

Maximizing expectation on vertex-disjoint cycle packing

João Pedro Pedroso

Technical Report Series: DCC-2013-5

U. PORTO

FC FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Departamento de Ciência de Computadores

Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre, 1021/1055,
4169-007 PORTO,
PORTUGAL

Tel: 220 402 900 Fax: 220 402 950
<http://www.dcc.fc.up.pt/Pubs/>

Maximizing expectation on vertex-disjoint cycle packing

João Pedro Pedroso^{a,b}

^aFaculdade de Ciências, Universidade do Porto,
Rua do Campo Alegre, 4169-007 Porto, Portugal

^bINESC Porto, Rua Dr. Roberto Frias 378, 4200-465 Porto, Portugal

Abstract

This paper proposes a method for computing the expectation for the length of the maximum set of vertex-disjoint cycles in a digraph where vertices and/or arcs are subject to failure with a known probability. This method has an immediate practical application: it can be used for the solution of a kidney exchange program in the common situation where the underlying graph is unreliable. Results for realistic benchmark instances are reported and analyzed.

Keywords: Kidney exchange programs, Cycle packing, Expectation optimization, Combinatorial optimization

1. Introduction

In many countries, recent legislation allows patients needing a kidney transplant to receive it from a living donor ready to provide one of her kidneys. When the transplant from the donor is not possible, due to blood type or other incompatibilities, the patient-donor pair may enter a kidney exchange program (KEP). The KEP may allow two (or more) patients with incompatible pairs to exchange their donors, in such a way that each recipient receives a compatible kidney from the donor of another pair (see Figure 1).

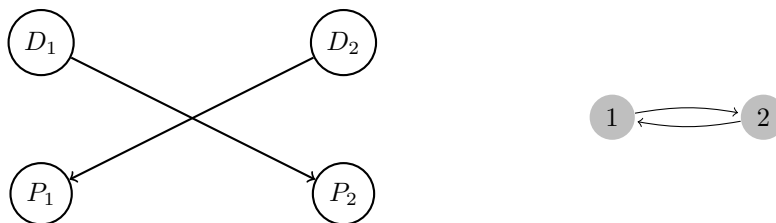


Figure 1: Left: incompatible pairs $P_1 - D_1$ and $P_2 - D_2$ exchange donors, allowing P_1 to receive a transplant from D_2 and vice versa. Right: graph representation of this situation, where vertices are patient-donor pairs, and arcs link a patient to compatible donors.

An instance of the KEP with 5 pairs is presented in Figure 2. The set of patients, donors and their interconnections is often called a *market*, and is represented as a directed graph $G(\mathcal{V}, \mathcal{A})$. In this graph, \mathcal{V} is the set of all incompatible patient-donor pairs. Two vertices i and j are connected by arc (i, j) if the donor of pair i is compatible with the patient of pair j . An *exchange* is defined as a cycle in the graph. In the situation of Figure 2 the possible exchanges are $1 - 2 - 5 - 3 - 1$, $1 - 2 - 3 - 1$, $1 - 2 - 4 - 1$, $2 - 5 - 3 - 2$, $2 - 3 - 2$ and $2 - 5 - 2$. A feasible exchange can be defined as a set of vertex-disjoint cycles. The size of an exchange is the sum of the lengths of its cycles. The maximum exchange in this example has size four, corresponding to the cycle $1 - 2 - 5 - 3 - 1$. In

Email address: jpp@fc.up.pt (João Pedro Pedroso)

many situations the length of each cycle in feasible exchanges is limited to a given value K , *e.g.*, due to limitations in the number of operation rooms that are simultaneously available. If cycles of length greater than $K = 3$ are not allowed, several solutions are possible; as we will see later, they may be non-equivalent.

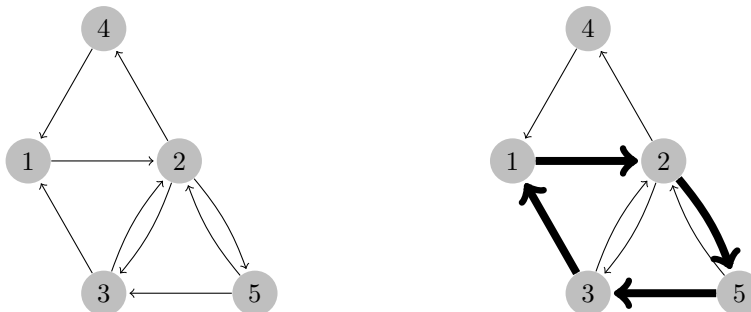


Figure 2: Instance with five pairs (left); the maximum number of transplants is four, corresponding to the cycle 1 – 2 – 5 – 3 – 1 (right).

Standard approaches to this problem consist of finding a set of vertex-disjoint cycles that maximizes the weighted sum of chosen cycles, where the weight of a cycle is typically its length; this corresponds to maximizing the number of transplants. In this work we will focus on maximizing the *expectation* of the size of an exchange, given that there may be withdrawal of vertices (*i.e.*, patients and/or donors may back out) and arcs (*e.g.*, incompatibilities that are detected in the last minute). In the occurrence of a withdrawal, vertices in the cycle involved may be rearranged between them if another cycle is possible, but other vertices cannot be involved; this is what in [9] is called a *contingency plan*.

A summary of models for solving a KEP has been presented in [3], where several formulations are presented, analyzed, and compared in terms of tightness and experimental performance. The topic to be developed in this paper — maximizing the expectation of the number of transplants in a given setting — has been initially raised in [9] and [2], which propose a model for maximizing the expected utility when arcs are subject to failure; 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. The study of the dynamics in a KEP, concerning the question of how often and how exactly should the exchange be run so as to make the mechanism efficient, has also been addressed in [12] and in references therein; more recently, issues on market design for kidney exchange have been analyzed in [11]. An analysis of the efficacy of chains in kidney exchange models, indicating that their length should be small — best results were obtained for a maximum length of four, in a dynamic setting —, is provided in [4].

