



Maximizing the expected number of transplants in kidney exchange programs with branch-and-price

Filipe Alvelos¹  · Xenia Klimentova²  ·
Ana Viana^{2,3} 

© Springer Science+Business Media, LLC 2017

Abstract In this paper, we propose a branch-and-price approach for solving the problem of maximizing the expected number of transplants in Kidney Exchange Programs (KEPs). In these programs, the decision on which transplants will be conducted is usually made with the support of optimization models with the assumption that all operations will take place. However, after a plan of transplants is defined, a pair may leave the KEP or a more accurate compatibility evaluation exam may invalidate a transplant. To model these possible events we consider probabilities of failure of vertices and of arcs and the objective of maximizing the *expected* number of transplants. The proposed approach is based on the so-called cycle formulation, where decision variables are associated with cycles. Built on the concept of *type of cycle* a branch-and-price algorithm is conceived. One subproblem is defined for each type of cycle. We present computational results of the proposed branch-and-price algorithm and compare them with solving directly the cycle formulation (with a general purpose mixed

This work is financed by the ERDF—European Regional Development Fund through the Operational, Programme for Competitiveness and Internationalization—COMPETE 2020, Programme and by National Funds through the Portuguese funding agency, FCT—Fundação para a Ciência e a Tecnologia within project “mKEP—Models and optimization algorithms for multi-country kidney exchange programs” (POCI-01-0145-FEDER-016677), is also financed by COMPETE: POCI-01-0145-FEDER-007043 and FCT within the Project Scope: UID/CEC/00319/2013 and FCT Project SFRH/BPD/101134/2014.

✉ Filipe Alvelos
falvelos@dps.uminho.pt

Xenia Klimentova
xenia.klimentova@inesctec.pt

Ana Viana
aviana@inesctec.pt

¹ Centro Algoritmi / Departamento de Produção e Sistemas, Universidade do Minho, 4710-057 Braga, Portugal

² INESC TEC, Campus da FEUP, 4200-465 Porto, Portugal

³ ISEP - School of Engineering, Polytechnic of Porto, 4200-072 Porto, Portugal

integer programming solver—CPLEX) showing that the proposed approach is the only one suitable for larger instances.

Keywords Kidney exchange problem · Expected number of transplants · Integer programming · Branch-and-price

1 Introduction

The problem addressed in this paper arises in the framework of Kidney Exchange Programs (KEPs). KEPs are meant to facilitate kidney transplants for patients who have someone willing to donate them a kidney but cannot do it because of blood and/or tissue incompatibilities. Those issues may be overcome by an exchange of donors between patients, as long as it is assured that each patient associated with a donor that provides a kidney also receives one.

The simplest form of this mechanism is for two incompatible pairs. The donor and patient of each pair are incompatible, but if the donor in one pair is compatible with the patient in the second pair and vice versa, the patients may exchange donors and the two transplants become possible. This mechanism can be generalized to exchanges involving more than two pairs although, in practice, exchanges involving a large number of pairs are not usually desirable, for logistics reasons and to reduce the impact of the cancellation of some planned transplants. The maximum number of transplants in a feasible exchange is represented by K .

In some KEPs donors that are not associated with any patient, but are willing to donate a kidney to someone in need (altruistic donors) are involved. Such donors initiate another type of exchanges: she/he gives a kidney to a patient, the patient's associated donor donates to a patient in another incompatible pair and so on. The last donor in this chain normally gives a kidney to the next compatible patient on the deceased donors waiting list.

In the presence of a large number of incompatible donor/patient pairs, optimization problems to decide which exchanges should be planned arise. Some of these problems have been tackled successfully by integer programming. The most common one has the objective of maximizing the number of transplants (Constantino et al. 2013; Klimentova et al. 2014) with the assumption that all planned transplants will actually be conducted. Other objectives can also be found in literature (Manlove and O'Malley 2014; Glorie et al. 2014), particularly embedded in lexicographic optimization. For integer programming models and a recent survey of approaches for KEP problems, the interested reader is referred to Dickerson et al. (2016) and Mak-Hau (2017).

Although these models consider that all planned transplants will be performed, typically that is not the case in practice. At the time the exchanges are planned, the information is incomplete in the sense that latter additional compatibility testing of donor and patient can elicit incompatibilities (so called, positive crossmatch) that were not detected before. Furthermore patients or donors may become unavailable, e.g. due to illness or to backing out, among other reasons. In such cases, some or all of the planned transplants may not proceed and the number of actual transplants may be severely impacted (Dickerson et al. 2013).

The main issue when dealing with this uncertainty is the rearrangement of pair assignment when some planned exchanges fail. Robust optimization has been used to address this uncertainty. In Glorie et al. (2014) various recourse policies that determine the allowed post-matching actions are proposed. For each recourse policy, a robust model is proposed and solution techniques are developed. A different approach is to give priority from the begin-

ning to exchanges which allow the rearrangement of transplants. For example, by hierarchical optimization where we maximize first the number of exchanges with three transplants that include exchanges with two transplants (if one transplant fails, an exchange of two transplants is still possible). After, other objectives can be considered in a lexicographical manner. This approach was proposed in [Manlove and O'Malley \(2012\)](#) where the first objective is to maximize the number of effective 2-cycles (exchanges with two pairs, or exchanges with three pairs that include at least one exchange with two pairs). Another objective is to maximize the number of exchanges with two transplants inside the exchanges with three transplants.

A third approach is to consider probabilities of failure related to the availability of pairs, and probabilities of failure related to additional incompatibility information. These probabilities are used to calculate the expected number of transplants in each potential exchange. The calculated values are then used in the objective of maximizing the expected number of overall transplants. The approach was studied in [Li et al. \(2014\)](#), [Klimentova et al. \(2016\)](#) and [Dickerson et al. \(2013\)](#). In those works, different probabilities of failure are assumed for each element. Some of the schemes for computing the expected value of each potential exchange are described in [Pedroso \(2014\)](#), [Li et al. \(2014\)](#) and [Klimentova et al. \(2016\)](#) based on the possibility of rearrangement of transplants when the exchange fails. The problem is solved by enumerating all the cycles, computing the expected number of transplants of each cycle, and including all the corresponding decision variables in the cycle formulation which is solved directly by a general purpose integer programming solver.

