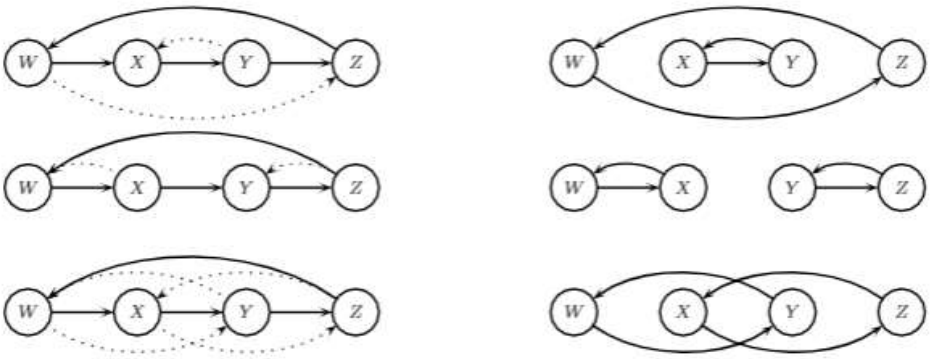
We also tested our best algorithm on the Dutch KEP objectives. As we did not have any information regarding the transplant centres nor on the waiting time experienced by the recipients, we only optimised the four first objective functions, which are: (i) maximise the number of transplants, (ii) maximise the number of transplants between a donor and recipient with the same blood group, (iii) prioritise the transplants to hard-to-match recipients, and (iv) minimise the length of the largest selected cycle.

A more precise interpretation of objective function (iii) was obtained through personal correspondence with van de Klundert (2020). For each recipient, their Match Probability (MP) is calculated based on their cPRA, their unacceptable HLA matches, and the distribution of HLA types in the current donor pool (Keizer et al. 2005). The Dutch system then attempts to find a solution that achieves optimal values on objectives (i) and (ii), and maximises the number of transplants to the recipients with the lowest MP, and subject to this, then maximises the number of transplants to recipients with the second-lowest MP, and so forth. As distributions of specific HLA types and unacceptable HLA matches were not available to us, we approximated a recipient's MP as one minus their cPRA value. We ordered the recipients by non-decreasing MP, and assigned to each recipient r a value v_r that indicates how many distinct MP values are less than or equal to the MP value of r. As an example, if the MP values of 5 recipients are $(0, 0, 0.15, 0.5, 0.5)$, then the v_r values are $(1, 1, 2, 3, 3)$. In our experiments we approximated objective (iii) by using a single objective function that minimises the sum of the v_r values of the recipients selected for transplant. This was done to avoid additional complexity, and in the knowledge that we would have been unable to generate realistic MP values in any case.

According to Biró et al. (2019), the Dutch KEP allows cycles and chains up to size four. Most of the techniques presented in Section 3 can be trivially extended to the Dutch KEP. As a noticeable

difference, we observe that since none of the objective functions involves maximising the number of cross arcs, PICEF can be used instead of the cycle formulation as a basic algorithm to compare with our best approach, which is now the diving algorithm without dominated cycles (but using PICEF instead of the cycle formulation for the models). Indeed, as there is no interest in switching from PICEF to the cycle formulation during the optimisation, there is no reason to try a hybrid approach for the Dutch KEP. Also, as mentioned in Section 3, it is not computationally interesting to forbid dominated chains in PICEF because it increases the model size. However, since we now have cycles of size up to four, and since forbidding a dominated cycle simply sets its corresponding value to 0 (and thus, reduces the model size), we detail in Figure EC.4 three generic cases of dominated cycles of length four. Note that in addition to the existence of specific cross arcs, we also have to make sure that any dominated cycle does not involve more transplants between a donor and a recipient with the same blood group than its dominating set (because of the second objective function).

Figure EC.4 Dominated cycles of length 4 for the Dutch KEP and their dominating sets (dominated cycles must not involve more transplants with the same blood group than their dominating set)



During our experiments on the Dutch KEP, we discovered that for both algorithms, it took the solver a significant amount of time to find a feasible solution when solving a model with integrality constraints. As a result, we used so-called *warm starts* as an additional feature: in the diving algorithm (respectively, in PICEF), the best integer solution obtained when optimising f_3 (respectively, f_1 to f_3) was saved and given to the solver as a warm start when optimising f_4 (respectively, f_2 to f_4). We also tested this feature for the UK and Spanish KEPs, but we did not notice any improvement in terms of computing time. Based on the solver outputs, our hypothesis

is that it is significantly harder for the solver to find a feasible integer solution in PICEF than it is in the cycle formulation. Considering that in the case of the UKLKSS objectives, the hybrid algorithm never obtains a feasible integer solution of PICEF for f_1 and very rarely obtains one for f_2, then in most instances there is no integer solution that could be used as warm start when optimising f_3 (before the switch to the cycle formulation for f_4 and f_5).

