A new matheuristic and improved instance generation for kidney exchange programmes

Maxence Delorme[‡], Sergio García[⋆], Jacek Gondzio[⋆], Joerg Kalcsics[⋆], David Manlove[†], William Pettersson[*][†], and James Trimble[†]

[‡]*Department of Econometrics and Operations Research, Tilburg University, The Netherlands*
[†]*School of Computing Science, University of Glasgow, United Kingdom*
[⋆]*School of Mathematics, University of Edinburgh, United Kingdom*

June 2, 2021

Abstract

Kidney exchange programmes increase the rate of living donor kidney transplants, and operations research techniques are vital to such programmes. These techniques, as well as changes to policy regarding kidney exchange programmes, are often tested using random instances created by a Saidman generator. We devise a new matheuristic that can optimally solve a benchmark set of Saidman instances in seconds: these instances have not been solved in under thirty minutes previously. This is possible as we take advantage of particular properties of these random instances that are noticeably different in real-world instances. We follow up this matheuristic with new techniques for generating random kidney exchange instances that are far more similar to real-world instances from the UK kidney exchange programme. This new process for generating random instances provides a more accurate base for comparisons of algorithms and models, and gives policy-makers a better understanding of potential changes to policy leading to an improved decision-making process.

1 Introduction

1.1 Background

Kidney disease affects millions of people worldwide, and the two known treatment options for end-stage kidney disease are dialysis and kidney transplantation. Of these options, kidney transplantation offers a better quality of life, better life expectancy, and is cheaper [6], but donor kidneys are in short supply. These kidneys are donated by either deceased or living donors, with better outcomes for the recipient correlated with living donors [18, 32]. However, it is difficult for a recipient to find a compatible and willing living donor on their own. A Kidney Exchange Programme (KEP) alleviates the need for recipients to be compatible with their own donors [28]. Instead, recipients join KEPs with a willing paired donor (or multiple willing paired donors), and at certain times the KEP performs a matching run, determining an optimal set of exchanges to perform such that a donor donates a kidney if their paired recipient receives a kidney (and at most one donor paired with any recipient donates a kidney). These exchanges involving recipients with paired donors form cycles: the length of these cycles is often limited for logistical reasons, but limiting this maximum length can reduce the efficiency of a KEP. Additionally, individuals can join a KEP to donate a kidney without expecting any reciprocal donation: such individuals are referred to as non-directed (or altruistic) donors, and donors paired with a recipient are called directed donors. Non-directed donors can initiate a chain of transplants, by donating to a recipient whose donor can then donate to a further recipient and so-on. We will use the term exchange to refer to either a cycle or a chain.

[*]`william.pettersson@glasgow.ac.uk`, Corresponding author

Identifying the potential transplants is a vital part of the operation of a KEP, and numerous operations research techniques and algorithms have been devised for finding such exchanges. The problem of finding a largest set of transplants is polynomial-time solvable in very restricted settings (such as limiting cycle and chain lengths to 2), but in general is NP-hard [1], and so integer programming (IP) is often used to find solutions. The first IP models used one variable for either each potential cycle [29] or each edge [1], but there have been numerous substantial improvements in this area, including compact formulations as well as column generation, branch-and-price, branch-and-price-and-cut, and failure-aware modelling [2, 3, 12, 13, 14, 21, 23, 25]. A summary of models and techniques currently in use by real-world KEPs is also available [9].

In this paper we use the UK Living Kidney Sharing Scheme (UKLKSS) as our real-world KEP example [23]. The UKLKSS is run by NHS Blood and Transplant, a special health authority of the National Health Service (NHS) within UK. The UKLKSS has been operating since 2007, and is currently the largest KEP (by both pool size and transplant count) in Europe [8].

We use the term *Kidney Exchange problem* (KE) to refer to the problem of finding a set of cycles and chains that yield the maximum number of transplants, given a dataset containing information about donors, recipients and compatibilities between them. The effectiveness of the techniques used to find optimal sets of exchanges is often compared using randomly-generated instances that are created to mimic real-world instances. Such random instances are also useful for policy-makers, as they allow changes to policies (such as prioritisation of highly-sensitised patients, or the introduction of a wider range of exchange structures) to be tested. In the literature, a generator due to Saidman et al., henceforth referred to as the *Saidman generator*, is commonly used to generate such instances [30]. The Saidman generator draws recipients and donors with various properties (such as blood-groups) based on the random distribution of such parameters in populations. A more thorough introduction to the Saidman generator is given in Section 5.2. We also highlight that PrefLib [24], an online repository of matching problems with preferences, includes a set of instances that have been generated using the Saidman generator [14]. We will refer to these as the PrefLib instances. The best known result in the literature for these particular instances has been demonstrated by Lam and Mak-Hau [21], who use a branch-and-price-and-cut algorithm to solve all 30 non-trivial instances when cycles are limited to length 3, and 19 of 30 non-trivial instances when cycles are limited to length 4. However, these PrefLib instances differ from real-world instances in a range of parameters (such as edge density), meaning that algorithms that can solve the PrefLib instances quickly may not be as applicable to real-world problems and vice-versa.

1.2 Related work

Many different approaches to improving the performance of KEPs have been investigated in the literature. We outline here two particular developments in the field, and highlight that for both of these, and indeed for almost any improvement to KEPs, the simulation of any proposed changes depends on the use of suitable random or benchmark instances for a baseline comparison.

The creation of transnational KEPs, either as new programmes or through coordination between or amalgamation of existing national programmes, allows KEPs to share their pools, thus increasing their overall effectiveness through more efficient allocation of resources. However, collaboration between countries when human organs are involved is a sensitive issue, both ethically and legally. Individual rationality has been proposed as one requirement for such collaboration [4, 5], and further research has suggested equality measures such as the Shapley value [7] as well as IP models that allow for, and have simulated, different constraints within different countries [26]. We note that KEPs within some countries (such as the USA) may exhibit similar characteristics without being transnational, as transplant centres (or groups of transplant centres) either collaborate or compete for kidney transplants [22].

Another important development in the field of kidney exchange is the introduction of robust, or fault-tolerant, KEPs. A potential transplant may be selected by an algorithm or policy for transplantation, but may fail due to any one of a number of reasons (e.g., donor or recipient illness, unexpected incompatibilities). Robust KEP algorithms aim to assign a likelihood of progression (alternatively, a likelihood of failure) to each potential transplant, and then select a set of exchanges that maximises the expected number of transplants. The difficulty of determining the likelihood of whether a potential transplant will proceed in practice is well known, and depends on cPRA (calculated Panel Reactive Antibodies, a score between zero and one hundred assigned to a recipient indicating the likelihood that a random blood-group compatible donor will be tissue-type incompatible, see Section 2.1 for more background), but cannot be explained by cPRA alone [16]. Section 5 in this paper provides a number of new insights into this relationship between cPRA and transplant compatibility. Recent models for KEPs have assigned a success probability to each potential transplant, and used these to find a set of exchanges that maximises the number of expected transplants [14], and further work has also included various recourse options that model fall-back strategies which may be used when transplants cannot proceed [11, 19].

1.3 Our contribution

We develop three upper bounds for the potential number of transplants in KE instances by utilising the available blood-group information of donors and recipients. We introduce a new ad-hoc matheuristic that aims to find a solution whose value matches one of the upper bounds. This is possible for every PrefLib instance, which allows us to empirically observe that our matheuristic is signicantly faster than known techniques. However for real-world instances used in our experiments, the solution found by the matheuristic does not attain the upper bound, meaning that it does not provide a guarantee of optimality, as is usually required in a KEP.

We then study the reasons why real-world instances differ from the PrefLib instances (and more generally, from any instance created by a Saidman generator) highlighting distinctions relative to a number of relevant parameters including edge density and maximum number of potential transplants. We then devise new methods for generating random KE instances, based on the Saidman generator but incorporating improvements that can be justified both statistically and by real-world scenarios. These improvements all involve the calculated Panel Reactive Antibody value (cPRA) of a recipient, which estimates what proportion of donors in the KEP will be tissue-type incompatible with that recipient. The improvements arise from examining:

- the distribution of cPRA amongst recipients when grouping recipients by their blood-group and the blood-group of their donor,
- the relationship between cPRA and transplant compatibility, and
- the transplant compatibility of recipients with a cPRA of zero.

We show that our improved generator creates random KE instances that more closely match real-world instances from the UK than the traditional Saidman generator across a wide variety of relevant parameters.

The paper is organised as follows. Section 2 gives a background on and defines terminology and notation used in kidney exchange programmes. Section 3 details our new upper bound methods for the PrefLib instances, while Section 4 introduces our new matheuristic and demonstrates its computational performance on the PrefLib instances. Section 5 then introduces our new techniques for generating random KE instances, and we close with a conclusion in Section 6.

2 Background

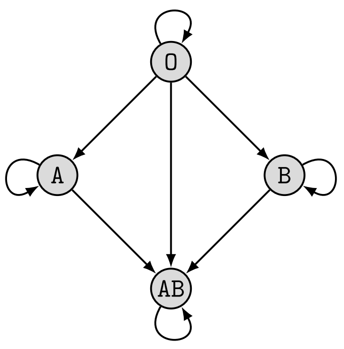
We present background on compatibility in KEPs in Section 2.1, and then in Section 2.2 we introduce representations of KE instances as well as terminology used to describe particular structures within such instances.

2.1 Compatibility

The compatibility of a transplant between a donor and a recipient is determined by a number of different factors. We will describe two in detail as they are relevant to our work, and briefly mention some others.

Possibly the most well-known factor is blood-group (or `ABO`) compatibility. A donor with blood-group `O` (also called universal donor) can potentially give their kidney to a recipient with any blood-group. A donor with blood-group `A` (respectively `B`) can potentially give their kidney to a recipient with blood-group `A` or `AB` (respectively `B` or `AB`). An `AB` donor can only give their kidney to an `AB` recipient (also called universal receiver). This compatibility is often presented as a graph such as Figure 1

Figure 1: Blood-group compatibility



Another factor for compatibility is tissue-type compatibility, determined by the presence of human leukocyte antigens (HLA) in a donor and potential presence of HLA antibodies in a recipient. As such, this is also known as HLA compatibility. Tissue-type compatibility is best determined with a lab-based crossmatch test, in which samples from a donor and recipient are combined and examined for reactions [27]. However, such tests are expensive and time-consuming, so it is often not feasible to test every single donor against every single recipient. Instead, some alternatives have been developed. A Panel Reactive Antibody (PRA) test determines a recipient's PRA value by performing crossmatch tests against a given panel of samples. The resulting PRA value of a recipient gives the proportion of donors that a recipient is likely to be incompatible with as a percentage (e.g., a recipient with a PRA of 0 is likely to be tissue-type compatible with all donors, while a recipient with a PRA of 95 is likely to be tissue-type compatible with only 5% of donors). A better understanding of HLA incompatibility has allowed for the testing of the presence or absence of individual antigens or antibodies, resulting in the calculated PRA (cPRA) parameter. This is calculated based on not only the presence of antigens in recipients and antibodies in donors, but also a better understanding of which specific antigen and antibody combinations are more likely to result in a failed transplant. Again, the actual cPRA value for a recipient aims to represent the proportion of donors with which the recipient will not be compatible with.

Many other factors influence compatibility, including but not limited to, the ages and general health of the donor and the recipient, the size of the donor kidney, and even potentially the number of veins or arteries present on the donor kidney.

2.2 Notation and terminology

We formally define a Kidney Exchange problem (KE) instance as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ (also called compatibility graph) in which vertex set $\mathcal{V}$ contains every recipient and arc set $\mathcal{A} = \{(i,j) : i,j \in \mathcal{V}\}$ contains the potential transplants: (i,j) belongs to $\mathcal{A}$ if the donor associated to recipient i is compatible with recipient j. Note that compatible recipient/donor pairs (represented by an arc $(i,i) \in \mathcal{A}$) are allowed in some KEPs. A feasible *exchange* is a cycle in the directed graph: a sequence of distinct vertices $(v_1, \ldots, v_k)$ such that there is an arc (v_i, v_{i+1}) for each $i \in \{1, \ldots, k-1\}$, and an arc (v_k, v_1). In an exchange, a donor paired with a recipient only donates a kidney if their paired recipient receives a kidney. For logistical purposes, there is often an upper bound on how many vertices can be part of any exchange (named L in the following).

Some KEPs also include non-directed donors (sometimes referred to as altruistic donors) who are willing to donate a kidney without being paired with a recipient. In the compatibility graph, non-directed donors are represented by a node in $\mathcal{V}$ with no incoming arcs and one outgoing arc to each of the recipients they are compatible with. In the presence of non-directed donors, an exchange can also be *a chain*, which is a sequence of distinct vertices $(v_1, \ldots, v_k)$ where v_1 corresponds to a non-directed donor, and for each $i \in \{1, \ldots, k-1\}$ there is an arc (v_i, v_{i+1}). Usually, the donor associated to recipient v_k gives a kidney to a deceased-donor waiting list.

Some KEPs also allow a recipient to be paired with multiple donors. In Sections 3 and 4, we focus on the PrefLib instances, in which: (i) every recipient is paired with exactly one donor and every donor is paired with exactly one recipient, (ii) there are no compatible recipient/donor pairs (i.e., no arcs

$(i,i) \in \mathcal{A}$), and (iii) there are no non-directed donors. In Section 5 we explicitly allow recipients to be paired with multiple donors, which more closely matches the real-world instances from the UKLKSS.

3 Upper bounds for PrefLib instances

In this section, we describe three upper bounding procedures for KE instances with no non-directed donors. The first bound U_1, which is valid for any instance, is derived from a relaxation of the cycle size limit constraints and can be computed in polynomial time. The two other bounds U_2 and U_3 use the donor and recipient blood-groups to identify a subset of recipient/donor pairs that cannot participate in a maximum-size exchange, and perform particularly well on PrefLib instances.

