21st International Congress on Modelling and Simulation, Gold Coast, Australia, 29 Nov to 4 Dec 2015
www.mssanz.org.au/modsim2015

Heuristic Approaches for Multi-Criteria Optimisation in Kidney Exchange Programs

L. Nickholds [a] **and V. Mak-Hau** [b]

[a] *School of Information Technology, Deakin University*
[b] *School of Information Technology, Deakin University*
Email: lukenickholds@gmail.com

Abstract: The Kidney Exchange Problem (KEP) is an optimisation problem that was first discussed in Rapaport (1986) but has only more recently been the subject of much work by combinatorial optimisation researchers. This has been in parallel with its increased prevalence in the medical community.

In the basic formulation of a KEP, each instance of the problem features a directed graph $D = (V, A)$. Each node $i \in V$ represents an incompatible pair wherein the patient needs to trade kidneys with the patient of another incompatible pair. The goal is to find an optimal set of cycles such that as many patients as possible receive a transplant. The problem is further complicated by the imposition of a cycle-size constraint, usually considered to be 3 or 4. Kidney exchange programs around the world implement different algorithms to solve the allocation problem by matching up kidneys from potential donors to patients. In some systems all transplants are considered equally desirable, whereas in others, ranking criteria such as the age of the patient or distance they will need to travel are applied, hence the multi-criteria nature of the KEP.

To address the multi-criteria aspect of the KEP, in this paper we propose a two-stage approach for the kidney exchange optimisation problem. In the first stage the goal is to find the optimal number of exchanges, and in the second stage the goal is to maximise the weighted sum of the kidney matches, subject to the added constraint that the number of exchanges must remain optimal. The idea can potentially be extended to multiple-objectives, by repeating the process in multiple runs.

In our preliminary numerical experiments, we first find the maximum number of kidney matches by using an existing open source exact algorithm of Anderson et al. (2015). The solution will then be used as an initial solution for the stage two optimisation problem, wherein two heuristic methods, steepest ascent and random ascent, are implemented in obtaining good quality solutions to the objective of maximizing total weight of exchanges. The neighbourhood is obtained by two-swaps. It is our intention in the future to implement a varying neighbourhood scheme within the same two heuristic framework, or within other meta-heuristic framework.

Keywords: *Kidney exchange, multicriteria optimisation, population generation*

1 INTRODUCTION

Unlike the other major organs in a human body, most people are typically born with more kidneys than is required for sustaining a healthy life. Chronic kidney disease (CKD) is increasingly common, particularly in developed countries. In the U.S.A. alone, from 1991 to 2004, the number of patients being treated for kidney failure with dialysis or a transplant doubled to approximately 472,000 people (Coresh et al. (2007)). For patients suffering from end-stage renal disease, one feasible option is for them to find a living donor. A generous spouse, family member or friend may decide to donate their second kidney to the patient. Kidney transplant is however a complicated process and even when a patient is able to find a willing donor, they may be unable to accept the donated kidney. Factors such as blood types or antibodies more broadly can rule out the prospect of accepting a particular kidney.

In these situations one option for the incompatible pair is to find another incompatible pair. Suppose Pair A and Pair B each consists of a donor-and-patient pair who are incompatible. Now, if the kidney of Donor A is compatible with Patient B, and that of Donor B is compatible with Patient A, then the two pairs can come to a mutual agreement to trade kidneys which may well lead to two successful transplants. Such an exchange is referred to as a 2-way exchange. A multi-way exchange can occur, for example if Donor A donates to Patient B, Donor B to Patient C, and so on, and finally the last donor donates to Patient A. As contracts regarding organ transplants are unenforceable or illegal in most countries, a donor can technically withdrawal from the program after his/her patient has received a kidney. For this reason, most multi-way kidney exchanges are carried out simultaneously. Due to logistic and human resource reasons, most kidney exchanges involve only 2- or 3-way exchanges, although recently a 9-way exchange was carried out successfully (Wollan (2015)). In recent years, many countries have set up programs to facilitate these exchanges, including the Netherlands, the United States, South Korea, and Australia (Ferrari et al. (2009)).

