MQT-6021-A2021 – Travail pratique 3

Michael Morin

2 novembre 2021

Instructions

- L'équipe doit être enregistrée dans le système de gestion des équipes.
- Les communications sur le sujet du travail ne sont permises qu'à l'intérieur d'une même équipe.
- Tous les fichiers nécessaires pour reproduire votre analyse doivent être déposés dans la boîte de dépôt (voir la section Livrables).
- Le travail est noté sur 100 points. La contribution du travail à votre note finale est telle qu'indiquée dans le plan de cours.
- Le plagiat est sanctionné par la note 0 et les mesures appropriées.
- N'oubliez pas de citer vos sources (s'il y a lieu) et d'accorder une importance particulière à la qualité de la langue et à la lisibilité.

Livrables

Ce travail pratique est constitué de deux livrables principaux :

- un rapport détaillé au format HTML, celui-ci devra être créé en utilisant R Markdown (voir les détails ci-dessous);
- l'ensemble des fichiers utilisés pour recréer votre analyse y compris les fichiers R Markdown.

Vous êtes notés à la fois sur les explications et sur le code. Vous devez appliquer de bonnes pratiques de programmation (code R le plus propre possible). Vous devez appliquer les bonnes pratiques pour l'analyse (même si ces pratiques ne sont pas toujours mentionnées sur l'énoncé). Par exemple:

- afficher un exemple de données dans le rapport;
- vérifier les valeurs manquantes après le chargement de données;
- vérifier les valeurs manquantes suite aux opérations de jointures de tables;
- revérifier les valeurs manquantes après leur traitement...

Rapport détaillé

Le travail pratique a plusieurs questions et chaque question pourrait avoir plusieurs sous-questions. Les questions sont indépendantes dans le sens où elles portent sur des données différentes.

Vous devez utiliser un R Markdown par question de façon à produire plusieurs petits rapports au format HTML (un par question). Notez que toutes les sous-questions se rapportant à une même (grande) question devraient être dans le même R Markdown et dans le même fichier HTML une fois le R Markdown compilé.

Chaque fichier HTML qui correspond au rapport devra avoir un entête contenant:

- le titre contenant le numéro de votre équipe, le numéro du travail pratique et le numéro de la question;
- le nom de chacun des membres de votre équipe et leur numéro matricule entre parenthèses;
- la date.

Cet entête doit être généré via l'entête YAML du format R Markdown.

Le rapport, pour chaque (grande) question, doit contenir une section par sous-question. Pour chaque question et sous-question, détaillez et justifiez la réponse proposée par votre équipe. L'absence de justification pour une sous-question entraînera des pénalités sur la note finale du travail.

L'utilisation de R Markdown est évaluée dans ce travail de sorte que des pénalités sont attribuées pour la non-utilisation ou la mauvaise utilisation de R Markdown.

Fichiers R Markdown et autres fichiers

Vous devez fournir tous les fichiers nécessaires pour permettre de reproduire votre analyse pour chacune des questions et des sous-questions.

Vous serez pénalisés pour les fichiers manquants puisque la note de 0 sera attribuée à une sous-question pour laquelle nous ne sommes pas en mesure de reproduire votre analyse. Ceci peut arriver pour différentes raisons, les plus fréquentes sont:

- il manque un ou plusieurs fichiers de données ou de code;
- il y a une erreur dans le code qui crée un bogue.

S'il y a un aspect aléatoire à votre analyse (peu importe la raison), SVP le mentionner. Le cas échéant, nous en tiendrons compte lors de nos tentatives de reproduction de votre analyse.

Notez que les fichiers de données que nous avons distribués pourraient être remplacés par les originaux pour tester votre code. Il ne faut pas changer le nom de ces fichiers lorsque vous les utilisez dans votre code.

Nous tenterons d'exécuter le code du R Markdown dans la séquence qu'il apparaît dans RStudio et aussi de compiler le R Markdown en HTML. Le code doit fonctionner et le R Markdown doit compiler dans cette séquence. Testez sur un environnement R vide.