2. Background: mathematical programming model

There are several possibilities for modeling the optimization problem of a KEP [3]. One of the most successful is the *cycle formulation* proposed in [1], where one needs to enumerate all cycles in the market graph $G = (\mathcal{V}, \mathcal{A})$ with length at most K . Notice that if $K = 2$ or K is unlimited the problem can be formulated as a matching and an assignment, respectively, and is solvable in polynomial time (see [5] and [8], respectively); otherwise, it is proven to be NP-hard [1]. For each cycle c in the set \mathcal{C} of cycles of the graph, let variable x_c be 1 if c is chosen for the exchange, 0 otherwise. The cycle model of the KEP is the following integer linear program:

$$\text{maximize} \quad \sum_c w_c x_c, \quad (1a)$$

$$\text{subject to} \quad \sum_{c:i \in c} x_c \leq 1, \quad \forall i \in \mathcal{V}, \quad (1b)$$

$$x_c \in \{0, 1\}, \quad \forall c \in \mathcal{C}. \quad (1c)$$

In the case of arcs with weight 1, the weight of a cycle is $w_c = |c|$, *i.e.*, its coefficient in the objective is its length; otherwise, it is typically the sum of the utilities assigned to arcs in the cycle. The objective function (1a) maximizes the weight of the exchange. Constraints (1b) ensure that every vertex is in at most one cycle (*i.e.*, each donor will donate, and the corresponding patient will receive, at most one kidney); a feasible solution is, thus, a vertex-disjoint cycle packing. The difficulty of this formulation is due to number of variables; the number of cycles of length at most K may be very large, even for graphs with a modest size. Very good results were mentioned in [1], obtained by implementing a column generation method within a branch-and-price scheme.

Enumerating cycles for this formulation (or for an associated column generation procedure) may be complemented with an appropriate assessment of the cycle's value, so as to encompass objectives other than the weighted sum of its of arcs. Indeed, we may evaluate the expectation of the number of transplants (possibly weighted) in a given subgraph of G induced by the vertices in a cycle. In other words, we propose to assess the expectation of the weight of a maximum packing of vertex-disjoint cycles *in the subgraph induced by each cycle*, given the probability of failure of its components (vertices and arcs). The value of this expectation will be used as the cycle's coefficient in the objective of the model above.

3. Unreliable graphs

3.1. Vertex withdrawal

One of the problems in the implementation of the solution of a KEP instance is that patients or donors may become unavailable, *e.g.*, due to illness or to backing out. In such cases, the previously found solution cannot be implemented in its totality; the cycle where failure occurred cannot be implemented, though the vertices involved may be rearranged if another (shorter) cycle within them can be formed (it is usually accepted that other cycles cannot be changed). In this situation, instead of using cycle's length, the value of a cycle can be better assessed by the expectation of the number of transplants it will lead to, given the probability of failure of each of its vertices.

As an example, consider again the graph of Figure 2. If the maximum cycle length is 3, there are three maximal solutions: $c_1 = 1 - 2 - 4 - 1$, $c_2 = 1 - 2 - 3 - 1$, and $c_3 = 2 - 5 - 3 - 2$, all of which are packings (*i.e.*, any other cycle leads to vertex overlap; see Figure 3). Even though the

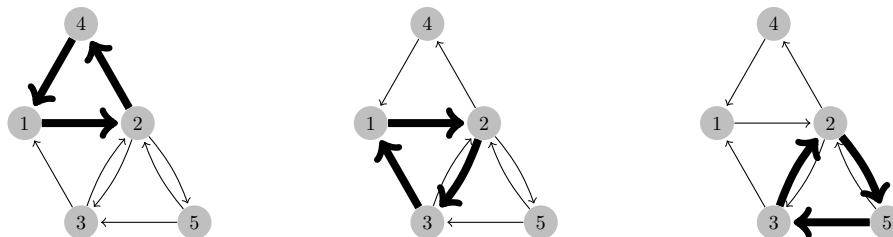


Figure 3: Cycles of length 3 in the example graph.

length is the same for all these cycles, the expected length may be different: in c_1 withdrawal of any of the vertices leads to no transplants, whereas in c_2 there can be two transplants (cycle $2 - 3 - 2$) in case (only) vertex 1 fails. In c_3 , either 3 or 5 may fail, and two transplants are still possible. Given the probability of failure p_i for each vertex in the graph, the expected length of each of these cycle configurations can be computed as follows:

$$E[c_1] = 3(1 - p_1)(1 - p_2)(1 - p_4) \quad (2)$$

$$\begin{aligned} E[c_2] &= 3(1 - p_1)(1 - p_2)(1 - p_3) + 2p_1(1 - p_2)(1 - p_3) \\ &= (3 - p_1)(1 - p_2)(1 - p_3) \end{aligned} \quad (3)$$

$$\begin{aligned} E[c_3] &= 3(1 - p_2)(1 - p_3)(1 - p_5) + 2(1 - p_2)p_3(1 - p_5) + 2(1 - p_2)(1 - p_3)p_5 \\ &= (p_2 - 1)(p_3p_5 + p_3 + p_5 - 3) \end{aligned} \quad (4)$$

Algorithm 1: Size expectation for a cycle packing (unreliable vertices).

