Computers & Operations Research 73 (2016) 1–11

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Maximising expectation of the number of transplants in kidney exchange programmes

Xenia Klimentova [a,*], João Pedro Pedroso [a,b], Ana Viana [a,c]

[a] INESC TEC, Campus da FEUP, Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal
[b] Faculdade de Ciências, Universidade do Porto, Rua do Campo Alegre, 4169-007 Porto, Portugal
[c] ISEP - School of Engineering, Polytechnic of Porto, Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal

ARTICLE INFO

Article history:
Received 6 March 2015
Received in revised form
20 January 2016
Accepted 8 March 2016
Available online 15 March 2016

Keywords:
Kidney exchange programmes
Cycle packing
Expectation optimisation

ABSTRACT

This paper addresses the problem of maximising the expected number of transplants in kidney exchange programmes. New schemes for matching rearrangement in case of failure are presented, along with a new tree search algorithm used for the computation of optimal expected values. Extensive computational experiments demonstrate the effectiveness of the algorithm and reveal a clear superiority of a newly proposed scheme, subset-recourse, as compared to previously known approaches.

1. Introduction

Kidney exchange programmes have been organised in many countries as an alternative mode of transplant for patients with kidney failure that have a donor willing to provide a kidney, but the pair is not physiologically compatible [9,4,24,25,1]. These programmes are based on the concept of "exchange" between patient–donor pairs: donors are allowed to provide a kidney to patients in the other pairs, if compatibility exists, so that patients in all pairs involved in the exchange benefit.

Fig. 1 (left) illustrates the simplest case with only two pairs, (P_1, D_1) and (P_2, D_2). Donor D_1 of the first pair is allowed to give a kidney to patient P_2 of the second pair, and patient P_1 may get a kidney from donor D_2. In the right-hand side of the figure an exchange between three incompatible pairs is possible: patient P_1 may receive the kidney from donor D_3, patient P_2 from donor D_1, and patient P_3 from donor D_2; in that case all patients are served. Notice that these graphs consider only preliminary compatibilities that will have to be reassessed prior to actual transplant through additional medical exams.

For logistical reasons, and also to reduce the number of affected patients when last-minute donor resignation occurs or new incompatibilities are detected, a limit k is imposed on the length of cycles.

The optimisation problem underlying a basic kidney exchange commonly considers the maximisation of the number of transplants [9,25]. But other criteria have been proposed to be maximised, e.g., the number of effective 2-cycles [18], the number of transplants with identical blood type [14], the overall score or utility assigned to each transplant [17,18], and the expected number of transplants [20]. For k limited and greater than 2, the problem of maximising the number of transplants is known to be NP-hard [4,1,15]. Integer programming (IP) is a natural framework for modelling this optimisation problem. A summary of IP models for the kidney exchange problem (KEP) has been presented in [8,16], where several formulations are analysed and compared in terms of tightness and experimental performance.

The most common formulations consider that there is no uncertainty associated with the data, which is not the case in practice. In fact, one of the main problems in the implementation of the solution of a KEP instance arises from data unreliability. Last-minute testing of donor and patient can elicit new incompatibilities (so-called, positive cross-matching) that were not detected before, causing some donations in a cycle to be cancelled; patients or donors may become unavailable, e.g., due to illness or to backing out. Data uncertainty is addressed by some authors by associating probabilities of failure to vertices (pairs) and arcs (compatibilities) and by considering the expected size of cycle, rather than the actual one.

A model considering a discounted utility of cycles was proposed in [11]. It takes into account a probability of failure, but rearrangement of vertices in case of failure is not considered (we will call this model "*no-recourse expectation*" in Section 3.1). A

* Corresponding author.
E-mail addresses: xenia.klimentova@inescporto.pt (X. Klimentova), jpp@fc.up.pt (J.P. Pedroso), aviana@inescporto.pt (A. Viana).

http://dx.doi.org/10.1016/j.cor.2016.03.004
0305-0548/

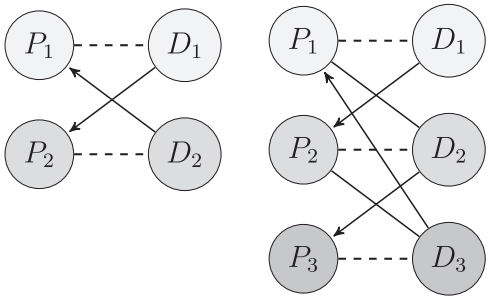

Fig. 1. An exchange between two (left) and three (right) incompatible pairs. Solid lines represent preliminary assessment compatibility and arrows define a possible exchange.

model for maximising the expected utility when arcs are subject to failure is proposed in [17] and [7]. A simulation system, where patient–donor pairs are generated and assigned to a cycle in a dynamic version of the KEP, is also proposed there. An approach for maximising expectation was studied in [20], considering both vertex and arc failure. A straightforward scheme was used for the computation of expectation, which turned out to be very limited; only experiments with k equal to 3 have been performed.

The main contributions of this paper are the following. We propose a new tree search algorithm for calculating the expected number of transplants in a KEP which allows instances of the KEP for larger values of k to be handled, as compared to the approach used in [20]. We also propose a new scheme for the rearrangement of vertices on cycles with failure — thus, with a different value for the expectation — where the recovery of broken cycles may be performed with rearrangements within a wider subset of vertices, which we believe is implementable in practice. A computational experiment was carried out to compare the different rearrangement schemes, as well as to test the algorithm used for calculating expectations. The results obtained show that by using the new rearrangement scheme a meaningful increase on the number of transplants is possible for most of the instances.

The remainder of this paper is organised as follows. A formal problem statement and IP formulation is provided in Section 2. In Section 3, three expectation schemes are presented, together with the algorithms for their computation. Results of computational experiments are presented in Section 4, which is followed by the conclusions.

2. Problem statement and IP formulation

Graph theory provides a natural framework for representing kidney exchange models. Given a directed graph $G=(V,A)$, the set of vertices V is the set of incompatible donor-patient pairs. Two vertices i and j are connected by arc $(i,j)\in A$ if donor of pair i is compatible with patient of pair j. To each arc (i,j) is associated a weight w_{ij}. If the objective is to maximise the total number of transplants, then $w_{ij}=1$, $\forall(i,j)\in A$.

The kidney exchange problem (KEP) can be defined as follows: *Find a packing of vertex-disjoint cycles with length at most k having maximum weight.*

There are several known integer programming (IP) formulations for the KEP [8]. The work presented in this paper is based on one of the computationally most effective formulations — the cycle formulation [23] — which can be described as follows. Let C be the set of all cycles in G with length at most k. We represent a cycle as an ordered set of arcs. Define variable $x_c=1$ if cycle $c\in C$ is selected for the exchange, $x_c=0$ otherwise. Denote by $V(c)\subseteq V$ the set of vertices which belong to cycle c and by w_c the weight of cycle: $w_c=\sum_{(i,j)\in c} w_{ij}$. The cycle formulation can be written as follows.

Problem $\mathcal{C}(\mathbf{k})$: (1)

$$\text{maximise} \quad \sum_{c\in C} w_c x_c \tag{2}$$

$$\text{subject to :} \quad \sum_{c:i\in V(c)} x_c \le 1 \quad \forall i\in V, \tag{3}$$

$$x_c\in\{0,1\} \quad \forall c\in C. \tag{4}$$

The objective function (2) maximises the weighted sum of the exchange. Constraints (3) ensure that each vertex is in at most one of the selected cycles (so that the corresponding patient/donor will at most receive/donate one kidney). The difficulty of this formulation is induced by the exponential number of variables in terms of k, related to the exponential number of cycles of length at most k (in the general case).