This paper follows this last type of approach and develops a branch-and-price algorithm to obtain the exchanges that should be planned so that the expected number of actual transplants is maximized. We consider that all pairs have the same probability of becoming unavailable and that all potential transplants have the same probability of being canceled due to incompatibility between the selected donor and patient. This assumption is based on two reasons. Firstly, distinct probabilities requires the definition of a procedure for estimating the probability of failure of each specific element, which is, in itself, a challenging open question. Using aggregated historic data for estimating a single probability value for failure of pairs and a single probability value for incompatibilities detected latter seems more suitable in practice. Secondly, distinct probabilities favour some (easier-to-match) patients with respect to others. Although justifiable by clinical reasons, at least partially, they may not be easily perceived as fair. In any case, the proposed branch-and-price approach can be extended to the case where different probabilities are considered, at the expense of spending more time in calculating the cycles expectations.

Another two approaches have been attempted for the maximization of the expected number of transplants when the probabilities are equal. In [Alvelos et al. \(2015\)](#), a compact integer programming model was proposed with decision variables associated with the assignment of vertices to positions in types of cycles. In [Alvelos et al. \(2016\)](#) a branch-and-price algorithm was proposed based on solving subproblems where the decision variables are associated with the assignment of arcs to positions in types of cycles. Preliminary tests showed that both approaches are not promising in computational terms, and therefore we focused in the branch-and-price approach described in this paper.

Branch-and-price is considered to be the most effective approach for the maximization of the number of planned transplants in KEPs ([Glorie et al. 2014](#); [Plaut et al. 2016](#); [Dickerson et al. 2016](#); [Klimentova et al. 2014](#)). Therefore it is a natural choice for addressing the maximization of the expected number of transplants. However, the modification of the objective function implies significant changes in the method. The main reason is that the contribution of an exchange is no longer obtained by the number of pairs involved in it, but by a non-linear

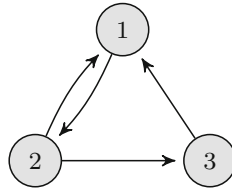


Fig. 1 Cycle with 3 vertices

function that measures the expected number of transplants (Klimentova et al. 2016; Pedroso 2014).

Following Klimentova et al. (2016) and given that the integer programming based models (Alvelos et al. 2015, 2016) proposed so far to solve the subproblems are not promising, we deal with this issue by enumerating all the cycles, while defining a restricted master problem with only a subset of them. This strategy is able to obtain solutions to large instances (1000 vertices) with very small optimality gaps.

The paper is structured as follows. In Sect. 2 the problem is defined together with the concept of expected number of transplants and type of cycle. The cycle model with types is presented in Sect. 3 and the branch-and-price algorithm detailed in 4. In Sect. 5, the computational experiments are reported and their results discussed. In Sect. 6 we provide some concluding remarks.

2 Problem definition and dealing with expectations

The problem addressed in this paper can be represented by a directed graph, $G(V, A)$, where 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 a donor from pair i is compatible with a patient from pair j . A (directed) cycle in G represents a feasible kidney exchange between the pairs associated with vertices belonging to the cycle. Since each incompatible pair can be included in at most one exchange, a solution to the KEP problem is a set of vertex-disjoint cycles of G . Furthermore, as mentioned before, only cycles with less than a given length K are accepted to be part of a solution. To the best of our knowledge, with some isolated exceptions, $K = 3$ is the current bound on the length of cycles in the most advanced KEPs, though the possibility of considering $K = 4$ has been discussed (Glorie et al. 2013; Dickerson et al. 2013, 2016).

We consider the problem where a probability of failure is associated to arcs and vertices and the objective is to maximize the expected number of transplants. We assume that the probability of arc failure is equal, for all arcs. The same holds for vertices:

$$p_i = p^v, \quad \forall i \in V, \quad p_{ij} = p^a, \quad \forall (i, j) \in A.$$

When calculating the expected number of transplants we also assume that the vertices of the cycle with failure may be rearranged if a shorter cycle is available within vertices of the cycle with failure.

The approach is detailed in Pedroso (2014) and Klimentova et al. (2016), so we present here just an illustrative example supported by Fig. 1. Assuming only vertices failure, the cycle presented leads to 3 transplants if none of the vertices fails, and to 2 transplants [cycle (1, 2)] in case of withdrawal of vertex 3 only.

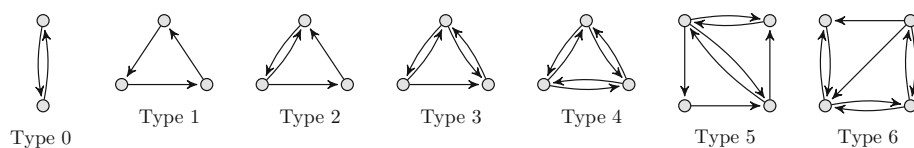


Fig. 2 Examples of *types of cycles*

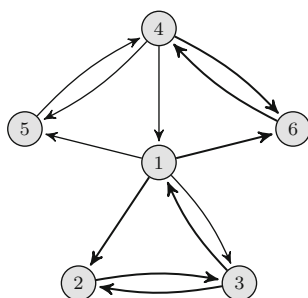


Fig. 3 Example of graph with three cycles with different *types of cycles*

The expected number of transplants for this cycle is:

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

Given that the expected number of transplants of a solution is the sum of the expected number of transplants of its cycles, a key issue in the proposed approach is the computation of the expected number of transplants of a single cycle. We model this as follows: cycles with the same length that have additional arcs (arcs between vertices of the cycle but not belonging to the cycle) in the same relative position are grouped in the same *type* of cycle. Formally, a type of a cycle of size k is an equivalence class of isomorphic graphs with k vertices containing at least a cycle of size k .

The only type of a cycle of size 2, all different types of cycles of size 3, and two types of cycles of size 4, are shown in Fig. 2. The number of different types of cycles grows very rapidly with the number of vertices: for $K = 4$ there are 61 different types and for $K = 5$ there are 3725 types. As mentioned above, $K = 4$ seems to be reasonable bound both from practical and theoretical points of view.

An illustrative example regarding the concept of types of cycles is given in Fig. 3. Cycles (1, 5, 4) and (1, 6, 4) are of type 2, and cycle (1, 2, 3) is of type 3.

For cycles of the same type the expected number of transplants is computed equivalently. Hence, for given equal probabilities of failure of vertices and arcs, the expected number of transplants of cycles of a type is equal for all those cycles.