Input: $G_c = (\mathcal{V}_c, \mathcal{A}_c)$, $p_i \forall i \in \mathcal{V}_c$

Output: Expectation for the input graph

```

(1)  $E := 0$  ← accumulator for expectation
(2)  $\mathcal{S} := 2^{\mathcal{V}_c}$  ← set of all subsets of  $\mathcal{V}_c$ 
(3) foreach  $\mathcal{R} \in \mathcal{S}$ : ← vertices remaining
(4)  $\mathcal{Q} := \mathcal{V}_c \setminus \mathcal{R}$  ← vertices quitting
(5)  $\mathcal{A}' := \{(i, j) \in \mathcal{A}_c : i \in \mathcal{R}, j \in \mathcal{R}\}$  ← arcs in subgraph induced by  $\mathcal{R}$ 
(6)  $z = \text{length of maximum vertex-disjoint cycle packing in } G' = (\mathcal{R}, \mathcal{A}')$ 
(7)  $E := E + z \prod_{i \in \mathcal{R}} (1 - p_i) \prod_{i \in \mathcal{Q}} p_i$ 
(8) return  $E$ 

```

The expectation of the length of the maximum set of vertex-disjoint cycles of a (sub)graph can be computed for the general case with Algorithm 1.

The purpose of this algorithm is to compute the expectation of a given cycle, to be used during the enumeration phase of the main optimization method when using the cycle formulation presented in Section 2. The evaluation of each cycle c (its coefficient w_c in the objective function) is, in this case, the *expected length* of the maximum set of vertex-disjoint cycle *in the subgraph induced by vertices in the current cycle*, $G_c = (\mathcal{V}_c, \mathcal{A}_c)$, which is given as a parameter to the algorithm. The key issue here is to check, for each subset of the set of vertices \mathcal{V}_c , if their withdrawal allows forming other cycles in G_c or not. This is done by solving a smaller optimization problem, in line (6). Notice that since we need not to be concerned with the size of the cycles here (the size of \mathcal{V}_c has been limited already, as we are considering only vertices in a cycle found in the enumeration phase) this is an assignment problem, and hence can be solved in polynomial time; we call it the *optimization subproblem*.

For helping illustrating the usage for this procedure, we present a list of possible distinct graphs of cycle length two and three in Appendix Appendix A. Figure 4 was prepared based on the four distinct cycles of size three, in order to visualize what happens to the *expected* size of the cycle when the probability of vertex withdrawal varies; identical probabilities for all vertices was assumed.

Figures 5 and 6 were prepared in order to provide an insight on the shape of solutions when probabilities of vertex withdrawal increase (once again, identical probability for all vertices was assumed). As can be seen, very significant changes in the solution structure may occur.

3.2. Arc withdrawal

Another source of unreliability comes from arc withdrawal: connections in the graph may become unavailable, mostly because of new sources of incompatibility between donors and patients discovered in a last, more thorough examination. As in the previous case, the cycle where failure occurred will have to be resolved. The expectation of the number of transplants corresponding to a given cycle configuration can be determined with an algorithm similar to Algorithm 1 but the sources of unreliability are arcs instead of vertices. For a given graph G , the data required for calculating expectation is, in this case, the probability of failure p_{ij} for each arc (i, j) in the graph; the expectation of the length of the maximum set of vertex-disjoint cycles of a (sub)graph can be computed with Algorithm 2.

As in the case of vertex withdrawal, it is interesting to see what happens in terms of solution shape when the probability of withdrawal of arcs in a graph increase. Figures 7 and 8 show how solutions vary with this probability, here considered identical for all arcs. Figure 7 presents a situation where vertex withdrawal and arc withdrawal may lead to different solutions. A more complex situation is presented in Figure 8, where there occur two changes in the solution for the graph on top, when shifting from deterministic to increasingly unreliable situations (bottom, from left to right).

3.3. Vertex and arc failure

Simultaneous vertex and arc failure can be handled by including in the sources of unreliability both vertices and arcs. In this case more than in the previous, the limiting part of the process of

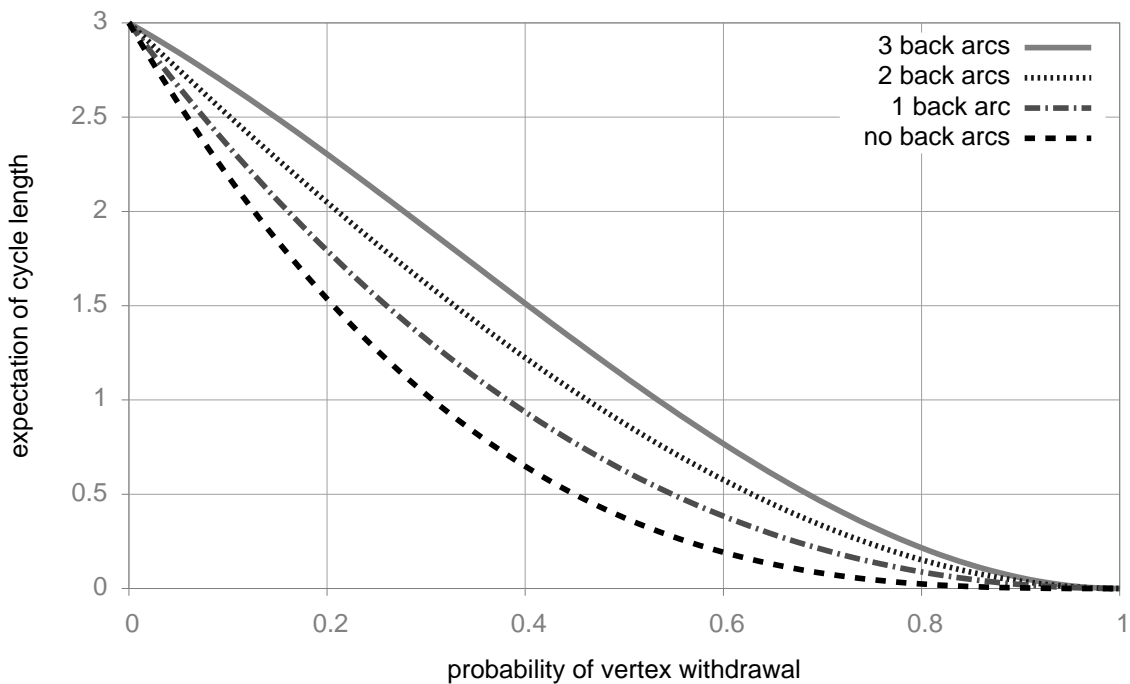


Figure 4: Expectation of cycle length as a function of vertex failure probability (considered identical for all vertices) for the four distinct cycle configurations with 3 vertices.