State-of-the-art kidney exchange programmes include altruistic donors, i.e., donors that are not associated with any patient, but are willing to donate a kidney to someone in need. In a kidney exchange programme, an altruistic donor initiates a chain, not a cycle: she/he gives a kidney to a patient and the recipient's donor is “dominoed” to add another incompatible pair to the chain and so on. The last donor in the chain normally gives a kidney to the next compatible patient on the deceased donors waiting list [19,13,22]. European programmes consider bounded chains with length at most k' (k' can be different from k) [18,14]. A discussion on how to extend the cycle formulation to include altruistic donors is provided in [8]. Alternatively, we may have a *non-simultaneous extended altruistic donor chain* [10,21,2,12] (also known as never-ending-altruistic-donor chain), where a kidney from the last donor in a chain (called a “bridge” donor) is not assigned to a patient in the deceased donors list; instead, the bridge acts as an altruistic donor for future matches. The cascading donor chain may continue indefinitely and the length of the chain is unbounded, unless a donor whose related recipient has already been transplanted drops out of the programme. Note that in this case simultaneity of operations is no longer a requirement. This possibility was adopted by some programmes in the USA. The inclusion of altruistic donor chains is not considered explicitly in this paper.

3. Unreliability and recourse policies

According to [11], in a particular kidney exchange programme running in the USA only 7% of the matches resulted in transplants, while 93% failed. For taking into account this observation – common in these programmes – alternative objectives for the KEP should be considered. Instead of using the cycle's length as a weight of the cycle (the most common approach [9,25]), one may consider using the *expectation* on the number of transplants that the cycle will lead to. This expectation can be computed if the probability of failure of each of the cycle's vertices and arcs is given. In this paper we will refer to the probability of failure of vertex $i\in V$ in the graph as p_i, and to the probability of failure of arc $(i,j)\in A$ as p_{ij}. Throughout the paper it is assumed that the failure events are statistically independent, and that arcs have unit weights, $w_{ij}=1$ $\forall(i,j)\in A$. We will consider three recourse schemes for the rearrangement of exchange cycles. We revisit the no-recourse and internal-recourse policies and propose a subset-recourse policy. We also propose algorithms for computing the corresponding expectations.

3.1. No-recourse expectation

In the simplest case, which we call *no-recourse* expectation, the vertices involved in any exchange cycle may not be rearranged in case of failure; the expectation of a cycle is computed as follows:

$$E_N(c) = \prod_{i \in V(c)} (1-p_i) \prod_{ij \in A(c)} (1-p_{ij})$$

where $V(c)$ and $A(c)$ are sets of vertices and arcs involved in cycle c, respectively. A similar objective was considered in [17,6,11].

To find a solution for the KEP having maximum overall no-recourse expectation, Problem $\mathcal{C}(k)$ with $w_c = E_N(c)$, $c \in C$, must be solved. We denote an optimum of this problem as E_N^*.

3.2. Internal-recourse expectations and their computation

More interesting cases of recourse policies consider the assumption that vertices involved in the cycle where failure occurs may be rearranged. In a first scheme for tackling this, it is supposed that the vertices of a cycle with failure are rearranged whenever another (shorter) cycle within them can be formed. We call this scheme and its expectations *internal-recourse*. This approach was studied in [20], and similar ideas have been presented in [17,7,6]. Here, we briefly present the most general scheme (refer to [20] for more details).

Consider the graph presented in Fig. 2. For the sake of simplicity, assume only vertex failures. The presented cycle c leads to three transplants if none of the three vertices fail, and to two transplants (cycle $(1,2)$) in case of withdrawal of vertex 3 only. Thus, the internal-recourse expectation for this cycle can be written as:

$$E(c) = 3(1-p_1)(1-p_2)(1-p_3) + 2(1-p_1)(1-p_2)p_3.$$

The expression with both vertex and arc failure has been derived in Appendix A.

More formally, the expectation for a cycle c with both vertices and arcs failure is computed as follows. For the joint set of vertices and arcs in the subgraph corresponding to the cycle, let $\Theta(c) = 2^{V(c) \cup \overline{A(c)}}$ be the set of all its subsets, where $\overline{A(c)} = \{(i,j) \in A | i,j \in V(c)\}$. Any $M \in \Theta(c)$ can be seen as the set of remaining elements of the cycle after a failure occurs. Consider the following definitions:

- $V' = V(c) \cap M$ is the set of vertices in M, i.e., the vertices remaining after failure of some components in cycle c;
- $T = V(c) \setminus V'$ is the set of failing vertices of cycle c;
- $A' = \{(i,j) \in \overline{A(c)} \cap M : i \in V', j \in V'\}$ is the set of remaining arcs of cycle c for which both vertices also remain;
- $Q = \overline{A(c)} \setminus (\overline{A(c)} \cap M)$ is the set of failing arcs.

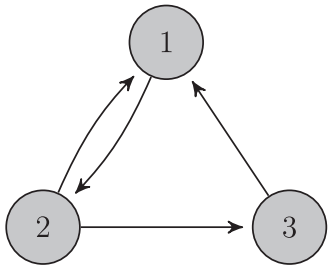

Fig. 2. Cycle c with 3 vertices.

The internal-recourse expectation for a cycle c can now be computed as follows:

$$E_I(c) = \sum_{M \in \Theta(c)} z \prod_{i \in V'} (1-p_i) \prod_{(i,j) \in A'} (1-p_{ij}) \prod_{i \in T} p_i \prod_{(i,j) \in Q} p_{ij} \quad (5)$$

where z is the optimum of Problem $\mathcal{C}(k)$ for $G' = (V', A')$, with a set of cycles $C' = \{c \in C : V' \subseteq V(c), A' \subseteq \overline{A(c)}\}$.

3.2.1. Previous approaches for computing internal-recourse expectation

Computation of internal-recourse expectation is faced with severe bottlenecks. The first difficulty arises when enumerating all the cycles in the graph representing a KEP; as their number increases exponentially with respect to the maximum cycle length, enumeration may be a limitation (which is shared by the standard cycle formulation). The second difficulty concerns formula (5) for computing each cycle's expectation. This assumes the enumeration of all subsets $M \in \Theta(c)$ of vertices and/or arcs; the number of such subsets also grows exponentially with the size of cycle. In practice, computing the expectation for given cycles is by far the limiting difficulty.

In [17] and [7] a simulation system has been implemented. However, the process of computation of expectation is not detailed. The approach of [20] details how this computation is dealt with, and is related to the concept of the *configurations* of a cycle.

Definition 1. A *configuration* of a cycle of size k is an equivalence class of isomorphic graphs with k vertices containing at least a cycle of size k.

All different configurations of a cycle of size 3, and two configurations of a cycle of size 4, are shown in Fig. 3. The approach in [20] was to first prepare a database of possible configurations of cycles up to a given size k; then, for each of them, the function calculating its expectation was stored (possibly after simplifying it, since it may become very large). For computing values of this expectation, all subsets of a set $\Theta(c)$ have been enumerated. This approach was used for maximum length of cycles $k=3$, as both the number of subsets of $\Theta(c)$ and the number of different configurations of the graph grow very rapidly with the number of vertices (the number of different configurations of size 4 is 61, and for size 5 it is 3725).