The concept of types of cycles can be extended to include chains initiated by altruistic donors and the computational algorithm developed in [Klimentova et al. \(2016\)](#) can be straightforwardly adapted for computation of expected number of transplants for chains. Regarding the present study, inclusion of chains would imply extension of number of types. However it would not imply any extra methodological developments, and is not addressed in this paper.

3 Cycle model with types

We use the following notation in the model proposed in this paper. The maximum length of a cycle is denoted by K . T is the set of types of cycles of sizes $k = 2, \dots, K$, ordered by increasing k ; types are indexed by t (e.g., if $k = 4$, $t = 5, \dots, 65$). Let v_t be the expected value (or weight) of type $t \in T$, i.e. the expected number of transplants for any cycle of type t computed as illustrated in the previous section and detailed in [Klimentova et al. \(2016\)](#) and [Pedroso \(2014\)](#).

For each type of cycle $t \in T$, let C^t be the set of all cycles of type t in graph G . Notice that $C^t \cap C^{t'} = \emptyset$, $\forall t, t', t \neq t' \in T$ and $\bigcup_{t \in T} C^t = C$, where C is set of all cycles of size up to K in graph G . Define the decision variables $y_c^t = 1$ if the c th cycle of type t is part of the solution, $y_c^t = 0$ otherwise, $\forall t \in T, \forall c \in C^t$. Parameter $a_{ic}^t = 1$ if cycle c of type t includes vertex i , $a_{ic}^t = 0$ otherwise, $\forall i \in V, \forall t \in T, \forall c \in C^t$.

The cycle model with types is as follows:

$$\text{Maximize} \quad \sum_{t \in T} \sum_{c \in C^t} v_t y_c^t \quad (1)$$

$$\begin{aligned} \text{subject to} \quad & \sum_{t \in T} \sum_{c \in C^t} a_{ic}^t y_c^t \leq 1, & \forall i \in V \\ & y_c^t \in \{0, 1\}, & \forall t \in T, \forall c \in C^t. \end{aligned} \quad (2)$$

The objective function (1) corresponds to maximizing the expected number of transplants. By constraints (2), a vertex belongs to at most one cycle.

4 Branch-and-price algorithm for the cycle model with types

Column generation allows solving a linear programming model, as the linear relaxation of the cycle model with types, without explicitly considering all the decision variables. Its combination with branch-and-bound (i.e. branch-and-price), allows solving the integer problem. In each iteration of a column generation algorithm, a problem with only a subset of variables (the Restricted Master Problem—RMP) is optimized. Based on the values of the dual variables of the RMP, subproblems are responsible for finding variables outside the RMP with positive (in case of maximization problem) reduced costs. Frequently, but not necessarily, each subproblem obtains the variable with highest reduced cost by solving an optimization problem. If no variable with positive reduced cost is found in any subproblem, the current solution of the RMP is also optimal for the full problem. For further details on branch-and-price the reader is addressed to [Barnhart et al. \(1998\)](#).

We propose to solve the linear relaxation of the cycle model with types using column generation by defining one subproblem for each type t . The reduced cost r_c^t of a cycle c of type t is:

$$r_c^t = v_t - \sum_{i \in V} a_{ic}^t \pi_i \quad (3)$$

where π_i is the dual value of constraint (2) of vertex i . The subproblem of type t consists in finding the cycle of that type with maximum reduced cost:

$$\text{Maximize} \quad \left(v_t - \sum_{i \in V} x_i^t \pi_i \right) \quad (4)$$

$$\begin{aligned} \text{Subject to:} \quad & x_i^t \text{ define a cycle of type } t \\ & x_i^t \in \{0, 1\}, \quad \forall i \in V, \end{aligned} \quad (5)$$

where $x_i^t = 1$ if vertex i is part of the selected cycle and $x_i^t = 0$, otherwise, $\forall i \in V$. Integer programming models to solve this subproblem can be derived from the models described in Alvelos et al. (2015, 2016). In this paper, we propose a procedure based on enumeration. This procedure turns out to be more efficient computationally than the integer programs referred above, as well as than a tree search algorithm that was also implemented. That algorithm enumerates all the cycles of a given type, and a subproblem is solved by choosing the one with the maximum reduced cost. For each vertex the branching is performed on a set of outgoing arcs, omitting the arcs that cannot be included due to the configuration of the type.

To solve an instance of the problem, we start by calculating the expected number of transplants associated with each type of cycle, v_t , according to the actual failure probabilities of the arcs (p^a) and of the vertices (p_v) of the instance. Then, cycles are enumerated and their types are identified. Algorithm 1 provides the pseudocode of this enumeration procedure. Its input is the maximum allowed length of cycle (K) and the function $B(c)$ that returns the type of a cycle c . The result of the algorithm are sets $C^t, t \in T$.

Algorithm 1: Enumeration of cycles.

Data: K maximum length of the cycle, $B(c)$ – function that returns a type t of cycle c .

Result: For each $t \in T$, C^t set of cycles of type t

```

1  $C^t = \emptyset, t \in T$ 
2 foreach  $i \in V$  do
3   Path  $P = (i)$ , length of path  $h = 1$                                 // initialize path
4   EnumerateCycles( $i, P, h$ )
5 return  $C^t$ 

6 EnumerateCycles( $u, P, h$ )
7   foreach  $v \in \delta^+(u)$  do
8     if  $v = p_1$  then                                                  // new cycle found
9       Cycle  $c := P$  is added to the set  $C^t$ , where  $t = B(c)$ .
10      continue
11     if  $h < K$  then
12        $P' := (P, v)$ , add vertex  $v$  to the path
13       EnumerateCycles( $v, P', h + 1$ )
  
```

Cycle enumeration is done recursively and is repeated for each $i \in V$ (lines 2–4). P stores the path from vertex i up to the current node as an ordered set of vertices (p_1, \dots, p_h). For each i the current path P is initialized with i , i.e. $p_1 = i$ (line 3). The branch is created for each vertex $v \in \delta^+(i)$, where $\delta^+(i) = \{j \in V : (i, j) \in A, j \geq i\}$, and is fathomed if a cycle is found (lines 8–10). Otherwise, vertex v is added to the current path and procedure EnumerateCycles is recursively called (lines 11–13). All the cycles of a given type are kept in memory. Solving the subproblem of a type corresponds to checking the enumerated cycles for maximum reduced cost. Each time a tree node with a fractional value is not fathomed, two descendant nodes are created. In one node a fractional variable is fixed to 1 and in the other the same fractional variable is fixed to 0. When solving the

subproblem, the corresponding cycle is not taken into account when selecting the most attractive variable.