Veuillez suivre la convention "EQn_TPz_Qx_nomFichier.ext" pour nommer vos fichiers où:

- EQ est la chaîne de caractères EQ suivie du numéro de votre équipe n;
- TP est la chaîne de caractères TP suivie du numéro de TP z;
- Q est le caractère Q suivi du numéro de question x;
- nomFichier peut être remplacé par un nom qui décrit sommairement le contenu du fichier, par exemple, le numéro de sous-question si c'est un fichier relié à une sous-question;
- ext est l'extension du fichier (p. ex., Rmd pour un R Markdown, csv pour un fichier CSV).

Voici un exemple de fichier nommé correctement pour le R Markdown de l'équipe 3 pour la question 2 du TP 12: "EQ3_TP12_Q2_analyse.Rmd". Pour la sous-question 1 de cette même question, l'équipe a produit un fichier de données sur les courges au format CSV. Elle l'a nommé correctement: "EQ3_TP12_Q2_1dataCourges.csv".

Des points seront retirés si la convention n'est pas respectée ou s'il y a confusion dans les fichiers.

Sur l'utilisation de R

Attention! Si vous travaillez avec la commande *setwd* dans un Rmd, c'est le moment de vous en départir. Pour que le répertoire de travail soit le répertoire courant du fichier Rmd, vous pouvez créer un fichier Rproj par question. Ensuite, ouvrez d'abord le Rproj, puis le Rmd. Ceci facilitera l'importation de données à partir d'un chemin relatif, c'est-à-dire, un chemin à partir du répertoire courant. La commande *setwd* n'a pas besoin d'être utilisée et provoque généralement des erreurs étranges.

Si votre programme génère des fichiers, il ne doit pas écraser les fichiers distribués pour une question donnée.

De plus, les fichiers de données fournis (le cas échéant) ne doivent pas être modifiés directement (pas de tri de données dans Excel, pas de changement manuel ou à l'aide de code au contenu des fichiers du sous-répertoire *data*). Ceci est vrai pour tous les travaux dans le cadre de ce cours: il convient d'éviter les traitements manuels. Si des modifications doivent être faites aux données, il faut les faire avec notre outil: R. Celui-ci permet d'automatiser le traitement de données.

Archive et structure du répertoire

Pour remettre le travail, vous devrez créer une archive zip (dossier compressé).

Placez les fichiers de chaque (grande) question i dans leur propre répertoire (aussi nommé dossier) nommé Qi. Placez tous les répertoires contenant les questions dans un répertoire nommé "EQn_TPz" où n est le numéro de votre équipe et z est le numéro du TP. Placez vos HTML compilés à la racine du répertoire. Faites le ménage des fichiers qui ne sont pas nécessaires (les retirer). La Figure 1 présente un exemple de répertoire bien structuré et de l'archive résultant de la compression de ce répertoire. Le nom du répertoire parent est

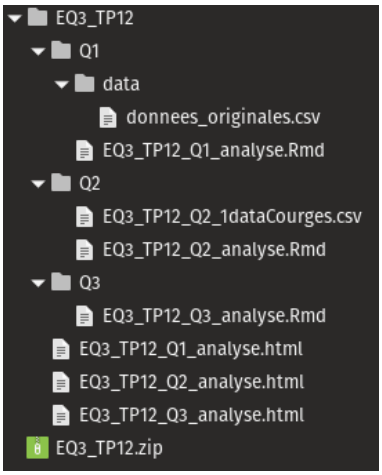


Figure 1: Exemple de répertoire bien structuré et de l'archive résultant de la compression

EQ3_TP12. Dans cet exemple, le travail pratique 12 avait 3 questions. L'équipe 3 a donc fait un répertoire par question: Q1, Q2, Q3. Dans chacun de ces répertoires, un fichier R Markdown a été créé. On remarque que chaque R Markdown a un HTML correspondant qui a été compilé par l'équipe et placé à la racine du répertoire EQ3_TP12. Chose intéressante, on remarque que des données étaient distribuées avec la question 1. Celles-ci sont disponibles dans l'archive. On remarque aussi que, pour la question 2.1, l'équipe a généré un fichier CSV nommé "EQ3_TP12_Q2_1dataCourges.csv".

