A New Branch-and-Price Approach for the Kidney Exchange Problem

Xenia Klimentova[1], Filipe Alvelos[2], and Ana Viana[1,3]

[1] INESC TEC,
Campus da FEUP, Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal
[2] Centro Algoritmi / Departamento de Produção e Sistemas,
Universidade do Minho,
Campus de Gualtar, 4710-057 Braga, Portugal
[3] ISEP - School of Engineering, Polytechnic of Porto,
Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal
xenia.klimentova@icc.ru, falvelos@dps.uminho.pt, aviana@inescporto.pt

Abstract. The kidney exchange problem (KEP) is an optimization problem arising in the framework of transplant programs that allow exchange of kidneys between two or more incompatible patient-donor pairs. In this paper an approach based on a new decomposition model and branch-and-price is proposed to solve large KEP instances. The optimization problem considers, hierarchically, the maximization of the number of transplants and the minimization of the size of exchange cycles. Computational comparison of different variants of branch-and-price for the standard and the proposed objective functions are presented. The results show the efficiency of the proposed approach for solving large instances.

Keywords: Kidney Exchange Problem, Integer Programming, Column Generation, Branch-and-price.

1 Introduction

The Kidney Exchange Problem arises in the framework of kidney exchange programs which were organized in different countries in recent years [18,6,26,27,1]. Patients with kidney failure can participate in these programs if they have a donor willing to donate him/her a kidney, but the pair is not physiologically compatible and because of that the transplantation can not be performed. In the most basic structure the idea is to organize exchanges between a number of such pairs so that a patient in one pair receives a kidney from a donor in the other pair and vice versa (see Fig. 1). The objective for optimization in a kidney exchange program is generally to maximize the collective benefit for a given pool of incompatible pairs, measured by the number of possible kidney exchanges [18,27], but other objectives may also be considered [19,10,22,15]. We will refer to this optimization problem as the *Kidney Exchange Problem* (KEP). Basically an exchange can be considered as a cycle in which the donor from one

B. Murgante et al. (Eds.): ICCSA 2014, Part II, LNCS 8580, pp. 237–252, 2014.

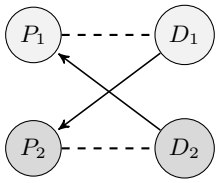

Fig. 1. The donor of the first pair D_1 gives a kidney to the patient P_2 of the second pair, and the patient P_1 gets a kidney from donor D_2

incompatible pair gives a kidney to a compatible recipient in another pair and so on, finally forming an alternating directed cycle (Fig. 1 illustrates an exchange cycle involving 2 pairs).

An important question for the KEP is the definition of an upper bound on the number of pairs that can be involved in a cycle. There are two main reasons that make this upper bound setting inevitable. First, because all operations in an exchange have to be performed simultaneously, the number of personnel and facilities needed for simultaneous operations bring several logistics problems. Second, because last-minute testing of donor and patient can elicit new incompatibilities that were not detected before, causing the donation and all possible exchanges in that cycle to be cancelled, it is preferred that cycles are of limited size. If only 2 pairs can be involved in an exchange cycle the KEP is a maximum cardinality matching problem which can be solved in polynomial time [27,21]. However when the maximum number of pairs is bounded by an integer number $k, 3 \leq k < n$, where n is number of pairs in pool, the problem is NP-hard [6,1,17].

Integer Programming (IP) is a natural framework for modeling the KEP and develop methods for seeking an optimal solution. There are two known non-compact formulations for the KEP [25], so called "edge formulation" and "cycle formulation". The former has exponential number of constraints and the later exponential number of variables. As an alternative in [7] compact formulations with the number of variables and constraints bounded by polynomials were proposed. The most effective from a computational point of view is the so called "extended edge formulation". In [1] for the cycle formulation a branch-and-price scheme (combination of column generation with branch-and-bound) is implemented. The authors carried out computational experiments on data generated by current state-of-the-art generator [26] with $k = 3$. A branch-and-price approach for the cycle formulation was also studied in [15].

In this paper we propose a new decomposition model entitled disaggregated cycle decomposition (DCD) which is related to the compact IP formulation proposed in [7] and to the cycle formulation. The proposed model is solved by a branch-and-price algorithm. We also introduce a new objective function with two components, hierarchically related: the first one is the commonly used maximization of the number of transplants, and the second is related with the minimization of the number of pairs involved in exchange cycles. Through use of

weights we can represent this hierarchy in a single objective function where the first component is made the most relevant one. Computational experience was carried out to test the proposed DCD and compare the standard and the newly proposed objective function on test instances of different size, for different values on the maximum length of the cycles (k). This new objective function is of practical interest as it increases the probability of the actual number of transplants i.e. the transplants that can still be performed after the last compatibility tests, that are only performed after pairs are matched (since the size of the cycles will tend to be smaller). It also makes branch-and-price much faster at reaching an optimal solution.