5 Computational tests

5.1 Implementation details

Computational experiments have been carried out to verify the effectiveness of the proposed method. Two different versions of the branch-and-price (B&P) algorithm are compared with the direct use of cycle formulation (1)–(2) using CPLEX solver. The following notation will be used in the remaining of this document to refer to the three approaches:

- CF —direct cycle formulation;
- $B\&P_{2SP}$ —version of B&P algorithm, where the variables associated with cycles of size two are generated by the subproblem when attractive;
- $B\&P_{2MP}$ —version of B&P algorithm that includes all 2-cycles from the first restricted master problem.

The expected values v_t for all possible types $t \in T$, of sizes $k = 2, \dots, 4$ are computed in advance. The enumeration procedure (Algorithm 1) is used in the beginning of each approach. The only difference between the approaches is that for CF all cycles are included in the model and the problem is solved directly, while for $B\&P_{2SP}$ and $B\&P_{2MP}$ cycles are included only when they are attractive.

In the implementation of the B&P algorithms we considered a search strategy that consists in a mix of depth search (when the current node is not fathomed) and best search (when the current node is fathomed). The branching variable is the one with a higher coefficient in the objective function from the ones with fractional value. In an attempt to find a good quality incumbent early, after the linear relaxation optimal solution is obtained and before branching starts, the variables of the RMP are set to integer and the resulting integer RMP is optimized.

5.2 Results

Computational tests were carried out in a computer with Intel Xeon processor at 3.00 GHz, 8 Gb of RAM and running Linux OS. MIP solver CPLEX 12.4 was used¹ for solving directly the cycle formulation and for solving the linear relaxations in the B&P approaches. B&P was implemented in SearchCol++.² The computational time allowed for solving each instance was limited to 3600 s (1 h).

The instances for the computational study were the ones from Klimentova et al. (2016) and Constantino et al. (2013) with $K = 3$ and $K = 4$. We concentrated our computational experiments on instances from 200 up to 1000 vertices, as the size of the instances up to 100 does not justify the use of branch-and-price (Klimentova et al. 2016). Although, to the best of our knowledge, the biggest KEPs consider pools of 200–250 patients (Dickerson et al. 2016), there is intention of joint collaboration between a number of European programs (ENCKEP 2017). Such a collaboration may increase the size of the pool significantly and therefore justify including in the computational analysis instances of a bigger size.

¹ IBM: ILOG CPLEX Optimizer. <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/> (last accessed in June, 2017).

² <http://searchcol.dps.uminho.pt/>.

Table 1 Comparison of performance of the three approaches for $K = 3$

| n | CF | | $B\&P_{2SP}$ | | $B\&P_{2MP}$ | |
|------------------------|-------|------|--------------|------|--------------|------|
| | #slvd | gap% | #slvd | gap% | #slvd | gap% |
| $p^v = 0, p^a = 0.2$ | | | | | | |
| 200 | 10 | 0 | 9 | 0.05 | 9 | 0.05 |
| 300 | 7 | 0.06 | 3 | 0.10 | 3 | 0.11 |
| 400 | 8 | 0.08 | 5 | 0.05 | 5 | 0.05 |
| 500 | 6 | 0.04 | 7 | 0.05 | 7 | 0.05 |
| 600 | 6 | 0.06 | 2 | 0.04 | 2 | 0.04 |
| 700 | 2 | 0.03 | 2 | 0.03 | 2 | 0.03 |
| 800 | 4 | 0.11 | 4 | 0.02 | 4 | 0.02 |
| 900 | 2 | 0.03 | 2 | 0.03 | 2 | 0.03 |
| 1000 | 4 | 0.12 | 4 | 0.03 | 4 | 0.04 |
| $p^v = 0.2, p^a = 0.4$ | | | | | | |
| 200 | 10 | 0 | 8 | 0.07 | 8 | 0.07 |
| 300 | 10 | 0 | 7 | 0.09 | 7 | 0.09 |
| 400 | 10 | 0 | 7 | 0.10 | 7 | 0.08 |
| 500 | 10 | 0 | 5 | 0.11 | 5 | 0.10 |
| 600 | 9 | 0.11 | 5 | 0.06 | 6 | 0.07 |
| 700 | 8 | 0.14 | 5 | 0.07 | 5 | 0.06 |
| 800 | 6 | 0.28 | 4 | 0.10 | 3 | 0.09 |
| 900 | 7 | 0.05 | 3 | 0.04 | 2 | 0.04 |
| 1000 | 7 | 0.33 | 3 | 0.08 | 3 | 0.08 |
| $p^v = 0.4, p^a = 0.4$ | | | | | | |
| 200 | 10 | 0 | 10 | 0 | 10 | 0 |
| 300 | 10 | 0 | 8 | 0.08 | 8 | 0.08 |
| 400 | 10 | 0 | 9 | 0.23 | 9 | 0.19 |
| 500 | 10 | 0 | 5 | 0.12 | 4 | 0.11 |
| 600 | 9 | 0.09 | 2 | 0.06 | 3 | 0.06 |
| 700 | 9 | 0.18 | 8 | 0.17 | 8 | 0.16 |
| 800 | 8 | 0.14 | 3 | 0.07 | 3 | 0.08 |
| 900 | 7 | 0.06 | 0 | 0.04 | 1 | 0.05 |
| 1000 | 7 | 0.15 | 3 | 0.09 | 3 | 0.10 |

Ten instances of each size were solved for three different combinations of values of probabilities of failure of vertices (p^v) and arcs (p^a): (i) $p^v = 0, p^a = 0.2$; (ii) $p^v = 0.2, p^a = 0.4$; (iii) $p^v = 0.4, p^a = 0.4$.

Table 1 reports the results for $K = 3$: n is the number of vertices in the instance; #slvd is the number of instances solved to optimality within the given time limit (3600 s); gap% is the average optimality gap for the set ($\text{gap} = \frac{BUB - \text{Inc}}{\text{Inc}} \times 100\%$), where Inc is the value of the incumbent solution, and BUB is the best found upper bound).

It can be seen that for $K = 3$ the cycle formulation performs better than the B&P approaches in terms of the number of instances solved to optimality. However, for the larger instances (600–1000 vertices), B&P approaches clearly outperform the direct approach in terms of quality of the solutions obtained, having smaller average gaps in all sets of instances. instances, B&P always provides very small optimality gaps.