3.2.2. A new algorithm for computing internal-recourse expectation

Here we propose a new effective tree search algorithm for computing the values of internal-recourse expectations. Different from the work in [20], the new approach allows an improved enumeration of the relevant subsets of $\Theta(c)$ by fathoming redundant branches of the enumeration tree.

To proceed with the algorithm description, we need to introduce the notion of *enclosed cycle*.

Definition 2. A cycle $c \in C$ is *enclosed* in cycle $s \in C$ iff $V(c) \subseteq V(s)$.

Algorithm 1 computes internal-recourse expectations for all cycles $c \in C$ of a graph $G = (V, A)$, given the probabilities of failure

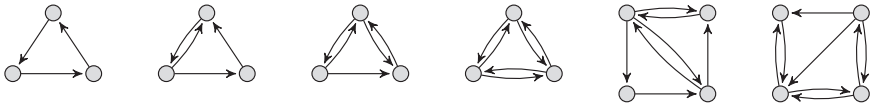

Fig. 3. Examples of configurations of cycles.

for its vertices (p_i) and arcs (p_{ij}). It computes a value of the expectation for G by implicitly enumerating all the relevant combinations of arcs and vertices failure.

Input: The algorithm starts working with the set of active nodes of the search tree $U \cup H$, where $U = V(c)$ is the sets of vertices and $H = \overline{A(c)}$ the set of arcs involved in the subgraph of considered cycle $c \in C$ (line 3); R is the set of enclosed cycles in cycle c (line 2). Throughout the algorithm, R represents the set of remaining cycles in each branch of the search tree, according to failure of elements of $U \cup H$ in parent nodes. The current expectation value s has value 1.

Branching: The set $U \cup H$ defines active nodes of the search tree for branching. On the left branch it is supposed that the chosen element t (arc or vertex) does not fail. This happens with probability $1-p_t$, and the corresponding value is multiplied by the current expectation s (s^0 in lines 20–23 and 25–28). On the right branch element t fails, with probability p_t. As a result of this failure, the sets of remaining cycles R, vertices U, and arcs H are updated (R^1, U^1, H^1, and s^1 in those lines). The chosen element t is removed from the set of active nodes (lines 21 and 27). The values obtained on the left and right branches are summed and returned (line 31).

Fathoming: A branch is fathomed in three cases:

(i) When the set of remaining cycles R is empty (line 7). The expectation for this node of the search tree is zero.
(ii) When R has a single (line 9). The corresponding expectation is computed similar to no-recourse expectation.
(iii) When a leaf of the tree is reached, i.e., the set of active nodes for branching is empty (line 13). In this case, Problem $\mathcal{C}(k)$ is solved to find a disjoint set of cycles with maximum total weight (line 15), and the optimum is multiplied by the current expectation s (line 16).

Algorithm 1. Computation of internal-recourse expectations for cycles.

Data: Set of cycles C, p_i, p_{ij}, where $i \in V, (i,j) \in A$.
Result: Expectation values $E_I(c)$ for each $c \in C$.

1 **foreach** $c \in C$ **do** `//prepare input to RecursionE`
2 $R \leftarrow \{r \in C : r \neq c, V(r) \subseteq V(c)\}$ // `set of enclosed cycles`
3 $U \leftarrow V(c)$ and $H \leftarrow \overline{A(c)}$ // `nodes for branching`
4 $E_I(c) \leftarrow$ `RecursionE`$(R, U, H, 1)$
5 **return** E_I

6 `RecursionE` (R, U, H, s)
7 **if** $R = \varnothing$ **or** $s = 0$ **then** // `fathoming`
8 **return** 0
9 **if** $|R| = 1$ **then**
10 $r \leftarrow$ element of R
11 $E \leftarrow s|V(r)| \prod_{i \in U \cap V(r)}(1-p_i)\prod_{(i,j) \in H \cap A(r)}(1-p_{ij})$
12 **return** E
13 **if** $U = \varnothing$ **and** $H = \varnothing$ **then**
14 $V' \leftarrow \bigcup_{r \in R} V(r), A' \leftarrow \bigcup_{r \in R} A(r)$
15 $z^* \leftarrow$ optimum of Problem $\mathcal{C}(k)$ on graph $G' = (V', A') E \leftarrow sz^*$
16 **return** E
17 select $t \in U \cup H$ // `branching`
18 **if** $t = u \in U$ **then** // `branching on vertex`
19 $R^0 \leftarrow R; R^1 \leftarrow \{r \in R | u \notin r\}$
20 $U^0 \leftarrow U \setminus \{u\}; U^1 \leftarrow U \setminus \{u\}$
21 $H^0 \leftarrow H; H^1 \leftarrow \{(i,j) \in H | i \neq u,\ j \neq u\}$
22 $s^0 \leftarrow s(1-p_u); s^1 \leftarrow sp_u$
23
24 **if** $t = (u,v) \in H$ **then** // `branching on arc`
25 $R^0 \leftarrow R; R^1 \leftarrow \{r \in R | (u,v) \notin r\}$
26 $U^0 \leftarrow U; U^1 \leftarrow U$
27 $H^0 \leftarrow H \setminus \{(u,v)\}; H^1 \leftarrow H \setminus \{(u,v)\}$
28 $s^0 \leftarrow s(1-p_{uv}); s^1 \leftarrow sp_{uv}$
29
30 $E^0 \leftarrow$ `RecursionE`(R^0, U^0, H^0, s^0)
31 $E^1 \leftarrow$ `RecursionE`(R^1, U^1, H^1, s^1)
return $E^0 + E^1$

Fig. 4 illustrates how the algorithm functions for a particular graph (left). For the sake of simplicity, only vertex failure is considered. The right-hand side of the figure presents the search tree for computing the expectation; decisions are taken on vertices of the left-hand graph with the same label. To each branch

corresponds an expression to be added during the computation of expectation (see lines 23, 28). Grey circles show leaves, where the values z are optima of the corresponding subproblems. Squares are branches fathomed, having either an empty set of cycles or a single cycle (indicated below them).

By following paths from the root of the tree down to each leaf, we can write the value of the expectation for this cycle. The relevant fathomed branches are marked by letters, and their expressions are highlighted with the corresponding label, as follows:

$$E_I(c) = \underbrace{2p_2(1-p_1)(1-p_4)}_{a} + \underbrace{2p_4(1-p_1)(1-p_2)(1-p_3)}_{b} + \underbrace{2p_3p_4(1-p_1)(1-p_2)}_{c} + \underbrace{4(1-p_1)(1-p_2)(1-p_3)(1-p_4)}_{d} + \underbrace{3p_3(1-p_1)(1-p_2)(1-p_4)}_{e}$$

For an illustration of the operation of the algorithm with both vertex and arc failure see Appendix A.

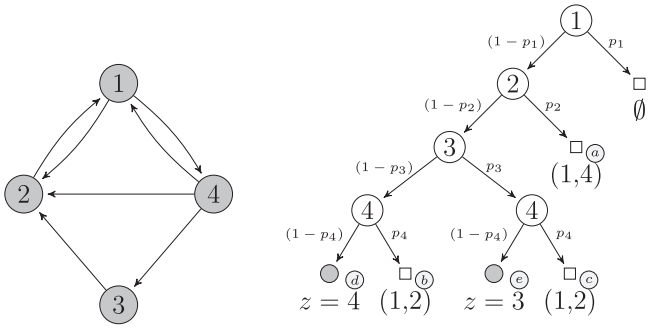

Fig. 4. Left: example of a cycle with 4 vertices. Right: search tree for calculating its expectation with *internal-recourse*.

