Collège Sciences et Technologies
UF Mathématiques et Interactions

Problème d'échange de rein

Martin Debouté
Chloé Guimberteau
Lin Hirwa Shema
Lucas Villenave

CMI OPTIM

2022 – 2023

Heuristic Vehicle Routing Solvers

1 Introduction

Aujourd'hui la transplantation rénale est le traitement le plus efficace pour guérir des maladies rendant les reins défectueux. Les patients sont inscrits sur une liste d'attente pour obtenir un rein d'un donneur compatible d'un point de vue biologique, le plus souvent des donneurs décédés.

Nous constatons une forte augmentation de personnes atteintes de ces maladies créant ainsi une pénurie de greffons. Une solution alternative à l'attente d'un donneur décédé est de réaliser une transplantation grâce à un donneur vivant. Les patients ont souvent un proche prêt à leur donner un rein mais généralement incompatible biologiquement. C'est sur ce fait que se basent plusieurs programmes créés dans le but de faire face à cette pénurie de greffon. Parmi eux, celui des dons croisés. Il est basé sur l'existence d'un donneur incompatible lié à un patient, tout deux forme une paire. Ce programme permet à chaque patient y participant d'échanger son donneur vivant incompatible contre un autre donneur vivant compatible pour eux.

De nombreuses contraintes imposent une limite sur le nombre d'échange, $K \in \mathbb{Z}_+$, en fonction des législations des pays elle oscille entre 2 et 4. Nous étudierons le cas général avec $K = 3$ échanges autorisés. Notons que pour des cycles de longueur 2, le problème peut être résolu de façon polynomiale comme étant un problème de *matching* grâce à l'algorithme d'Edmonds, mais dès que $K > 2$, le problème se révèle NP-complet. La chaîne de dons est une solution alternative non soumise à ces contraintes. Le principe est d'initier une chaîne par un donneur seul, c'est à dire un donneur ne faisant pas partie d'une paire. Ce donneur peut aussi bien être un donneur décédé qu'un donneur vivant, dit altruiste, faisant don d'un de ses reins. Le donneur du patient ayant reçu un rein par le donneur seul peut donc donner son rein à un patient d'une autre paire, et ainsi de suite. La limite d'une chaîne de dons est donnée par $L \in \mathbb{Z}_+$ avec souvent $L \geqslant K$.

Le problème d'échange de rein, en anglais *Kidney Exchange Problem* (KEP), prend en compte ces deux contraintes représentées réciproquement par un cycle et une chaîne dans un graphe orienté ayant comme sommets les donneurs et les patients et les arcs symbolisant les compatibilités.

2 Formalisation du problème

Les données du KEP s'apparentent à un graphe orienté $G := (P \cup A, E)$. Nous avons les paires patient-donneur P définissant les sommets où le patient peut recevoir un rein, un arc entrant, et le donneur peut donner un rein à un autre patient, un arc sortant. Des sommets particuliers représentent les donneurs altruistes A qui peuvent seulement donner un rein, ils n'ont que des arcs sortant. Chaque arc de ce graphe $uv \in E$ représente la compatibilité d'un point de vue biologique d'une transplantation entre le donneur du sommet d'origine $u \in P \cup A$ et le receveur du sommet d'arrivée $v \in P$, cette arc peut éventuellement être valué d'un poids $w_{uv} \in \mathbb{R}$.

La figure 1 montre un exemple de données où les paires patient-donneur sont représentées par des cercles rouges et les donneurs altruistes par des losanges jaunes.