3.1 Unbounded cycle lengths upper bound

It is well known that determining the maximum number of transplants in a kidney exchange instance for general values of L is $\mathcal{NP}$-hard. Abraham et al. [1] outlined two cases where the problem can be solved in polynomial time:

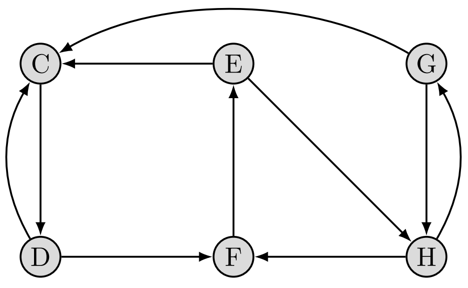
- When $L = 2$, the compatibility graph can be transformed into an undirected graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ in which edge $\{i,j\}$ belongs to $\mathcal{E}'$ if and only if both arcs (i,j) and (j,i) belong to $\mathcal{A}$ in the original graph. The problem now is to find a matching of maximum size and this can be done in polynomial time using Edmonds' algorithm [15].

- When $L = \infty$, the compatibility graph can be transformed into a bipartite graph $\mathcal{G}'' = (\{\mathcal{L}'', \mathcal{R}''\}, \mathcal{E}'')$ where $\mathcal{L}''$ is the set of donors and $\mathcal{R}''$ is the set of recipients. There is an edge $\{i,j\}$ in $\mathcal{E}''$ with cost 1 if donor i is compatible with recipient j and an edge $\{i,j\}$ in $\mathcal{E}''$ with cost 0 if i and j are in the same recipient/donor pair. The problem now is to find a perfect matching of maximum weight, and again, this can be done in polynomial time using the Hungarian algorithm [20].

For a given kidney exchange instance with cycle size limit L, the maximum number of transplants cannot decrease as L increases. As a result, solving the problem with $L = \infty$ gives an upper bound on the maximum number of pairs selected in an exchange for any value of $L \in \mathbb{Z}$. We call this upper bound U_1.

Example 1. *Let us consider the following kidney exchange instance with 6 recipients and the following compatibility graph:*

Figure 2: Compatibility graph for Example 1



- *If $L \geq 6$, an optimal solution will match the six recipient/donor pairs in either one or two cycles: $\{[G,C,D,F,E,H]\}$ or $\{[C,D,F,E],\ [H,G]\}$;*

- *If $L = 4$ or 5, an optimal solution will match the six recipient/donor pairs in two cycles: $[C,D,F,E]$ and $[H,G]$;*

- *If $L = 3$, an optimal solution will match five recipient/donor pairs in two cycles: $[E,H,F]$ and $[D,C]$;*

- *If $L = 2$, an optimal solution will match four recipient/donor pairs in two cycles: $[H,G]$ and $[D,C]$.*

3.2 Data-driven upper bound

Our second upper bound U_2 uses blood-group compatibility (see Section 2) to calculate the maximum number of transplants that can possibly take place. Note that blood-group compatibility is a necessary but not sufficient condition for transplant compatibility. In the following, we call:

- A `*/*` pair a recipient/donor pair in which the recipient and the donor have any blood group, and we denote by n the number of `*/*` pairs.
- An `X/*` pair a recipient/donor pair in which the recipient has blood-group `X` and the donor has any blood group, and we denote by $n_{\texttt{X/*}}$ the number of `X/*` pairs.
- A `*/Y` pair a recipient/donor pair in which the recipient has any blood-group, and the donor has blood-group `Y` and we denote by $n_{\texttt{*/Y}}$ the number of `*/Y` pairs.
- An `X/Y` pair a recipient/donor pair in which the recipient has blood-group `X` and the donor has blood group `Y`, and we denote by $n_{\texttt{X/Y}}$ the number of `X/Y` pairs.

Upper bound U_2 makes the initial assumption that each of the recipient/donor pairs will participate in the exchange unless stated otherwise. It then uses the two following rules to adjust its value:

- Since only cycles are allowed, if an `O/*` pair p cannot be matched with any `*/O` pair, then p cannot participate in the exchange.
- Similarly, if a `*/AB` pair p cannot be matched with any `AB/*` pair then p cannot participate in the exchange.

Even though it might be difficult to detect with precision which `O/*` and which `*/AB` pairs will be excluded from the matching, we know that if $n_{\texttt{O/*}} > n_{\texttt{*/O}}$ (i.e., if there are more `O/*` than `*/O` pairs), then at least $n_{\texttt{O/*}} - n_{\texttt{*/O}}$ pairs cannot participate in the exchange. Similarly, we know that if $n_{\texttt{*/AB}} > n_{\texttt{AB/*}}$, then at least $n_{\texttt{*/AB}} - n_{\texttt{AB/*}}$ pairs cannot participate in the exchange. If we now define $\ell_1 = \max\{0, n_{\texttt{O/*}} - n_{\texttt{*/O}}\}$ as the minimum number of `O/*` pairs that cannot participate in the exchange and $\ell_2 = \max\{0, n_{\texttt{*/AB}} - n_{\texttt{AB/*}}\}$ as the minimum number of `*/AB` pairs that cannot participate in the exchange, then we can formally introduce the bound $U_2 = n - \ell_1 - \ell_2 + \min\{n_{\texttt{O/AB}}, \ell_1, \ell_2\}$, where the last term identifies the number of `O/AB` pairs that were counted both in ℓ_1 and in ℓ_2. Note that a solution with value U_2 is only reachable if the following assumptions are met: (i) if $\ell_1 > 0$ (which is the case for every PrefLib instance), then every `O` donor is matched with an `O` recipient (otherwise more than ℓ_1 pairs of the form `O/*` would not be able to participate in the exchange), and (ii) if $n_{\texttt{O/AB}}$ is less than or equal to both ℓ_1 and ℓ_2 (which is also the case for every PrefLib instance), then no `O/AB` pairs should participate in the exchange (otherwise, such a pair should be counted both in ℓ_1 and in ℓ_2). These assumptions will be used henceforth.

Example 2. *Let us consider the kidney exchange instance from the PrefLib repository "MD-00001-00000191", an instance with $n = 512$ and the following blood-group distribution:*

Table 1: Blood-group distribution for Example 2

Rec. / Don.	./O	./A	./B	./AB	./*
O/.	66	134	70	13	283
A/.	51	34	59	11	155
B/.	10	53	4	1	68
AB/.	3	1	1	1	6
*/.	130	222	134	26	512

Out of the 512 recipient/donor pairs, we know that 153 (ℓ_1 = 283 - 130) `O/*` pairs cannot participate in the matching because there are not enough `*/O` pairs to accommodate them. We also know that 20 (ℓ_2 = 26 - 6) `*/AB` pairs cannot participate in the matching because there are not enough `AB/*` pairs to accommodate them. Out of the 173 removed pairs, we know that up to 13 `O/AB` pairs could have potentially been counted twice (i.e., both in ℓ_1 and ℓ_2). This gives an upper bound U_1 equal to 352 (512 - 153 - 20 + 13). An optimal solution to this instance has size 351 if $L = 3$, and size 352 if $L \geq 4$, showing that, for this particular instance, the bound is very close to the optimal solution value.

3.3 ILP-based upper bound

After some preliminary analysis, we found that the difference observed between upper bounds U_2 and the optimal solutions of the tested PrefLib instances always came from some `*/AB` pairs that we assumed could be part of the exchange while in reality they could not (in other words, we underestimated ℓ_2). This can be explained by the fact that `AB` is the least common blood-group, so there are fewer opportunities to match `*/AB` pairs with `AB/*` pairs. In order to correct the wrong assumptions, we designed a tailored ILP model that maximises the number of `X/AB` pairs that can possibly be included in the exchange (`X` $\in$ {`A`, `B`, `AB`}) to determine a refined upper bound U_3. Note that the other assumptions derived from U_2, namely (i) that any `*/O` pair must be followed by an `O/*` pair in the exchange, and (ii) that no `O/AB` pairs should be included in the exchange, still hold. Every time one of these two assumptions is violated, the exchange size is reduced by one unit. In the following, we first describe the ILP model details for $L = 3$ and then we explain how to modify it for $L = 4$. The model needs a list of every cycle solely composed of `*/AB` pairs. We distinguish:

- C_2^2, the set of 2-cycles with two `*/AB` pairs – these can only be in the form [`AB/AB - AB/AB`].
- C_3^3, the set of 3-cycles with three `*/AB` pairs – these can only be in the form [`AB/AB - AB/AB - AB/AB`].

For instance, if all the pairs in Example 1 were `AB/AB` pairs, then there would be two cycles in C_2^2: $[C, D]$ and $[G, H]$, and one cycle in C_3^3 : $[E, H, F]$. The model also needs a list of every cycle partly composed of `*/AB` pairs. We distinguish:

- P_2^1, is the set of *partial* cycles comprising two pairs, among which exactly one is a `*/AB` pair – these are in the form [`X/AB - AB/Y`], where `X` $\in$ {`A, B, AB`} and `Y` $\in$ {`O, A, B`}. These cycles do not necessarily need to be complete, so we make sure that either the donor of the second pair is compatible with the recipient of the first pair, or that there exists at least one additional pair that can complete the cycle;
- C_3^2, the set of 3-cycles with exactly two `*/AB` pairs – these are in the form [`X/AB - AB/AB - AB/Y`], where `X` $\in$ {`A, B, AB`} and `Y` $\in$ {`A, B`};

By introducing binary decision variables x_c taking value 1 if cycle c is selected and 0 otherwise ($c \in \mathcal{C}$, where $\mathcal{C} = \{C_2^2 \cup C_3^3 \cup P_2^1 \cup C_3^2\}$), the maximum number of `X/AB` pairs that can possibly be included in the exchange (`X` $\in$ {`A, B, AB`}) denoted as z can be obtained through the following ILP model:

$$\max \quad z = \sum_{c \in C_2^2} 2x_c + \sum_{c \in C_3^3} 3x_c + \sum_{c \in P_2^1} x_c + \sum_{c \in C_3^2} 2x_c \tag{1}$$

$$\text{s.t.} \quad \sum_{c \in \mathcal{C}: i \in V(c)} x_c \leq 1, \qquad \forall i \in \mathcal{V}, \tag{2}$$

$$x_c \in \{0, 1\}, \qquad c \in \mathcal{C}, \tag{3}$$

where $V(c)$ indicates the set of recipient/donor pairs that belong to cycle c. Upper bound U_3 can be defined as $U_3 = n - \ell_1 - \ell_2'$ where $\ell_2' = (n_{\texttt{*/AB}} - z)$. Note that it is not necessary to consider cycles [`O/AB - AB/Y`] in P_2^1 because selecting such cycle would increase z (and thus, U_3) by one unit at best, but would also break one of our assumptions, decreasing by one unit the exchange size. Similarly, it is not necessary to consider cycles [`O/AB - AB/AB - AB/O`] in C_3^2 because selecting such cycle would only increase U_3 by one unit (two units in z with a one unit penalty due to the broken assumption), which is as effective as selecting the truncated [`AB/AB - AB/O`] cycle that already exists in P_2^1.

The model can be extended to take into account a cycle size limit of four. In this case, we need:

- An additional set of cycles C_4^4 with four `*/AB` pairs – these can only be in the form [`AB/AB - AB/AB - AB/AB - AB/AB`].
- An updated set of partial cycles P_2^1 with two pairs having exactly one `*/AB` pair in which we additionally make sure that either the donor of the second pair is compatible with the recipient of the first pair, or that there exists at least one pair or a couple of pairs that can complete the cycle.

- An updated set of partial cycles P_3^2 with three pairs having exactly two `*/AB` pair – these are in the form [`X/AB - AB/AB - AB/Y`] where `X` $\in$ {`A`, `B`, `AB`} and `Y` $\in$ {`O`, `A`, `B`}. These cycles do not necessarily need to be complete, so we make sure that either the donor of the third pair is compatible with the recipient of the first pair, or that there exists at least one pair that can complete the cycle.

- An additional set of 4-cycles C_4^3 with exactly three `*/AB` pairs [`X/AB - AB/AB - AB/AB - AB/Y`], where `X` $\in$ {`A`, `B`, `AB`} and `Y` $\in$ {`A`, `B`}.

3.4 Bound performance on PrefLib instances

We tested each of the bounds on a set of 30 PrefLib instances without non-directed donors, which is the subset that was used by Lam and Mak-Hau [21] for testing their branch-and-price-and-cut algorithm. The results are available in Table 2. The “Instance” column gives the name of the instance, column n gives the number of recipient/donor pairs in the instance, the four following columns gives the upper bounds and optimal value for $L = 3$, and the last four columns give the same information for $L = 4$. The optimal values were either obtained from [21] or were deduced from a lower bound obtained in [21] combined with one of the upper bounds we introduced.