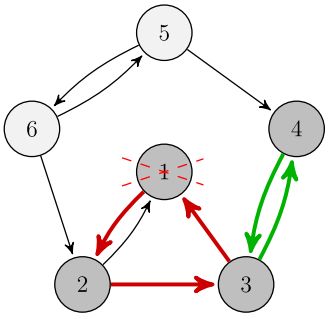

Fig. 5. Example of subset-recourse.

Branching rule: It is well-known that the choice of the element for branching (line 18) is crucial for the effectiveness of tree search. We started with random choice of the elements first from the set U and, when $U = \emptyset$, from H. However, after preliminary tests the following strategy has appeared more effective:

- If $U \neq \emptyset$, choose the vertex included in the maximum number of remaining cycles R.
- If $U = \emptyset$, choose any arc in the list H.

Solution method for problem $\mathcal{C}(k)$: When a leaf of the search tree is reached, Problem $\mathcal{C}(k)$ needs to be solved for the corresponding subgraph G' (line 15). For this purpose any IP solver could be used; however, due to the small size of graphs G' (at most k vertices), enumeration turned out to be more efficient. An enumeration algorithm was implemented and used for the computational experiments presented in Section 4.

Having found the expectation $E_I(c)$ for all cycles $c \in C$, to find a solution of the KEP that has maximum overall internal-recourse expectation value, Problem $\mathcal{C}(k)$ must be solved for the original graph with weights $w_c = E_I(c)$, $c \in C$. We denote an optimum of this problem as E_I^*.

3.3. Subset-recourse expectations

In this section we propose another possibility for the rearrangement upon failure, which considers the possibility of involving vertices not enclosed in the cycle in the rearrangement, as long as they had not been selected for a different exchange. This leads to another scheme for computing expectations, which we call *subset-recourse expectations*. It is a generalisation of the scheme used for internal-recourse expectations, presented in the previous section.

Let us consider the example in Fig. 5. In the case of internal-recourse, if vertex 1 fails then no transplantation can be performed for vertices in cycle $(1, 2, 3)$. Under subset-recourse expectations, we can consider the subset of vertices $\overline{V} = \{1, 2, 3, 4\}$, and we allow vertex 3 to be reassigned to cycle $(3, 4)$. Upon failure of vertex 1, this recourse policy still allows two transplants. We believe that this new kind of rearrangement is useful in practice; one of the main results of this paper is showing its benefits, as compared to no rearrangement and to internal rearrangement.

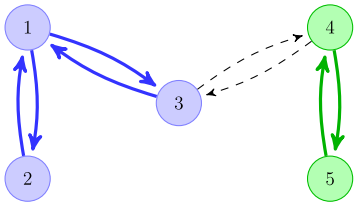

Fig. 7. Limitation of the subset-recourse expectations.

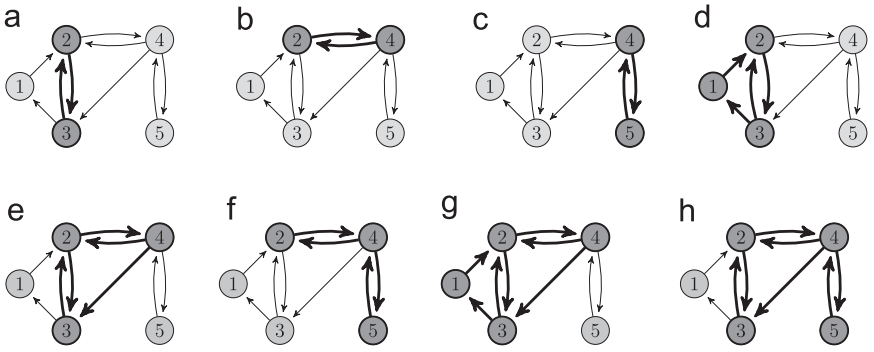

Fig. 6. Relevant subsets for an example of graph with five vertices, $k=3$, $q=1$.

Under subset-recourse rearrangement, the expected number of transplants for the set of vertices $\overline{V}$ is given as follows (again, for the sake of clarity, in this example we consider only vertex failure):

$$\begin{aligned} E_S(\overline{V}) = {} & 2(p_1+(1-p_1)p_2)(1-p_3)(1-p_4) \\ & +2p_3(1-p_1)(1-p_2)+3p_4(1-p_1)(1-p_2)(1-p_3) \\ & +4(1-p_1)(1-p_2)(1-p_3)(1-p_4) \end{aligned}$$

Recursion: Vertices of a cycle $c \in \overline{C}$ together with vertices in S form a new relevant subset if they have a nonempty intersection and satisfy the cardinality requirement (line 10). This condition provides strong connectivity of the new subset S', in accordance with Definition 3. While the size of the subset S' is less than the given limit $k+q$ (line 13), the procedure is recursively called with S' and the set of candidate cycles $\hat{C}$, modified accordingly, as input parameters (lines 14, 15).

Algorithm 2. Enumeration of all relevant subsets.

Data: Set of cycles C, max. cycle size k, max. size of relevant subsets $k+q$
Result: Set of subsets Ω for input data

```
1   Ω ← ⋃_{c∈C} {V(c)}  // all sets of vertices of cycles
2   foreach c ∈ C do    // prepare input to RecursionS
3   |  S ← V(c)                    //  current nodes in subset
4   |  C̄ ← {s ∈ C | V(s) ⊈ V(c)}   //  set of candidate cycles
5   |  if |S| < k+q then
6   |  |  RecursionS Ω, S, C̄, k, q
7   return Ω

8   RecursionS Ω, S, C̄, k, q
9   |  for c ∈ C̄ do
10  |  |  if V(c) ∩ S ≠ ∅ and |V(c) ∪ S| ≤ k+q then
11  |  |  |  S' ← V(c) ∪ S
12  |  |  |  Ω ← Ω ∪ {S'}
13  |  |  |  if |S'| < k+q then
14  |  |  |  |  Ĉ = {c ∈ C̄ | V(c) ⊈ S'}
15  |  |  |  |  RecursionS(Ω, S', Ĉ, k, q)
```

To take into account practical considerations (e.g., that a crossmatch validation test must be made among all pairs in a subset), it is assumed that rearrangements can be made only within vertices of one subset. Furthermore to limit the combinatorial explosion, it is assumed that rearrangements to be considered for a cycle with failures are restricted to a small subset of extra vertices. These are limited in number and, to be relevant, must form at least one cycle of acceptable size within the subset. For example, for the graph of Fig. 5, the subset of vertices $\{1,2,3,4\}$ is relevant; however, subset $\{1,2,3,5\}$ is not, as vertex 5 does not form a cycle within vertices of $\{1,2,3\}$. The formal definition of the relevant subset for this work is provided next, where strong connectivity is discussed, e.g., in [5].

Definition 3. *A relevant subset S of size* (k,q) is the set of at most $(k+q)$ vertices in graph G inducing a strongly connected subgraph such that any arc of the paths that provide the strong connectivity belongs to some cycle c of size at most k, and $V(c) \subseteq S$.

Note that within this definition for any cycle c in a subset S, i.e., $V(c) \subseteq S$, the number of extra vertices to make rearrangements in case of failures is equal to $(|S| - |V(c)|)$.

A method for the recursive construction of the set of relevant subsets Ω is presented in Algorithm 2.

Input: The set of relevant subsets Ω is initialised with the sets of vertices of each cycle $c \in C$ (line 1). Then, for each cycle $c \in C$ an initial subset S is given by $V(c)$ (line 3). The set of cycles whose vertices might augment the current subset S are stored in $\overline{C}$ (line 4).