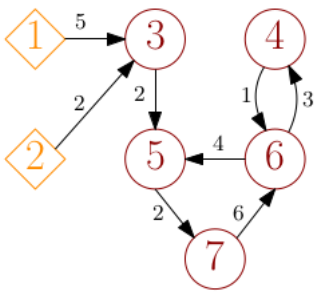

FIGURE 1 – Exemple d'instance

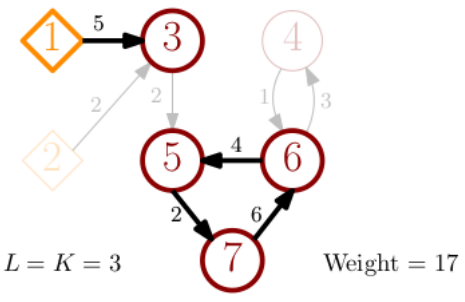

FIGURE 2 – Exemple de solution

L'objectif du modèle consiste à maximiser le poids total des transplantations à réaliser, cela revient à choisir les donneurs et patients qui sont le plus compatibles tout en respectant les contraintes. Un cycle peut contenir au plus K transplantations et une chaîne au plus L. Un donneur peut donner seulement un rein à au plus un patient, par conséquent un donneur altruiste ou une paire patient-donneur ne peut être sélectionné qu'une fois dans un cycle ou une chaîne. Un donneur altruiste peut initier une chaîne mais ne peut pas être dans un cycle. Un donneur dans une paire ne peut donner son rein que si son patient lié reçoit lui aussi un rein. La figure 2 montre un exemple de solution respectant toutes les contraintes. Ces figures sont extraites de l'article de Pansart, Cambazard et Catusse, 2022.

3 Modélisation

Pour un ensemble de paires patient-donneur P, un ensemble de donneurs altruistes A et des entiers K et L donnés, le KEP est représenté par un graphe orienté $G = (P \cup A, E)$ avec $\forall uv \in E$, $u \in P \cup A$ et $v \in P$. Nous avons modélisé et résolu le problème à l'aide de deux outils, OR-Tools et LocalSolver, le premier est *open-source* et le second payant.

3.1 OR-Tools

OR-Tools est un outil développé par Google afin de modéliser et résoudre des problèmes d'optimisations mathématiques. Son module de résolution de problèmes de type *Constraint Programming* (CP) `pywrapcp` donne accès à un solveur spécifique dédié à la modélisation et résolution de problèmes de routage de véhicules : `pywrapcp.RoutingModel`. C'est avec ce module que nous avons modélisé notre problème d'échange de reins comme un problème de tournées de véhicules avec capacité (*Capacited Vehicle Routing Problem* ou CVRP). Nous nous sommes basés sur l'exemple simple de modélisation CVRP donné sur le site d'OR-Tools [1] que nous avons su adapter à notre problème.

Dans cette modélisation, chaque chemin emprunté par un véhicule constitue un chemin ou cycle du problème sauf les chemins ne visitants qu'un seul noeud, ce qui correspond à ne sélectionner ce donneur dans aucun cycle ou chemin.

3.1.1 Transformation du graphe

Avant de pouvoir modéliser notre problème comme un CVRP, nous devons d'abord transformer notre graphe de compatibilité d'échange de reins en un graphe adapté au CVRP. Tout d'abord, nous créons un noeud dépôt 0 connecté à chaque noeud du graphe par un arc entrant et un arc sortant. On donne le coût 0 à chacun de ces arcs pour le moment.

Dans notre problème, nous cherchons à maximiser le coût total des échanges sélectionnés, ce qui revient à maximiser la compatibilité de ces échanges. Cependant le module de *Routing* d'OR-Tools semble imposer que l'objectif soit toujours celui de minimiser le coût total des routes sélectionnées. On doit alors transformer les coûts des arêtes pour que les échanges les plus compatibles soient les moins coûteux pour le solveur. Transformer c_{ij} en $-c_{ij}$ n'est pas une option car ces coûts sont utilisés comme les distances entre les noeuds et notre modélisation CVRP ne permet pas à celles-ci d'être négatives.

Ainsi, nous avons opté pour l'option de calculer le coût maximal $c_{max} = \sum_{(i,j)\in E} c_{ij}$ et nous transformons chaque coût en $\tilde{c}_{ij} = c_{max} - c_{ij} + 1$. Par contre, les coûts des arcs entrants au dépôt restent inchangés (0) et les coûts des arcs sortants du dépôt restent à 0 pour ceux dirigés vers les noeuds altruistes, et deviennent $\tilde{c}_{0j} = \max_{i\in V} \tilde{c}_{ij} + 1$ pour tout sommet $j \in P$ non-altruiste.

1. https://developers.google.com/optimization/routing/cvrp

On fait ceci pour qu'il soit plus coûteux de visiter un noeud seul (ce qui correspond à ne pas sélectionner le donneur) plutôt que de le visiter dans une chaîne ou dans un cycle.

Enfin, pour obtenir une matrice de coûts complète, nous posons $\tilde{c}_{ij} = U, \forall (i,j) \notin E$ où U est une valeur très grande par rapport aux autres coûts (ainsi ces arcs ne seront jamais sélectionnés).