Table 2: U_1, U_2, and U_3 values for PrefLib instances with $L = 3$ and $L = 4$

Instance	n	$L = 3$				$L = 4$			
		U_1	U_2	U_3	OPT	U_1	U_2	U_3	OPT
MD-00001-00000191	512	352	352	**351**	351	**352**	**352**	**352**	352
MD-00001-00000192	512	**337**	**337**	**337**	337	**337**	**337**	**337**	337
MD-00001-00000193	512	**300**	301	**300**	300	**300**	301	**300**	300
MD-00001-00000194	512	**313**	**313**	**313**	313	**313**	**313**	**313**	313
MD-00001-00000195	512	**322**	**322**	**322**	322	**322**	**322**	**322**	322
MD-00001-00000196	512	**313**	**313**	**313**	313	**313**	**313**	**313**	313
MD-00001-00000197	512	**334**	337	**334**	334	**334**	337	**334**	334
MD-00001-00000198	512	**332**	333	**332**	332	**332**	333	**332**	332
MD-00001-00000199	512	**313**	314	**313**	313	**313**	314	**313**	313
MD-00001-00000200	512	**312**	313	**312**	312	**312**	313	**312**	312
MD-00001-00000231	1024	659	659	**658**	658	**659**	**659**	**659**	659
MD-00001-00000232	1024	**662**	**662**	**662**	662	**662**	**662**	**662**	662
MD-00001-00000233	1024	**635**	636	**635**	635	**635**	636	**635**	635
MD-00001-00000234	1024	**641**	642	**641**	641	**641**	642	**641**	641
MD-00001-00000235	1024	**660**	**660**	**660**	660	**660**	**660**	**660**	660
MD-00001-00000236	1024	**655**	**655**	**655**	655	**655**	**655**	**655**	655
MD-00001-00000237	1024	**597**	**597**	**597**	597	**597**	**597**	**597**	597
MD-00001-00000238	1024	672	672	**671**	671	**672**	**672**	**672**	672
MD-00001-00000239	1024	**673**	674	**673**	673	**673**	674	**673**	673
MD-00001-00000240	1024	**639**	**639**	**639**	639	**639**	**639**	**639**	639
MD-00001-00000271	2048	**1284**	**1284**	**1284**	1284	**1284**	**1284**	**1284**	1284
MD-00001-00000272	2048	**1249**	**1249**	**1249**	1249	**1249**	**1249**	**1249**	1249
MD-00001-00000273	2048	**1232**	**1232**	**1232**	1232	**1232**	**1232**	**1232**	1232
MD-00001-00000274	2048	**1242**	**1242**	**1242**	1242	**1242**	**1242**	**1242**	1242
MD-00001-00000275	2048	**1301**	**1301**	**1301**	1301	**1301**	**1301**	**1301**	1301
MD-00001-00000276	2048	**1311**	**1311**	**1311**	1311	**1311**	**1311**	**1311**	1311
MD-00001-00000277	2048	**1316**	**1316**	**1316**	1316	**1316**	**1316**	**1316**	1316
MD-00001-00000278	2048	**1268**	**1268**	**1268**	1268	**1268**	**1268**	**1268**	1268
MD-00001-00000279	2048	**1271**	1272	**1271**	1271	**1271**	1272	**1271**	1271
MD-00001-00000280	2048	**1295**	**1295**	**1295**	1295	**1295**	**1295**	**1295**	1295

We observe that bound U_1 displays a very good quality, as it is equal to the optimal solution in 27 instances out of 30 for $L = 3$ and on all instances for $L = 4$. In fact, by simply using U_1 with the solution obtained by the algorithm of Lam and Mak-Hau [21] for $L = 3$, we could solve the 11 benchmark instances with $L = 4$ that they left open in their paper. Upper bound U_2 is also very promising, finding the optimal objective value in 39 instances out of 60. It is not surprising that the values of U_1 and U_2 do not change when the cycle size limit increases as L is not taken into account when the bounds are computed. We also notice that upper bound U_3 displays outstanding performance on the PrefLib instances as it is always equal to the optimal solution value both for $L = 3$ and $L = 4$. It is worth mentioning that each bound was calculated in less than one second of computing time. The experiments were run on an Intel Xeon E5-2680W v3, 2.50GHz with 192GB of memory, running under Scientific Linux 7.5, and using Gurobi 7.5.2 to solve the ILP models.

4 Matheuristic for solving PrefLib instances

Lam and Mak-Hau [21] observed that the cycle formulation obtained poor results on PrefLib instances. This can be explained by the very large number of cycles that have to be generated to run the model due to the cycle size limit ($L = 3$ or 4), the density of the compatibility graph (close to 0.25), and the large number of recipient/donor pairs (from 512 to 2048).

In this section, we introduce a 7-step matheuristic specifically tailored for solving PrefLib instances that takes advantage of the high density in the compatibility graph and of the large number of recipient/donor pairs. When used with upper bound U_3 presented in the previous section, it is able to solve all the tested instances in a few seconds of computing time, which is much faster than the algorithm of Lam and Mak-Hau [21].

Matheuristic is a neologism originating from the contraction of “math” and “heuristic” and is often used as an alternative to mathematical models when the latter become too large to solve difficult combinatorial optimisation problems. Matheuristics usually obtain good quality solutions but do not have a proof of optimality. A common approach in matheuristics is to solve a mathematical model with a reduced set of variables once or several times in a row. In the following, we first describe our matheuristic for $L = 3$ and then we explain how to modify it for $L = 4$.

4.1 Matheuristic overview

As we observed that our best upper bound was tight on the PrefLib instances, the matheuristic aims to find a solution with value U_3 exactly. To do so, it keeps the same assumptions form earlier, namely that every pair participates in the exchange with the exception of ℓ_1 `O/*` pairs and ℓ'_2 `*/AB` pairs. It also assumes that no `O/AB` pairs should be included in the exchange and that every `O` donor should be matched with an `O` recipient. We provide in Algorithm 1 an overview of our approach.

Algorithm 1 `7-step matheuristic`

```
1: Identify the types of cycles and recipient/donor pairs used in the exchange and process the AB/* and */AB pairs
2: Process the B/B pairs
3: Process the A/B and B/A pairs
4: Process the A/A pairs
5: Process the O/O pairs
6: Process the B/O and O/B pairs
7: Process the A/O and O/A pairs
```

The matheuristic starts by determining the types of cycles that should be used if a solution of value U_3 (hence optimal) exists (e.g., x_1 cycles of the form [`A/B`−`B/A`], x_2 cycles of the form [`A/B`−`B/O`−`O/A`], . . .). Then, for each combination of recipient/donor pairs, it uses the compatibility graph to enumerate every feasible cycle or partial cycle of the given types and uses an ILP formulation to select those that are necessary to reach the desired solution. At each step, we define:

- An *objective*, which is the goal of the ILP model that is solved at the current step.
- A list of *cycles to complete*, which contains the cycles and incomplete cycles determined at a previous step that need to be completed at the current step if we want to reach a solution of value U_3 because it is the last opportunity to do so.
- A list of *pairs to use*, which contains the pairs that need to be assigned to a cycle at the current step if we want to reach a solution of value U_3 because it is the last opportunity to do so.
- a list of *types of cycles allowed*, which contains the types of cycles that are enumerated in the ILP formulation at the current step;
- A list of *types of incomplete cycles allowed*, which contains the types of incomplete cycles (i.e., with no compatibility requirements between the donor of the second pair and the recipient of the first pair) that are enumerated in the ILP formulation at the current step.

The model is inspired by the cycle formulation (see [29]) and associates a binary variable to every cycle and every incomplete cycle allowed. It maximises the defined “objective” under the constraints that every "cycle to complete" must be completed and every "pair to use" must be assigned to a cycle. Note that the cycles selected at a given step are assumed to be fixed at each subsequent step while the partial cycles selected at a given step must be completed (but cannot be broken) in a subsequent step.

4.2 Matheuristic steps

Step 1: Identifying the types of cycles in the exchange

The first step consists of finding how to split the 16 kinds of recipient/donor pairs (4 × 4, one for each possible blood-group combination) into cycle types that our matheuristic will seek in subsequent steps in order to obtain a solution with value U_3.

Out of these 16 kinds of pairs, we process the `AB/*` and `*/AB` pairs first as these are critical to finding a solution with the desired value. Similarly to what was done for U_3, we enumerate every cycle (complete or incomplete) containing a `*/AB` pair (i.e., we reuse set $\mathcal{C} = C_2^2 \cup C_3^3 \cup P_2^1 \cup C_3^2$). We also define $P_2^{1,\texttt{X/Y}}$ ($\texttt{X} \in \{\texttt{O,A,B}\}$, $\texttt{Y} \in \{\texttt{A,B}\}$), a subset of P_2^1 containing every partial cycle [`Y/AB` − `AB/X`], meaning that an `X/Y` pair might be needed at a later stage to complete the cycle. For practical reasons, cycles [`AB/AB` − `AB/X`] are included in the set $P_2^{1,\texttt{X/A}}$, as there are significantly more `A` donors than `A` recipients in the PrefLib instances. We also add the possibility to "transform" any `AB/X` pair ($\texttt{X} \in \{\texttt{O,A,B}\}$) into a `A/X` pair or a `B/X` pair to allow it to participate in the exchange even if it is incompatible with every `*/AB` pair. The `AB/X` pairs are gathered in $\mathcal{V}_{\texttt{AB/X}}$ and the `X/AB` pairs are gathered in $\mathcal{V}_{\texttt{X/AB}}$, forming all together set $\mathcal{V}$.

Out of the 9 remaining kinds of pairs, we remove the `O/O`, `A/A`, and `B/B` combinations, as we assume they will always participate in the exchange, either to complete a partial cycle, or to extend a 2-cycle to a 3-cycle (e.g., an `O/O` pair used in the middle of an [`A/O` − `O/A`] or a [`B/O` − `O/B`] cycle), or to form a cycle with other pairs of the same type. The remaining 6 combinations are spread among the following 5 types of cycles:

- [`A/B` − `B/A`];
- [`A/O` − `O/A`];
- [`B/O` − `O/B`];
- [`A/B` − `B/O` − `O/A`];
- [`B/A` − `A/O` − `O/B`].

By using binary decision variables x_c taking value 1 if cycle c is selected and 0 otherwise ($c \in \mathcal{C}$), five integer variables $x_{\texttt{ABBA}}, x_{\texttt{AOOA}}, x_{\texttt{BOOB}}, x_{\texttt{ABBOOA}}, x_{\texttt{BAAOOB}}$ indicating the number of cycles [`A/B` − `B/A`], [`A/O` − `O/A`], [`B/O` − `O/B`], [`A/B` − `B/O` − `O/A`], [`B/A` − `A/O` − `O/B`] selected in the exchange, respectively, binary variable $t_i^{\texttt{AB}\to\texttt{X}}$ ($\texttt{X} \in \{\texttt{A,B}\}, \texttt{Y} \in \{\texttt{O,A,B}\}, i \in \mathcal{V}_{\texttt{AB/Y}}$) taking value 1 if `AB/Y` pair i is transformed into an `X/Y` pair, the objective function of the ILP model maximising the number of exchanges is defined as follows:

$$\max z_1 = \sum_{c\in\mathcal{C}} s_c x_c + 2(x_{\texttt{ABBA}} + x_{\texttt{AOOA}} + x_{\texttt{BOOB}}) + 3(x_{\texttt{ABBOOA}} + x_{\texttt{BAAOOB}}) + \sum_{\texttt{X}\in\{\texttt{O,A,B}\}} n_{\texttt{X/X}} + \sum_{\substack{i\in\mathcal{V}_{\texttt{AB/X}}\\ \texttt{X}\in\{\texttt{A,B}\}}} t_i^{\texttt{AB}\to\texttt{X}}, \tag{4}$$

where s_c indicates the size of cycle c, which is 2 if $c \in C_2^2$, 2 if $c \in P_2^{1,\texttt{A/A}} \cup P_2^{1,\texttt{B/B}}$ (even if these cycles are not necessarily complete, every `A/A` and `B/B` pair is counted in the objective function in the second summation), 3 if $c \in P_2^{1,\texttt{A/B}} \cup P_2^{1,\texttt{B/A}} \cup P_2^{1,\texttt{O/A}} \cup P_2^{1,\texttt{O/B}}$ (these cycles need to be completed by a third pair), and 3 if $c \in C_3^3 \cup C_3^2$. The last summation adds to the objective function the `AB/A` pairs transformed to `A/A` and the `AB/B` pairs transformed to `B/B` (we recall that `AB/O` pairs cannot be transformed into `O/O` pairs as we do not have enough `O` donors to satisfy the current `O` recipients in the PrefLib instances).

The six capacity constraints related to each of the six types of pairs of the form `A/B`, `B/A`, `A/O`, `O/A`, B/O, O/B are as follows:

$$x_{\texttt{ABBA}} + x_{\texttt{ABBOOA}} + \sum_{c \in P_2^{1,\texttt{A/B}}} x_c \leq n_{\texttt{A/B}} + \sum_{i \in \mathcal{V}_{\texttt{AB/B}}} t_i^{\texttt{AB}\to\texttt{A}}, \tag{5}$$

$$x_{\texttt{ABBA}} + x_{\texttt{BAAOOB}} + \sum_{c \in P_2^{1,\texttt{B/A}}} x_c \leq n_{\texttt{B/A}} + \sum_{i \in \mathcal{V}_{\texttt{AB/A}}} t_i^{\texttt{AB}\to\texttt{B}}, \tag{6}$$

$$x_{\texttt{AOOA}} + x_{\texttt{BAAOOB}} \leq n_{\texttt{A/O}} + \sum_{i \in \mathcal{V}_{\texttt{AB/O}}} t_i^{\texttt{AB}\to\texttt{A}}, \tag{7}$$

$$x_{\texttt{AOOA}} + x_{\texttt{ABBOOA}} + \sum_{c \in P_2^{1,\texttt{O/A}}} x_c \leq n_{\texttt{O/A}}, \tag{8}$$

$$x_{\texttt{BOOB}} + x_{\texttt{ABBOOA}} \leq n_{\texttt{B/O}} + \sum_{i \in \mathcal{V}_{\texttt{AB/O}}} t_i^{\texttt{AB}\to\texttt{B}}, \tag{9}$$

$$x_{\texttt{BOOB}} + x_{\texttt{BAAOOB}} + \sum_{c \in P_2^{1,\texttt{O/B}}} x_c \leq n_{\texttt{O/B}}. \tag{10}$$

For each X/Y combination (X,Y $\in$ {O,A,B}, X $\neq$ Y), a dedicated constraint makes sure that the number of X/Y pairs used in the five types of cycles plus the number of X/Y pairs required to complete the cycles in P_2^1 is less than or equal to the total number of X/Y pairs available plus the number of AB/Y pairs transformed into X/Y pairs. Note that (i) sets $P_1^{\texttt{A/O}}$ and sets $P_1^{\texttt{B/O}}$ do not exist because the matheuristic assumes that no O/AB pairs should be included in the exchange, and (ii) variables $t_i^{\texttt{AB}\to\texttt{O}}$ do not exist because the matheuristic assumes that every O donor should be matched with an O recipient. The rest of the model is as follows:

$$\sum_{c \in \mathcal{C}: i \in V(c)} x_c + t_i^{\texttt{AB}\to\texttt{A}} + t_i^{\texttt{AB}\to\texttt{B}} \leq 1, \qquad \forall i \in \mathcal{V}_{\texttt{AB/O}} \cup \mathcal{V}_{\texttt{AB/A}} \cup \mathcal{V}_{\texttt{AB/B}}, \tag{11}$$

$$\sum_{c \in \mathcal{C}: i \in V(c)} x_c \leq 1, \qquad \forall i \in \mathcal{V}_{\texttt{A/AB}} \cup \mathcal{V}_{\texttt{B/AB}} \cup \mathcal{V}_{\texttt{AB/AB}}, \tag{12}$$

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

$$t_i^{\texttt{AB}\to\texttt{A}}, t_i^{\texttt{AB}\to\texttt{B}} \in \{0,1\}, \qquad \forall i \in \mathcal{V}_{\texttt{AB/O}} \cup \mathcal{V}_{\texttt{AB/A}} \cup \mathcal{V}_{\texttt{AB/B}}, \tag{14}$$

$$x_{\texttt{ABBA}}, x_{\texttt{AOOA}}, x_{\texttt{BOOB}}, x_{\texttt{ABBOOA}}, x_{\texttt{BAAOOB}} \in \mathbb{Z}^+, \tag{15}$$

where constraints (11) and (12) ensure that pairs AB/* and */AB can be used at most once. After running model (4)-(15) on instance "MD-00001-00000191" from the PrefLib repository, we obtain the following solution:

- [pair 9 - pair 307], a [B/AB - AB/O] cycle to be completed by an O/B pair,
- [pair 253 - pair 248], an [A/AB - AB/O] cycle to be completed by an O/A pair,
- [pair 258 - pair 259], an [AB/AB - AB/O] cycle to be completed by an O/A pair,
- [pair 306 - pair 91], an [A/AB - AB/A] cycle that may be completed by an A/A pair,
- [pair 356 - 244], an [A/AB - AB/B] cycle to be completed by a B/A pair,
- 52 × [A/B – B/A],
- 51 × [A/O – O/A],
- 3 × [B/O – O/B],
- 7 × [A/B – B/O – O/A],
- 0 × [B/A – A/O – O/B],
- 34 A/A, 4 B/B, and 66 O/O pairs,

- 0 transformation of AB/* pairs,

for a total of 351 pairs, which matches bound U_3 ($3+3+3+2+3+2\times 106+3\times 7+104=351$). The partial cycles involving */AB pairs will be carried over to subsequent steps and extended to full cycles. The remainder of the solution indicates the numbers of cycles of the given types that could be computed; attempts to construct the cycles themselves are made in subsequent steps. Note that the solution given by the model can possibly be adjusted to convert two cycles [A/B - B/O - O/A] and [B/A - A/O - O/B] into three cycles [A/B - B/A], [A/O - O/A], and [B/O - O/B] as 2-cycles are easier to handle. We underline that at this point, we are not sure that a solution with value 351 exists: the output of this step is a skeleton solution which, if it can be built up using the subsequent steps, will have value 351.