1.1 Basic formulation of the kidney exchange problem

In its basic form, a Kidney Exchange Problem (KEP) can be defined on a directed graph, with the set of vertices representing patient-donor pairs. An arc from one node to another indicates that the donor in the first pair is compatible with the patient in the second pair, which indicates that a transplant may be possible.

As mentioned earlier, due to logistical issues a cycle-size constraint is applied. Usually the cycles are limited to the size of 2 or 3, Manlove and O'Malley (2012), however depending on the circumstances, higher size limit may be desirable as it improves the possible outcomes.

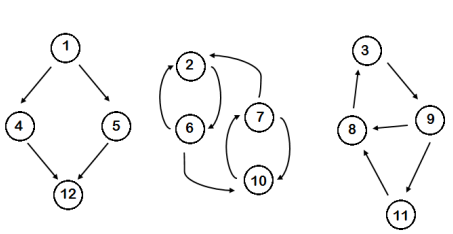
1.2 Possible variations of the basic kidney exchange problem

As research in the field has progressed, variations on the basic KEP have emerged in line with the medical profession. One variation discussed has been the potential to include compatible pairs in the pool (which will be represented by loops on the directed graph). In practice this variant has not reached widespread adoption yet, as there are ongoing discussions about the ethical issues involved in the practice, as discussed in papers such as Fortin (2013). For the test populations used in this paper we have assumed that the pool does not contain any compatible pairs, but the tools used to develop the populations (which are available online at https://github.com/lukenick/KidneyExchangePublic) can be used to generate pools that do contain compatible pairs.

Another variant that has been studied is the inclusion of altruistic donors to the pool. An altruistic donor is a donor who wants to donate a kidney to a patient, without expecting anything in return. The altruistic donor will start a chain (also known as a domino) by donating to a patient in a pair, whose partner then donates to a patient in another pair, whose partner then donates to another pair and so on. See Figure 1 for an example of a KEP instance.

Exact implementations of altruistic donors can differ, in some programs the goal is to make the donor kidney go as far as possible. In these programs an NEAD chain is formed, known either as a Never-Ending-Altruistic-Donor or Nonsimultaneous-Extended-Altruistic-Donor chain. In these exchanges the chain continues until something breaks it, such as a failed transplant or a donor backing out.

The other main implementation sees a series of donations occurring with incompatible pairs, before a final donation to a patient on the terminal wait-list. The terminal waitlist comprises of patients who need a kidney transplant but do not have a partner willing to donate in exchange. Where our test-populations include altruistic donors, we use this interpretation.



$$\begin{vmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Figure 1. An example of a Kidney Exchange Pool with one altruist node (1), ten paired nodes (2-11) and one terminal node(12)

The user is able to specify the number of altruistic nodes and nodes that represent incompatible pairs to generate a problem instance, and the program will also generate a number of terminal 'dummy' nodes, one per altruistic node. These dummy nodes simulate the large wait-list of unpaired patients, so these nodes are able to receive kidneys from any patient but cannot donate. For a general exposition of existing exact method for KEPs of various forms, and in particular, of integer programming models and solution methodologies, see Mak-Hau (2015).

1.3 The multi-objective nature of the kidney exchange problem

There is a lack of uniformity in the various Kidney Exchange Programs around the world, with each adopting different algorithms or methods of organising their system that reflect local practice. Some programs consider that all transplants are equally desirable and that the program should seek only to optimise the total number of kidney exchanges that can occur. Others use detailed ranking criteria to determine the desirability of different solutions to a given KEP instance, see for example Malik and Cole (2014) for a discussion of the Canadian System amongst others Glorie et al. (2014), Manlove and O'Malley (2012). These criteria utilise both arc and node weights, incentivising exchanges between younger patients or patients that live close to one another. As these programs optimise the weighted sum of the solution, this can lead to a solution where less exchanges occur but the solution is deemed as more desirable.