3.1.2 Modélisation en CVRP

Dans notre modèle CVRP, chaque noeud donné a une demande de 1 sauf le dépôt qui a une demande de 0. En fixant la capacité de chaque véhicule à L, on assure avec ces demandes que chaque tournée de véhicules visite au plus L noeuds, ce qui permet donc de respecter la contrainte sur la longueur des chaînes. Nous créons $|P| + |A|$ véhicules, car dans le pire cas aucune transplantation n'est réalisée, ce qui correspond à visiter chaque noeud avec un véhicule différent. Ceci fait, il ne reste plus qu'à ajouter la contrainte sur les cycles, notamment le fait que chaque chemin ne commençant pas par un altruiste doit être un cycle et ne peut pas visiter plus de K sommets.

Nous n'avons pas trouvé de manière de modéliser les contraintes sur les cycles en modifiant juste les données du problème de CVRP. Nous avons alors dû les ajouter comme des contraintes supplémentaires au CVRP. Formellement, les contraintes que nous voulons modéliser sont les suivantes (on dénote par V^+ l'ensemble de noeuds excluant le dépôt) :

for $i, j \in V^+ \times V^+$
  if $veh(i) == veh(j)$ **and** $cumulVar(j,d) == 0$ **and** $next(i) == 0$
  then $isAltruist(j)$ **or (** $(i,j) \in E$ **and** $cumulVar(i,d) \leqslant K-1$ **)**

Ces contraintes signifient que pour tout couple de noeuds i et j différents du dépôt, s'ils appartiennent au même chemin sur lequel j est le premier visité et i est le dernier visité, alors soit il s'agit d'une chaîne (j est altruiste) ou soit il s'agit d'un cycle (il existe un arc allant de i à j) de longueur inférieur à K.

Malheureusement, le module de routage n'offre pas de manière plus subtile de renforcer certaines contraintes sous conditions que des variables prennent certaines valeurs. Par contre, le fait que ça reste un module de résolution par programmation par contraintes nous donne une technique pour contourner cette limite : la multiplication de contraintes. En créant une variable booléenne pour chacune des expressions utilisées dans la contrainte, nous pouvons faire de l'arithmétique de variables booléennes pour obtenir la contrainte suivante qui est valide pour le solveur :

for $i, j \in V^+ \times V^+$, `add constraint (`
  $(veh(i) == veh(j)) \times (cumulVar(j,d) == 0) \times (next(i) == 0)$
  $\leqslant$
  $isAltruist(j) +$ **(** $((i,j) \in E) \times (cumulVar(i,d) \leqslant K-1)$ **)**
)

Ici, $cumulVar(j,d)$ désigne la consommation cumulée de la capacité du véhicule (la dimension d) sur l'arc entrant au sommet j.

Avec ces modifications de la formulation de base du CVRP pour OR-Tools, nous obtenons une formulation du problème d'échange de reins valide pour le *routing solver* d'OR-Tools.

3.2 LocalSolver

LocalSolver est un autre outil de modélisation et résolution de problèmes d'optimisations mathématiques. En utilisant les ensembles décrit précédemment, nous avons un graphe $G = (V = A \bigcup P, E)$ avec A les donneurs altruistes, P les paires donneur-receveur et E l'ensemble des compatibilités entre deux couples donneur-receveur, ainsi que K et L la taille maximale d'un cycle et d'une chaîne.

On introduit les notations suivantes :

- $NL := \lfloor |V|/2 \rfloor + 1$, le nombre de liste correspondant aux chemins possibles. Nous avons besoin d'au plus une liste pour 2 sommets pour avoir un chemin valide, *i.e.* contenant au moins une transplantation à réaliser, et nous ajoutons une dernière liste non contrainte qui correspond aux sommets non sélectionnés.
- $AM_{(i,j)\in V^2}$, la matrice d'adjacence égale à 1 si $(i,j) \in E$, 0 sinon.
- $W_{(i,j)\in V^2}$, la matrice de poids égale à w_{ij} si l'arête (i,j) existe, 0 sinon.

On utilise les variables suivantes :

- $CS_{c\in[1..NL]}$, l'ensemble de listes d'éléments composants une chaîne/un cycle.
- $IA_{c\in[1..NL]}$, une variable binaire égale à 1 si la séquence est une chaîne, 0 si c'est un cycle.