Fig. 6 illustrates all subsets for a given graph, for $k=3$ and $q=1$. Vertices belonging to each subset are shown in dark grey, and cycles having all their vertices included into a subset are shown with bold arcs. Subsets (a), (b), (c), (d) and (e) directly correspond to cycles, while subsets (f), (g), (h) are constructed as described above.

Let us denote by $\Omega(k,q)$ the set of all relevant subsets for a given graph and given values k and q. For each subset $S \in \Omega(k,q)$, its expectation value under subset-recourse is calculated by means of `RecursionE`, defined in Algorithm 1; this value can be found in a straightforward fashion after preparing the input as shown in Algorithm 3.

Algorithm 3. Computation of subset-recourse expectations.

Data: Set of cycles C of size up to k, set of subsets Ω, p_i, p_{ij}, for $i \in V, (i,j) \in A$.
Result: Expectation values $E_S(S)$ for each $S \in \Omega$.

```
1  foreach S ∈ Ω do    // prepare input to RecursionE
2  |  R ← {c ∈ C : V(c) ⊆ S}    // set of internal cycles
3  |  H ← ⋃_{c∈R} A(c)          //nodes for branching
4  |  E_S(S) ← RecursionE(R, S, H, 1)
5  return E_S
```

To find a solution to the KEP based on subset-recourse expectation, we may use a formulation similar to the cycle formulation. Taking into account that rearrangements must be done within the vertices of one subset, we seek a set of disjoint subsets providing

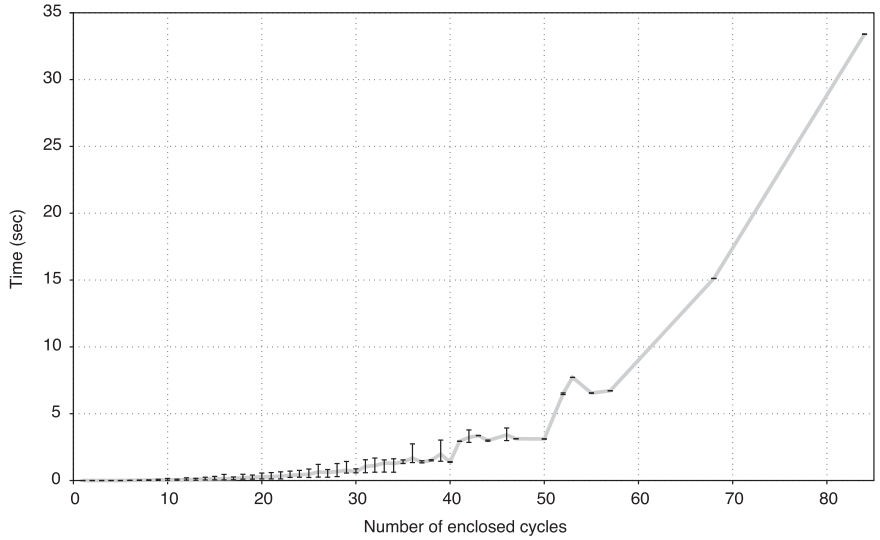

Fig. 8. Time needed for computing expectations for all configurations of size 5.

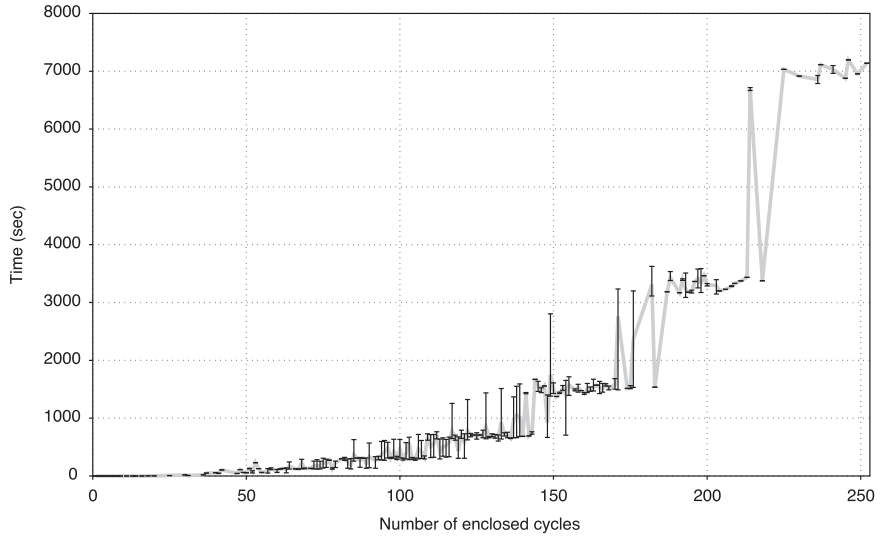

Fig. 9. Time needed for computing expectations for configurations of size 6.

the maximum expectation. This may be done by solving the following problem, where variable y_S is 1 if subset $S \in \Omega$ is selected, 0 otherwise.

Problem $\mathcal{P}(k,q)$: (6)

$$\text{maximise} \quad \sum_{S \in \Omega(k,q)} E_S(S) y_S \tag{7}$$

$$\text{subject to :} \quad \sum_{S: i \in S} y_S \leq 1 \quad \forall i \in V, \tag{8}$$

$$y_S \in \{0,1\} \quad \forall S \in \Omega(k,q). \tag{9}$$

We denote the optimum for problem $\mathcal{P}(k,q)$ as E^*_S.

In kidney exchange programmes after solution obtention, the pairs involved in the chosen cycles must be informed of their selection for transplantation. For an exchange based on the subset-recourse solution this has to be handled differently, because not all pairs involved in a selected subset will be actually subject to transplantation. Hence, firstly one must decide, for a selected subset S, which cycles to use. This can be done in several ways, depending on the time of observation of the failures. For example, the cycle packing leading to the maximum number of possible transplants in the subgraph defined by the set of vertices S may be attempted, and, if some failures are observed, recourse cycles in S are sought.

Because not all possible rearrangements are considered in the subset-recourse procedure this policy presents some limitations. To illustrate those limitations consider the graph presented in Fig. 7. Let $k=2$ and $q=1$. An optimal solution for the subset-recourse formulation is given by subsets $\{1,2,3\}$ and $\{4,5\}$. In case of failure of vertices 5 and 1, the value determined for the expectation is zero; however, cycle $(3,4)$, with 2 transplants, is still available. Cycle $(3,4)$ is neither considered in the expectation value of the subset $\{1,2,3\}$, nor of the subset $\{4,5\}$. Hence, in case this arrangement is possible, the value computed is a lower bound to the true expectation. This limitation is overcome by considering larger subsets: e.g., when $q \geq 3$, the expectation computed for the subset $\{1,2,3,4,5\}$ is the correct one.

4. Computational experiments

In this section we report computational experiments that were carried out to evaluate the performance of the tree search method proposed for computing expectations, and to analyse the results obtained with the various schemes. All the results were obtained with a computer with an Intel Xeon processor at 3.00 GHz, 8 GB of RAM and running Linux. The algorithms presented were implemented in the Python language, version 2.7, using Gurobi version 5.0 as the optimiser.[1]

Computational experiments were carried out in three main parts:

- Section 4.1 presents test results of the efficiency of Algorithm 1.
- Section 4.2 shows a comparison of the results obtained with the deterministic approach to those obtained by maximising the expected number of transplants under no-recourse and internal-recourse.
- Section 4.3 compares subset-recourse to the best expectation scheme among those tested in the previous step.