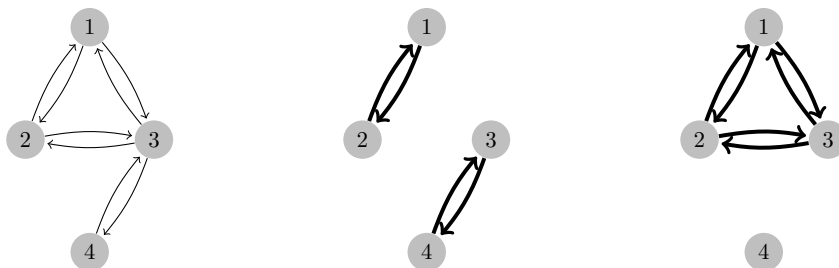


Figure 5: Left: original graph. Center: solution for high vertex reliability ($p_i < 1/3$). Right: solution for low vertex reliability ($p_i > 1/3$).

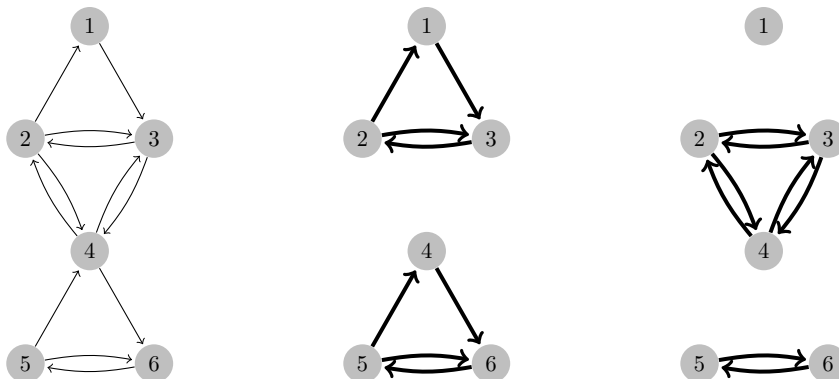


Figure 6: Left: original graph. Center: solution for high vertex reliability ($p_i < 0.2$). Right: solution for low vertex reliability ($p_i > 0.2$).

Algorithm 2: Size expectation for a cycle packing (unreliable arcs).

Input: $G_c = (\mathcal{V}_c, \mathcal{A}_c)$, $p_{ij} \forall (i, j) \in \mathcal{A}_c$

Output: Expectation for the input graph

- (1) $E := 0$ \leftarrow accumulator for expectation
- (2) $\mathcal{S} := 2^{\mathcal{A}_c}$ \leftarrow set of all subsets of \mathcal{A}_c
- (3) **foreach** $\mathcal{R} \in \mathcal{S}$: \leftarrow arcs remaining
- (4) $\mathcal{Q} := \mathcal{A}_c \setminus \mathcal{R}$ \leftarrow arcs quitting
- (5) $z =$ length of maximum vertex-disjoint cycle packing in $G' = (\mathcal{V}_c, \mathcal{R})$
- (6) $E := E + z \prod_{(i,j) \in \mathcal{R}} (1 - p_{ij}) \prod_{(i,j) \in \mathcal{Q}} p_{ij}$
- (7) **return** E

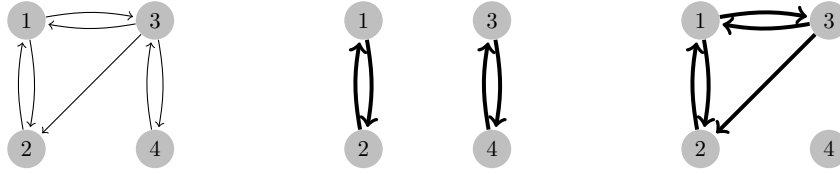


Figure 7: Differences between vertex and arc reliability: for vertex withdrawal, left hand graph has only the solution shown in the center; for arc withdrawal the solution may be the one in the center (for $p_{ij} < 0.1670\dots$), as well as the one in the right (for larger p_{ij}).

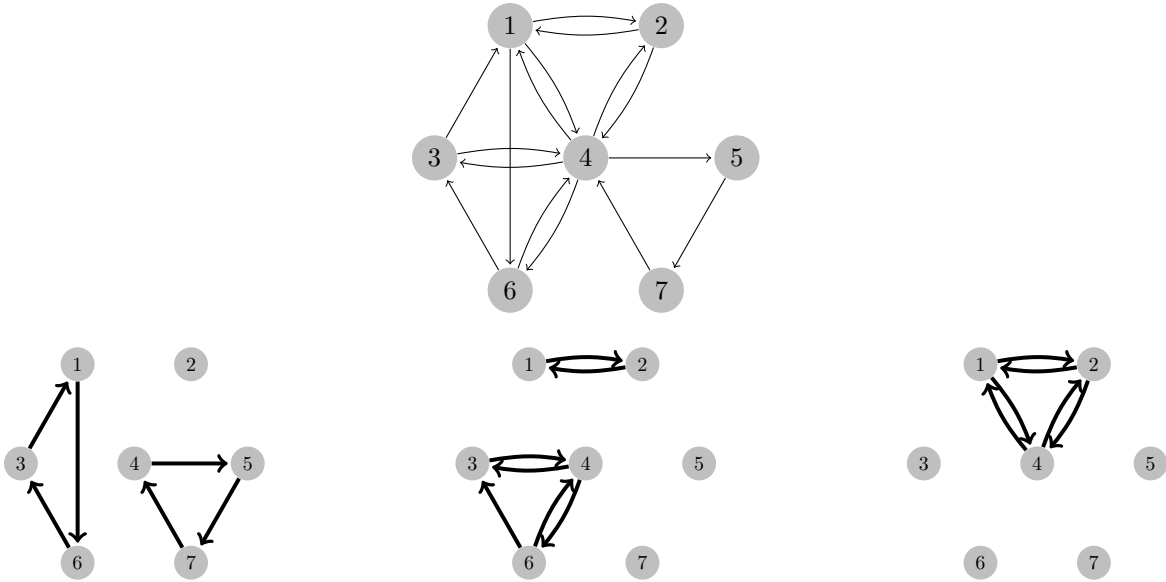


Figure 8: Example of a more complex graph (top): solutions from high to low reliability (bottom, from left to right).

calculating the expectation is the generation and analysis of the set of subsets of sources of potential failure; indeed, the set of subsets of \mathcal{S} in Algorithm 3 becomes extremely large, forbidding the direct use of this algorithm. An alternative to this will be presented in the next section.