Nous obtenons le modèle suivant :

$$min : \sum_{c=1}^{NL-1} \sum_{i=0}^{|CS_c|-1} W_{i,i+1} + IA_c * W_{|CS_c|,1} \tag{1}$$

$$partition(CS, V) \tag{2}$$

$$\forall_{c\in[1..NL-1]} \; \forall_{i\in[1..|CS]-1]} \; AM_{CS_{c,i},CS_{c,i+1}} = 1 \tag{3}$$

$$\forall_{c\in[1..NL-1]} \; AM_{CS_{c,|CS|},CS_{c,1}} \geqslant (1 - IA_c) \tag{4}$$

$$\forall_{c\in[1..NL-1]} \; |CS_c| \leqslant K + (L - K) * IA_c \tag{5}$$

$$\forall_{c\in[1..NL-1]} \; IA_c = (CS_{c,0} \in A) \tag{6}$$

(1) On somme pour tout cycle et chaîne le poids de toutes les arêtes utilisées.
(2) L'ensemble des listes et chaînes doivent partitionner les sommets.
(3) Les arêtes entre chaque élément du cycle/de la chaîne doivent exister.
(4) L'arête de retour d'un cycle doit exister.
(5) Les cycles et chaînes ne doivent pas dépasser K et L respectivement.
(6) La variable IA est fixée en fonction du premier élément du cycle/de la chaîne.

4 Expérimentation et résultats

L'ensemble du code réalisé et utilisé pour effectuer ces tests comparatifs est disponible en libre accès sur ce dépôt Github : https://github.com/mdeboute/kidney_exchange_optim. L'implémentation a été faite en Python et les tests ont étés éffectués sur un MacBook Pro possédant un Intel Core i5 de quatre cœurs cadencé à 2,4 GHz.

Nous avons utilisé un sous-ensemble des 310 instances de *Kidney Exchange* qui sont disponibles en libre accès sur *PrefLib*[2] grâce à M. John Dickerson. Les données ont été produites à l'aide d'une méthode de génération de *pool* de donneurs décrite dans Saidman et al., 2006. Dans toutes les instances sauf la première qui est originale (la numéro 0), les poids des arcs sont tous égaux à 1. En effet, nous avons conçu notre code pour qu'il puisse résoudre tout type d'instance du KEP.

Pour nos tests, nous avons décidé de fixer : $K = 3$ et $L = 4$ et de se limiter aux 100 premières instances. Notons que la taille des instances ($|V|$, $|E|$, $\%nb\_altruist$) est croissante avec leur index. Les solveurs étant des solveurs heuristiques, ils ont utilisé l'entièreté du temps mis à leur disposition pour trouver la meilleure solution réalisable possible. Nous avions fait un premier *benchmark* avec une limite de temps de 30 secondes puis un second avec 60 secondes de recherche. Cependant, l'ajout de 30 secondes supplémentaires de recherche n'a servi qu'à améliorer seulement 5 des solutions de LocalSolver et 4 pour OR-Tools, et ce de manière non significative dans les deux cas. C'est pour cette raison que nous présentons uniquement les résultats du second *benchmark*.