Step 2: Process the B/B pairs

In the second step, we deal with B/B pairs. We start by these pairs because B is the least common blood-group in the PrefLib instances apart from AB (already processed in Step 1). At this step:

- **Objective**: maximise the number of B/B pairs inserted in a cycle.
- **Cycles to complete:** [B/AB-AB/B] (fixed in Step 1).
- **Pairs to use:** none.
- **Types of cycles allowed:** [B/AB-AB/B], [B/AB-AB/B-B/B], [B/B-B/B], [B/B-B/B-B/B].

We identify the structures that were fixed in a previous step by underlining them. We try to find a suitable cycle for as many B/B pairs as possible under the constraint that every [B/AB-AB/B] cycle found in Step 1 should now be completed. This means that either the donor of the second pair is compatible with the recipient of the first pair, or that a B/B pair is needed to complete the cycle. After running Step 2 on instance "MD-00001-00000191", two B/B pairs could be matched into a [B/B-B/B] cycle, meaning that the other two become "pairs to use" in a subsequent step.

Step 3: Process the A/B and B/A pairs

In the third step, we deal with A/B and B/A pairs:

- **Objective:** maximise the number of [A/B-B/A] cycles
- **Cycles to complete:** [A/AB-AB/B], [B/AB-AB/A] (fixed in Step 1).
- **Pairs to use:** B/B (from Step 2).
- **Types of cycles allowed:** [A/AB-AB/B-B/A], [B/AB-AB/A-A/B], [A/B-B/A], [A/B-B/B-B/A].
- **Types of incomplete cycles allowed:**[A/B-B/O], [B/A-A/O], [A/B-B/A].

We try to find as many [A/B-B/A] cycles as expected under the constraint that some of these cycles should incorporate the "pairs to use" B/B that could not be matched in Step 2, that some pairs A/B and B/A should be used to complete the [B/AB-AB/A] and [A/AB-AB/B] cycles selected in Step 1, and that we have to create the partial cycles [A/B-B/O] (or [B/A-A/O]) that will be completed by O/A pairs at Steps 7 (or by O/B pairs at Step 6). It is also possible to create incomplete cycles [A/B-B/A] that will be completed by an A/A pair at Step 4. After running Step 3 on instance "MD-00001-00000191", cycle [356-244] was completed, the seven [A/B-B/O] partial cycles were found, and all 52 cycles containing one A/B pair and one B/A pair could be found: 50 [A/B-B/A] 2-cycles and 2 [A/B-B/B-B/A] 3-cycles, the latter containing the two remaining B/B pairs from Step 2.

Step 4: Process the A/A pairs

In the fourth step, we deal with A/A pairs. At this step:

- **Objective:** maximise the number of A/A pairs inserted in a cycle.
- **Cycles to complete:** [$\underline{\texttt{A/AB-AB/A}}$] (fixed in Step 1), [$\underline{\texttt{A/B-B/A}}$] (fixed in Step 3).
- **Pairs to use:** none.
- **Types of cycles allowed:** [$\underline{\texttt{A/AB-AB/A}}$], [$\underline{\texttt{A/AB-AB/A}}$-A/A] [$\underline{\texttt{B/A}}$-A/A-$\underline{\texttt{A/B}}$], [A/A-A/A].

We try to find a suitable cycle for as many A/A pairs as possible under the constraint that every cycle [A/AB-AB/A] found in Step 1 and cycle [A/B-B/A] found in Step 3 should now be completed. Note that we do not generate the cycles containing three A/A pairs because it makes the model significantly bigger. After running Step 4 on instance "MD-00001-00000191", 33 out of the 34 A/A pairs could be matched: 32 pairs into 16 [A/A-A/A] cycles, and one pair to complete cycle [306-91]. This means that the remaining A/A pair will be dealt with in a subsequent step.

Step 5: Process the O/O pairs

In the fifth step, we deal with O/O pairs. At this step:

- **Objective:** maximise the number of O/O pairs inserted in a cycle.
- **Cycles to complete:** none.
- **Pairs to use:** none.
- **Types of cycles allowed:** [O/O - O/O].

We try to find a suitable cycle for as many O/O pairs as possible. Note that we do not generate the cycles containing three O/O pairs because it makes the model significantly bigger. After running Step 5 on instance "MD-00001-00000191", all the 66 O/O pairs could be matched into 33 [O/O-O/O] cycles.

Step 6: Process the B/O and O/B pairs

In the sixth step, we deal with B/O and O/B pairs. We use these pairs at the end of the matheuristic because there are many O/B pairs that will not participate in the exchange due to a lack of O donors. At this step:

- **Objective:** maximise the number of [B/O-O/B] cycles.
- **Cycles to complete:** [$\underline{\texttt{B/AB-AB/O}}$] (fixed in Step 1), [$\underline{\texttt{B/A-A/O}}$] (fixed in Step 3).
- **Pairs to use:** none.
- **Types of cycles allowed:** [$\underline{\texttt{B/AB-AB/O}}$-O/B], [B/O-O/B], [B/O-$\underline{\texttt{O/O}}$-O/B], [$\underline{\texttt{B/A-A/O}}$-O/B].

We try to find as many [B/O-O/B] cycles as expected under the constraint that some O/B pairs should be used to complete the [B/AB-AB/O] cycles found in Step 1 and that we have to complete the partial cycles [B/A-A/O] from Step 3 with O/B pairs. Note that some [B/O-O/B] cycles could also be used to insert the O/O pairs that could not be matched in Step 5, but there are so few B/O pairs in PrefLib instances that it is not often possible to do so. This also explains why we consider that B/B are "pairs to use" in Step 3 while they could also theoretically be used in this step. After running Step 6 on instance "MD-00001-00000191", all 3 [B/O-O/B] cycles could be found and cycle [9-307] was completed.

Step 7: Process the `A/O` and `O/A` pairs

In the last step, we deal with `A/O` and `O/A` pairs. We finish with these pairs because `O/A` pairs are the most frequent pairs in PrefLib instances, and many of them will not participate in the exchange due to the lack of `O` donors. At this step:

- **Objective:** maximise the number of [`O/A-A/O`] cycles.
- **Cycles to complete:** [`A/AB-AB/O`] (fixed in Step 1), [`O/AB-AB/A`] (fixed in Step 1), [`A/B-B/O`] (fixed in Step 3).
- **Pairs to use:** `O/O` (from Step 5), `A/A` (from Step 4).
- **Types of cycles allowed:** [`A/AB-AB/O-O/A`], [`A/O-O/A`], [`A/O-O/O-O/A`], [`A/O-O/A-A/A`], [`A/B-B/O-O/A`].

We try to find as many [`O/A-A/O`] cycles as expected under the constraint that some of these cycles should incorporate the "pairs to use" `A/A` and `O/O` that could not be matched in Steps 4 and 5, that some `O/A` pairs should be used to complete the [`A/AB-AB/O`] cycles found in Step 1 and that we have to complete the partial cycles [`A/B-B/O`] from Step 3 with `O/A` pairs. Note that it is not necessary to consider [`O/AB-AB/A`] cycles as we assumed that `O/AB` pairs were excluded from the exchange. After running Step 7 on instance "MD-00001-00000191", cycles $[253-248]$ and $[258-259]$ were completed, the 7 incomplete [`A/B-B/O`] cycles from Step 3 were completed by an `O/A` pair to form 7 complete [`A/B-B/O-O/A`] cycles, all 51 cycles containing one `A/O` pair and one `O/A` pair could be found: 50 [`A/O-O/A`] cycles and one [`A/O-O/A-A/A`] cycle, the latter containing the remaining `A/A` pair from Step 4.

Extending the matheuristic for $L = 4$

We were able to solve the PrefLib instances with $L = 4$ by only updating Step 1 in the same way we updated U_3 in Section 3, i.e., by updating the set containing the `AB/*` and `*/AB` pairs as $\mathcal{C} = C_2^2 \cup C_3^3 \cup C_4^4 \cup P_2^1 \cup P_3^2 \cup C_4^3$. It was not necessary to consider any supplementary types of cycles (e.g., [`B/B-B/A-A/A-A/B`] or [`O/O-O/O-O/O-O/O`]).

4.3 Matheuristic performance on PrefLib instances and discussion

We measured the performance of our matheuristic on the 30 PrefLib instances used by Lam and Mak-Hau [21] for testing their branch-and-price-and-cut algorithm and the results are displayed in Table 3. Column "L" gives the cycle size limit, column "n" gives the number of recipient/donor pairs in the instance, the three following columns indicate the number of optimal solutions found by the matheuristic (out of 10), the average lower bound (which was always equal to U_3 in all tested instances), and the computing time in seconds (using the same computational environment as the one described in Section 3). The six following columns report the number of optimal solutions found and the computing time required by the branch-and-price-and-cut and two ILP formulations as reported in the experiments of Lam and Mak-Hau [21].

Table 3: Matheuristic performance on PrefLib instances

Parameters		Matheuristic			PIEF		PICEF / Cycle		B-P-C	
L	n	OPT	LB	Time	OPT	Time	OPT	Time	OPT	Time
3	512	10	322.7	0.2	10	8	10	82	10	1
3	1024	10	649.1	1.4	4	1304	5	1308	10	0
3	2048	10	1276.9	3.0	0	1800	0	1800	10	5
4	512	10	322.8	0.2	0	1800	0	1800	10	21
4	1024	10	649.3	0.7	0	1800	0	1800	9	800
4	2048	10	1276.9	3.4	0	1800	0	1800	0	1800

We observe that the matheuristic can solve every instance in a few seconds. In particular, it is able to solve the 11 instances left open by Lam and Mak-Hau [21]. Even though these results are very good, a word of caution is in order: the matheuristic is tailored for large-size PrefLib instances. These are characterised by a very large graph density (around 25%) and a relatively large number of recipient/donor pairs, which makes it very likely that the cycles predicted in Step 1 can be constructed in subsequent steps. Small-size PrefLib instances could not be solved by the matheuristic but were qualified as"trivial" by Lam and Mak-Hau [21] (i.e., solvable in seconds by an ILP formulation). Real-world instances from the UKLKSS seem to indicate that graph densities are usually smaller (around 10%).

The main objective of this section is to show that benchmark instances, which are important for developing new algorithms, can easily favour one kind of algorithm. PrefLib instances have a high graph density, which penalises the methods that enumerate every feasible cycle (such as the cycle formulation and PICEF) and benefits methods using lower and upper bounding techniques (like our matheuristic). In the following, we suggest some improvements for the Saidman generator (that was used for generating the PrefLib instances) in order to make it produce instances that match more closely the real-world instances of the UKLKSS.

5 Improved generation of KE instances

This section introduces a new method for generating KE instances, based on the Saidman generator. This generator, which we describe in Section 5.2, is widely used to generate random KE instances for evaluating both computational techniques and policy changes. We develop a number of improvements to the Saidman generator, and then show that using our refined version creates instances that are substantially closer to the real-world instances from the UK in a number of important measures (such as number of edges or sizes of solutions) when compared to those constructed by the original Saidman generator. Each of our improvements is based on statistical evidence from the UKLKSS, and accompanied by a justification based on real-world scenarios. This is followed by an experimental evaluation of these improvements, and a conclusion discussing the outcomes of the experiments.

5.1 Performance evaluation and data presentation

We evaluate the performance of our improvements by generating random instances and comparing them to real-world data from the UKLKSS. We gathered the blood-group and cPRA distributions from UKLKSS instances from January 2012 to October 2019 (32 instances in total as UKLKSS runs matchings quarterly), and these are summarised in Appendix A. As these real-world instances do not have a constant size, we cannot report the results of our experiments by comparing the averages of various parameters. Instead, for each real-world instance, we create 20 random instances with the same number of recipients and non-directed donors (the precise number of directed donors is determined by the generator). For each such randomly-generated instance and its associated real-world instance, and for each parameter we measure, we calculate the ratio of the value of parameter in the random instances divided by the value of the parameter in the real-world instance (e.g., if the random instance has 5000 edges, but the real-world instance has 4000 edges, we report the ratio $\frac{5000}{4000} = 1.25$). This results in a set of 640 ratios for each parameter measured. We then plot the distribution of these ratios as box-and-whisker plots. These plots show a coloured box denoting the interquartile range (IQR), a black vertical bar denoting the median, and a white cross denoting the mean. The horizontal lines extend a distance of 1.5 IQR from either side of the coloured box (if no data points extend this far, these lines stop at the last data point), with data points further away being marked as outliers with solid dots. We limit data and plots presented in this section to show only the ratios corresponding to the number of edges in the compatibility graphs and the size of the largest set of identified transplants. We include in Appendix B results on a much larger variety of parameters.