2 Problem Statement and Formulations

Graph theory can provide a natural framework for representing the KEP 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 a donor from pair i is compatible with the patient of pair j. To each arc (i, j) is associated a weight w_{ij}. If the objective is maximizing total number of transplants $w_{ij} = 1$, $\forall (i, j) \in A$.

The Kidney Exchange Problem can be defined as follows:

Find a maximum weight packing of vertex-disjoint cycles with length at most k.

In the remaining of this document, without loss of generality, we will consider $w_{ij} = 1$, $\forall ij \in A$.

The work presented in this paper is related with two formulations previously presented in the literature, the cycle formulation and the reduced extended edge formulation, that can be described as follows.

2.1 Cycle Formulation

Let $\mathcal{C}$ be the set of all cycles in G with length at most k. We assume that a cycle is an ordered set of arcs. Define a variable $z_c = 1$ if cycle c is selected for the exchange $c \in \mathcal{C}$. Denote by $V(c) \subseteq V$ the set of vertices which belong to cycle c and the number vertices in the cycle is $|c| = |V(c)|$. The cycle formulation is written as follows:

$$\text{maximize} \quad \sum_{c \in \mathcal{C}} |c| z_c \tag{1}$$

$$\text{Subject to:} \quad \sum_{c : i \in V(c)} z_c \leq 1 \qquad \forall i \in V, \tag{2}$$

$$z_c \in \{0, 1\} \qquad \forall c \in \mathcal{C}. \tag{3}$$

The objective function (1) maximizes the number of transplants. Constraints (2) ensure that every vertex is in at most one of the selected cycles (so the donor will

only donate one kidney and the patient will receive only one). The difficulty of this formulation is induced by exponential number of variables defined by exponential (in general case) number of cycles of length at most k.

2.2 Reduced Extended Edge Formulation

Let L be an upper bound on the number of cycles in any solution (e.g. the number of vertices). The cycles in the solution can be represented by an index l, with $1 \leq l \leq L$, the problem variables for the extended edge formulation being x_{ij}^l: $x_{ij}^l = 1$ if arc (i,j) is selected to be in a cycle with index l, $x_{ij}^l = 0$, otherwise.

To eliminate redundant variables and avoid symmetry of extended edge formulation some reduction procedures were implemented in [7]. It is imposed that a cycle l in the solution must have node l, and any other nodes in that cycle must have an index larger than l. Moreover, based on this restriction the variables can be eliminated when l cannot be in the same cycle with a node i or an arc $(i,j) \in A$. Denote by $\tilde{V}^l = \{i \in V \ : \ i \geq l\}$ and by d_{ij}^l the shortest path distance in terms of number of arcs in graph G from i to j for $i,j \in \tilde{V}^l$ such that the path passes only through vertices of set $\tilde{V}^l$. Let $d_{ij}^l = +\infty$ if there is no such path from i to j. For each vertex $l \in V$, let us build the set of vertices:

$$V^l = \{i \in V | i \geq l \ \text{ and } \ d_{li}^l + d_{il}^l \leq k\}.$$

It can happen that for some $l \in \{1,\ldots,L\}$, $V^l = \emptyset$. Denote by $\mathcal{L} \subseteq \{1,\ldots,L\}$ the set of indices l such that $V^l \neq \emptyset$. Define the set of arcs as:

$$A^l = \{(i,j) \in A \mid i,j \in V^l \text{ and } d_{li}^l + 1 + d_{jl}^l \leq k\}$$

Considering subgraphs $G^l(V^l, A^l)$ for all $l \in \mathcal{L}$, the reduced extended edge formulation is given as:

$$\text{maximize} \quad \sum_{l \in \mathcal{L}} \sum_{(i,j) \in A^l} x_{ij}^l \tag{4}$$

$$\text{subject to} \quad \sum_{j:(j,i) \in A^l} x_{ji}^l = \sum_{j:(i,j) \in A^l} x_{ij}^l \qquad \forall i \in V^l, \forall l \in \mathcal{L}, \tag{5}$$

$$\sum_{l \in \mathcal{L}} \sum_{i:(i,j) \in A^l} x_{ij}^l \leqslant 1 \qquad \forall j \in \bigcup_{l \in \mathcal{L}} V^l, \tag{6}$$

$$\sum_{(i,j) \in A^l} x_{ij}^l \leqslant k \qquad \forall l \in \mathcal{L} \tag{7}$$

$$\sum_{j:(i,j) \in A^l} x_{ij}^l \leqslant \sum_{j:(l,j) \in A^l} x_{lj}^l \qquad \forall i \in V^l, \forall l \in \mathcal{L} \tag{8}$$

$$x_{ij}^l \in \{0,1\}. \qquad \forall (i,j) \in A, \forall l \in \mathcal{L} \tag{9}$$

The objective (4) is to maximize the total number of arcs in the set of all subgraphs of the graph. Constraints (5) guarantee that in each cycle l the number of kidneys received by patient i is equal to number of kidneys given by donor i. Constraints (6) make sure a donor donates only once. Constraints (7) state that in each cycle a maximum number of k arcs is allowed. Finally constraints (8) assure that whenever an arc (i, j) is in cycle l a node with index l is also included in this cycle.