For Sections 4.2 and 4.3 the instances used have been created by the most commonly used generator, described in [24]. The generator creates random graphs based on probabilities of blood type and of donor–patient tissue compatibility. Default values for the generator's parameters were used. The probabilities of failure p_{ij}, $(i,j) \in A$, and p_i, $i \in V$ were generated randomly with uniform distribution in $[0, 1]$. The same test instances have been used in the computational experiments reported in [8,16], and values for probabilities are those of [20]. Graphs have different number of vertices n, starting from 10 up to 300. For each size up to 100 vertices, 50 instances are considered. For bigger n there were 10 instances of each size. Experiments were made for values of k ranging from 3 to 6.[2]

4.1. Efficiency of the tree search method

To evaluate the effectiveness of the proposed algorithm for computing expectations for particular cycles, we computed the value of the expectation for different configurations (see Definition 1 in Section 3.2.1). The configurations were generated dynamically by adding all possible combinations of arcs to the cycle of size k and checking whether it is isomorphic to any of the currently generated configurations. Due to the number of different configurations and to the complexity of computation of the expected values, for $k=6$ we randomly selected only 500 different graphs.

When considering internal-recourse, the number of enclosed cycles for a given configuration has a crucial influence on the behaviour of tree search. In this study we analysed the time required for computing the expectation in terms of the number of enclosed cycles, for different configurations of cycles of a given length. The number of enclosed cycles has been determined by enumeration.

For configurations of size 3 and 4 the expectations for all possible configurations were computed in less than 1 s; tree search is very fast for these cases, which hence are not further analysed. Fig. 8 illustrates the time spent by the algorithm to compute expectations for configurations of size 5 as a function of the number of enclosed cycles.

There usually exist several configurations with the same number of enclosed cycles; in these cases the average time is plotted, and a bar between maximum and minimum computational times is presented. Often, there is no difference in these values, and the bar appears as a point. For configurations with up to 50 enclosed cycles the expectations were computed within 5 s. For more dense configurations, the computational time grows rapidly, up to 33 s for the configuration corresponding to a complete graph with 5 vertices. In this case there are 84 enclosed cycles.

Results for configurations of size 6 are presented in Fig. 9. The time used by the algorithm was limited to 2 h (7200 s). Due to this limitation, only instances with at most 250 enclosed cycles have been considered. As some instances with less than 250 cycles could not also be solved within the time limit, cases of more dense graphs have not been attempted. Note that the configuration corresponding to a complete graph of size 6 has 409 enclosed cycles. As in the case of graphs of size 5, we can observe a drastic rise in the time used: about 1500 s are enough for graphs with up to 170 cycles; close to 3500 s are required for graphs with 190–210 cycles; and 7000 s or more have been used for graphs with more than 220 cycles. In contrast with the results for size 5, the computational time for graphs with the same number of enclosed cycles may vary significantly; e.g., for graphs with 138 cycles (7 different configurations have been considered), computing the expectation required from 662 to 1551 s, with an average of 1054 s. Moreover, as compared to configurations of size 5, much steeper spikes can be observed in this plot.

For configurations of size 7 only a small part of the graphs could be handled within the time limit of 2 h, and therefore we do not present results for this case.

4.2. Deterministic case, no-recourse, and internal-recourse

The aim of the second computational experiment was to compare results for the KEP in a deterministic setting to the no-recourse and internal-recourse cases. These three approaches are based on the same model for solving the main problem (cycle formulation $\mathcal{C}(k)$); they differ only in the weight w_c attributed to each of the cycles:

1. Deterministic approach: $w_c = |c|$, maximising the attainable number of transplants; let z^* denote the optimum of this problem.
2. No-recourse expectation: $w_c = E_N(c)$; the optimum of this problem is denoted by E_N^*.
3. Internal-recourse expectation: $w_c = E_I(c)$; the optimum is denoted by E_I^*.

For the sake of comparison, we also computed E_D to be the expected number of transplants of the optimal solution for the deterministic problem, if internal-recourse is applied afterwards. The time used by the algorithm was limited to 1 h (3600 s) for each instance. In Table 1 we show in parenthesis the number of instances out of 50 or out of 10 that were not solved within this time frame. Note that for graphs with 10 vertices, out of the 50 cases considered, only 36 had non-null solutions; the remaining 14 did not have any feasible cycle. All the instances with a larger number of vertices had non-null solutions. This table shows the average values of z^*, E_D, E_N^*, and E_I^* for the set of instances of each size.

First of all we should mention that the proposed computational algorithm is less effective for $k=3$ than the approach in [20]. That paper reports the computational experiments with instances up to 1000 vertices; we are only considering instances with up to 300 vertices. However our approach is useful for bigger values of

[1] Gurobi Optimisation. http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview/ (last accessed in September, 2014).

[2] Currently implemented kidney exchange programmes work in general with a maximum value of $k=3$. However, this value has already been exceeded in practice: the maximum number of simultaneous transplants that we are aware of is 6 [3].

Table 1
Average optima obtained for deterministic and non-deterministic models.

k	n		$\overline{z^*}$	$\overline{E_D}$	$\overline{E_N^*}$	$\overline{E_I^*}$
3	10		3.8	0.24	0.28	0.34
	20		8.2	0.40	0.88	1.01
	30		12.3	0.71	1.30	1.50
	40		17.3	1.05	1.88	2.21
	50		23.5	1.31	2.91	3.35
	60		27.6	1.66	3.64	4.26
	70		34.0	1.89	4.24	4.93
	80		39.5	2.28	5.43	6.26
	90		46.2	2.63	6.45	7.47
	100		51.3	2.86	7.20	8.31
	200		21.4	1.09	3.75	4.28
	300	(3)	23.7	1.21	4.29	4.87
4	10		3.8	0.23	0.28	0.37
	20		8.6	0.53	0.88	1.19
	30		12.7	0.75	1.30	1.77
	40		18.1	1.00	1.88	2.54
	50		24.3	1.52	2.92	3.92
	60		28.8	1.86	3.65	4.91
	70		35.1	1.92	4.25	5.74
	80		40.4	2.42	5.45	7.39
	90	(3)	46.6	2.64	6.31	8.53
5	10		3.8	0.26	0.28	0.39
	20		8.7	0.49	0.88	1.28
	30	(3)	12.5	0.85	1.24	1.79
	40	(8)	17.4	1.08	1.78	2.57
	50	(30)	23.1	1.70	2.81	4.00
6	10		3.8	0.28	0.28	0.40
	20	(25)	6.6	0.33	0.64	0.92

$k=4,5,6$. The average expectation $\overline{E_D}$ for the optimal deterministic solution is significantly inferior to the solutions obtained by models that explicitly consider rearrangements, either with no-recourse or internal-recourse. Overall, the expected no-recourse values exceed the deterministic approach by a factor of 1.7 on average. The internal-recourse scheme allows an improvement over E_D by a factor of 2.4. Most strikingly, for larger instances with 80 and 90 vertices with $k=4$, the average value $\overline{E_I^*}$ exceeds $\overline{E_D}$ more than 3 times. Given this superiority, the internal-recourse scheme was selected as the approach for comparison with subset-recourse expectations.