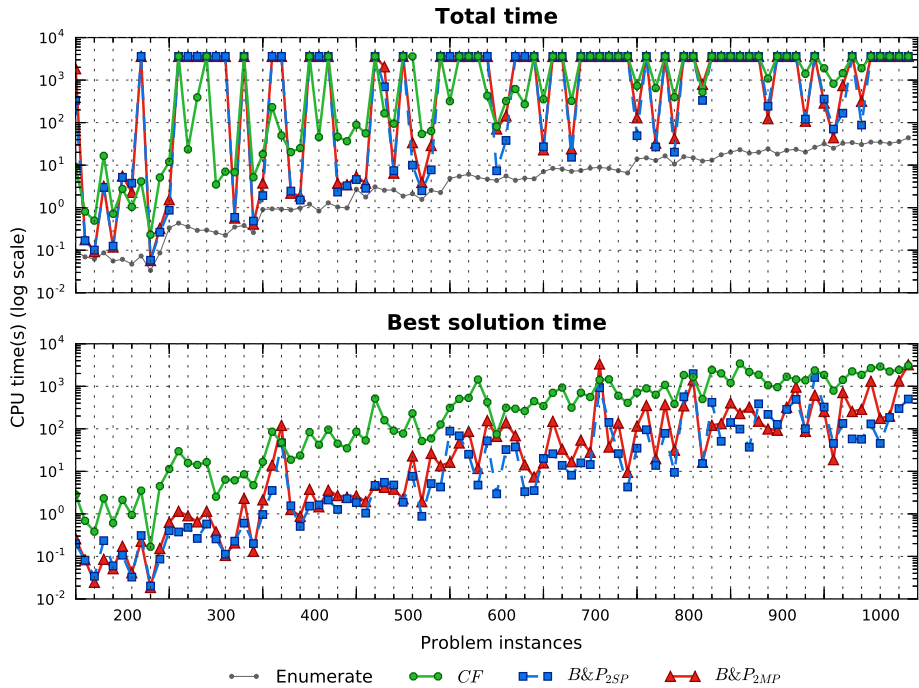


Fig. 4 Total time and time when the best solution was obtained for each instance with $K = 3$, $p^v = 0$, and $p^a = 0.2$

The instances with smaller probabilities values ($p^v = 0$, $p^a = 0.2$) turned out to be more difficult to solve. The number of instances solved with those probabilities is less than that with other combinations of p^v , p^a . Also noteworthy is that B&P approaches managed, in some cases, to solve more instances of bigger size (compare $n = 300$, and $n = 400, 500$ for $p^v = 0$, $p^a = 0.2$; $n = 500, 600$ and $n = 700$ for $p^v = 0.4$, $p^a = 0.4$). For $n = 500$, $p^v = 0$ and $p^a = 0.2$, B&P were able to solve one instance that was not solved by CF.

Figure 4 shows the computational time, in logarithmic scale, to solve each instance with $p^v = 0$, $p^a = 0.2$ and to reach the best solution. Similar figures for the other combinations of probabilities are provided in Appendix. The graph *Enumerate* plots the CPU time spent to enumerate all cycles. That value is at most 40–50 s for instances with 1000 vertices. The time for unsolved instances is 3600 s.

As shown in the graphs, both B&P methods work very similarly, so the CPU time lines almost coincide. Furthermore, as already mentioned, the CF solves more instances to optimality, however it can be seen that B&P finds a good quality solution (the best solution found) faster. not negligible but is not relevant also, the time spent was at most 40–50 s for instances of size 1000. The graphs on Fig. 4, down, show the time when the final solution was obtained for each approach (with logarithmic scale). Compared to CF both B&P approaches manage to obtain the best solution significantly faster almost for all instances and sets of probabilities (see also Appendix).

A compilation of results obtained for $K = 4$ is presented in Table 2. In addition to the previously mentioned notation, we also use $\#feas$ for the number of instances where a feasible solution was found. present For $K = 3$ feasible solutions were always found with

Table 2 Comparison of performance of the three approaches for $K = 4$

| n | CF | | | $B\&P_{2SP}$ | | | $B\&P_{2MP}$ | | |
|------------------------|-------|-------|-------|--------------|-------|------|--------------|-------|------|
| | #slvd | #feas | gap% | #slvd | #feas | gap% | #slvd | #feas | gap% |
| $p^v = 0, p^a = 0.2$ | | | | | | | | | |
| 200 | 8 | 10 | 0.37 | 3 | 10 | 0.29 | 3 | 10 | 0.29 |
| 300 | 3 | 10 | 2.32 | 1 | 10 | 0.18 | 1 | 10 | 0.19 |
| 400 | 0 | 0 | NA | 0 | 10 | 0.16 | 0 | 10 | 0.16 |
| 500 | 0 | 0 | NA | 1 | 10 | 0.25 | 1 | 10 | 0.14 |
| 600 | — | — | NA | 0 | 10 | 0.55 | 0 | 10 | 0.40 |
| 700 | — | — | NA | 0 | 10 | 0.68 | 0 | 10 | 0.41 |
| 800 | — | — | NA | 0 | 10 | 1.22 | 0 | 10 | 0.85 |
| $p^v = 0.2, p^a = 0.4$ | | | | | | | | | |
| 200 | 6 | 10 | 0.21 | 3 | 10 | 0.18 | 3 | 10 | 0.18 |
| 300 | 5 | 10 | 1.94 | 2 | 10 | 0.16 | 2 | 10 | 0.16 |
| 400 | 0 | 1 | 47.50 | 2 | 10 | 0.16 | 2 | 10 | 0.15 |
| 500 | 0 | 0 | NA | 0 | 10 | 0.11 | 0 | 10 | 0.11 |
| 600 | — | — | NA | 1 | 10 | 1.11 | 1 | 10 | 0.41 |
| 700 | — | — | NA | 0 | 10 | 1.58 | 1 | 10 | 0.57 |
| 800 | — | — | NA | 0 | 10 | 2.11 | 0 | 10 | 1.56 |
| $p^v = 0.4, p^a = 0.4$ | | | | | | | | | |
| 200 | 9 | 10 | 0.27 | 4 | 10 | 0.17 | 4 | 10 | 0.14 |
| 300 | 8 | 10 | 4.37 | 5 | 10 | 0.09 | 5 | 10 | 0.11 |
| 400 | 0 | 1 | 20.27 | 3 | 10 | 0.12 | 3 | 10 | 0.13 |
| 500 | 0 | 0 | NA | 2 | 10 | 0.11 | 1 | 10 | 0.16 |
| 600 | — | — | NA | 1 | 10 | 1.09 | 2 | 10 | 0.11 |
| 700 | — | — | NA | 0 | 10 | 1.69 | 0 | 10 | 0.77 |
| 800 | — | — | NA | 0 | 10 | 2.06 | 0 | 10 | 1.45 |

all the methods for all instances. The time for enumeration of cycles is reasonably big for $n \leq 700$ (at most 1400 s) and dramatically increases for instances with 800 vertices (almost 1 h). Therefore instances with $n > 800$ were not considered for $K = 4$.