3 Disaggregated Cycle Decomposition

Exact methods based on the decomposition of the cycle formulation were implemented in [26,1,15]. Here we propose a different decomposition based on disaggregation of the cycles according to the subgraphs $G^l(V^l, A^l)$ defined above. For each $l \in \mathcal{L}$ let $\mathcal{C}^l$ be the set of cycles in subgraph G^l with length at most k arcs and including vertex l. For each cycle c in set $\mathcal{C}^l$ define a decision variable $y^{lc} = 1$ if cycle c of set $\mathcal{C}^l$ is chosen and $y^{lc} = 0$, otherwise. Similarly to the cycle formulation denote by $V^l(c) \subseteq V^l$ the set of vertices which belong to cycle $c \in$, $\mathcal{C}^l$ and the number of vertices in the cycle is $|c|^l = |V^l(c)|$.

The disaggregated cycle decomposition (DCD) is given as follows:

$$\text{maximize} \quad \sum_{l \in \mathcal{L}} \sum_{c \in C^l} |c|^l y^{lc} \tag{10}$$

$$\text{subject to} \quad \sum_{l \in \mathcal{L}} \sum_{c \in \mathcal{C}^l : i \in V^l(c)} y^{lc} \leq 1 \quad \forall i \in V \tag{11}$$

$$y^{lc} \in \{0, 1\} \qquad \forall l \in \mathcal{L}, \forall c \in C^l. \tag{12}$$

As in the cycle formulation, the number of variables of the DCD grows exponentially with the size of the underlying graph. Therefore approaches based on column generation (e.g. branch-and-price) are the most suitable for obtaining optimal solutions.

Column generation (CG) method allows obtaining an optimal solution of the linear relaxation of a decomposition without all the decision variables. CG alternates between solving a problem with a restricted set of variables (the so called restricted master problem - RMP), and solving subproblems where variables that may potentially improve the current solution of the RMP are identified by using their linear programming reduced costs. In the DCD a subproblem SP(l) is associated with each vertex $l \in \mathcal{L}$ and a solution of SP(l) is a cycle in graph G^l of length at most k and including vertex l. Using the variables x^l_{ij}, $(i, j) \in A^l$, $l \in \mathcal{L}$ defined above for the extended edge formulation, subproblem SP(l), $l \in \mathcal{L}$ is written as follows:

$$\text{maximize} \quad \sum_{(i,j)\in A^l} x_{ij}^l - \sum_{j\in V^l} \lambda_j \sum_{(i,j)\in A^l} x_{ij}^l, \tag{13}$$

$$\text{subject to} \quad \sum_{j:(j,i)\in A^l} x_{ji}^l = \sum_{j:(i,j)\in A^l} x_{ij}^l \qquad \forall i \in V^l, \tag{14}$$

$$\sum_{i:(i,j)\in A^l} x_{ij}^l \leq 1 \qquad \forall j \in V^l,\ j \neq l, \tag{15}$$

$$\sum_{i:(i,j)\in A^l} x_{il}^l = 1, \tag{16}$$

$$\sum_{(i,j)\in A^l} x_{ij}^l \leq k \tag{17}$$

$$x_{ij}^l \in \{0,1\} \qquad \forall (i,j) \in A^l,$$

Here λ_j, $j \in V$ are the dual variables for the packing constraints (11) of the DCD.

By construction of subgraphs G^l there always exists a feasible cycle including node l for each $l \in \mathcal{L}$ (otherwise $l \notin \mathcal{L}$) and the SP(l) always has an optimal solution. Hence the constraints (6) and (8) can be modified into (15) and (16). The other constraints are similar to constraints of the formulation (4)–(9).

A major issue when using relaxations in integer programming models is the quality of the bound they provide. It can be derived that the optimal value of the linear relaxation of the DCD is the same as the optimal value of the linear relaxation of the cycle formulation. Moreover the DCD is stronger (the optimal value of its linear relaxation is equal or better) than the extended edge formulation. This can be proved by noting that the DCD can be seen as a Dantzig-Wolfe decomposition [8] of the extended edge formulation represented by (4)–(9) where constraints (6) are the only constraints kept in the master problem (10)-(12), and convexity constraints are redundant. Since the subproblems do not have the integrality property (i.e. the polyhedrons of their linear relaxations have fractional extreme points), the bound provided by the linear relaxation of the decomposed model dominates the bound provided by the linear relaxation of the original model (as proved in general in [14], and by the equivalence between Lagrangean Relaxation and Dantzig-Wolfe decomposition).