This paper proposes and experiments a lexicographical approach to tackle the bi-criteria optimisation of maximizing both the number of matches and the total arc weight. In our work we have considered only arc weights but the heuristics would be easily adapted to a system with node weights. We define this as an approach in which the KEP is split into two phases. In the first phase we set all valid arcs to a weight of 1 and find the optimal number of transplants possible. In the second phase, we solve again with non-binary weights and introduce the additional constraint that the total number of exchanges must be the same as in our solution for the first phase.

2 POPULATION GENERATION

The populations used in this research have been developed using a model as described in Saidman et al. (2006). This model considers ABO+ blood type distribution in generating patients and donors, as well as the additional consideration of HLA cross-matching as summarised in Table 1. For full details of the model refer to Saidman et al. (2006) work.

Using this model we have generated four groups of test populations. Each group uses the reported Australian distribution of ABO+ blood types and comprises of 30 populations (i.e. 30 data instances). The name of a group is in the form of "AUS_X_Y", for X the number of altruist nodes and Y the number of incompatible pair nodes. The four groups of populations are: "AUS_10_190", "AUS_0_200", "AUS_5_95" , and "AUS_0_100".

3 HEURISTIC METHODS

In this section, we explain our solution methodology. The implementation is only bi-criteria at this stage, but the idea can be extended for multi-criteria objectives. We use a lexicographical approach, in the first instance to apply the most advanced exist algorithm Anderson et al. (2015) to find the optimal objective value that maximizes the number of kidney matches, and then conduct an objective propagation by adding the optimal

Table 1. PRA Population Model

Low PRA	PRA <10%	5% chance of a positive crossmatch
Medium PRA	PRA 10%-80%	45% chance of a positive crossmatch
High PRA	PRA >80%	90% chance of a positive crossmatch

objective value as a constraint, and solve a second optimisation problem with the objective of maximizing total arc weights by using the steepest ascent and the random ascent methods with problem specific designs.

For the first priority objective function, the optimisation problem is solved by the code provided by Anderson et al. (2015) which is available online at https://github.com/rma350/kidneyExchange. Solution to a KEP can be expressed in multiple formats, for our work we have represented them in the form [["Chain", 1, 4, 12], ["Cycle", 2, 6], ["Cycle", 7, 10], ["Cycle", 3, 9, 8]].

The two meta-heuristic algorithms we have attempted are random ascent, and steepest ascent.

3.1 Random Ascent

The Random-Ascent Heuristic uses a simple method to randomly generate new solutions based upon the given initial solution, and then repeats this for multiple iterations. In each iteration it compares the new solution generate to the old one, and uses the solution with greater weight as the basis for the next iteration. The method used to generate new solutions is based upon the 2-Opt heuristic initially proposed by Croes (1958), applied in areas such as the Travelling Salesman Problem and Vehicle Routing.

Algorithm 1 The Random Ascent Heuristic

```
1:  procedure HILL-CLIMBING(population, initial_solution)
2:      Set best_solution to initial_solution
3:      Set count = 0
4:      while count < 1,000 do
5:          while new_solution is not a feasible solution do
6:              Select a random node first_node from population
7:              Select a random node second_node from population
8:              Generate a new solution by switching first_node and second_node in best_solution
9:              Check if new_solution is a feasible solution.
10:         if Weight of new_solution is greater than Weight of best_solution then
11:             Set best_solution = new_solution
12:         Increment count
13:     return best_solution
```

The use of 10,000 iterations is a parameter that can be tweaked based upon the size of the population. Ideally the value should be high enough that one can be relatively confident that the algorithm will have hit a local maximum by that stage. For populations of 200 nodes 10,000 has been a safe choice, however for populations of 100 a much lower value of 1,000 would be more appropriate. The suitable function is yet to be found, and in our future implementation, the number of iterations will be calculated as a function of size of population.