Les fichiers HTML, c'est ce que nous lirons pour comprendre votre analyse. Les fichiers R Markdown, c'est ce que nous exécuterons puis compilerons en document HTML pour savoir si votre code fonctionne.

Je vous conseille de bien structurer vos répertoires dès le départ et de faire le ménage à la fin du travail pratique avant de créer l'archive. Pour faire le ménage et faire l'archive, travaillez sur une copie du répertoire (ainsi vous pourrez revenir en arrière si vous effacez quelque chose de trop). Ensuite, créez votre archive. Pour vous assurer que tout fonctionne, décompressez votre archive dans un répertoire temporaire et retestez vos fichiers R Markdown et vos scripts à partir d'un environnement R vide.

Note finale

Évitez d'utiliser des caractères spéciaux pour nommer vos fichiers et vos répertoires. Utilisez des lettres sans accents, des chiffres et le caractère _ (appelé underscore ou caractère de soulignement) pour remplacer les espaces.

Prenez bien le temps de lire l'énoncé. Notez que certaines questions sont plus courtes que d'autres.

Pour terminer, j'offre un remerciement spécial aux auxiliaires de MQT-6021 pour leur contribution aux questions de ce TP!

Bon travail!

Michael Morin

1 Programmation mathématique (40 points)

Vous cherchez à réinvestir la somme de 550 000 dollars issue de la vente d'un des brevets de votre startup. Vous ne souhaitez pas emprunter pour avoir un effet levier et vous ne considérez que des placements sûrs. Votre but est de maximiser vos gains totaux sur les 5 prochains mois (de janvier à mai) en sachant que vous aurez besoin de la totalité des fonds (incluant les intérêts et les revenus entre janvier et juin) au début juin. En d'autres termes, au début juin, aucun montant ne doit être placé.

Trois options s'offrent à vous au cours des 5 prochains mois:

- un placement d'une durée de 2 mois à 2.5 %;
- un placement d'une durée de 3 mois à 4.0 %;
- un placement d'une durée de 4 mois à 6.0 %.

Ces trois types placements ne peuvent être pris qu'au premier jour de chaque mois: pas durant le mois. Pour un placement donné, les intérêts sont reçus automatiquement à la fin de la durée du placement. Chaque placement a une durée fixe (selon l'option) et n'est pas accessible durant le mois. Par exemple, supposons qu'un placement de 1000 dollars est fait dans la première option au 1er février. Dans ce cas:

- Le montant est non disponible pour février et mars.
- Au 1er avril, le montant de 1000 dollars plus 25 dollars d'intérêt sont disponibles. Ce montant peut être placé de nouveau.

Vous pouvez prendre chaque option de placement autant de fois que vous le désirez (tant qu'aucun montant n'est placé au début juin).

Vous devez planifier vos investissements en considérant les dépenses suivantes:

- 55 000 dollars en janvier;
- 192 500 dollars en février;
- 110 000 dollars en avril;
- 165 000 dollars en mai.

Ces fonds doivent être disponibles le premier jour du mois indiqué. Par exemple, au premier jour de février, vous devez absolument avoir 192 500 dollars.

Vous aurez aussi les revenus additionnels suivants:

- 55 000 dollars en plus disponibles au premier jour de janvier;
- 60 000 dollars en plus disponibles au premier jour de mars.

Les sous-questions suivantes portent sur ce cas.

1.1 Modélisation

Formulez un programme linéaire avec `CVXR` qui vous permettra de trouver la meilleure série d'investissements possible en considérant les besoins et les entrées de fonds. Vous devez déterminer, pour chaque début de mois, si un investissement est fait, combien et dans quelle option. Toutes les contraintes décrites dans l'énoncé doivent être respectées. Notez que maximiser vos gains totaux revient à maximiser la somme en main au premier juin.

Vous aurez besoin de variables pour:

- suivre la valeur en main (excluant les montants placés);
- décider quel montant investir à quel mois et dans quelle option.

Ce problème doit être modélisé avec des variables continues seulement.

Décrivez bien votre modèle dans votre rapport (voir l'Annexe A pour un exemple de description de modèle).

1.2 Résolution

Résoudre le programme mathématique formulé en 1.1 à l'aide de `CVXR` en utilisant le solveur `GLPK_MI`.