Remark 1. State-of-the-art kidney exchange programs include altruistic donors, i.e., donors that are not associated to any patient, but willing to donate a kidney to someone in need. Included into a kidney exchange program an altruistic donor 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 [20,13,24]. European programs consider bounded chains with length at most k' (k' can be different from k) [19,15]. In programs running in the USA so called Never-Ending-Altruistic-Donor chains are considered, where a kidney from the last donor in a chain is not assigned to a patient in the deceased

donor list [9,23,3,12]. Instead, the last donor acts as an altruistic donor in 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 program. To take into account the European variant of the KEP with altruistic donors some constraints similar to (7) are to be added into the extended edge formulation. For the cycle formulation an extension on the set of cycles $\mathcal{C}$ with respect to inclusion of chains needs to be performed (see [7] for more details). As for DCD the sets $\mathcal{C}^l$ where l are altruistic donors have to be constructed with respect to constant k' (similarly to the cycle formulation), and k' substitutes k in the constraints (17) of the corresponding subproblems SP(l). These modifications do not change significantly the structure of the problem. Therefore the inclusion of altruistic donor chains is not considered in this paper. However all the implemented techniques can be adapted to the problem with bounded length chains as well.

4 A New Objective Function

As mentioned above the bound k on the number of incompatible pairs involved in exchange cycles is important because last-minute tests may find out incompatibilities that were not detected before, causing all possible exchanges in that cycle to be cancelled. Hence among all possible optimal solutions of the problem the ones having smaller cycles are preferable. This goal can be achieved by introducing an additional objective function related with the number of cycles (which is desired to be high) and considering the objectives hierarchically. Given that the number of cycles is $\sum_{l\in\mathcal{L}}\sum_{c\in\mathcal{C}^l} y^{lc}$, the problem with the second objective can be written as:

$$\text{maximize} \quad \sum_{l\in\mathcal{L}}\sum_{c\in\mathcal{C}^l} y^{lc} \tag{18}$$

$$\text{subject to} \quad \sum_{l\in\mathcal{L}}\sum_{c\in\mathcal{C}^l:i\in V^l(c)} y^{lc} \le 1 \quad \forall i \in V, \tag{19}$$

$$\sum_{l\in\mathcal{L}}\sum_{c\in C^l} |c|^l y^{lc} \ge v^*, \tag{20}$$

$$y^{lc} \in \{0,1\} \qquad \forall l \in \mathcal{L}, \forall c \in C^l. \tag{21}$$

where v^* is an optimal value of the problem DCD. Denote the hierarchical problem defined by DCD and (18)–(21) as H. It can be derived that the objective functions (10) and (18) may be combined into a single one, keeping their hierarchy, with help of weights as follows:

$$\sum_{l\in\mathcal{L}}\sum_{c\in\mathcal{C}^l}(|V||c|^l+1)y^{lc}. \tag{22}$$

The weight $|V|$ is given to maximize the number of transplants, implicitly stating that this objective is more important than the maximization of the number of

cycles. An optimal value for hierarchical problem H may be obtained by solving the DCD with objective function (22) (we will denote this problem (22), (11)–(12) as DCD_{nc}). Indeed, as the number of cycles in a solution cannot exceed $|V|$, an optimal solution of the DCD_{nc} will lead to the same number of transplants obtained when solving DCD. However, thanks to the second component of (22), the solution for the DCD_{nc} will have equal or bigger number of cycles, implying that cycles are of equal or smaller sizes. Therefore, it can be stated that by solving the problem DCD_{nc}, we also solve the problem DCD.

5 Solution Methods

As the DCD has exponential number of variables branch-and-price based on CG is an adequate method to obtain optimal solutions for the problem. Implementation details are presented in this section. They are preceded by a preprocessing phase, that generates subgraphs $G^l(V^l, A^l)$. The shortest paths d^l_{ij} used for construction of the subgraphs are calculated using Floyd-Warshall algorithm [11].

5.1 Column Generation

The general scheme of the implemented column generation is given by Fig 2.

```
Step 0. Initialize the RMP.
Step 1. If a heuristic to solve subproblems is used
        {
           Do {
                Solve the RMP with a linear programming solver.
                Solve the subproblems SP(l) heuristically.
              }
           while(at least one subproblem provides one attractive column)
        }
        Else
           Go to Step 2.
Step 2. Do {
           Solve the RMP with a linear programming solver.
           Solve the subproblems SP(l) exactly.
           }
        while(at least one subproblem provides one attractive column)
STOP
```

Fig. 2. Column generation general scheme

Initialization of RMP

Although it is reported in [1] that it is effective to initialize RMP with some set of initial columns, our preliminary experiments showed that for the implementation proposed in this work the most effective methods are the ones which do not use any initial columns. So in the computational experiments presented in section 6 the RMP was initialized with solutions of sub-problems SP(l) obtained with 0 values of dual variables λ_j, $j \in V^l$, $l \in \mathcal{L}$.