5.2 The Saidman generator

The Saidman generator [30] is widely used to generate instances of KEP problems [12, 19]. We now describe how the Saidman generator functions, and then compare the output from the Saidman generator to real world instances.

5.2.1 Generation process

The Saidman generator takes as input the number of recipients, the number of non-directed donors, as well as the distributions of the donor and recipient blood-groups (these may be identical, if distinct distributions are not known), the distribution of number of donors per recipient[1] and the distribution of cPRA values of the recipients. These distributions are commonly gathered from a long-running KEP; we will be using values from the UKLKSS. The Saidman generator first produces donor-recipient pairs by independently drawing a blood-group and cPRA for a new recipient, as well as drawing the number of donors paired with this recipient, and a blood-group for each of these donors.

For any donor and recipient, their compatibility is determined as follows:

1. If the donor is blood-group incompatible with a given recipient, then they are not compatible.
2. Otherwise, draw uniformly at random a number r from $[0, 1]$. The donor and recipient are compatible if and only if $r > \frac{cPRA}{100}$.

If a donor is determined to be compatible with their paired recipient, the Saidman generator will discard both the recipient and all paired donors to avoid the generation of compatible pairs. Whilst never specifically addressed in the literature, it is commonly accepted that this is done so that the pool of donor-patient pairs more accurately reflects the real-world data. For instance, the proportion of type AB recipients in kidney exchange pools is often lower than the proportion of people with type AB in a general population, as it is easier for type AB recipients to find a willing and compatible donor as they are blood-group compatible with all potential donors. Discarding compatible pairs mimics this effect, and similar effects for recipients with a low cPRA.

Once sufficient recipients have been generated, the Saidman generator then draws a blood-group for each non-directed donor required.

Once this is complete, the compatibility of any donor with any recipient (other than their paired recipient, if applicable) is determined using the above algorithm, and a corresponding arc is added to the compatibility graph if the donor and recipient are compatible.

There are two important factors to note when discussing the Saidman generator. Firstly, the blood-groups of donors and recipients and the cPRA values are all assumed to be independent. This is but one of the aspects we will address later in this section. Secondly, the discarding of compatible pairs means that the distributions of the generated data will not match the input distributions (i.e., low cPRA recipients are more likely to be discarded, so low cPRA recipients will be under-represented in the generated data). To avoid this, the inputs to the Saidman generator can be tuned. This can be achieved by running the generator to produce an instance which is then analysed for some property or set of properties (e.g., the distribution of cPRA levels amongst recipients). If this property is deemed not sufficiently close to some desired target (e.g., the proportion of recipients with a low cPRA is deemed too low), then the inputs can be adjusted (e.g., the probability of a recipient drawing a low cPRA can be increased), and the process is repeated. Alternative tuning options include sweeping across a broad range of parameters to find the input distributions that give output distributions deemed best [31]. In this paper, we tune the distributions of donor blood-groups, recipient blood-groups, recipient cPRA levels, and number of donors paired with a recipient.

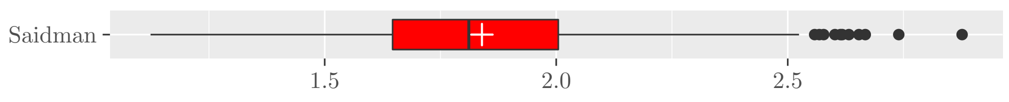
5.2.2 Performance of the Saidman generator

Recall from Section 5.1 that we generate 640 random instances, and for each random instance and each parameter we calculate the ratio comparing the parameter in the random instance to the parameter in the real-world instance with the same size. Figure 3 gives boxplots of these ratios for two parameters, showing how the Saidman generator compares to the real-world data. Better generators therefore have means and medians closer to one, and with less variance, although some variance is to be expected. We see that the Saidman generator produced instances with substantially more edges than present within the UKLKSS data, which corresponds to more transplants in a largest set of transplants. A comparison of a larger set of parameters is given in Appendix B.

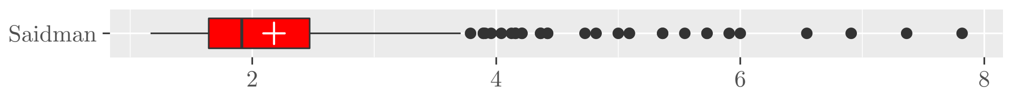
5.3 New generator configurations

In the rest of this section we introduce changes based on three different assumptions made by the Saidman generator. These are: the independence of donor and recipient blood-groups and the recipient cPRA, the

[1]The UKLKSS, and many other KEPs, allows a recipient to join a KEP with multiple donors, so as to increase their probability of finding a match.



(a) Distribution of the ratio of number of edges in randomly generated instances



(b) Distribution of the ratio of largest set of transplants in randomly generated instances.

Figure 3: Boxplots outlining the performance of the Saidman generator

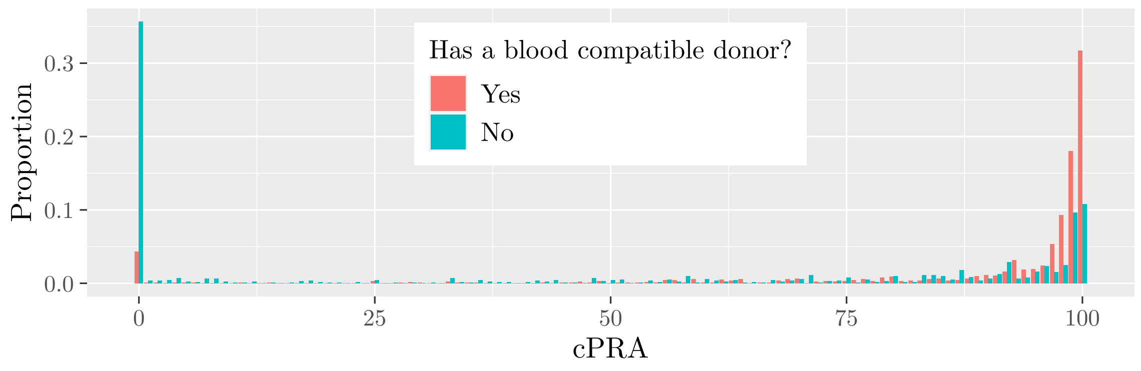


Figure 4: Distributions of cPRA dependent on whether a recipient has a blood-group compatible donor

direct linear relationship between cPRA and donor compatibility, and the equivalence of all recipients with a given cPRA value.

5.3.1 Donor and recipient blood-groups and cPRA distributions

As noted earlier, the Saidman generator assumes that the blood-groups of the donors and recipients, and the cPRA levels of the recipients, are independent. From a clinical point of view, however, it would not be surprising that recipients paired with blood-group compatible donors would be more likely to have tissue-type incompatibility, as donors and recipients who are both blood-group and tissue-type compatible are likely to be medically compatible and thus have proceeded with a transplant outside of a KEP. We ran a *t*-test of the corresponding Wald statistic [10] to compare our hypothesis that there is a correlation between a recipient having a blood-group compatible donor and the recipient's cPRA level against a null hypothesis. The resulting *p*-value of approximately 1.79×10^{-7} is a strong indicator that we can reject the null hypothesis, suggesting that different cPRA distributions should be used when generating recipients depending on whether a recipient has been generated with a blood-group compatible donor or not. We show in Figure 4 the distributions of cPRA levels of recipients from the UKLKSS broken down according to whether or not the recipient is paired with a blood-group compatible donor.

An obvious requirement for modelling this phenomenon is the ability to accurately model the correct proportion of blood-group compatible pairs. To achieve this, we calculate the distribution of blood-groups of the recipients. We then calculate five distributions of donor blood-groups, one for each of the four possible recipient blood-groups, and one for non-directed donors. This data is shown in Figure 5, and we see that there are substantial differences between these distributions.

We implement a generator that includes these adaptations, and use the term SplitPRA to refer to this adaptation. Two box plots comparing the performance of the adaptation SplitPRA to the Saidman generator are shown in Figure 6. We see from these that SplitPRA creates random instances that are

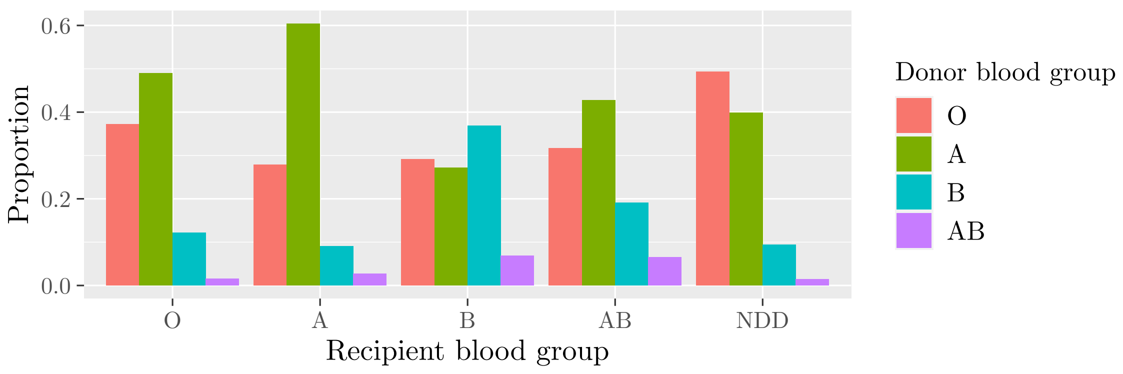
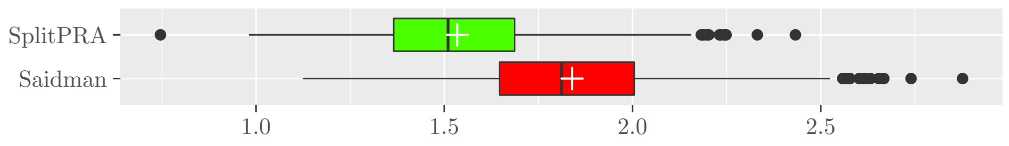
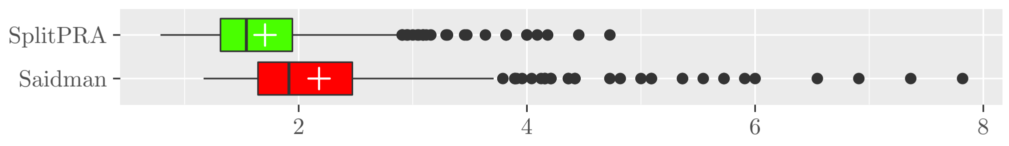


Figure 5: Distributions of donor blood-group by recipient blood-group



(a) Distribution of number of edges as proportion of edges in real-world instances



(b) Size of largest set of transplants

Figure 6: Comparison of Saidman and SplitPRA generators

closer to the real-world instances than those produced by the Saidman generator both in terms of number of edges and size of largest set of transplants.

5.3.2 Link between cPRA and transplant compatibility

For any donor and recipient who are blood-group compatible, the Saidman generator uses cPRA as a proxy for transplant compatibility. In real-world applications, transplant compatibility does require tissue-type compatibility (and cPRA is a good measure of this), but there are also external factors such as the size of the kidney, or the age or overall health of the donor. As such factors are much harder to randomly sample, the Saidman generator draws transplant compatibility uniformly at random with probability $(1 - \text{cPRA}/100)$. However, cPRA by definition only considers tissue-type compatibility, and ignores these other possible causes of incompatibility that may arise. For each recipient in the UKLKSS data, we calculate the proportion of blood-group compatible donors appearing in a matching run simulatenously with the recipient that are also identified as potential donors for the recipient. We plot this data in Figure 7. From this, it is clear to see that while there is a strong negative relationship between cPRA and compatibility (confirmed with a Spearman rank correlation coefficient of approximately -0.519, which corresponds to a p-value of $< 2.2 \times 10^{-16}$), it is not necessarily as simple as $(1 - \text{cPRA}/100)$. Indeed, a linear least-squares model weighted by the distribution of cPRA amongst the recipients gives the equation $P(\text{Compat.}) = 0.58 - 0.55 \times \text{cPRA}/100$. This linear model is shown as a solid blue line in Figure 7, with the shaded region showing the 95% confidence interval. The triangles denote each data point, with one data point per cPRA value. Note that the number of samples represented by each data point is not visually represented in Figure 7 to make the plot easier to read.

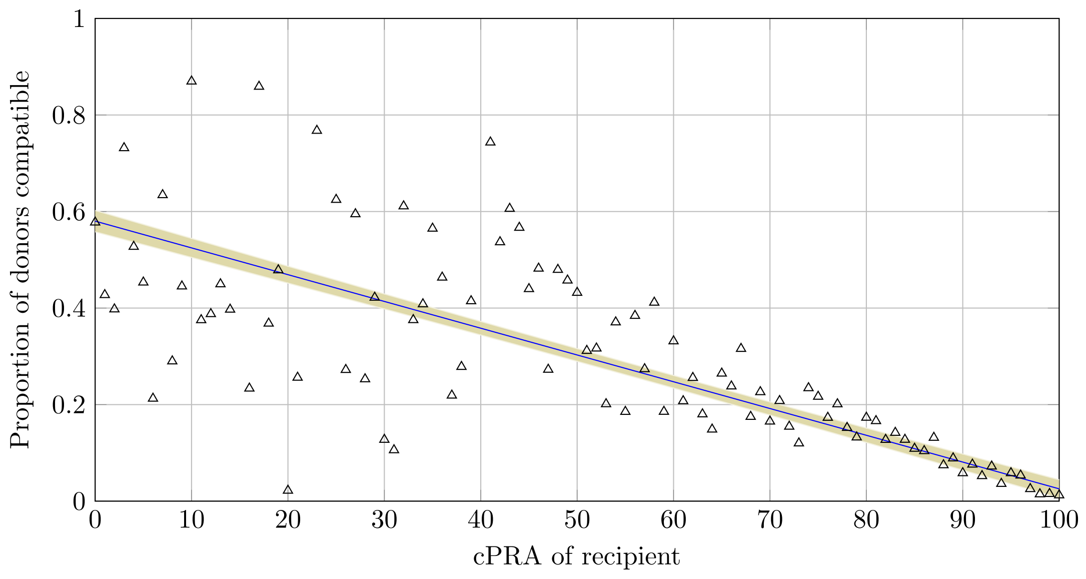


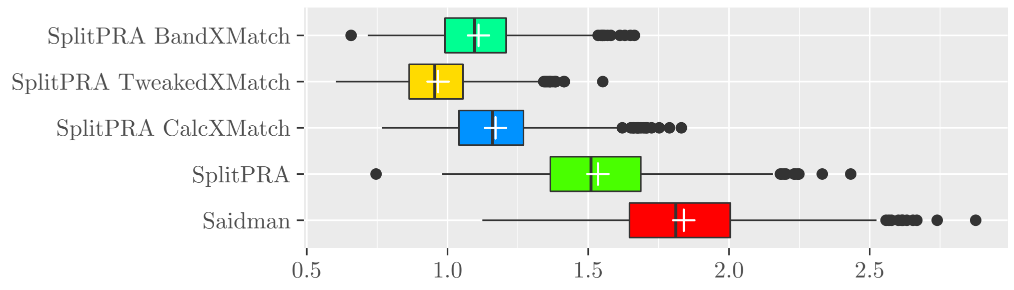
Figure 7: Proportion of UKLKSS donors that are compatible with a blood-group compatible recipient

As such, the reader should be aware that the level of correlation between cPRA and the variance of compatibility as seen in Figure 7 may be misleading. By looking at the data we see that while there is a greater variance at lower values of cPRA (we discuss this in detail for recipients with a cPRA of exactly 0 in Section 5.3.3), much of the variance visually depicted in Figure 7 is due to the relative difference in populations represented at each cPRA value (c.f. Figure 4 which shows the relative distribution of cPRA values among recipients).