4. Solution procedure

The limitations of algorithms 1 to 3 are due to the need of enumerating the set of all subsets of sources of unreliability (*i.e.*, \mathcal{V} , \mathcal{A} , of $\mathcal{V} \cup \mathcal{A}$). In other words, the limiting part is the very rapid growth of the set of subsets of \mathcal{S} with the cardinality of \mathcal{S} , in line (2) of each algorithm. Indeed, this makes the method not viable except for very small graphs. However, here we are dealing with subgraphs induced by vertices in a cycle of length at most K , found when enumerating cycles in the

Algorithm 3: Size expectation for a cycle packing (unreliable vertices and arcs).

Input: $G_c = (\mathcal{V}_c, \mathcal{A}_c)$, $p_i \forall i \in \mathcal{V}_c$, $p_{ij} \forall (i, j) \in \mathcal{A}_c$

Output: Expectation for the input graph

- (1) $E := 0$ \leftarrow accumulator for expectation
- (2) $\mathcal{S} := 2^{\mathcal{V}_c \cup \mathcal{A}_c}$ \leftarrow set of all subsets of $\mathcal{V}_c \cup \mathcal{A}_c$
- (3) **foreach** $\mathcal{R} \in \mathcal{S}$: \leftarrow vertices and arcs remaining
- (4) $\mathcal{V}' := \mathcal{V}_c \cap \mathcal{R}$ \leftarrow vertices in \mathcal{R}
- (5) $\mathcal{T} := \mathcal{V}_c \setminus \mathcal{V}'$ \leftarrow vertices quitting
- (6) $\mathcal{A}' := \{(i, j) \in \mathcal{A}_c \cap \mathcal{R} : i \in \mathcal{V}', j \in \mathcal{V}'\}$ \leftarrow arcs in induced subgraph
- (7) $\mathcal{Q} := \mathcal{A}_c \setminus \mathcal{A}'$ \leftarrow arcs quitting
- (8) $z =$ length of maximum vertex-disjoint cycle packing in $G' = (\mathcal{V}', \mathcal{A}')$
- (9) $E := E + z \prod_{i \in \mathcal{V}'} (1 - p_i) \prod_{(i, j) \in \mathcal{A}'} (1 - p_{ij}) \prod_{i \in \mathcal{T}} p_i \prod_{(i, j) \in \mathcal{Q}} p_{ij}$
- (10) **return** E

original graph. As the length K of cycles of the original graph to consider is, in practice, limited to a small number (due to limitations on the number of simultaneous operations), the algorithms have practical relevance if properly used.

An alternative to directly using the previous algorithms, in settings where the output is required with minimum delay, is to have previously prepared a database of possible configurations of graphs up to a given size, and for each of them store the function calculating the expression given by the algorithms (possibly after simplifying this expression, since it may become very large). As the graphs that we are dealing with here are small (the maximum number of vertices is K) and their number is limited, with appropriate data structures it is computationally acceptable to search this database for a graph which is isomorphic to the one under consideration, *i.e.*, isomorphic to the subgraph induced by nodes in a cycle of the original graph, itself found during the enumeration phase when setting up the main optimization problem. A list of all possible graphs with cycle length up to three, and the expressions for computing the corresponding expectations, is presented in Appendix Appendix A. Notice that the number of different graphs grows very rapidly with the number of vertices; for graphs with four vertices there are 61 distinct, non-isomorphic graphs that contain (at least) a cycle of length 4; for graphs with five vertices, this number is 3725 (see [7]). Maximum cycle length of five may, indeed, be the practical limit for this method; interestingly, it is also a common limit to the number of simultaneous transplants in kidney exchange programs.

Algorithm 4: Database construction.

- (1) Modify algorithms 1 to 3 for returning the *expression* for computing expectation;
- (2) Enumerate possible graph configurations up to the desired size;
- (3) For all distinct graphs (*i.e.*, non-isomorphic to one already in the database):
- (4) Apply the relevant algorithm (1, 2, or 3);
- (5) Optionally, simplify the expression obtained with mathematical computation software;
- (6) Store in a database the graph as a key and the expression as the data;
- (7) (Use this database for setting up instances of Problem (1a)–(1c).)

In order to use this approach, a preliminary step is the construction of a database of all possible graphs containing at least a cycle of relevant length (excluding isomorphic graphs) and their corresponding expressions, as detailed in Algorithm 4. It turns out that, even with a careful (though straightforward) implementation, the most time consuming operation in this algorithm is the precomputation of formulas for expectations for each of the graph configurations.

After the database is constructed, it can be used for setting up the optimization problem, using Algorithm 5. Here, the expectation formula determined with Algorithm 4 and stored in the database is used in line (7) for computing the objective coefficient for each cycle appearing in the instance. For this purpose, a cycle in the instance is matched with an isomorphic cycle previously enumerated and

stored in the database; *i.e.*, a key that matches the current cycle is searched for, the corresponding expectation formula is extracted, and the probabilities from the instance’s cycle are mapped to the cycle stored. Finally, a numeric value for the expectation is computed with that formula, and used as the cycle’s coefficient in the objective.