Solving Sub-problems

In this paper two special procedures are implemented for finding "good" feasible and optimal solutions for each subproblem. The objective of each SP(l) is to find a cycle including vertex l with maximum weight. This problem can be solved by finding maximum weight paths from vertex l to all other vertices and adding the weight of the arc from each vertex to l (if it exists). By following this procedure the cycle with maximum weight can be discovered. It is known that Belman-Ford's algorithm is effective to find a shortest path in a graph with weighted arcs. A general scheme of the algorithm was adopted for finding a cycle of length up to k.

A heuristic to find a feasible cycle including node l for subproblem SP(l) is presented in Figure 3. Let h^l_{ij}, $(i,j) \in A^l$ be the current coefficients of the objective function of SP(l): $h^l_{ij} = 1 - \lambda_j$. The heuristic performs a random walk on subgraph G^l, starting from vertex l (Step 0). The trace of the performed walk is kept in δ which is an ordered set of nodes (Step 5). The method chooses a new current vertex u in each iteration from the set of active nodes U such that the corresponding arc has the maximum weight (Step 2). Afterwards it checks if there is a back arc to vertex l (Step 4). The method stops whenever the weighted cycle is found. The performed walk δ will define a solution of SP(l) with weight equal to the sum of coefficients h_{ij} for corresponding arcs.

```
Step 0. Initialize U = V^l\{l}, u = l, δ = (l).
Step 1. If U = ∅ STOP.
Step 2. If {i ∈ U : (u,i) ∈ A^l} = ∅, then U = U\{u},
        u =vertex in walk δ previous to u, GoTo Step 1.
Step 3. Choose node i ∈ U, such that h_ui = max_{k:(u,k)∈A^l} h^l_uk,
        U = U\{i}
Step 4. If (i,l) ∈ A^l then δ is a solution of SP(l), STOP
Step 5. If   (length of δ equal to k − 1), then
             GoTo Step 2,
        Else add i to walk: δ = (δ,i), u = i, GoTo Step 1
```

Fig. 3. Heuristic for solving SP(l)

5.2 Branch and Price Based Approaches

Branch-and-price (B&P) is the combination of column generation and branch-and-bound methods used to obtain optimal solutions to decomposition models [4]. In this work we explored two ways of potentially increasing the efficiency of a pure branch-and-price scheme: (i) to solve the integer RMP of the root node by a general purpose solver with different time limits, and (ii) to apply local search based heuristics in the root or in all nodes of the search tree.

The second approach is related with the framework *SearchCol: Metaheuristic search by column generation* [2]. Within that framework global solutions to a problem are considered to be formed by selecting one solution from each subproblem (in some cases the null solution). SearchCol defines many problem independent metaheurisitic components such as constructive procedures, neighbourhoods, and evaluation functions. When combined with B&P, (meta)heuristics may be applied in chosen nodes of the tree after column generation solves the relaxation of the node. In the current work we used Local Search (LS) and Variable Neighbourhood Search (VNS) [16] based on the definition of n-neighbour — a solution which differs from the current one by n cycles. For LS $n = 1$ was used. The shaking phase of VNS was accomplished by using higher values of n. The LS and VNS were attempted to be used in the root node and in all nodes of the search tree. Initial solutions were obtained by selecting, for each subproblem, the cycle with higher value in the optimal RMP of the CG of the node.

6 Computational Tests

Computational experiments were carried out to evaluate the proposed decomposition for the KEP and to investigate the difference from a computational point of view of two objective functions of the problem. CPU times were obtained on a computer with Intel Xeon processor at 3.00 GHz, 8 Gb of RAM and running Linux OS. MIP solver CPLEX 12.4 was used.

We used SearchCol++ (http://searchcol.dps.uminho.pt/) to implement all the approaches developed. SearchCol++ is a C++ set of classes which implements column generation based algorithms (B&P and SearchCol) in a general way. Only problem specific components such as the exchange of information between the RMP and subproblems solvers must be implemented by the user. Thus branching, column management, the application of metaheuristics, and perturbations are hidden and controlled through parameters. For the sake of generality branching is performed in the original subproblem variables.

To generate the instances for the computational study current state-of-the-art generator [26] was used. It creates random graphs based on probability of blood type and of donor–patient compatibility. Instances with different number of nodes (starting from 100) were generated, 10 instances of each size. The problem was studied for different values of parameter k ranging from 3 to 6[1].

[1] Currently implemented kidney exchange programs work in general with a maximum value of $k = 3$. However this value has already been exceeded by large: the maximum number of simultaneous transplantations performed up to now being 6 [5].