To generate a new solution, we simply switch the place of first_node and second_node in the original solution. For instance, if our original solution is [["Chain", 1, 4, 12], ["Cycle", 2, 6], ["Cycle", 7, 10], ["Cycle", 3, 9, 8]] and first_node is 2 and second_node is 8, then our new solution would be [["Chain", 1, 4, 12], ["Cycle", 8, 6], ["Cycle", 7, 10], ["Cycle", 3, 9, 2]].

To check whether a given solution is feasible, we need to check whether all the arcs that it contains are valid. We could iterate through all of the arcs, however if we have generated it from a previous solution that we know is feasible we can simplify the process by only checking the new arcs, reducing it from linear to constant time. In the example given above the newly generated arcs are $8 \to 6, 6 \to 8, 9 \to 2$ and $2 \to 3$.

3.2 Steepest Ascent

This heuristic is similar to the previous one, with the major difference being that it examines every possible swap at each stage rather than just a single random swap. In each iteration, the Steepest-Ascent Heuristic considers swapping every possible pair of nodes within the population. For those swaps that would result in a feasible solution, it select the solution with the greatest weight and uses that as the basis of the next iteration. It continues this until it reaches a local maximum where there are no valid swaps that would increase the weight of the solution.

Algorithm 2 The Steepest Ascent Heuristic

```
1:  procedure STEEPEST ASCENT(population, initial_solution)
2:      Set old_solution to initial_solution
3:      while flag == True do
4:          Set best_solution to old_solution
5:          for all first_node ∈ V(D) do
6:              for all second_node ∈ V(D) do
7:                  Generate a new solution by switching first_node and second_node in best_solution
8:                  if new_solution is a feasible solution then
9:                      if Weight of new_solution is greater than Weight of best_solution then
10:                         Set best_solution = new_solution
11:         if Weight of old_solution > Weight of best_solution then
12:             flag = False
13:         else
14:             old_solution = best_solution
15:     return old_solution
```

4 EXPERIMENTAL RESULTS

4.1 Methods

The experiments detailed below were conducted using servers hosted by the website Digital Ocean. They were run using instances of their basic droplet, with 2.4Ghz CPU and 512mb RAM.

4.2 Results

To test the performance of each heuristic we used the test libraries "AUS_10_190", "AUS_0_200", "AUS_5_95" and "AUS_0_100", each of which contain 30 instances.

For "AUS_10_190" and "AUS_0_200" we ran the Random-Ascent Heuristic five times (as it is inherently probabilistic) upon each instance to better capture its expected performance. Each time we recorded the time taken and the weight of the final solution found. We calculated the average run-time and average final weight for each instance, as well as the average % improvement over the initial solution. We also calculate the total run-time across the five runs for each instance, and the best weight found during the five runs and the % improvement this represented.

For the Steepest-Ascent Heuristic we ran it upon the 30 populations in each test library, and recorded the run-time, final weight and % improvement for each instance.

Full details of the experiments are available in the online supplement (https://github.com/lukenick/KidneyExchangePublic), Table 2 summarises the results for each heuristic with the average across the 30 instances.

Table 3 summarises the results of similar experiments for the populations "AUS_5_95" and "AUS_0_100". This has however been extended to demonstrate the importance of selecting an appropriate number of iterations of Random Ascent to be performed. For these smaller populations, running Random-Ascent for 1000 iterations as we had previously done with the larger populations did not yield significantly better results than running it for 250 iterations, in the table above from 0.05% to 0.5% improvement for quadruple the runtime. Instead, the user would be better to run 4 times as many Random Ascents with 250 iterations and take the best result found amongst those.