For $K = 4$ and $n = 200$ –300 vertices, the cycle formulation obtains more optimal solutions than B&P. However, for larger instances (400–800 vertices), the CF dramatically fails—only for 1 instance of size 400 a feasible solution was found with a very large gap. was not run.

As for the B&P approaches, they obtained feasible solutions with very small gaps in all the instances with all probabilities (average values for 10 instances are always less than or equal to 0.25% with $n \leq 500$, and never exceed 3% for bigger n). Moreover for bigger values of probabilities ($p^v = 0.4, p^a = 0.4$) the B&P methods were able to solve to optimality 3 out of 10 instance of size 400, 2 ($B\&P_{2SP}$) and 1 ($B\&P_{2MP}$) of size 500, and even 1 ($B\&P_{2SP}$) and 2 ($B\&P_{2MP}$) of size 600, while CF did not provide any feasible solution.

Figure 5 presents the total time (in logarithmic scale) and the time when the best solution was obtained (normal scale) for all methods for $p^v = 0.2$ and $p^a = 0.4$. The conclusions with respect to computational times conducted for $K = 3$ are valid also here. In general the B&P methods significantly outperform the cycle formulation in obtaining a good quality

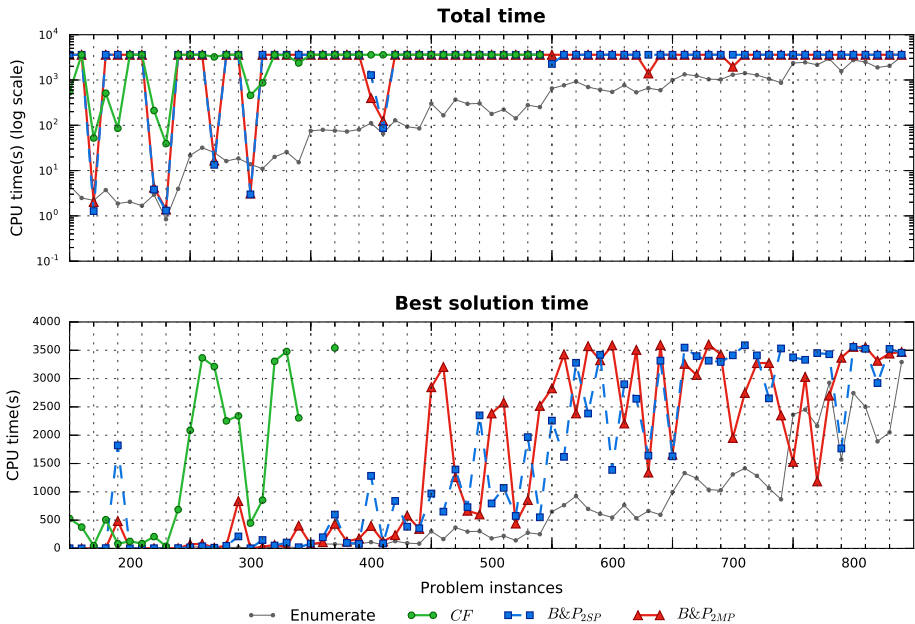


Fig. 5 Total time and time when the best solution was obtained for each instance with $K = 4$, $p^v = 0.2$, and $p^a = 0.4$

solution faster. As to the total time, the developed methods loose for smaller instances, having however better performance for the bigger ones.

Summing up, the proposed B&P approach clearly outperforms the CF approach consisting in solving directly the cycle formulation in the larger instances (for cycle lengths up to 3 and 600–1000 vertices, and cycle lengths up to 4 and 400–800 vertices) in terms of the quality of the solutions obtained. For the largest instances among these (cycle lengths up to 4, 400–800 vertices) B&P still obtains solutions with very small gaps, while the cycle formulation fails to obtain a single feasible solutions.

The cycle formulation is suitable for small instances (cycle lengths up to 3 and less than 500 vertices), as it provides optimal solutions for almost all of those instances. In any case, even in small instances, B&P obtained solutions with very small gaps.

6 Conclusions

In this paper we propose a branch-and-price approach for the kidney exchange problem, when there are probabilities of failure associated both to vertices and to arcs. In such case the objective is to find a solution that maximizes the expected number of transplants. We assign equal probabilities of failure to all vertices, as well as to all arcs.

The branch-and-price algorithm proposed is based on the concept of type of cycle. The efficiency of the proposed approach is compared with solving directly a corresponding model with a general purpose solver (CF). Total enumeration of the variables is used in the branch-and-price approach to solve the subproblem (which is difficult to solve efficiently given the non-linearity arising from the expectations calculations). In one version of branch-and-price

algorithm, $B\&P_{2SP}$, the variables associated with cycles of size 2 are generated by the subproblem, when attractive. In another version, $B\&P_{2MP}$, all 2-cycles are included in the first restricted master problem.

Both approaches were tested in 390 realistic instances with up to 1000 incompatible donor/patient pairs. For smaller instances, up to 500 vertices, $K = 3$, or up to 300 vertices, $K = 4$, CF outperforms $B\&P$ versions. However, for bigger instances, branch-and-price obtains feasible solutions for all the tested instances with an average optimality gap always less than 2.11%. For such instances, with 500 incompatible pairs or more and cycles up to length four, the direct solving of the model fails to provide a feasible solution due to memory limitation. Therefore, computational results validate the importance of the B&P approach hereby proposed, when the size of the pool of a Kidney Exchange Program increases above a given number of pairs.

Acknowledgements We would like to thank Dr. James Trimble from the University of Glasgow, UK for his valuable comments.