Affichez la solution et la valeur de l'objectif à optimalité. Affichez aussi le statut du solveur: la solution est elle optimale? Interprétez la solution de façon sommaire.

2 Optimisation combinatoire (60 points)

Les véhicules semi-autonomes sont de plus en plus présents sur les routes. On peut facilement imaginer que les véhicules totalement autonomes prendront une grande place sur nos routes dans quelques années. De même, la "cohabitation" des véhicules conduits par des humains et des véhicules totalement/partiellement autonomes créera (très probablement) de nouveaux besoins au niveau des infrastructures.

Un futur probable: La mégapole de Montbec (région urbaine continue formée des métropoles de Québec, Drummondville, Trois-Rivières, Sherbrooke et Montréal) fourmille de véhicules totalement autonomes. La conduite manuelle est toujours possible selon la loi, mais de moins en moins de véhicules conduits manuellement sont présents sur les routes.

Avec l'avènement des véhicules totalement autonomes, on a vu grandir d'autres technologies dont certaines automatisent la sécurité et la détection d'infractions sur les routes. L'une de ces technologies est le système de détection d'infractions aux intersections de la compagnie InterCov. Il va sans dire que les infractions aux intersections (p. ex., non-respect des feux de circulation) sont majoritairement liées aux conducteurs humains ou à des pannes (extrêmement rares) du système de conduite d'un véhicule autonome.

Problème général: Montbec souhaite se prémunir du système d'InterCov pour certains de ses quartiers les plus peuplés.

InterCov est un système décentralisé qui utilise des boîtes noires positionnées aux intersections. Lorsqu'une boîte noire est positionnée à une intersection, elle peut détecter les infractions à cette intersection et aux intersections directement visibles (sans obstructions visuelles) à partir de celles-ci. Pour les quartiers qu'elle souhaite couvrir, Montbec souhaite acheter le moins de boîtes noires possible pour couvrir l'ensemble des intersections du quartier.

Exemple d'instance du problème et de solution au problème: Une *instance* du problème de Montbec/InterCov peut entièrement être décrite à l'aide d'une matrice M qui indique, pour chaque paire d'intersections (i, j), si oui ou non l'intersection j est visible à partir de l'intersection i. Si j est visible à partir de i alors $M_{ij} = 1$, sinon $M_{ij} = 0$.

Voici un exemple d'une telle matrice stockée dans le fichier data/Q2_tiny_interproblem_adjacency_matrix.rds.

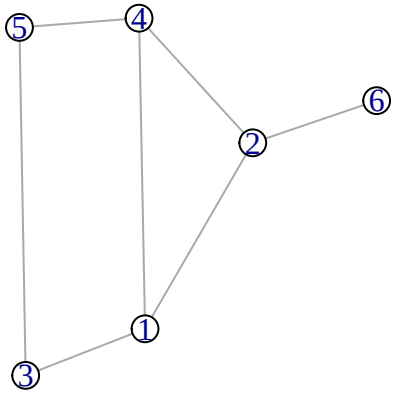
```
tiny_intercov_matrix
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    1    1    0    0
## [2,]    1    1    0    1    0    1
## [3,]    1    0    1    0    1    0
## [4,]    1    1    0    1    1    0
## [5,]    0    0    1    1    1    0
## [6,]    0    1    0    0    0    1
```

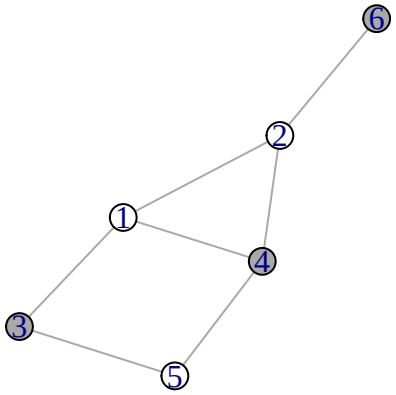
Dans cet exemple, l'intersection 3 est visible à partir de l'intersection 1, $M_{ij} = 1$.