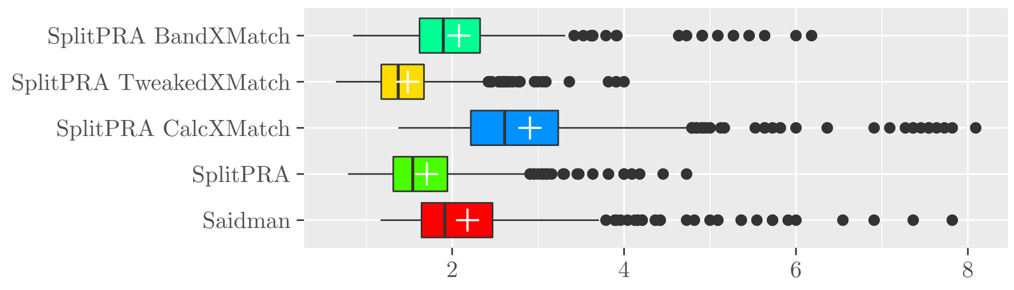
We see that this linear model does approximate the relationship between cPRA and compatibility, but many data points are outside the 95% confidence interval. We also highlight that recipients with a cPRA of 0 (i.e., those recipients who theoretically should have no tissue-type compatibility problems) are only compatible with approximately 58% of the blood-compatible donors in their matching runs. We investigate this particular aspect further in Section 5.3.3. Other studies model the link between cPRA and tissue-type compatibility through the cumulative distribution function (CDF) of a standard normal distribution [17]. However, these studies only consider tissue-type compatibility, whereas we are aiming to model actual compatibility, giving one potential explanation for this difference. Based on the data we observe in Figure 7, neither a linear relationship nor CDF nor any straight-forward model can accurately represent the link between cPRA and transplant compatibility, so we opt for the linear model which is able to more closely match the low but non-zero probability of compatibility that is observed with recipients with a cPRA of 100.

We test an adaptation to the generator that uses $P(\text{Compat.}) = 0.58 - 0.55 \times \text{cPRA}/100$ (i.e., the linear least-squares model) to determine compatibility between blood-compatible donors and recipients; we call this CalcXMatch. Full experimental results are shown in Appendix B, but Figure 8 shows that CalcXMatch does reduce the number of edges (bringing it closer to the real-world instances) but at the cost of significantly increasing the number of transplants identified. A closer examination of the compatibility equation shows that CalcXMatch will give recipients with a cPRA of 100 a 3% chance of compatibility, whereas the Saidman generator gives such recipients a 0% chance of compatibility. Thus, CalcXMatch has fewer incoming edges from the recipients with a low cPRA, and more incoming edges to recipients with cPRA values above 98, compared to the Saidman generator, resulting in more identified transplants.

We test a second adaptation that uses the formula $P(\text{Compat.}) = 0.55 - 0.55 \times \text{cPRA}/100$ to determine compatibility (forcing recipients with a cPRA of 100 to have a 0% chance of compatibility). We call this adaptation TweakXMatch, and its performance is included in Figure 8. This, as expected, reduces the number of edges, but brings the mean number of edges very close to the mean number of edges in the real-world data. Whilst this looks impressive, we do highlight that this particular generator, like the Saidman generator and just using SplitPRA, gives a recipient with a cPRA of 100 zero chance of being



(a) Distribution of number of edges as proportion of edges in real-world instances



(b) Size of largest set of transplants

Figure 8: Comparison of four methods of calculating compatibility

compatible, and so no recipients with a cPRA of 100 are ever identified for transplantation.

We also create a third adaptation that does not use one equation to determine compatibility for all recipients. Instead, we split the range of potential cPRA values into the bands $[0, 50)$, $[50, 95)$, $[95, 96)$, $[96, 97)$, $[97, 98)$, $[98, 99)$, $[99, 100)$ and 100. Then a linear regression is performed on the real-world recipients from the UKLKSS who fit into each band of cPRA values, giving one equation (or value, if the band contains just one cPRA value) per cPRA band. These equations are then used to determine compatibility for generated recipients, based on their cPRA values. We call this configuration BandXMatch, and its performance is also included in Figure 8. We see that, as expected, it has a similar number of edges to CalcXMatch, but on average it has a smaller-sized largest set of transplants.

5.3.3 The compatibility of cPRA 0 recipients

One oddity that arises from Figure 7 is that recipients with a cPRA of 0 are only compatible with approximately 58% of their blood-group compatible donors. This is somewhat less than may have been anticipated, so we investigated it further. Figure 9 shows a histogram of the compatibility of real-world recipients with a cPRA of 0, where compatibility is calculated as the proportion of potential transplants in the instances divided by the number of blood-group compatible donors in said instances. We see that approximately 18% of these cPRA 0 recipients are compatible with all of their blood-group compatible donors, and that approximately 40% of these recipients are compatible with at least 95% of their blood-group compatible donors. However, almost 26% of recipients with a cPRA of 0 are compatible with less than 25% of their blood-compatible donors.

Some hypotheses have been presented for this phenomenon:

- Paediatric recipients may have stricter compatibility requirements, beyond the simple blood and tissue-type compatibility, or,
- if a recipient is known to have a low (or zero) cPRA, then this recipient (or their clinician) may be more stringent with regards to other parameters such as age or health of the donor.

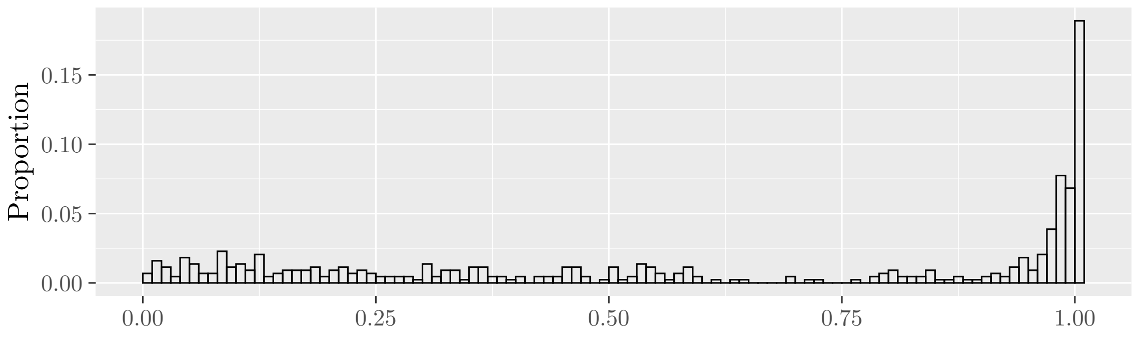


Figure 9: Histogram of compatibility for cPRA 0 recipients

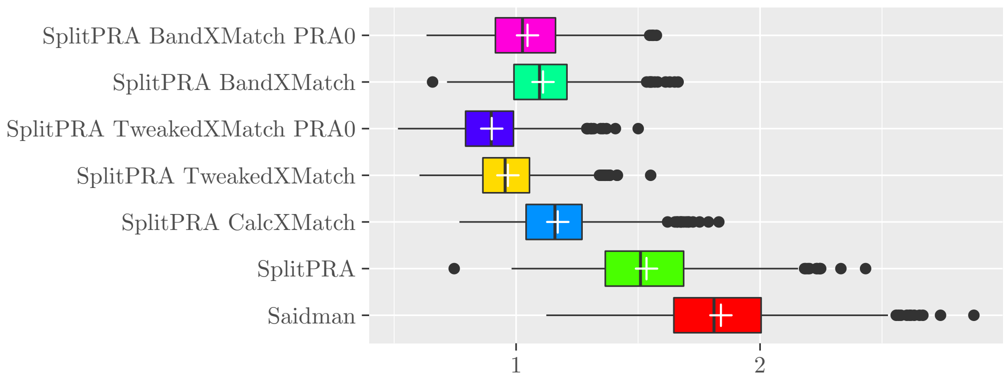
We account for this phenomenon in our generator as follows. For each recipient with a cPRA of 0, we randomly sample from a smoothed variant of the distribution shown in Figure 9 the compatibility proportion of this recipient. This compatibility proportion is then used directly as the probability that the recipient is compatible with a blood-group compatible donor. In this way, we generate instances wherein approximately 40% of recipients with a cPRA of 0 are compatible with over 95% of their blood-group compatible donors, and so on.

Figure 10 shows the performance of this adaptation, which we denote by PRA0 in the plots. We see that when combined with either TweakXMatch or BandXMatch, this adaptation slightly reduces both the number of edges and size of the largest set of transplants.

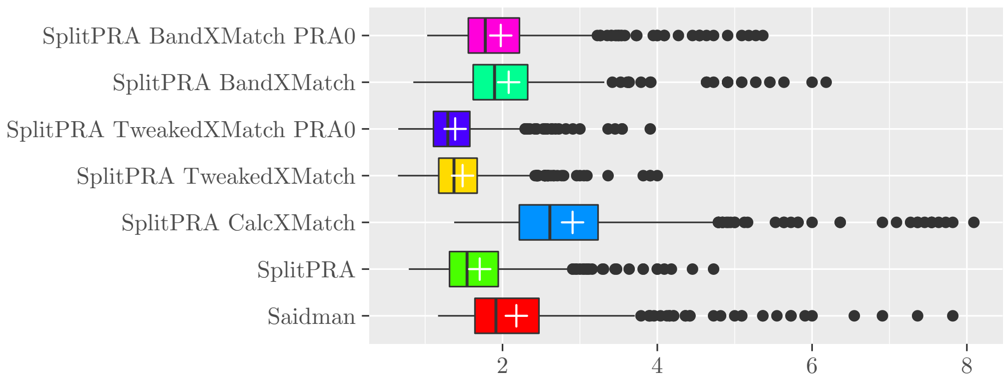
5.4 Discussion

We have demonstrated a number of improvements to the process of generating random KE instances. We have shown that the recipients who have a blood-group compatible donor have a markedly different distribution of cPRA values to those who do not have a blood-group compatible donor, and that utilising this when generating instances will result in random instances that more closely resemble real-world instances. This approach can be utilised by simply calculating and using two distinct cPRA distributions for the two populations. We have also shown that the correlation between cPRA and transplant compatibility, whilst present, is not necessarily as simple as $P(\text{compatibility}) = 1 - \text{cPRA}/100$, and taking a more nuanced approach results in random instances that more closely resemble real-world instances. Lastly, we show that, within the UKLKSS, recipients with a cPRA of 0 have an unusual distribution of transplant compatibility. Figure 10 shows that two particular combinations of improvements, SplitPRA TweakXMatch PRA0 and SplitPRA BandXMatch PRA0, give the best peformance. It is important to note that, while both Figure 10 and further plots in Appendix B show that SplitPRA TweakXMatch PRA0 generally produces instances with properties closer to instances from the UKLKSS, SplitPRA TweakXMatch PRA0 gives recipients with a cPRA of 100 zero chance of finding a compatible donor, while SplitPRA BandXMatch PRA0 gives each such recipient approximately 1% chance of being compatible with each blood-compatible donor in the pool. With that in mind, to simulate the UKLKSS, we recommend the use of SplitPRA BandXMatch PRA0.

We ran the first step of the matheuristic from Section 4 on instances generated with our new methods, and the bound obtained was generally found to be far above the optimal solution (e.g., the maximum number of transplants, as determined by the cycle formulation, was 364.1 on average for 20 instances with 1024 recipient/donor pairs while the matheuristic was looking for a solution with value 668.2 on average). This indicates that, in such instances, the maximum number of transplants does not solely depend on the blood groups anymore, which explains why the value was overestimated in Step 1 of our approach. Note that one could update the matheuristic to find a feasible solution even if it does not match the bound found in Step 1 by modelling the "pairs to use" and the "cycles to complete" as parts of the objective function instead of within constraints. Even though such an adaptation could potentially provide good-quality solutions, the lack of a guarantee of optimality would be a major drawback, as exact approaches are strongly favoured for optimisation problems where the objective function may impact on



(a) Distribution of number of edges as proportion of edges in real-world instances



(b) Size of largest set of transplants

Figure 10: Comparison of all six new methods of calculating compatibility

human lives.

Many of the conclusions we have drawn are based on a study of UK data, but we expect that at least some of our adaptations (in particular, highlighting the correlation between blood compatibility and cPRA distributions, and the relationship between cPRA and transplant compatibility) will be useful when generating random instances for other KEPs.