Algorithm 5: Solution procedure.

- (1) Read instance: compatibility between pairs, withdrawal probabilities
- (2) Prepare compatibility graph
- (3) Enumerate cycles of relevant size
- (4) Setup optimization model:
- (5) Create one variable for each cycle
- (6) Constraints: each vertex in at most one cycle
- (7) Objective coefficients: for each cycle, the corresponding expectation
- (8) Solve optimization model, Problem (1a)–(1c)

After the optimization instance is prepared, it is solved in line (8) of the algorithm; this is done by means of a general-purpose mixed-integer programming solver. Computationally, this turned out to be relatively inexpensive, compared to the effort required for setting up the instance. The solution of the optimization problem can afterwards be used for the actual implementation of the KEP: contacting the selected pairs, verifying if the solution is indeed possible (*i.e.*, checking if there are no back outs), making a last-minute compatibility check for each pair, possibly rearranging the solution, and finally executing the transplants.

4.1. Computational experiment

Algorithm 5 has been used to optimize size expectation for vertex-disjoint cycle packing on benchmark instances used in kidney exchange programs, available in [3]. This set, named *blood-type test instances*, includes graphs constructed by means of a generator referred to in [10], which creates random graphs based on probabilities of blood type and donor–patient compatibility. Instances tested here vary on size, from 10 to 1000 vertices; hence, they correspond to realistic KEP situations. Sizes up to 100 vertices include 50 different instances each; for larger sizes there are 10 different instances per size. Each instance’s data of [3] was complemented with probabilities for vertex and arc withdrawal, putting together the *non-deterministic* version of the instance¹. The maximum cycle size considered was three; this allowed the complete enumeration of all relevant cycles for the instances considered, and thus the usage of the proposed algorithms in a straightforward setting (*i.e.*, there was no need of recourse to more sophisticated methods, such as decomposition and column generation, for solving the main optimization problem).

Table 1 presents average results for instances of each size. The average number of cycles of length three, for all instances of each size, is shown in column N . Values on column \bar{z}^D are average objective values obtained when maximizing the number of transplants in a deterministic setting, *i.e.*, with $w_c = |c|$. Column \bar{z}^N lists, for the same instances, the average objective values obtained for the non-deterministic case. This is the average, for each instance size, of the maximum expected number of transplants with withdrawal probabilities on vertices and arcs, obtained with values for w_c as determined by algorithms 4 and 5.

Let us consider an optimal solution to the deterministic version of a given instance. Besides its objective value z^D , we can compute the expectation of the number of transplants that it would lead to in a corresponding non-deterministic setting; let us denote it by y^D . Similarly, let z^N be the objective value for the non-deterministic version of an instance, and y^N the number of transplants that the solution would lead to in the corresponding deterministic setting. The percentage of missed operations when the deterministic solution is used in a probabilistic setting can be calculated as $M^D = 100(z^N - y^D)/z^N$. Analogously, the percentage of missed operations if the non-deterministic solution is used in a completely reliable setting can be calculated as $M^N = 100(z^D - y^N)/z^D$.

¹Instances’ data are available at the author’s home page.

Averages for these values are listed in the corresponding columns in Table 1. These results show that solutions for maximum expectation are much more accurate and robust in a deterministic setting than the inverse. This tendency seems to be more pronounced on large instances (see also Figure 9).

Size	$ \overline{\mathcal{C}} $	\bar{z}^D	\bar{y}^D	\bar{M}^D	\bar{z}^N	\bar{y}^N	\bar{M}^N
10	5.62	2.720	0.170	30.3	0.243	2.420	11.0
20	46.02	8.200	0.396	60.9	1.013	7.000	14.6
30	109.1	12.26	0.712	52.4	1.495	10.58	13.7
40	221.8	17.28	1.046	52.7	2.213	14.76	14.6
50	442.2	23.48	1.312	60.9	3.352	20.04	14.7
60	722.6	27.64	1.658	61.0	4.256	24.46	11.5
70	1117.	34.04	1.886	61.8	4.932	30.02	11.8
80	1686.	39.54	2.279	63.6	6.260	34.00	14.0
90	2472.	46.18	2.626	64.8	7.467	40.36	12.6
100	3264.	51.28	2.864	65.5	8.314	44.96	12.3
200	24401.	107.1	5.431	74.6	21.40	97.20	9.2
300	87342.	163.6	8.570	75.7	35.21	150.1	8.3
400	224763.	221.3	13.06	75.3	52.93	206.8	6.6
500	419670.	280.6	16.00	77.5	71.00	264.0	5.9
600	791068.	341.8	20.00	77.5	88.73	322.4	5.7
700	1057876.	383.3	21.49	77.9	97.43	364.3	5.0
800	1586344.	441.8	23.84	79.5	116.22	425.1	3.8
900	2343290.	498.8	29.87	78.3	137.56	481.0	3.6
1000	2952990.	548.3	33.60	78.1	153.54	530.3	3.3

Table 1: Graph size of benchmark instances and average number of cycles of sizes 2 or 3 ($|\overline{\mathcal{C}}|$); average values for the objective value in deterministic and non-deterministic settings, and performance of the corresponding solutions in the reverse situation.

Concerning CPU times required for solving these problems, the limiting part was cycle enumeration and the evaluation of the corresponding expectation. Indeed, solving the integer optimization problems using Gurobi version 5.0.1 [6] required less than 0.1 second for the 10-vertex instances, less than 1 second for 100 vertices, and about 400 seconds for the 1000-vertex instances. Cycle enumeration and evaluation, implemented in the Python language, also used less than one second for instances with 100 or fewer vertices, but the time required increased up to more than 5000 seconds for the largest, 1000-vertex instances. The system used was the following: a computer with an Intel Xeon processor at 3.0 GHz, running Linux version 2.6.32; only one thread was assigned to this experiment.