Cette instance peut-être représentée par un graphe où les intersections sont des noeuds et la visibilité est indiquée par des arêtes (voir la figure ci-dessous). Les boucles, c.-à-d., les cas où une intersection est visible

à elle-même, ne sont pas représentés pour alléger la figure. De même, ici, la visibilité est mutuelle, c.-à-d., $M_{ij} = M_{ji}$, de sorte qu'une seule arête est nécessaire par paire d'intersections mutuellement visibles.



Une *solution réalisable* à ce problème est une sélection d'intersections qui permet de "voir" l'ensemble des intersections. Par exemple, dans l'instance décrite ci-dessus, une solution réalisable (qui couvre l'ensemble des intersections est la suivante:



Dans la solution réalisable ci-dessus, les boîtes noires sont positionnées aux intersections 3, 4 et 6.

Une *solution optimale* est une solution réalisable avec un nombre minimal de boîtes noires.

Et vous dans tout ça: Vous êtes analyste(s) pour la ville de Montbec. Ultimement, vous devez offrir des recommandations sur le nombre de boîtes noires à acheter et sur la position de celles-ci dans différents quartiers. Par contre, vous savez que certains quartiers de la ville contiennent beaucoup d'intersections. Ce serait très difficile de trouver une solution manuellement et, de toute façon, vous savez qu'il est préférable d'automatiser la tâche et de laisser travailler l'ordinateur. Par conséquent, vous décidez de modéliser puis de résoudre ce problème d'analytique prescriptive. Sachant que, pour certains cas, le problème pourrait demander beaucoup de temps de calcul, vous décidez de tester plusieurs approches enseignées dans votre cours d'analytique. Cette question et ses sous-questions portent sur la modélisation et sur la résolution du problème Montbec/InterCov.

2.1 Modélisation: Programme en nombres entiers avec CVXR

Utilisez `CVXR` pour formuler un programme en nombres entiers pour résoudre l'instance du problème contenue dans le fichier data/Q2_tiny_interproblem_adjacency_matrix.rds.

Assurez-vous que votre formulation est générale. Elle devrait pouvoir résoudre le problème, peu importe l'instance. Décrivez bien votre modèle dans votre rapport (voir l'Annexe A pour un exemple de description de modèle).

Indice pour faire une formulation générale: Créez une fonction qui prend une matrice d'adjacence en paramètre

et qui retourne une liste d'objets: (1) les variables du problème (pour un accès facile); (2) l'objet créé par l'appel à `CVXR::Problem`. Voici un exemple de code à compléter:

```
model_coverage_dom_set <- function(adjacency_matrix) {
  # TODO: Code du modèle ici
  # ...

  # Objet modèle
  problem <- CVXR::Problem(objective,
                           constraints)

  result = list(variables = TODO_remplacer_par_variables,
                problem = problem)

  return(result)
}
```

Avec un code tel que celui ci-dessus, vous pourrez faire appel à la fonction ainsi (où `TODO_votre_instance` est votre instance telle que décrite par sa matrice d'adjacence):

```
# Appel a la fonction de modelisation:
my_problem <- model_coverage_dom_set(adjacency_matrix = TODO_votre_instance)
```

Ensuite, vous pouvez utiliser la sortie `my_problem` comme ceci:

```
# Aller chercher les variables:
my_problem$variables

# Aller chercher le modele:
my_problem$problem
```

2.2 Résoudre le programme

Résolvez le programme mathématique formulé à la Section 2.1 avec `CVXR` en utilisant le solveur `GLPK_MI`.

Assurez-vous que votre résolution est générale. Elle devrait pouvoir fonctionner sur n'importe quel `CVXR::Problem`.

Indice pour faire une résolution générale: Créez une fonction qui prend en entrée un objet `CVXR::Problem` et qui retourne la sortie de l'appel à `CVXR::psolve`.

Utilisez la fonction créée ci-dessus pour résoudre le problème pour l'instance du fichier suivant:

- data/Q2_tiny_interproblem_adjacency_matrix.rds.

Affichez aussi le statut du solveur: la solution est elle optimale? Affichez la solution et expliquez celle-ci.

2.3 Modélisation générale à l'aide de fonctions et de variables globales

Créez une modélisation générale du problème à l'aide de fonctions et de variables globales. (*Indice:* Vous pouvez vous inspirer du Tutoriel 10b). La modélisation doit être générale dans le sens où il devrait être possible de résoudre plusieurs instances du problème.