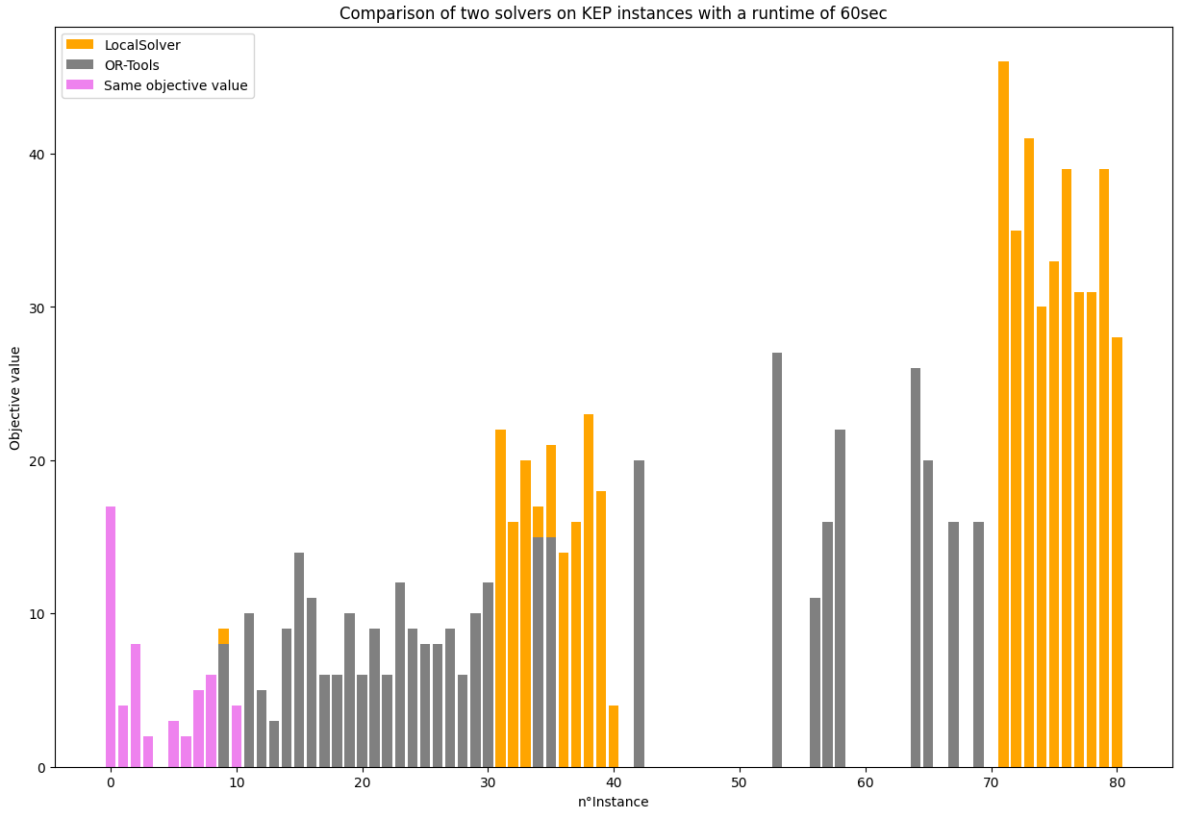

Figure 3 – Résultats

2. *PrefLib* est une bibliothèque de données maintenue par Nicholas Mattei et Simon Rey. La version originale de ce site Web a été développée par Nicholas Mattei et Toby Walsh.

Nous remarquons dans un premier temps que toutes les instances n'ont pas été résolues dans le temps imparti pour les deux solveurs, notamment les instances n°43 à n°52 qui ont l'air particulièrement compliquées à résoudre. De plus, comme nous pouvons le constater, les deux solveurs sont très complémentaires dans la mesure où ils n'ont réussi qu'à résoudre que 12 instances en commun dont 9 où ils ont trouvé une solution de même coût et seulement 3 où LocalSolver s'est trouvé être meilleur qu'OR-Tools. En effet, sur les mêmes instances résolues par les deux solveurs, OR-Tools n'a jamais fait mieux que LocalSolver. Cependant, on constate qu'OR-Tools réussit à trouver des solutions pour des instances de tailles plus variées, allant pour certaines de l'instance n°0 à l'instance n°69. LocalSolver quant à lui performe plus sur les instances de tailles plus conséquentes à savoir les instances de tailles moyenne, n°31 à n°40 où OR-Tools à plus du mal, et les instances de grande taille, n°71 à n°80 où LocalSolver a su se démarquer.

5 Conclusion

Pour conclure, nous avons vu comment nous pouvons modéliser le problème d'échange de rein dit KEP comme étant un problème de tournées de véhicules, et comment nous pouvons implémenter et résoudre ce modèle mathématique grâce à des outils libres de droits ou sous licence, respectivement OR-Tools et LocalSolver. Enfin, nous avons procédé à une comparaison expérimentale de ces deux solveurs afin d'évaluer leurs performances sur un jeu de données réputé réaliste et difficile. Les résultats se sont montrés surprenant dans la mesure où ces deux solvers semblent, à temps de recherche égale, complémentaires dans leurs résultats sur les instances testées. Il serait maintenant intéressant d'effectuer de nouveaux tests avec un temps de recherche plus conséquent et ceux sur un plus grand nombre d'instances afin de voir si la tendance se confirme.

Références

Mattei, Nicholas et Toby Walsh (2013). « PrefLib : A Library of Preference Data http ://preflib.org ». In : *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT 2013)*. Lecture Notes in Artificial Intelligence. Springer.

Pansart, Lucie, Hadrien Cambazard et Nicolas Catusse (jan. 2022). « Dealing with elementary paths in the Kidney Exchange Problem ». In : *Elsevier*.

Saidman, Susan et al. (avr. 2006). « Increasing the Opportunity of Live Kidney Donation by Matching for Two- and Three-Way Exchanges ». In : *Transplantation* 81, p. 773-82. doi : 10.1097/01.tp.0000195775.77081.25.