4.3. Subset-recourse

The last stage of the computational experiment is devoted to comparing expectations obtained with internal-recourse and subset-recourse models. For each instance, Problem $\mathcal{P}(k,q)$ was solved for $k=3,4,5$, $q=0,\ldots,7$, the optimum being denoted by E_S^q*. Then, for all consecutive pairs $(q-1,q)$, $q=1,\ldots,7$ the expectation values have been compared by computing the relative percentage improvement from $E_S^{(q-1)}*$ to E_S^q*:

$$\Delta_q = 100 \times \left(\frac{E_S^q*}{E_S^{(q-1)}*} - 1 \right),$$

and $q=0$ was compared to internal-recourse expectation E_I^*, denoted in Table 2 by I:

$$\Delta_0 = 100 \times \left(\frac{E_S^0*}{E_I^*} - 1 \right).$$

The computational time allowed for solving each instance was limited to 3600 s. Column $I \rightarrow 0$ reports improvement from internal-recourse to subset-recourse with $q=0$; the following columns $(q-1) \rightarrow q$, for $q=1,\ldots,7$, concern improvement from $q-1$ to q. Each column presents values for each graph size in two lines:

- upper line values have the form N/M, where N is the number of instances considered and M is the number of cases where there was an improvement (i.e., the number of instances with $\Delta_q > 0$);
- in the lower line is the average value of Δ_q, i.e., the average of percentage improvements obtained.

Dashes indicate that no instance has been considered, due to the time limitation.

The limit on the computational time allocated for the solution of each instance allowed the most interesting cases to be solved. Large improvements occur with small values of q; for greater q,

Table 2
Comparison of optimum expectations for internal-recourse (I) to subset-recourse, with increasing q. Columns $q \rightarrow (q+1)$ show improvements when going from subset recourse with q, to $q+1$ extra vertices.

k	n	$I \rightarrow 0$	$0 \rightarrow 1$	$1 \rightarrow 2$	$2 \rightarrow 3$	$3 \rightarrow 4$	$4 \rightarrow 5$	$5 \rightarrow 6$	$6 \rightarrow 7$
3	10	50/19	50/24	50/17	50/16	50/11	48/8	47/1	46/0
		11.0%	12.2%	3.8 %	1.7%	0.9%	0.4%	$<0.1\%$	–
	20	50/41	50/46	50/46	43/35	27/19	18/9	17/5	15/2
		11.7%	13.4%	6.8%	3.9%	1.7%	0.6%	0.3%	0.3%
	30	50/46	50/50	50/49	27/25	11/8			
		9.9%	12.8%	7.6%	5.1%	1.6%	–	–	–
	40	50/48	50/50	50/49					
		9.0%	13.2%	6.4%	–	–	–	–	–
	50	50/50	50/50	44/44					
		8.9%	13.7%	7.3%	–	–	–	–	–
	60	50/50	50/50	35/35					
		8.9%	12.7%	7.4%	–	–	–	–	–
	70	50/50	50/50	25/25					
		8.9%	13.3%	8.5%	–	–	–	–	–
	80	50/50	50/50						
		8.7%	14.4%	–	–	–	–	–	–
	90	50/50	50/50						
		8.3%	13.9%	–	–	–	–	–	–
	100	50/50	50/50						
		8.5%	13.7%	–	–	–	–	–	–
4	10	50/20	50/17	50/16	49/12	45/7	45/1		
		18.2%	4.4%	2.3%	1.0%	0.3%	$<0.1\%$		
	20	50/44	50/46	37/31	19/11	14/5	13/3		
		10.3%	8.1%	4.4%	1.8%	0.2%	0.1%		
	30	50/47	50/49	24/22	6/3	3/1	3/0		
		6.3%	9.1%	6.2%	1.6%	$<0.1\%$	–		
	40	50/49	50/49	11/11					
		9.3%	8.7%	7.1%	–	–	–		
	50	50/50	43/43						
		7.4%	8.2%	–	–	–	–		
	60	50/50	32/32						
		7.6%	8.9%	–	–	–	–		
	70	48/48	19/19						
		7.5%	9.7%	–	–	–	–		
	80	34/34	5/5						
		6.8%	10.4%	–	–	–	–		
	90	20/20							
		6.8%	–	–	–	–	–		
5	10	50/24	50/16	49/12	45/7	45/2			
		20.4%	2.2%	1.0%	0.3%	$<0.1\%$			
	20	50/46	37/31	18/10	14/5	13/3			
		11.5%	4.5%	1.5%	0.2%	0.1%			
	30	50/47	23/21	4/2	3/1	3/0			
		8.3%	6.0%	2.8%	$<0.1\%$	–			
	40	46/45	9/9						
		10.0%	7.7%	–	–	–			
	50	26/26	2/2						
		7.6%	8.0%	–	–	–			

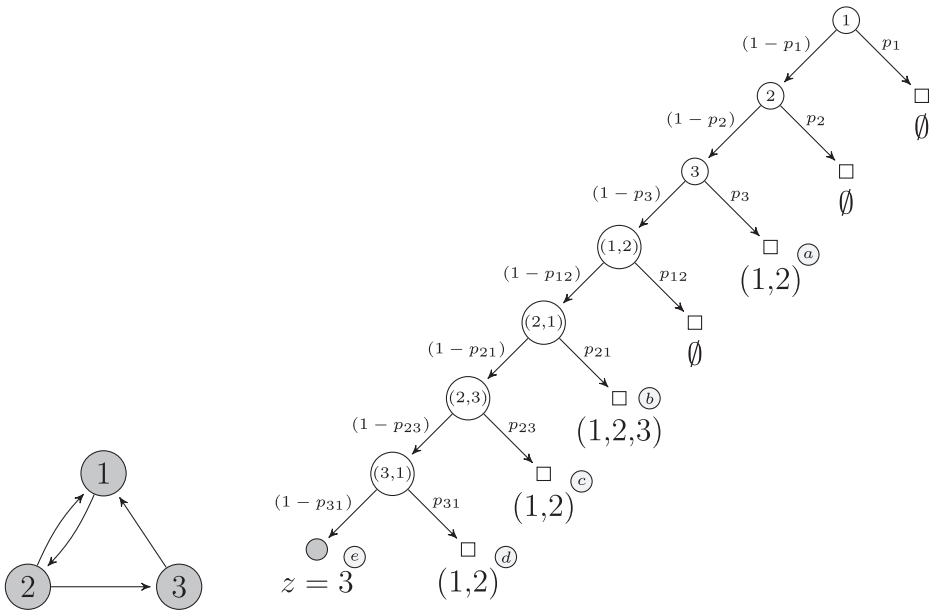

Fig. 10. Left: example of a cycle with 3 vertices. Right: search tree for calculating its expectation with *internal-recourse*.

smaller improvements on expectations were obtained, and for fewer instances. Improvements from I to $q=0$ (column $I \rightarrow 0$), and from $q=0$ to $q=1$ (column $0 \rightarrow 1$), show a clear superiority of subset-recourse with respect to simple internal-recourse.

5. Conclusions

This paper is devoted to a variant of the kidney exchange problem where the objective is to find exchanges that maximise the expected number of transplants. Data for this problem are a graph and probabilities of failure of its components (vertices or arcs). The graph describes patient–donor pairs (the set of vertices) and compatibilities between donors and patients of other pairs (the set of arcs). Failure probabilities concern the possibility of patient–donor pairs withdrawing (e.g., due to backing out), and of arcs connecting them to vanish (e.g., due to incompatibilities found in a last-minute check).

In this context, for determining the expectation for the number of transplants associated to a given cycle, the form of reorganisation of a cycle upon failure of some of its components must be defined. Three rearrangement possibilities have been considered: no-recourse, internal-recourse, and, newly proposed in this paper, subset-recourse. This last scheme considers an extended subset of vertices that can be used for rearrangements after failure of a cycle. We have proposed an algorithm for computing internal-recourse and subset-recourse expectations, based on tree search. These values are required for setting up the optimisation problem of finding the feasible exchanges that maximise the overall expected number of transplants.