6 Conclusion

We have presented new upper bounds and new matheuristic for identifying large sets of transplants in KE instances. We show that, on a standard set of randomly-generated instances, combining our upper bound and our matheuristic achieves optimal solutions much faster than any algorithm from the literature. This particular set of instances, despite its common use in the literature, differs from real-world instances in a number of vital characteristics (particularly edge density), and our upper bound and matheuristic are not guaranteed to converge, and indeed do not converge, on all real-world instances. We then demonstrated a number of improvements to the standard method of generating random KE instances which are shown to produce instances that are much closer to real-world instances in many characteristics. We have made our generator, and relevant distributions and parameters from the UKLKSS available to allow people to test both new KEP algorithms and new KEP policies on data that closely matches real-world instances from the UKLKSS.

We hope to continue work with this generator by collaborating with other KEPs to determine which adaptations are applicable globally and obtain new adaptations for improved generation of KE instances.

Acknowledgements

The authors would like to thank NHS Blood and Transplant for sharing their data and experience with us. This research was supported by the Engineering and Physical Science Research Council through grant numbers EP/P028306/1 (Manlove and Pettersson), EP/P029825/1 (Delorme, Garcìa, Gondzio, and Kalcsics), and EP/R513222/1 (Trimble).

References

[1] D.J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. In *Proceedings of EC '07: the 8th ACM Conference on Electronic Commerce*, pages 295–304. ACM, 2007.

[2] F. Alvelos, X. Klimentova, and A. Viana. Maximizing the expected number of transplants in kidney exchange programs with branch-and-price. *Annals of Operations Research*, 272(1-2):429–444, 2019.

[3] R. Anderson, I. Ashlagi, D. Gamarnik, and A.E. Roth. Finding long chains in kidney exchange using the traveling salesman problem. *Proceedings of the National Academy of Sciences*, 112:663–668, 2015.

[4] I. Ashlagi and A. Roth. Individual rationality and participation in large scale, multi-hospital kidney exchange. In *Proceedings of EC '11: the 12th ACM Conference on Electronic Commerce*, pages 321–322. ACM, 2011.

[5] I. Ashlagi and A. Roth. New challenges in multihospital kidney exchange. *American Economic Review*, 102(3):354–59, 2012.

[6] D. A. Axelrod, M. A. Schnitzler, H. Xiao, W. Irish, E. Tuttle-Newhall, S.-H. Chang, B. L. Kasiske, T. Alhamad, and K. L. Lentine. An economic asssessment of contemporary kidney transplant practice. *Americal Journal of Transplantation*, 18:1168–1176, 2018.

[7] P. Biró, M. Gyetvai, X. Klimentova, J. Pedroso, W. Pettersson, and A. Viana. Compensation scheme with Shapley value for multi-country kidney exchange programmes. In M. Steglich, C. Mueller, G. Neumann, and M. Walther, editors, *Proceedings of the 34th International ECMS Conference*

on Modelling and Simulation, ECMS 2020, Wildau, Germany, June 9-12, 2020, pages 129–136. European Council for Modeling and Simulation, 2020.

[8] P. Biró, B. Haase-Kromwijk, T. Andersson, E. I. Ásgeirsson, T. Baltesová, I. Boletis, C. Bolotinha, G. Bond, G. Böhmig, L. Burnapp, K. Cechlárová, P. Di Ciaccio, J. Fronek, K. Hadaya, A. Hemke, C. Jacquelinet, R. Johnson, R. Kieszek, D. R. Kuypers, R. Leishman, M.-A. Macher, D. Manlove, G. Menoudakou, M. Salonen, B. Smeulders, V. Sparacino, F. C. R. Spieksma, M. O. Valentín, N. Wilson, and J. van der Klundert. Building Kidney Exchange Programmes in Europe — An Overview of Exchange Practice and Activities. *Transplantation*, 103:1514–1522, 2019.

[9] 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. Costa, B. Smeulders, F. Spieksma, M. Valentin, and A. Viana. Modelling and optimisation in European Kidney Exchange Programmes. *European Journal of Operational Research*, 291(2):447–456, 2019.

[10] A. Buse. The likelihood ratio, Wald, and Lagrange multiplier tests: An expository note. *The American Statistician*, 36(3a):153–157, 1982.

[11] M. Carvalho, X. Klimentova, K. Glorie, A. Viana, and M. Constantino. Robust models for the kidney exchange problem. *INFORMS Journal on Computing*, 2020.

[12] 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*, 231:57–68, 2013.

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

[14] J. Dickerson, A.D. Procaccia, and T. Sandholm. Optimizing kidney exchange with transplant chains: Theory and reality. In *Proceedings of AAMAS '12: the 11th International Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 711–718. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[15] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[16] K. Glorie. Estimating the probability of positive crossmatch after negative virtual crossmatch. Technical Report EI 2012-25, December 2012.

[17] K. Glorie. *Clearing Barter Exchange Markets: Kidney Exchange and Beyond.* PhD thesis, November 2014.

[18] A. Hart, J. M. Smith, M. A. Skeans, S. K. Gustafson, D. E. Stewart, W. S. Cherikh, J. L. Wainright, A. Kucheryavaya, M. Woodbury, J. J. Snyder, B. L. Kasiske, and A. K. Israni. OPTN/SRTR 2015 annual data report: Kidney. *American Journal of Transplantation*, 17:21–116, 2017.

[19] X. Klimentova, J. Pedroso, and A. Viana. Maximising expectation of the number of transplants in kidney exchange programmes. *Computers & Operations Research*, 73:1–11, 2016.

[20] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[21] E. Lam and V. Mak-Hau. Branch-and-cut-and-price for the cardinality-constrained multi-cycle problem in kidney exchange. *Computers & Operations Research*, 115(104852), 2020.

[22] Z. Li, N. Gupta, S. Das, and J. Dickerson. Equilibrium behavior in competing dynamic matching markets. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 389–395. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[23] D. Manlove and G. O'Malley. Paired and altruistic kidney donation in the UK: Algorithms and experimentation. *ACM J. Exp. Algorithmics*, 19, 1 2015.

[24] N. Mattei and T. Walsch. Preflib.org: A library for preferences. `https://www.preflib.org/data/matching/kidney/` (accessed 23 September 2020), 2020.

[25] D. McElfresh, H. Bidkhori, and J. Dickerson. Scalable robust kidney exchange. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1077–1084, 7 2019.

[26] R. Mincu, P. Biró, M. Gyetvai, A. Popa, and U. Verma. IP solutions for international kidney exchange programmes. *Central European Journal of Operations Research*, pages 1–21, 2020.

[27] W. Mulley and J. Kanellis. Understanding crossmatch testing in organ transplantation: A case-based guide for the general nephrologist. *Nephrology*, 16(2):125–133, 2011.

[28] A.E. Roth, T. Sönmez, and M.U. Ünver. Kidney exchange. *Quarterly Journal of Economics*, 119(2):457–488, 2004.

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

[30] S.L. Saidman, A.E. Roth, T. Sönmez, M.U. Ünver, and S.L. Delmonico. Increasing the opportunity of live kidney donation by matching for two and three way exchanges. *Transplantation*, 81(5):773–782, 2006.

[31] N. Santos, P. Tubertini, A. Viana, and J. P. Pedroso. Kidney exchange simulation and optimization. *Journal of the Operational Research Society*, 68(12):1521–1532, 2017.

[32] R. A. Wolfe, E. C. Roys, and R. M. Merion. Trends in organ donation and transplantation in the United States, 1999–2008. *American Journal of Transplantation*, 10:961–972, 2010.

A Parameters for generating instances similar to real-world instances from the UKLKSS

In this section we list a number distributions relevant to the generation of random instances of KEP problems. These distributions are all taken from the UKLKSS between 2012 and 2018 inclusive. For more details on how these may be used, see Section 5.

Table 4 shows the distribution of number of donors paired with a recipient in the UKLKSS. Recall that in the UKLKSS a donor can be paired with at most one recipient, while a recipient may be paired with multiple donors. In any exchange, only one donor of a recipient is selected (unless the donor is not-directed, in which case they have no paired donor).

Table 4: Distribution of number of donors paired with a recipient in UKLKSS

Number of donors	Proportion
1	0.9112
2	0.0769
3	0.0105
4	0.0015

Table 5 shows the distribution of blood groups amongst recipients within the UKLKSS. Tables 6, 7, 8, and 9 then show the distribution of donor blood groups within the UKLKSS. These distributions are dependent on the blood group of a paired recipient. Table 10 shows the distribution of blood groups amongst non-directed (altruistic) donors within the UKLKSS.

Table 5: Blood group distribution of recipients in the UKLKSS

Blood group	Proportion
O	0.6293
A	0.2325
B	0.1119
AB	0.0263

Table 6: Blood group distribution of donors paired with a recipient with blood group O

Blood group	Proportion
O	0.3721
A	0.4899
B	0.1219
AB	0.1610

Table 11 shows the distribution of cPRA amongst recipients who are paired with at least one blood-group compatible donor. Table 12 shows the distribution of cPRA amongst recipients who are not paired with any blood-group compatible donor. Table 13 shows the relationship between cPRA and likelihood of transplant compatibility with a blood-group compatible donor within the UKLKSS. Where an equation is given, these were determined by linear regression. Table 14 shows the distribution of likelihood of a transplant being incompatible for recipients within the UKLKSS with a cPRA of 0.

B Additional comparisons of various generators

In this section we give extensive results on our tests of various generator configurations. Recall that for each real-world instance, we generated 20 random instances for each of our test generators with the same number of recipients as the corresponding real-world instance. We then calculate, for each

Table 7: Blood group distribution of donors paired with a recipient with blood group A

Blood group	Proportion
O	0.2783
A	0.6039
B	0.0907
AB	0.0270

Table 8: Blood group distribution of donors paired with a recipient with blood group B

Blood group	Proportion
O	0.2910
A	0.2719
B	0.3689
AB	0.0683

parameter, the proportion of the value of the parameter in each random instance divided by the value of the parameter for the real-world instance. These are then plotted as box-and-whisker plots, with a coloured box denoting the interquartile range (IQR), a black vertical bar denoting the median, and a white cross denoting the mean. The horizontal lines extend up to a distance of 1.5 IQR from either side of the coloured box (if no data points extend this far, these lines stop at the last data point), with data points further away being marked as outliers with solid dots.

See Section 5.1 for a more detailed explanation.

Table 9: Blood group distribution of donors paired with a recipient with blood group AB

Blood group	Proportion
O	0.3166
A	0.4271
B	0.1910
AB	0.0653

Table 10: Blood group distribution of non-directed (altruistic) donors in the UKLKSS

Blood group	Proportion
O	0.3333
A	0.2698
B	0.0635
AB	0.0095

Table 11: Distribution of cPRA amongst recipients who have a blood-group compatible paired donor in the UKLKSS

cPRA range	Proportion
0	0.0435
$[1, 10)$	0.0064
$[10, 20)$	0.0027
$[20, 30)$	0.0060
$[30, 40)$	0.0084
$[40, 50)$	0.0107
$[50, 60)$	0.0217
$[60, 70)$	0.0291
$[70, 80)$	0.0391
$[80, 85)$	0.0257
$[85, 90)$	0.0308
90	0.0114
91	0.0107
92	0.0157
93	0.0318
94	0.0191
95	0.0197
96	0.0241
97	0.0535
98	0.0929
99	0.1802
100	0.3170

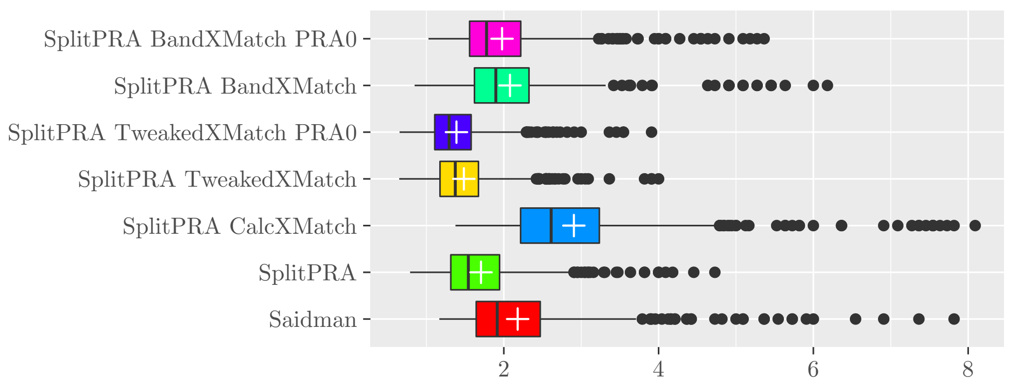


Figure 11: Size of largest set of transplants as a proportion of the size of largest set in real-world instances

Table 12: Distribution of cPRA amongst recipients who do not have a blood-group compatible paired donor in the UKLKSS

cPRA range	Proportion
0	0.3568
$[1, 10)$	0.0390
$[10, 20)$	0.0134
$[20, 30)$	0.0107
$[30, 40)$	0.0210
$[40, 50)$	0.0244
$[50, 60)$	0.0336
$[60, 70)$	0.0306
$[70, 80)$	0.0428
$[80, 85)$	0.0355
$[85, 90)$	0.0458
90	0.0065
91	0.0126
92	0.0286
93	0.0065
94	0.0076
95	0.0157
96	0.0237
97	0.0153
98	0.0252
99	0.0966
100	0.1081

Table 13: Proportion of blood-group compatible donors marked as transplant compatible broken down by cPRA of the recipient

cPRA range	Compatibility
$[0, 50)$	$0.5651 - 0.3301 \times \frac{\text{cPRA}}{100}$
$[50, 95)$	$0.6578 - 0.6419 \times \frac{\text{cPRA}}{100}$
95	0.0583
96	0.0534
97	0.0253
98	0.0145
99	0.0152
100	0.0124

Table 14: Distribution of the proportion of blood-group compatible donors marked as transplant compatible for recipients with a cPRA of 0

Incompatibility range	Proportion of recipients
0	0.1891
$(0, 0.01]$	0.0683
$(0.01, 0.02]$	0.0774
$(0.02, 0.03]$	0.0387
$(0.03, 0.04]$	0.0205
$(0.04, 0.10]$	0.0547
$(0.10, 0.25]$	0.0592
$(0.25, 0.50]$	0.0911
$(0.50, 0.75]$	0.1412
$(0.75, 1.00]$	0.2597