We tested the adapted diving algorithm without dominated cycles on a subset of the randomly generated instances created for the UKLKSS objectives and we compared its results with those obtained by PICEF. The subset contained 750 instances with the number of recipients $|\mathcal{R}| \in \{50, 100, 200, 400, 600\}$ and the proportion of non-directed donors $q \in \{0.01, 0.05, 0.10, 0.15, 0.20\}$. These values were chosen so that each algorithm could solve all the instances to optimality in a reasonable time. We did not test the adapted diving algorithm without dominated cycles on larger instances because there were too many cycles of length four that were generated in the early stages of the approach. Similar to what was presented in Algorithm 2, our new approach dived until f_3, and checked at the same time if the bounds T_1 and T_2 were correct.

Table EC.6 displays the time spent on average by PICEF and the diving algorithm without dominated cycles (attribute "-DCl") optimising each objective function, the model size when optimising f_4, and the average number of times the bound T_k $(k = 1, \ldots, 4)$ had to be updated for instances with up to 600 recipients.

Table EC.6 Information on the diving algorithm without dominated cycles and PICEF for instances with up to 600 recipients

$\|\mathcal{R}\|$	Algorithm	**TT**	T1	T2	T3	T4	nb. var.	nb. cons.	nb. nzs.	nb. F1	nb. F2	nb. F3	nb. F4
50	PICEF	**0.07**	0.01	0.02	0.02	0.02	340	196	1917	-	-	-	-
	Diving algoithm - DCl	**0.03**	0.00	0.01	0.01	0.01	114	132	597	0.01	0.01	0.06	0
100	PICEF	**0.37**	0.08	0.08	0.09	0.11	2466	1487	16 971	-	-	-	-
	Diving algoithm - DCl	**0.09**	0.02	0.01	0.03	0.02	479	410	2842	0	0	0.05	0
200	PICEF	**4.11**	0.78	0.94	1.21	1.16	22 671	16 879	178 082	-	-	-	-
	Diving algoithm - DCl	**0.67**	0.24	0.06	0.25	0.09	2666	1840	17 950	0	0.01	0.18	0
400	PICEF	**100.13**	17.35	23.42	39.60	19.49	299 249	269 390	2 564 640	-	-	-	-
	Diving algoithm - DCl	**14.40**	4.82	1.18	6.56	1.55	31 769	25 029	253 554	0	0.01	0.39	0
600	PICEF	**894.61**	105.6	135.29	564.42	88.07	1 310 222	1 235 630	11 465 402	-	-	-	-
	Diving algoithm - DCl	**389.22**	24.10	7.48	344.89	11.49	207 614	181 770	1 757 257	0	0	0.44	0

We observe that even for the Dutch KEP, our approach is faster than a plain ILP formulation such as PICEF. This is particularly true for instances with 400 recipients or more. For example,

it takes 100 seconds on average for PICEF to solve an instance with 400 recipients while it only takes a bit less than 15 seconds on average to solve the same instance for the diving algorithm. We observe a significant reduction in the number of variables when optimising f_4 which can mainly be attributed to the cycle/arc deactivation algorithms. Indeed, for instances with 600 recipients, out of 1 200 292 cycles of size four that were created on average, only 10 913 were dominated, meaning that the 1.1 million difference in the average number of variables between the two models is due to cycle/arc deactivation. We also observe that the number of variables grows very fast as the number of recipients increases. This can be explained by the fact that the Dutch KEP allows cycles of size four, that PICEF requires one variable for each feasible cycle, and that not so many cycles of size four are necessarily removed by the cycle/arc deactivation algorithm at an early stage of the algorithm since none of the first objective functions requires to limit the use of these large cycles. In order to reduce even further the number of variables, an interesting approach would be to extend the column generation procedure developed by Dickerson et al. (2016) for PICEF to tackle Dutch KEP objectives.

Finally, we observe that the number of backtracking steps is barely noticeable for f_1 and f_2 and more significant for f_3, indicating that the diving paradigm used (diving until f_3 and set $\bar{\Lambda}_2$ and $\bar{\Lambda}_3$ to 1) seems appropriate. We also notice that it can be time-consuming for the diving algorithm to optimise f_3, which is mainly explained by the fact that, in most cases, the algorithm does not have an integer solution to use as a warm start when optimising f_3. Note that a heuristic is not an easy way to alleviate this problem since the solution given as a warm start needs to satisfy the bounds obtained when optimising f_1 and f_2.

References

Biró P, Haase-Kromwijk B, Andersson T, Ásgeirsson EI, Baltesová T, Boletis I, Bolotinha C, Bond G, Böhmig G, Burnapp L, Cechlárová K, Ciaccio PD, Fronek J, Hadaya K, Hemke A, Jacquelinet C, Johnson R, Kieszek R, Kuypers DR, Leishman R, Macher MA, Manlove D, Menoudakou G, Salonen M, Smeulders B, Sparacino V, Spieksma FCR, Valentín MO, Wilson N, van der Klundert J (2019) Building kidney exchange programmes in Europe–an overview of exchange practice and activities. *Transplantation* 103:1514–1522.

Dickerson J, Manlove D, Plaut B, Sandholm T, Trimble J (2016) Position-indexed formulations for kidney exchange. *Proceedings of the 2016 ACM Conference on Economics and Computation*, 25–42, EC '16 (New York, NY, USA: ACM).

Keizer K, de Klerk M, Haase-Kromwijk B, Weimar W (2005) The Dutch algorithm for allocation in living donor kidney exchange. *Transplantation Proceedings* 37:589–591.

van de Klundert J (2020) Personal Communication.