Appendix

Figures 6–9 complement the computational results presented in the paper in terms of total computational time to solve each instance and time to reach the best solution for different combinations of probabilities, for $K = 3$ (Figs. 6–7) and for $K = 4$ (Figs. 8–9).

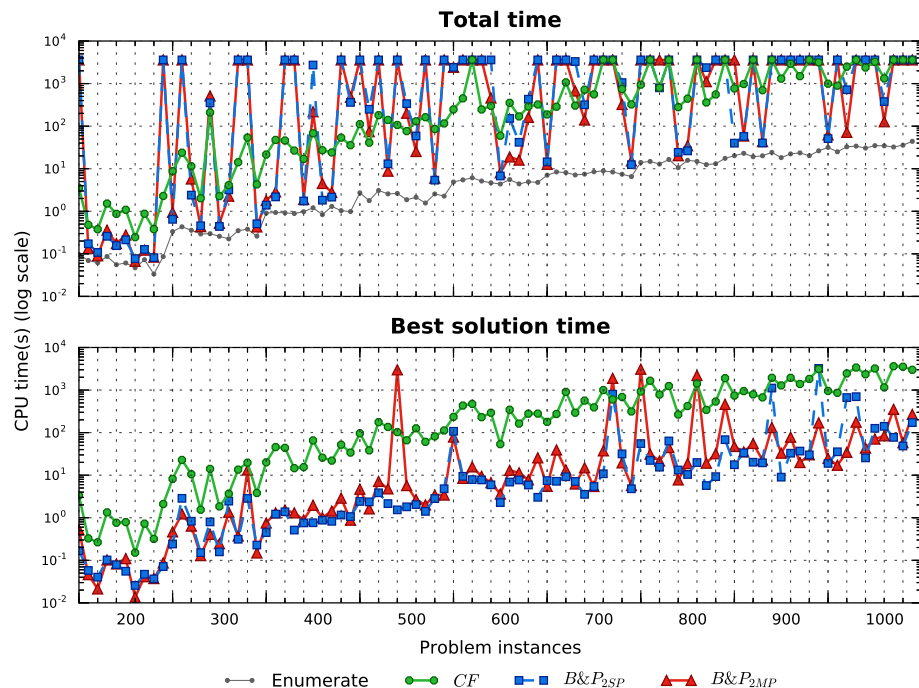


Fig. 6 Total time and time when the best solution was obtained for each instance with $K = 3$, $p^v = 0.2$, and $p^a = 0.4$

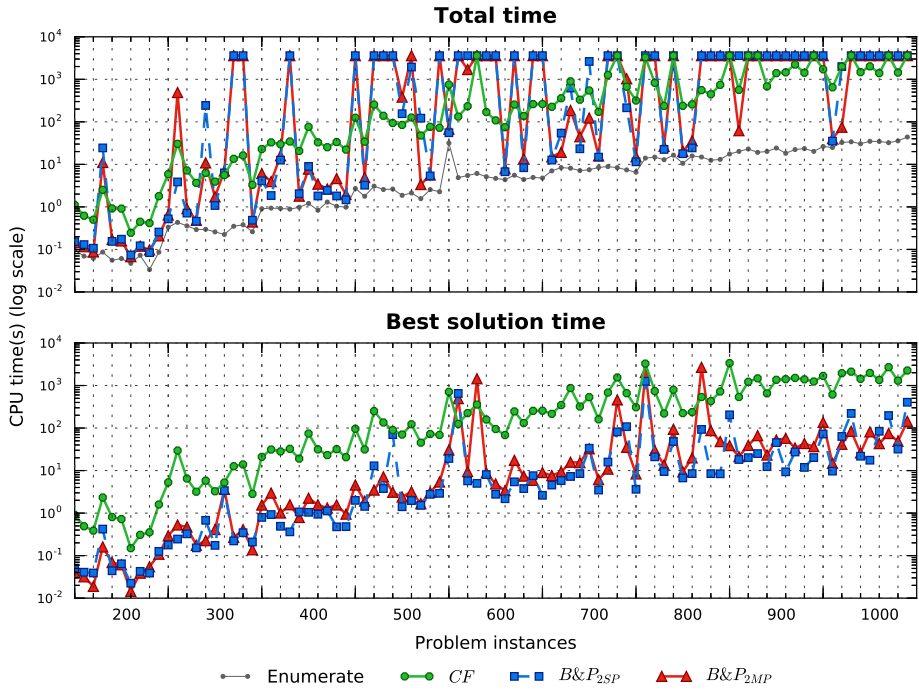


Fig. 7 Total time and time when the best solution was obtained for each instance with $K = 4$, $p^v = 0.4$, and $p^a = 0.4$

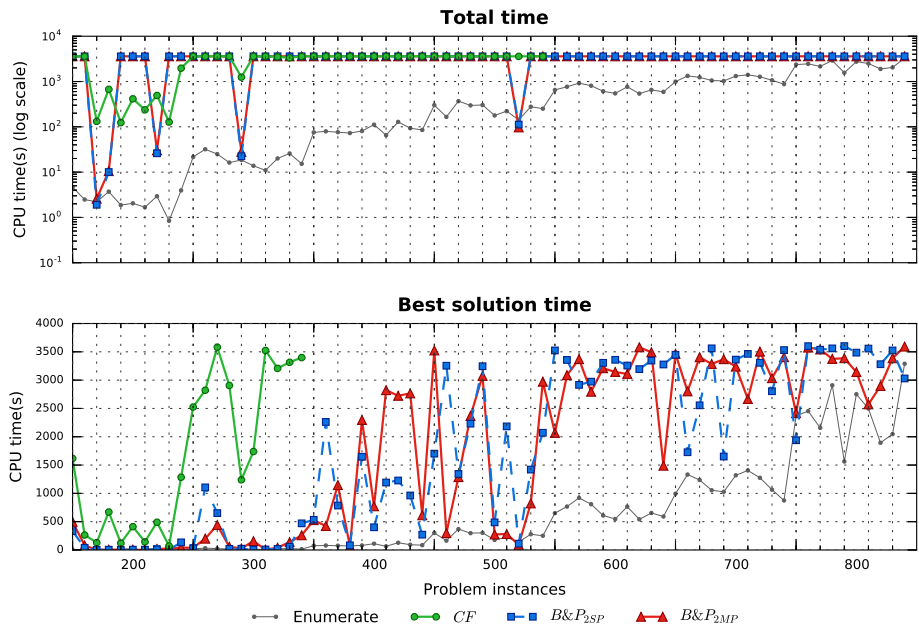


Fig. 8 Total time and time when the best solution was obtained for each instance with $K = 4$, $p^v = 0$, and $p^a = 0.2$