Average times to find an optimal solution (excluding instances where a given CPU time limit was reached) with different methods for the problem with the standard and the proposed objective function (see (22) and section 4) are shown in tables 1 to 4; maximum CPU time was set to 1800 seconds for all methods. The following notation is used in the tables:

- $|V|$ is the number of nodes in the graph;
- t_{prep} is the average CPU time (in seconds) for preprocessing the instances;
- $B\&P$ is the average CPU time to find an optimal solution of the KEP with standard objective function (the DCD) by the pure Branch-and-Price method;
- $rMIP$ is the average CPU time for solving DCD by CG combined with MIP solver in root node;
- $rVNS$ is the average CPU time for the CG combined with VNS in root node;
- $rMIP_H$ is the average CPU time for the CG combined with MIP solver in root node, using an heuristic for solving Sub-Problems;
- $rMIP_{LS}$ is the average CPU time for the CG combined with MIP solver, with LS running in all nodes;
- $B\&P_{nc}$ is the average CPU time to find an optimal solution for the problem with objective function (22) (the DCD_{nc}) by the pure Branch-and-price method;
- $rMIP_{nc}$ is the average CPU time to find an optimal solution of the DCD_{nc} by the $rMIP$ method.

In some cases there were instances that could not be solved within the CPU time limit. For such cases we show in parenthesis for each method the number of instances out of 10 that were solved.

Note that, as mentioned in section 4, the optimal value of the problem with standard objective function may be derived from the optimal value of the problem with the new function (22). Thus methods $B\&P_{nc}$ and $rMIP_{nc}$ for the latter may be considered as solution methods for the problem with standard objective and compared with other approaches for this problem.

Computational results for $k = 3$ to $k = 6$ are presented in Tables 1 to 4.

Results show that, for $k = 3$, preprocessing times are almost equal or even exceed the time to solve a problem with e.g. $rMIP_{nc}$. Still, they do not exceed 2 minutes and for the harder instances, with bigger values of k, they are smaller than the time required to solve the problem. It can also be seen that, for $k = 3$, $rMIP$ outperforms $B\&P$ and $rVNS$. Moreover, no improvements are found when heuristics are used for solving SPs and local search is run in each node of the tree (see $rMIP$ versus $rMIP_H$ and $rMIP_{LS}$). In fact CPU times increase. Therefore these variants of $rMIP$ as well as $rVNS$ are not used in experiments with bigger values of k.

When the new objective is considered, $rMIP_{nc}$ cannot solve some instances within available CPU time. This method will also be discarded for computational tests with $k > 3$.

Table 1. CPU time (in seconds) for $k = 3$

$\|V\|$	t_{prep}	$B\&P$	$rMIP$	$rVNS$	$rMIP_H$	$rMIP_{LS}$	$B\&P_{nc}$	$rMIP_{nc}$
100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
200	0.1	0.0	0.1	0.1	0.1	0.1	0.0	0.0
300	0.3	0.3	0.2	0.4	0.3	0.2	0.2	0.2
400	0.7	0.6	0.6	0.9	0.5	0.4	0.5	1.1 (9)
500	1.4	1.4	1.6	2.0	1.9	1.0	1.0	0.8
600	2.5	3.7	1.5	4.9	3.3	2.5	2.3	1.2 (7)
700	3.6	5.5	4.0	7.5	3.9	2.5	3.9	1.8 (7)
800	5.5	10.6	4.0	13.4	6.6	5.2	4.5	2.6 (8)
900	8.0	10.3	5.8	14.8	10.3	6.3	8.3	4.1 (8)
1000	10.4	18.5	9.3	24.3	21.1	13.1	11.7	5.2
1500	35.6	105.4	20.9	127.4	71.1	47.9	57.1	16.3 (7)
2000	87.6	513.2	89.8	574.4	299.8	91.6	206.0	51.4 (7)

Results for $k = 4$ are presented in Table 2. With this value of k the problem becomes harder to solve. Therefore the time spent by MIP solver in the $rMIP$ method was limited to a number of seconds obtained by multiplying a parameter by the number of vertices of the network. Extra computational tests were carried out for values of 0.1 and 0.01 of that parameter. They are presented in columns $rMIP_{0.1}$ and $rMIP_{0.01}$, respectively.

Table 2. CPU time (in seconds) for $k = 4$

$\|V\|$	$B\&P$	$rMIP$	$rMIP_{0.1}$	$rMIP_{0.01}$	$B\&P_{nc}$
100	0.0	0.1	0.1	0.1	0.0
200	0.7	2.5	2.5	1.5	0.2
300	5.0	34.8	19.0	7.8	1.1
400	15.7	214.4	79.7	20.7	3.8
500	46.3	369.3 (8)	131.5	47.9	6.8
600	118.0	296.4 (3)	201.3	109.1	15.5
700	176.6	-	333.6	198.2	30.1
800	375.5	-	729.7	392.4	50.5
900	746.4	-	903.6 (9)	748.3	144.4
1000	1315.5 (8)	-	1388.7 (6)	1375.2 (7)	82.4

It is obvious that $rMIP$ is not efficient. $rMIP_{0.01}$ with the tightest bound on the time spent by the MIP solver performs almost as $B\&P$ and these two are the most efficient among the methods for the problem with the standard objective function. However it is interesting that all those methods lose to the Branch-and-Price run for the problem with the newly proposed objective function (column $B\&P_{nc}$). It handles all instances with up to 1000 nodes in significantly shorter time.