5. Conclusions

This paper proposes a method for computing the maximum expectation for the length of a set of vertex-disjoint cycles in a digraph where vertices and/or arcs are subject to failure with a known probability. This problem has a practical application on maximizing the expected number of transplants in a kidney exchange program. It makes use of a formulation where all cycles of feasible length are enumerated. An intermediate step in the solution procedure is the construction of a database associating cycle configurations to the corresponding expectations, for all small graphs of relevant size; this is employed for setting up the main optimization problem.

A set of well-known benchmark instances was used for testing the procedure, considering a maximum cycle-size of three; the experiment unveiled a superior robustness of models maximizing expectation, as opposed to models maximizing total cycle length in a deterministic setting. This is a clear indication that probabilistic information should be taken into account for the solution of this problem.

In usual situations the difficulty of the solution of NP-hard optimization problems is stressed; it should be noted that in the current context optimization of the main problem took only a small

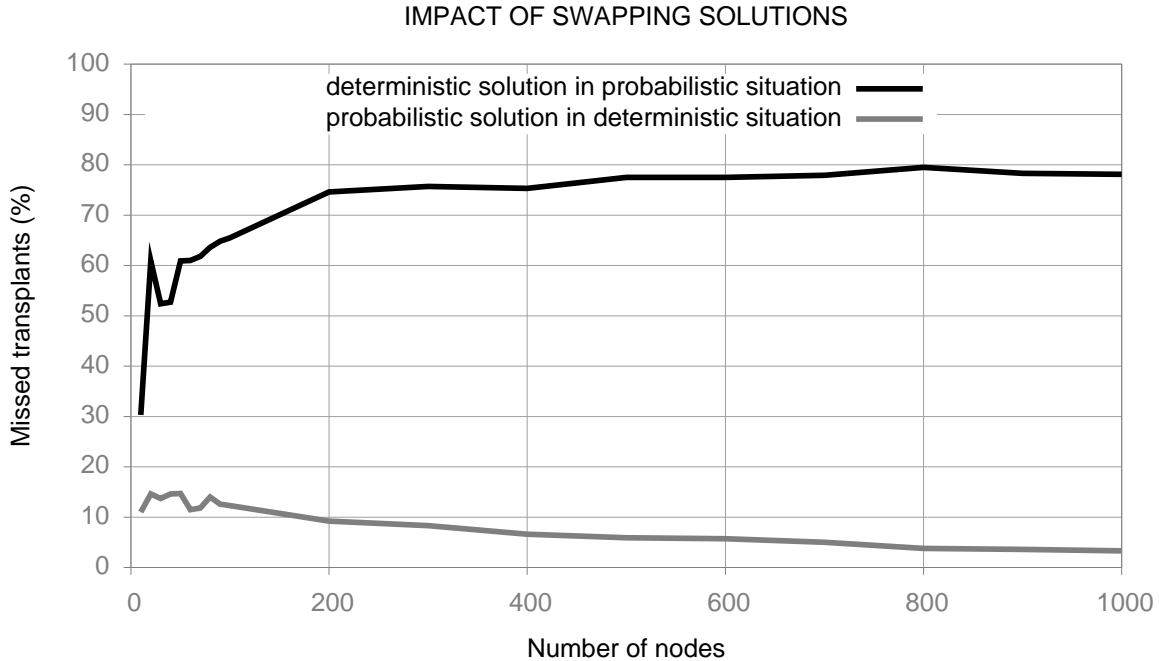


Figure 9: Percentage of missed operations if deterministic and non-deterministic solutions are used in the reverse situation as a function of the instance size.

fraction of the computational effort required. Most of the effort was used in the identification of cycle configurations and in determining the corresponding contribution to the expectation.

For a given graph, the efficient computation of the expectation of the number of arcs in a vertex-disjoint cycle packing (even its simple computation, not its maximization) is an interesting subject for future research. Though we were able to do this systematically for every distinct graph with up to 5 vertices having a cycle of its size, doing it for the general case does not seem straightforward.

Acknowledgements

We would like to thank Dr. Margarida Carvalho, Dr. Xenia Klimentova and Prof. Ana Viana, from INESC TEC, for their valuable comments.

References

- [1] D. J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. *Proceedings of the 8th ACM conference on Electronic commerce*, pages 295–304, 2007.
- [2] Y Chen, Y Li, J D Kalbfleisch, Y Zhou, A Leichtman, and P X Song. Graph-based optimization algorithm and software on kidney exchanges. *Biomedical Engineering, IEEE Transactions on*, 59(7):1985–1991, 2012.
- [3] Miguel Constantino, Xenia Klimentova, Ana Viana, and Abdur Rais. New insights on integer-programming models for the kidney exchange problem. Technical Report 4, Centro de Investigao Operacional, Universidade de Lisboa, 2012.
- [4] J P Dickerson, A D Procaccia, and T Sandholm. Optimizing kidney exchange with transplant chains: theory and reality. In Conitzer, Winikoff, Padgham, and van der Hoek, editors, *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*,

pages 711–718, Valencia, Spain, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

- [5] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics. Journal Canadien de Mathématiques*, 17:449–467, 1965.
- [6] Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual, Version 5.0*. <http://www.gurobi.com>, 2012.
- [7] Frank Harary and Edgar M. Palmer. *Graphical Enumeration*. Academic Press, New York, 1973.
- [8] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.
- [9] Y. Li, J. Kalbfleisch, P. Song, Y. Zhou, A. Leichtman, and M. Rees”. Optimization and simulation of an evolving kidney paired donation (KPD) program. Working Paper Series 90, Department of Biostatistics, University of Michigan, May 2011. <http://www.bepress.com/umichbiostat/paper90>.
- [10] S.L. Saidman, A.E. Roth, T. Sonmey, et al. Increasing the opportunity of live kidney donation by matching for two- and three-way exchanges. *Transplantation*, 81:773–782, 2006.
- [11] T. Sönmez and M.U. Ünver. *Oxford Handbook of Market Design*, chapter Market Design for Kidney Exchange. Oxford University Press, 2013.
- [12] M Ünver. Dynamic kidney exchange. *Review of Economic Studies*, 77(1):372–414, 2010.