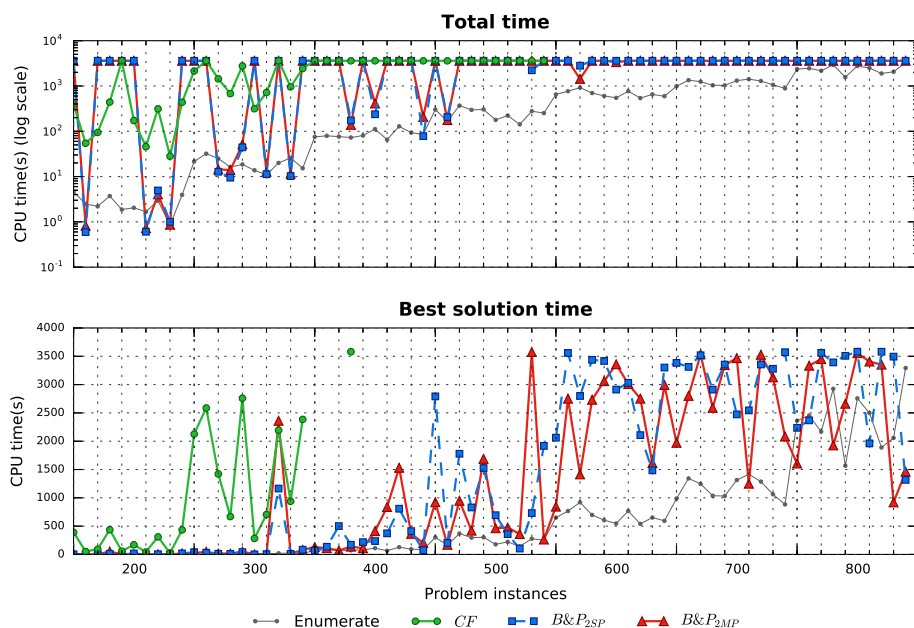


Fig. 9 Total time and time when the best solution was obtained for each instance with $K = 4$, $p^v = 0.4$, and $p^a = 0.4$

References

- Alvelos, F., Klimentova, X., Rais, A., & Viana, A. (2015). A compact formulation for maximizing the expected number of transplants in kidney exchange programs. *Journal of Physics: Conference Series*, 616(1), 012011.
- Alvelos, F., Klimentova, X., Rais, A., & Viana, A. (2016). Maximizing expected number of transplants in kidney exchange programs. In *Electronic notes in discrete mathematics. INOC 2015—7th international network optimization conference*, vol. 52, pp. 269–276.
- Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46, 316–329.
- Constantino, M., Klimentova, X., Viana, A., & Rais, A. (2013). New insights on integer-programming models for the kidney exchange problem. *European Journal of Operational Research*, 231(1), 57–68.
- Dickerson, J., Procaccia, A. D., & Sandholm, T. (2013). Failure-aware kidney exchange. In *EC-13: Proc. 14th ACM conference on electronic commerce*, June.
- Dickerson, J. P., Manlove, D. F., Plaut, B., Sandholm, T., & Trimble, J. (2016). Position-indexed formulations for kidney exchange. In *Conference on economics and computation (EC)*.
- ENCKEP. (2017). European network for collaboration on kidney exchange programmes. <http://www.enckep-cost.eu/>.
- Glorie, K. M., Carvalho, M., Bouman, P., Viana, A., & Constantino, M. (2014). *Clearing barter exchange markets: Kidney exchange and beyond*, Chapter VI. Ph.D. thesis, Erasmus University Rotterdam.
- Glorie, K. M., de Klerk, M., Wagelmans, A. P. M., van de Klundert, J. J., Zuidema, W. C., Claas, F. H. J., et al. (2013). Coordinating unspecified living kidney donation and transplantation across the blood-type barrier in kidney exchange. *Transplantation Journal*, 96(6), 814–820.
- Glorie, K. M., van de Klundert, J. J., & Wagelmans, A. P. M. (2014). Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manufacturing & Service Operations Management (MSOM)*, 16(4), 498–512.
- Klimentova, X., Alvelos, F., & Ana Viana, A. (2014). A new branch-and-price approach for the kidney exchange problem. In B. M. S. Misra, A. M. A. R. C. Torre, J. G. R. M. I. Falcão, D. T. B. O. Apduhan, & O. Gervasi (Eds.), *Computational science and its applications—ICCSA 2014*, volume 8580 of *Lecture Notes in Computer Science*, pp. 237–252. Springer International Publishing.

- Klimentova, X., Pedroso, J. P., & Viana, A. (2016). Maximising expectation of the number of transplants in kidney exchange programmes. *Computers & Operations Research*, 73, 1–11.
- Li, Y., Song, P. X., Zhou, Y., Leichtman, A., Rees, M., & Kalbfleisch, J. (2014). Optimal decisions for organ exchanges in a kidney paired donation program. *Statistics in Biosciences*, 6(1), 85–104.
- Mak-Hau, V. (2017). On the kidney exchange problem: Cardinality constrained cycle and chain problems on directed graphs: A survey of integer programming approaches. *Journal of Combinatorial Optimization*, 33(1), 35–59.
- Manlove, D., & O'Malley, G. (2012). Paired and altruistic kidney donation in the UK: Algorithms and experimentation. *Lecture Notes in Computer Science*, 7276, 271–282.
- Manlove, D., & O'Malley, G. (2014). Paired and altruistic kidney donation in the UK: Algorithms and experimentation. *ACM Journal of Experimental Algorithmics*, 19(2), 2.6:1.1–2.6:1.21.
- Pedroso, J. P. (2014). Maximizing expectation on vertex-disjoint cycle packing. In B. M. S. Misra, A. M. A. R. C. Torre, J. G. R. M. I. Falcão, D. T. B. O. Apduhan & O. Gervasi (Eds.), *Computational science and its applications—ICCSA 2014*, volume 8580 of *Lecture Notes in Computer Science*, pp. 32–46. Springer International Publishing.
- Plaut, B. Dickerson, J. P., & Sandholm, T. (2016). Fast optimal clearing of capped-chain barter exchanges. In *AAAI conference on artificial intelligence (AAAI)*.