Table 3. CPU time (in seconds) for $k = 5$

$\|V\|$	$B\&P$	$rMIP_{0.1}$	$rMIP_{0.01}$	$B\&P_{nc}$
100	0.1	0.3	0.3	0.0
200	2.0	23.0	5.8	0.4
300	17.9	71.0	27.3	3.3
400	66.8	146.5	80.1	10.1
500	255.0	355.8	211.7	14.6
600	618.1	741.5	579.8	36.8
700	1040.0	1107.5 (9)	906.3	87.5
800	1694.6 (2)	-	1453.1 (5)	117.5
900	-	-	-	125.4 (6)

Table 4. CPU time (in seconds) for $k = 6$

$\|V\|$	$B\&P$	$rMIP_{0.01}$	$B\&P_{nc}$
100	0.3	0.6	0.1
200	9.4	9.8	0.6
300	96.6	63.0	3.9
400	187.3	252.6	12.5
500	994.1	619.7	21.5
600	1085.5	1276.4	63.8
700	1688.6 (2)	1706.8 (1)	114.1
800	-	-	155.6
900	-	-	214.6 (4)

Similar conclusions can be made when $k = 5$ and $k = 6$ (see Tables 3 and 4). $rMIP_{0.01}$ with stronger limit on time spent by MIP solver and $B\&P$ work similar, with slight dominance of $rMIP_{0.01}$ for bigger graphs and $k = 5$ (5 instances with 800 nodes solved by $rMIP_{0.01}$ versus 2 instances solved by $B\&P$). Furthermore, the $B\&P_{nc}$ is significantly more efficient than the others, solving 6 instances with 900 nodes, none of which having been solved by any other method.

For $k = 6$ the Branch-and-Price method applied to the problem with the proposed objective function is again the most efficient method. It was able to solve 4 instances with 900 nodes, while the other methods could only handle at most 2 problems with 700 nodes.

It is worth reminding that although problems DCD and DCD_{nc} are different – they differ in the objective function – the latter also solves the former and therefore it is reasonable to compare CPU times between the formulations. This allows us to conclude that using the proposed decomposition with the new objective function DCD_{nc}, problems of significantly bigger size can be solved especially for larger k. When compared to computational experiments carried out in [7] on a computer with a Quad-Core Intel Xeon processor at 2.66 GHz, 16 Gb of RAM and running Mac OS X 10.6.6 with the same version of MIP solver

the maximum size of solved instances was 1000 nodes for $k = 3$, and only 200 nodes for $k = 4$ and 100 nodes for $k = 5, 6$.

7 Conclusions

This paper proposes a new efficient approach for seeking an optimal solution for large-scale instances of the kidney exchange problem. The approach is based on a new decomposition model entitled disaggregated cycle decomposition, two component objective function and branch-and-price. An optimal solution for the proposed objective function has maximum number of transplants, trying also to reduce the size of the cycles involved in the exchanges.

Computational results show that the addition of the second objective component related with minimizing the sizes of cycles to the standard objective function of maximizing the number of transplants allows solving smaller instances faster and also handling larger instances. Within the proposed approach instances with 2000 pairs and a limit of 3 pairs involved in an exchange cycle were solved in 200 seconds in average. Instances with 1000 pairs and a limit of 4 pairs in a cycle were solved in 80 seconds, and instances with 800 pairs and a limit of 5 and 6 pairs in 120 and 155 seconds respectively. These results clearly outperform those obtained when using the standard objective function of maximizing the number of transplants, as well as the ones obtained by using a general purpose solver for the compact model in [7]. Thus, by solving the problem with the new objective function we can indirectly find a solution of the original problem, with standard objective function, with much less computational effort.

Acknowledgements. We would like to thank Dr. Nicolau Santos from the INESC TEC, Porto, Portugal for useful comments. This work is financed by the ERDF — European Regional Development Fund through the COMPETE Programme (operational programm for competitiveness), by National Funds through the FCT — Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project "KEP - New models for enhancing the kidney transplantation process. /FCT ref: PTDC/EGE-GES/110940/2009", by the North Portugal Regional Operational Programme (ON.2 O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds through FCT within project "NORTE-07-0124-FEDER-000057", and has also been supported by FCT through projects "SearchCol: Metaheuristic search by column generation" (PTDC/EIAEIA/100645/2008) and PEst-OE/EEI/UI0319/2014.