Appendix A. Graphs for vertex failure

Given the probability p_i for failure of vertex i , and/or probability p_{ij} of failure of arc (i, j) , expressions of the expectation for the size of the cycle is presented for all cycle configurations in two- and three-vertex graphs. The original, more lengthy expressions were simplified using the mathematical computation software *Mathematica*.

Appendix A.1. Two-vertex graphs


Graph	Expectation with failure on
	vertices: $2(p_1 - 1)(p_2 - 1)$ arcs: $2(p_{12} - 1)(p_{21} - 1)$ both: $2(p_1 - 1)(p_2 - 1)(p_{12} - 1)(p_{21} - 1)$

Table A.2: Graphs with two vertices (1 configuration).

Appendix A.2. Three-vertex graphs

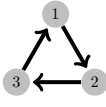
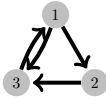
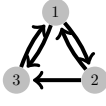
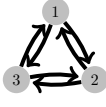
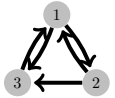
Graph	Expectation with failure on
	vertices: $-3(p_1 - 1)(p_2 - 1)(p_3 - 1)$ arcs: $-3(p_{12} - 1)(p_{23} - 1)(p_{31} - 1)$ both: $3(p_1 - 1)(p_2 - 1)(p_3 - 1)(p_{12} - 1)(p_{23} - 1)(p_{31} - 1)$
	vertices: $-(p_1 - 1)(p_2 - 3)(p_3 - 1)$ arcs: $-(p_{12}(2p_{13} + 1)(p_{23} - 1) - (2p_{13} + 1)p_{23} + 3)(p_{31} - 1)$ both: $(p_1 - 1)(p_3 - 1)(p_2(p_{12} - 1)(2p_{13} + 1)(p_{23} - 1) - p_{12}(2p_{13} + 1)(p_{23} - 1) + 2p_{13}p_{23} + p_{23} - 3)(p_{31} - 1)$
	vertices: $(p_1 - 1)(p_3 + p_2(p_3 + 1) - 3)$ arcs: $3 + (2p_{13}p_{21} + 1)p_{23}(p_{31} - 1) - 2p_{21}p_{31} - p_{31}$ $-p_{12}(p_{23}(p_{31} - 1) + 2p_{13}(p_{21}p_{23} - 1)(p_{31} - 1) - 2p_{21}p_{31} + p_{31} + 1)$ both: (see Appendix A.2.1)
	vertices: $-p_3 - p_2(p_3 + 1) + p_1(-p_3 + p_2(3p_3 - 1) - 1) + 3$ arcs: (see Appendix A.2.1) both: (this expression would not fit in one A4 page)

Table A.3: Graphs with two vertices (4 configurations).

Appendix A.2.1. Longer expectation expressions

First case: unreliable vertices and arcs are considered for graph

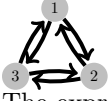


$$G = (\{1, 2, 3\}, \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1)\}).$$

The expression for expectation when vertices and arcs are subject to failure is the following:

$$\begin{aligned}
 &(p_1 - 1)(2p_{13}p_{12} - 2p_{13}p_{21}p_{23}p_{12} - p_{23}p_{12} - 2p_{13}p_{31}p_{12} - 2p_{21}p_{31}p_{12} + 2p_{13}p_{21}p_{23}p_{31}p_{12} \\
 &\quad + p_{23}p_{31}p_{12} + p_{31}p_{12} + p_{12} + 2p_{13}p_{21}p_{23} + p_{23} - p_3(-p_{23} + p_{21}(2 - 2p_{13}p_{23})) \\
 &\quad + p_{12}(-2p_{21} + p_{23} + 2p_{13}(p_{21}p_{23} - 1) + 1) + 1)(p_{31} - 1) + 2p_{21}p_{31} - 2p_{13}p_{21}p_{23}p_{31} \\
 &\quad - p_{23}p_{31} + p_{31} + p_2(p_{12} - 1)(-p_{31}p_{23} + p_{23} - 2p_{13}(p_{21}p_{23} - 1)(p_{31} - 1) \\
 &\quad + p_3(-2p_{21} + p_{23} + 2p_{13}(p_{21}p_{23} - 1) + 1)(p_{31} - 1) + 2p_{21}p_{31} - p_{31} - 1) - 3)
 \end{aligned}$$

Second case.: reliable vertices and unreliable arcs are considered for graph



$$G = (\{1, 2, 3\}, \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)\}).$$

The expression for expectation when only arcs are subject to failure is the following:

$$\begin{aligned} & -p_{21}p_{23} - p_{21}p_{31}p_{23} + p_{21}p_{32}p_{23} + p_{21}p_{31}p_{32}p_{23} + p_{31}p_{32}p_{23} - p_{32}p_{23} - p_{21}p_{31} \\ & -p_{21}p_{31}p_{32} - p_{31}p_{32} + p_{13} (p_{23} (p_{31} - 1) (-p_{32} + p_{21} (p_{32} + 1) + 1) - (p_{21} - 1) p_{31} (p_{32} - 1)) \\ & + p_{12} (- (p_{23} (p_{31} - 1) + p_{31} + 1) p_{32} + p_{21} (p_{23} (p_{31} - 1) (p_{32} - 1) + p_{32} + p_{31} (p_{32} + 1) - 1) \\ & \quad + p_{13} (p_{23} (p_{31} + 1) (p_{32} - 1) + (p_{31} - 1) (p_{32} + 1) + p_{21} ((p_{31} - 1) (p_{32} - 1) \\ & \quad \quad + p_{23} (p_{31} (1 - 3p_{32}) + p_{32} + 1)))) + 3 \end{aligned}$$