A thorough computational experiment indicated that the expected number of transplants obtained by internal-recourse significantly improve on no-recourse, confirming results obtained by related approaches; these had been presented in [11,17,7,6], albeit for more restricted situations. The algorithm proposed allowed internal-recourse expectations to be computed for problems with maximum length of cycles 4, 5 and 6; this complements [20], where only results for cycles with length 3 are reported.

Results obtained for subset-recourse indicate a significant, additional improvement with respect to internal-recourse expectations, showing evidence of the outcomes that can be obtained if subset-recourse is put in practice.

Acknowledgements

We would like to thank two anonymous reviewers for their constructive comments on a previous version of this paper. This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) within projects PTDC/IIMGES/2830/2014 (mKEP - Models and optimisation algorithms for multi-country kidney exchange programs) and SFRH/BPD/101134/2014.

Appendix A. Illustration of Algorithm 1 functioning with both vertex and arc failure

Fig. 10 (right) illustrates the functioning of algorithm 1 for the graph in Fig. 10 (left).

The expressions below are highlighted with the corresponding labels of the relevant fathomed branches of the tree:

$$E_I(c) = \underbrace{2p_3(1-p_1)(1-p_2)(1-p_{12})(1-p_{21})}_{a}$$
$$+\underbrace{3p_{21}(1-p_1)(1-p_2)(1-p_3)(1-p_{12})(1-p_{23})(1-p_{31})}_{b}$$
$$+\underbrace{2p_{23}(1-p_1)(1-p_2)(1-p_3)(1-p_{12})(1-p_{21})}_{c}$$
$$+\underbrace{2p_{31}(1-p_1)(1-p_2)(1-p_3)(1-p_{12})(1-p_{21})(1-p_{23})}_{d}$$
$$+\underbrace{3(1-p_1)(1-p_2)(1-p_3)(1-p_{12})(1-p_{21})(1-p_{23})(1-p_{31})}_{e}.$$

In the minimal form the expression is written as follows:

$$E_I(c) = (1-p_2)(1-p_1)(1-p_{12})\big[3-p_3(1-p_{23})(2p_{21}+1)(1-p_{31})$$
$$-p_{23}(2p_{21}+1)(1-p_{31})-2p_{21}p_{31}-p_{31}\big].$$

References

[1] Abraham D, Blum A, Sandholm T. Clearing algorithms for Barter exchange markets: enabling nationwide kidney exchanges. In: Proceedings of the 8th ACM conference on electronic commerce, June 13–16; 2007. p. 295–304.

[2] Ashlagi I, Gilchrist D, Roth A, Rees M. Nonsimultaneous chains and dominos in kidney paired donation - revisited. Am J Transplant 2011;11(5):984–94.

[3] BBC. BBC news website. six-way kidney transplant first (9/04/2008). ⟨http://news.bbc.co.uk/1/health/7338437.stm⟩(last accessed in December, 2012); 2008.

[4] Biro P, Manlove D, Rizzi R. Maximum weight cycle packing in directed graphs, with application to kidney exchange programs. Discrete Math Algorithms Appl 2009;1(4):499–517.

[5] Bondy J, Murty U. Graph theory with applications. New York: North-Holland; 1976.

[6] Bray M, Wang W, Song P, Leichtman A, Rees M, Ashby V, et al. Planning for uncertainty and fallbacks can increase the number of transplants in a kidney-paired donation program. Am J Transplant 2015;15(10):2636–45.

[7] Chen Y, Li Y, Kalbfleisch J, Zhou Y, Leichtman A, Song P-K. Graph-based optimization algorithm and software on kidney exchanges. IEEE Trans Biomed Eng 2012;59(7):1985–91.

[8] Constantino M, Klimentova X, Viana A, Rais A. New insights on integer-programming models for the kidney exchange problem. Eur J Oper Res 2013;231(1):57–68.

[9] de Klerk M, Keizer K, Claas F, Haase-Kromwijk B, Weimar W. The Dutch national living donor kidney exchange program. Am J Transplant 2005;5:2302–5.

[10] Dickerson J, Procaccia A, Sandholm T. Optimizing kidney exchange with transplant chains: Theory and reality. In: AAMAS-12: Proceedings of 11th international joint conference on autonomous agents and multiagent systems, June; 2011.

[11] Dickerson J, Procaccia A, Sandholm T. Failure-aware kidney exchange. In: EC-13: Proceedings of 14th ACM conference on electronic commerce, June; 2013.

[12] Gentry S, Montgomery R, Segev D. Kidney paired donation: fundamentals, limitations, and expansions. Am J Kidney Dis 2010;57(1):144–51.

[13] Gentry S, Montgomery R, Swihart B, Segev D. The roles of dominos and nonsimultaneous chains in kidney paired donation. Am J Transplant 2009;9:1330–6.

[14] Glorie K, Wagelmans A, van de Klundert J. Iterative branch-and-price for large multi-criteria kidney exchange. Econometric institute report, 2012–11; 2012.

[15] Huang C. Circular stable matching and 3-way kidney transplant. Algorithmica 2010;58:137–50.

[16] Klimentova X, Alvelos F, Viana A. A new branch-and-price approach for the kidney exchange problem. In: Murgante B, Misra S, Rocha AMA, Torre C, Rocha JG, Falcão MI, et al., editors. Computational science and its applications ICCSA 2014, Lecture notes in computer science, vol. 8580. Switzerland: Springer International Publishing; 2014. p. 237–52.

[17] Li Y, Song P, Zhou Y, Leichtman A, Rees M, Kalbfleisch J. Optimal decisions for organ exchanges in a kidney paired donation program. Stat Biosci 2014;6(1):85–104.

[18] Manlove D, O'Malley G. Paired and altruistic kidney donation in the UK: algorithms and experimentation. ACM J Exp Algorithmics, 2014; 19(2):2.6:1.1–21.

[19] Montgomery R, Gentry S, Marks W, Warren D, Hiller J, Houp J, et al. Domino paired kidney donation: a strategy to make best use of live non-directed donation. Lancet 2006;368(9533):419–21.

[20] Pedroso JP. Maximizing expectation on vertex-disjoint cycle packing. In: Murgante B, Misra S, Rocha AMA, Torre C, Rocha JG, Falcão MI, Taniar D, Apduhan BO, Gervasi O, editors. Computational science and its applications ICCSA 2014, Lecture notes in computer science, vol. 8580. Switzerland: Springer International Publishing; 2014. p. 32–46.

[21] Rees M, Kopke J, Pelletier R, Segev D, Rutter M, Fabrega A, et al. A non-simultaneous, extended, altruistic-donor chain. N Engl J Med 2009;360:1096–101.

[22] Roth A, Sönmez T, Ünver M. Kidney exchange. Q J Econ 2004;119(2):457–88.

[23] Roth A, Sönmez T, Ünver M. Efficient kidney exchange: coincidence of wants in markets with compatibility-based preferences. Am Econ Rev 2007;97(3):828–51.

[24] Saidman S, Roth A, Sönmez T, Ünver M, Delmonico F. Increasing the opportunity of live kidney donation by matching for two- and three-way exchanges. Transplantation 2006;81:773–82.

[25] Segev D, Gentry S, Warren D, Reeb B, Montgomery R. Kidney paired donation and optimizing the use of live donor organs. J Am Med Assoc 2005;293(15):1883–90.