Nous souhaitons avoir une modélisation sous la forme de plusieurs fonctions:

- Une fonction qui calcule la valeur de fonction objectif nommée `val_obj`.
- Une fonction qui vérifie si une solution est réalisable nommée `is_sol_feasible`.
- Une fonction qui compare deux solutions et choisit la meilleure nommée `is_sol_x_better_than_y`.

Attention! Les fonctions `is_sol_feasible` et `val_obj` devraient prendre une solution en entrée alors que `is_sol_x_better_than_y` devrait en prendre deux. Les paramètres de l'instance du problème doivent être des variables globales pour permettre, comme dans le Tutoriel 10b, de facilement changer l'instance du problème.

En plus de définir les fonctions, vérifiez que les valeurs de retour de `val_obj`, `is_sol_feasible` et `is_sol_x_better_than_y` sont correctes pour l'instance définie dans le fichier suivant:

- data/Q2_tiny_interproblem_adjacency_matrix.rds.

en utilisant les solutions suivantes:

```
x <- c(0, 0, 1, 1, 0, 0)
y <- c(0, 0, 1, 1, 0, 1)
z <- c(1, 1, 1, 1, 1, 1)
```

Pour `is_sol_x_better_than_y`, comparez toutes lrs solutions entre-elles dans tous les ordres possibles et assurez-vous que les valeurs sont bonnes.

2.4 Deux algorithmes de recherche aléatoire

Vous devez créer un algorithme "efficace" de recherche aléatoire pour ce problème. Pour ce faire, créez et testez deux versions de la recherche aléatoire.

Attention! Voici une astuce pour vos recherches aléatoires: Vous allez vite remarquer que si vous initialisez `best_incumbent` à une solution où aucune intersection n'a de boîte noire la recherche aléatoire a beaucoup de difficulté à trouver une solution réalisable. Pour vos deux recherches aléatoires, initialisez `best_incumbent` à une solution où toutes les intersections ont une boîte noire. Au moins, la première solution rencontrée sera réalisable.

Voici, intuitivement, une description des deux variantes de la recherche aléatoire attendues:

- Dans une première version de votre algorithme, sélectionnez les valeurs de la solution candidate aléatoirement avec une probabilité uniforme sur le domaine des variables.
- Dans une seconde version de votre algorithme, lorsque vous créez votre solution candidate aléatoirement, trouvez un moyen de faire en sorte que le nombre d'intersections sans boîte noire soit légèrement plus petit que dans la première version.

Vos algorithmes doivent être généraux dans le sens où ils doivent pouvoir fonctionner même si l'instance du (même) problème change (voir le Tutoriel 10a qui contient une telle implémentation).

Vérifiez que vos algorithmes fonctionnent avec l'instance suivante:

- `data/Q2_large_interproblem_adjacency_matrix.rds`.

Pour ce faire, affichez si oui ou non la solution retournée par chacun est réalisable et affichez sa valeur de fonction objectif.

2.5 Expérimentations

Vos deux recherches aléatoires devraient contenir un paramètre correspondant au nombre d'itérations (nombre de solutions générées aléatoirement lors de l'exploration). Fixez ce nombre d'itérations à 100. Vous pouvez supposer que ce nombre d'itérations a été fixé par des expérimentations préliminaires lors d'une phase de configuration.

Ici, on vous demande d'exécuter des expérimentations dans le but de tester vos deux recherches aléatoires et de les comparer sur les instances définies dans les fichiers:

- `Q2_huge1_interproblem_adjacency_matrix.rds`
- `Q2_huge2_interproblem_adjacency_matrix.rds`

- `Q2_huge3_interproblem_adjacency_matrix.rds`
- `Q2_huge4_interproblem_adjacency_matrix.rds`
- `Q2_huge5_interproblem_adjacency_matrix.rds`
- `Q2_huge6_interproblem_adjacency_matrix.rds`
- `Q2_huge7_interproblem_adjacency_matrix.rds`

Pour ce faire, vous devez rouler chaque algorithme stochastique 3 fois sur chaque instance (vous pouvez faire plus d'exécutions si vous le désirez, mais ce sera très long).