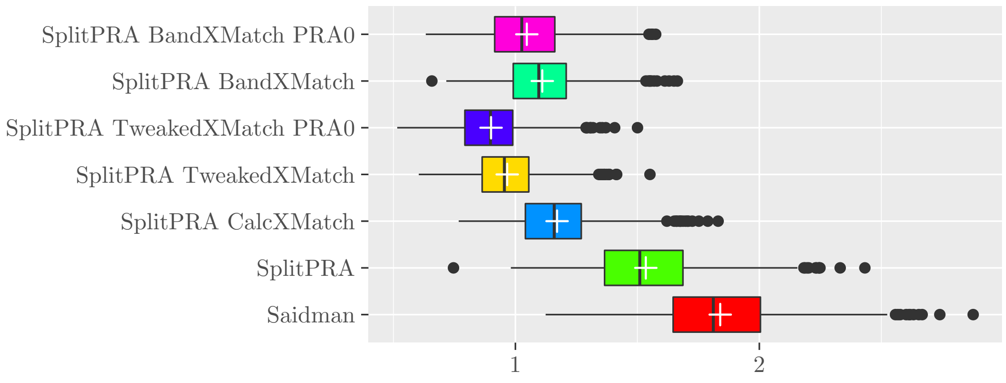


Figure 12: Number of edges as a proportion of number of edges in real-world instances

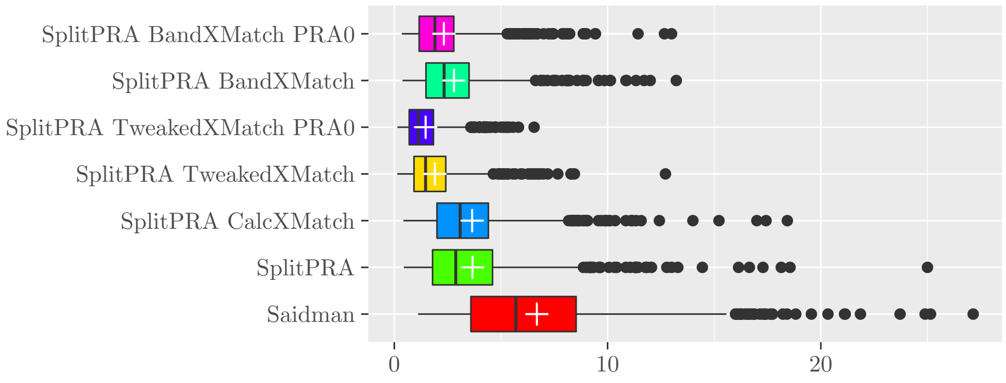


Figure 13: Number of two-cycles as a proportion of the number of two-cycles in real-world instances

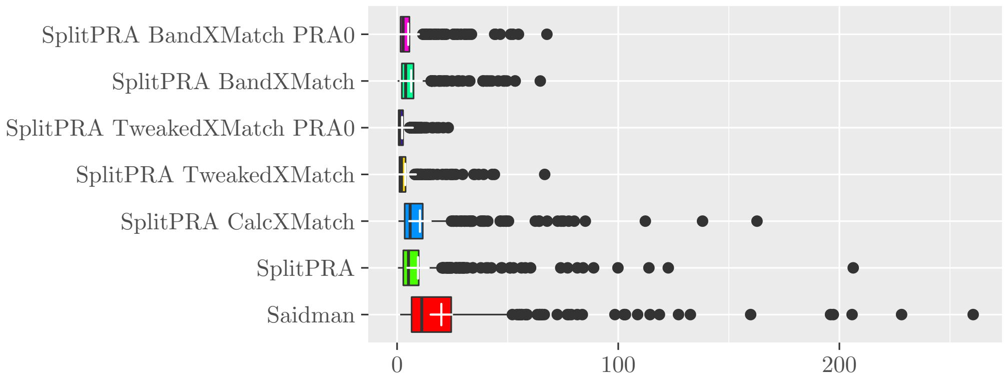


Figure 14: Number of three-cycles as a proportion of the number of three-cycles in real-world instances

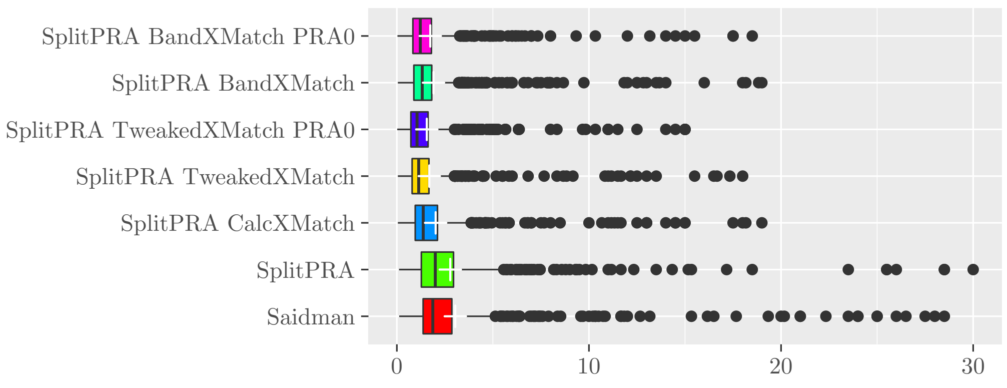


Figure 15: Number of short chains as a proportion of the number of short chains in real-world instances

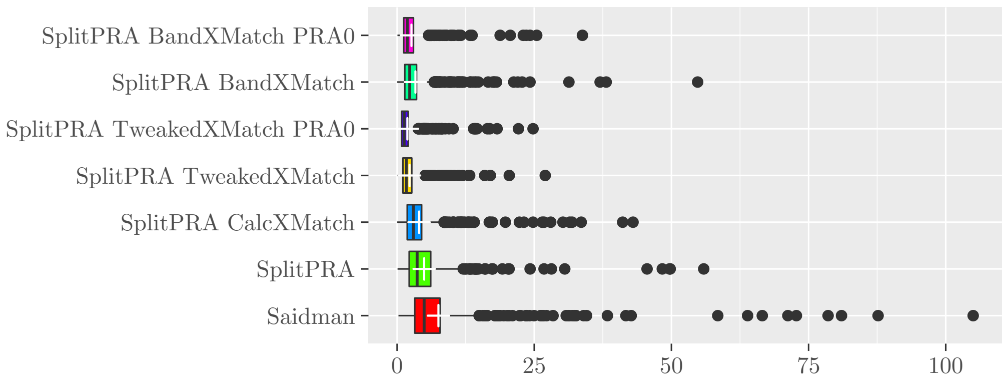


Figure 16: Number of long chains as a proportion of the number of long chains in real-world instances

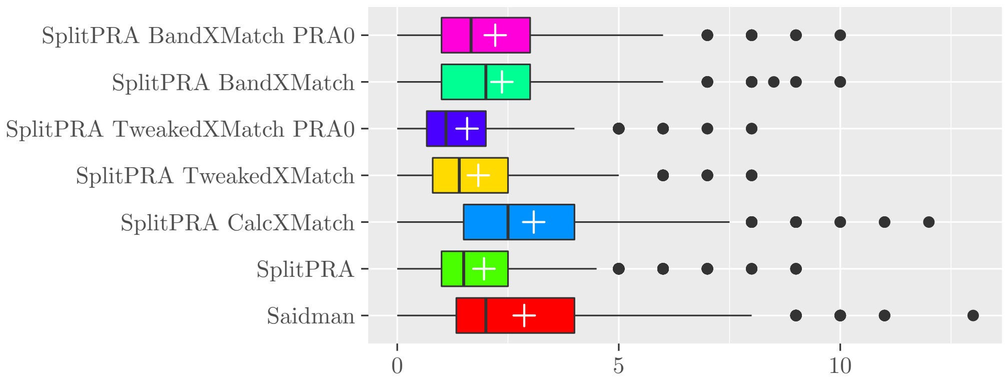


Figure 17: Number of selected two-cycles as a proportion of the number of selected two-cycles in real-world instances

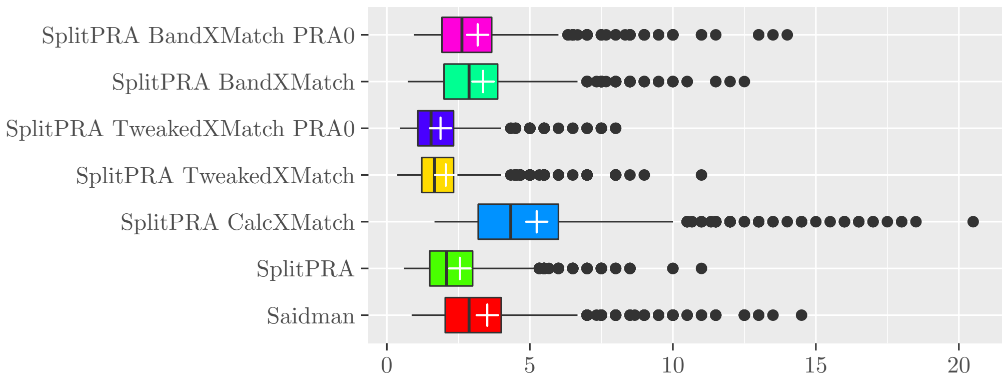


Figure 18: Number of selected three-cycles as a proportion of the number of selected three-cycles in real-world instances

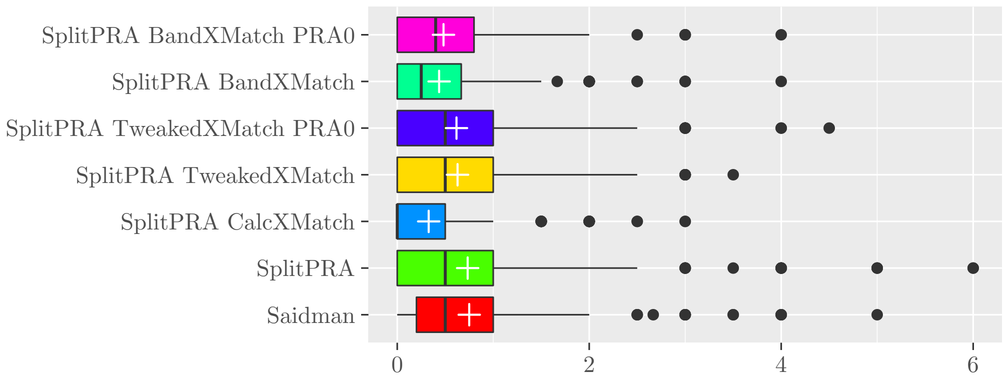


Figure 19: Number of selected short chains as a proportion of the number of selected short chains in real-world instances

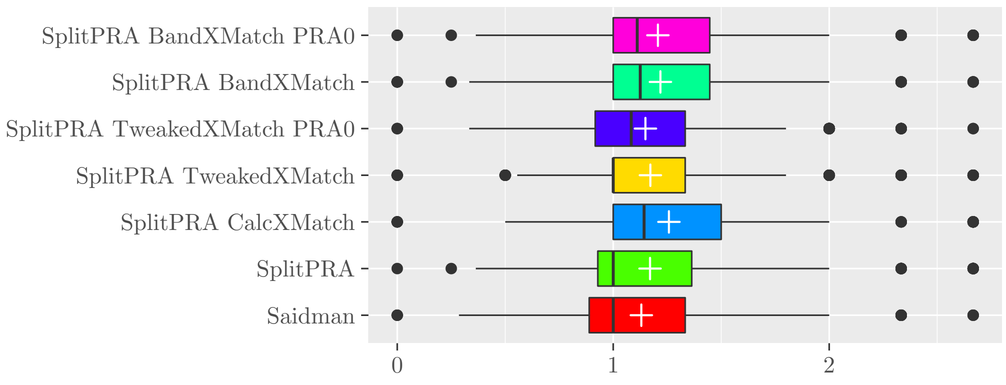


Figure 20: Number of selected long chains as a proportion of the number of selected long chains in real-world instances

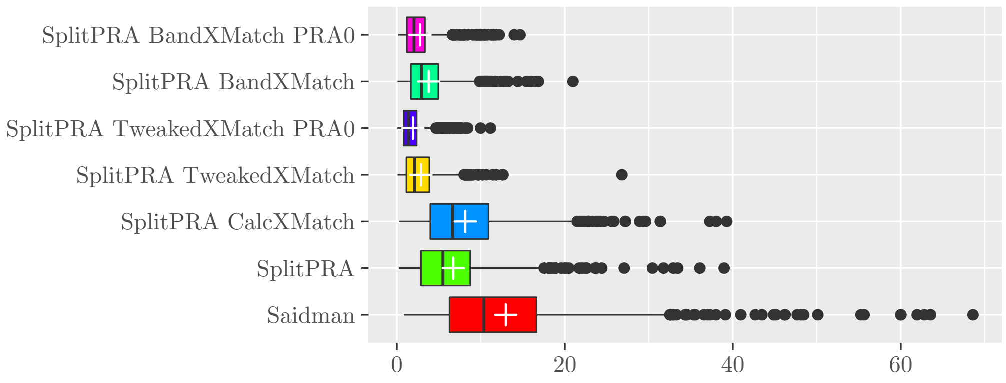


Figure 21: Time to solve IP as a proportion of the time to solve the IP for real-world instances

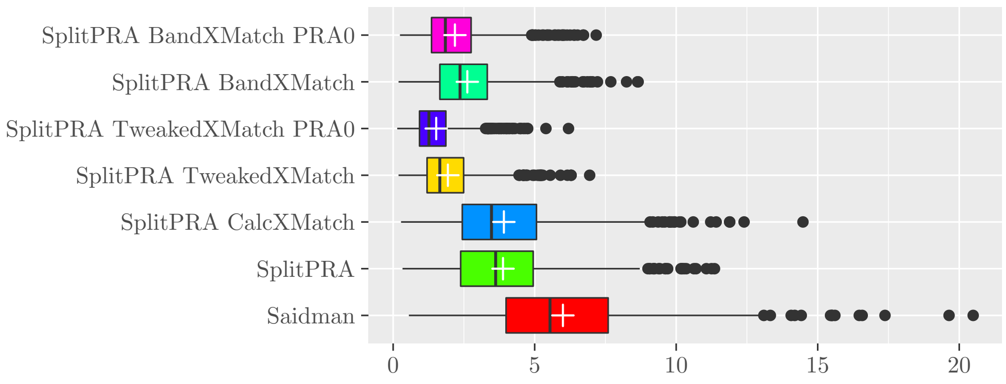


Figure 22: Time to solve total problem as a proportion of the time to solve the total problem for real-world instances