Table 2. Summary of Heuristic Results for Larger Populations

Heuristic	AUS_0_200			AUS_10_190		
	Time	Final Weight	% Improvement	Time	Final Weight	% Improvement
Steepest Ascent	37.0416	12418.567	27.972%	46.749	12148	31.893%
Avg Random Ascent	12.243	12287.193	26.622%	14.790	11975.573	30.004%
Best Random Ascent	61.217	12530.7	29.134%	73.955	12220.133	32.669%

Table 3. Summary of Results for Smaller Populations

		AUS_0_100			AUS_5_95		
		Time	Final Weight	% Improvement	Time	Final Weight	% Improvement
	Steepest Ascent	1.649	5422.069	16.675%	1.924	5318.267	17.867%
count	Avg Random Ascent	4.465	5366.166	15.496%	4.649	5270.71	16.762%
=250	Best Random Ascent	22.324	5489.31	18.170%	23.248	5386.517	19.344%
count	Avg Random Ascent	17.627	5389.945	16.005%	17.967	5282.048	16.990%
=1000	Best Random Ascent	88.137	5508.552	18.527%	89.836	5391.241	19.396%

For all the populations examined, both heuristics took slightly longer to run when the population contained altruists but also found a comparatively higher result.

5 FURTHER DIRECTIONS

The results presented above demonstrate that both the Random-Ascent Heuristic and the Steepest-Ascent Heuristic can be used to obtain a good quality solution for the second priority objective function of maximizing total arc weights, with the first priority objective of maximizing number of kidney matches propagated as a constraint. The computation time required is substantially less than it would take for obtaining an exact optimal solution using integer programming models presented in, e.g., Mak-Hau (2015).

Further work is needed to investigate the behaviour of these two heuristics and their suitability for different input populations. Further research is also suggested to investigate the appropriateness of other heuristics, with initial attempts to apply simulated annealing being so far unable to obtain substantially improved results.

Another direction of computational experiment is to take consideration into a third or even a fourth priority objective, with propagation of higher priority objectives, and investigate the computational effort required to obtain a good quality solution with those objectives.

It is our intention in the future to implement a varying neighbourhood scheme, instead of just two-swaps, and incorporate the VNS within the steepest ascent and random ascent framework and other well tested meta-heuristic framework.

REFERENCES

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

Coresh, J., E. Selvin, and L. Stevens (2007). Prevalence of chronic kidney disease in the united states. *JAMA 298*(17), 2038–2047.

Croes, G. A. (1958, November). A method for solving traveling-salesman problems. *Operations Research 6*(6), 791–.

Ferrari, P., C. Woodroffe, and F. Christiansen (2009). Paired kidney donations to expand the living donor pool: the western australia experience. *Medical Journal of Australia 190*(12), 700–703.

Fortin, M.-C. (2013, 2013). Is it ethical to invite compatible pairs to participate in exchange programmes. *Journal of medical ethics 39*(12), 743–747.

Glorie, K. M., J. J. van de Klundert, and A. P. M. Wagelmans (2014). Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manufacturing & Service Operations Management 16*(4), 498–512.

Mak-Hau, V. (2015). On the kidney exchange problem: cardinality constrained cycle and chain problems on directed graphs: a survey of integer programming approaches. *Journal of Combinatorial Optimisation*.

Malik, S. and E. Cole (2014, April). Foundations and principles of the canadian living donor paired exchange program. *Canadian Journal of Kidney Health and Disease 1*, 6–.

Manlove, D. and G. O'Malley (2012). *Experimental algorithms. Lecture notes in computer science*, Chapter Paired and altruistic kidney donation in the UK: algorithms and experimentation, pp. 271–282. Springer, Berlin.

Rapaport, F. (1986). The case for a living emotionally related international kidney donor exchange registry. *Transplant Proceedings 18*(3), 5–9. 1986.

Saidman, S., A. Roth, T. Sonmez, U. Unver, and F. Delmonic (2006). Increasing the opportunity of live kidney donation by matching for two- and three- way exchanges. *Transplantation 81*(5), 773–782. 2006.

Wollan, M. (2015, April 30). The great american kidney swap. New York Times.