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, pp. 295–304 (2007)

2. Alvelos, F., de Sousa, A., Santos, D.: Combining Column Generation and Metaheuristics. In: Talbi, E.-G. (ed.) Hybrid Metaheuristics. SCI, vol. 434, pp. 285–334. Springer, Heidelberg (2013)
3. Ashlagi, I., Gilchrist, D., Roth, A., Rees, M.: Nonsimultaneous chains and dominos in kidney paired donation - revisited. American Journal of Transplantation 11(5), 984–994 (2011)
4. Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P.: Branch-and-price: column generation for solving huge integer programs. Operations Research 46, 316–329 (1998)
5. BBC: BBC news website. six-way kidney transplant first (9/04/2008) (2008), `http://news.bbc.co.uk/1/health/7338437.stm` (last accessed in December 2012)
6. Biro, P., Manlove, D., Rizzi, R.: Maximum weight cycle packing in directed graphs, wiht application to kidney exchange programs. Discrete Mathematics, Algorithms and Applications 1(4), 499–517 (2009)
7. Constantino, M., Klimentova, X., Viana, A., Rais, A.: New insights on integer-programming models for the kidney exchange problem. European Journal of Operational Research 231(1), 57–68 (2013)
8. Dantzig, G., Wolfe, P.: Decomposition principle for linear programs. Operations Research 8, 101–111 (1960)
9. Dickerson, J., Procaccia, A., Sandholm, T.: Optimizing kidney exchange with transplant chains: Theory and reality. In: AAMAS 2012: Proc. 11th Intl. Joint Conference on Autonomous Agents and Multiagent Systems (June 2011)
10. Dickerson, J., Procaccia, A., Sandholm, T.: Failure-aware kidney exchange. In: EC 2013: Proc. 14th ACM Conference on Electronic Commerce (June 2013)
11. Floyd, R.: Algorithm 97: Shortest path. Communications of the ACM 5(6), 345 (1962)
12. Gentry, S., Montgomery, R., Segev, D.: Kidney paired donation: Fundamentals, limitations, and expansions. American Journal of Kidney Disease 57(1), 144–151 (2010)
13. Gentry, S., Montgomery, R., Swihart, B., Segev, D.: The roles of dominos and nonsimultaneous chains in kidney paired donation. American Journal of Transplantation 9, 1330–1336 (2009)
14. Geoffrion, A.: Lagrangean relaxation for integer programming. Mathematical Programming Study 2, 82–114 (1974)
15. Glorie, K., Wagelmans, A., van de Klundert, J.: Iterative branch-and-price for large multi-criteria kidney exchange. Econometric Institute report (2012-11) (2012)
16. Hansen, P., Mladenovic, N., Perez, J.: Variable neighbourhood search: methods and applications. Annals of Operations Research 175, 367–407 (2010)
17. Huang, C.: Circular stable matching and 3-way kidney transplant. Algorithmica 58, 137–150 (2010)
18. de Klerk, M., Keizer, K., Claas, F., Haase-Kromwijk, B., Weimar, W.: The Dutch national living donor kidney exchange program. American Journal of Transplantation 5, 2302–2305 (2005)
19. Manlove, D.F., O'Malley, G.: Paired and altruistic kidney donation in the UK: Algorithms and experimentation. In: Klasing, R. (ed.) SEA 2012. LNCS, vol. 7276, pp. 271–282. Springer, Heidelberg (2012)
20. Montgomery, R., Gentry, S., Marks, W., Warren, D., Hiller, J., Houp, J., Zachary, A., Melancon, J., Maley, W., Simpkins, H.R.C., Segev, D.: Domino paired kidney donation: a strategy to make best use of live non-directed donation. The Lancet 368(9533), 419–421 (2006)

21. Nemhauser, G., Wolsey, L.: Integer and Combinatiorial Optimization. A Wiley-Interscience Publication (1999)
22. Pedroso, J.: Maximizing expectation on vertex-disjoint cycle packing. Technical Report DCC-2013-5 (2013)
23. Rees, M., Kopke, J., Pelletier, R., Segev, D., Rutter, M., Fabrega, A., Rogers, J., Pankewycz, O., Hiller, J., Roth, A., Sandholm, T., Ünver, M., Montgomery, R.: A nonsimultaneous, extended, altruistic-donor chain. The New England Journal of Medicine 360, 1096–1101 (2009)
24. Roth, A., Sönmez, T., Ünver, M.: Kidney exchange. Quarterly Journal of Economics 119(2), 457–488 (2004)
25. Roth, A., Sönmez, T., Ünver, M.: Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. The American Economic Review 97(3), 828–851 (2007)
26. 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 81, 773–782 (2006)
27. Segev, D., Gentry, S., Warren, D., Reeb, B., Montgomery, R.: Kidney paired donation and optimizing the use of live donor organs. The Journal of the American Medical Association 293(15), 1883–1890 (2005)