La comparaison des algorithmes devra être faite à la Section 2.6. Ici, on ne souhaite avoir que le code qui permet de rouler les expérimentations.

Vous devez aussi exécuter le solveur `GLPK_MI` sur chacune des instances à l'aide des fonctions générales de création du modèle et de résolution définies précédemment. Les fontions créées aux Sections 2.1 et 2.2 devraient vous permettre de rouler `GLPK_MI` sur chacune de ces instances.

Vous devez stocker les résultats de vos expérimentations dans une table de données `tibble`. Au minimum, il faut avoir l'information suivante:

- nom de l'algorithme (ou du solveur);
- numéro de l'instance;
- numéro de l'exécution de l'algorithme sur l'instance;
- temps en secondes pour l'exécution de l'algorithme (ou du solveur);
- la solution est-elle optimale?;
- la solution est-elle réalisable?;
- la valeur de la fonction objectif.

Sauvegardez les résultats de vos expérimentations dans un fichier RDS. Votre expérimentation ne doit rouler que si le fichier RDS est absent.

Attention!! Si vous utilisez un portable pour exécuter vos expérimentations, je conseille de le brancher au préalable. De même, il faut qu'il soit bien "ventilé." De plus, si certains cas sont difficiles à résoudre, vous pouvez utiliser une limite de temps de 1 minute ou plus pour limiter le temps d'exécution du solveur (voir les tutoriels).

2.6 Comparaison des méthodes de résolution

Attention! Si vous n'avez pas été en mesure d'exécuter vos algorithmes pour générer les résultats au numéro précédent alors vous pouvez charger `Q2_experiments_coverint_huge.rds` qui est fournit avec l'énoncé. Dans ce cas, vous n'aurez pas les points pour le numéro précédent, mais vous pourrez quand même tracer des graphiques. Dans `Q2_experiments_coverint_huge.rds`, la recherche aléatoire de base se nomme `rs`, la recherche aléatoire améliorée se nomme `cirs` et les observations qui correspondent à la résolution du programme mathématique avec `GLPK_MI` sont nommées `GLPK_MI`.

Comparez les méthodes de résolution en utilisant plusieurs visualisations de leur performance. Laquelle des méthodes devriez-vous choisir pour l'implémentation d'un module de recommandations pour la ville de Montbec? Pouvez-vous imaginer un cas où ce choix ne serait pas idéal? Expliquez.

Incluez au moins un graphique avec une information de la valeur de la fonction objectif pour la solution finale trouvée par l'algorithme par rapport à l'optimale (si elle a été trouvée par `GLPK_MI`). Vous devez visualiser l'information sur:

- le temps de calcul requis avant que l'algorithme ou le solveur termine;

- la valeur de la solution en termes de l'objectif du problème original: nombre de boîtes noires dont on a besoin pour implémenter la solution dans la "vraie vie vraie."

En plus, au moins l'un de vos graphiques devrait permettre de répondre aux questions suivantes pour chacune des solutions trouvées par chacun des algorithmes pour chaque instance:

- La solution est-elle réalisable?
- La solution est-elle optimale?

Vos graphiques devraient permettre de voir si le comportement est identique pour chaque instance ou non.

Décrivez brièvement ce que vous observez.

Annexes

A Exemple de description d'un modèle mathématique

Dans certaines questions de ce TP, on vous demande d'écrire un programme mathématique en utilisant `CVXR` *et de le décrire. Cette annexe présente un exemple de modèle mathématique dans CVXR et une description du modèle. Les modèles que vous avez à faire sont différents de celui-ci, mais ils suivent le même principe et nous nous attendons à avoir une présentation similaire de vos modèles.*

Pour décrire un modèle dans le rapport, il faut décrire ses composants et les identifier en plus de présenter le code:

- Identifiez les groupes de paramètres, les variables et les contraintes ainsi que la fonction objectif par un commentaire dans votre code. Des exemples de commentaires pour identifier les composants du modèle figurent dans le code ci-dessous. Par exemple, la première contrainte est identifiée par C1 alors que le troisième paramètre est identifié par P3.
- Définissez chaque groupe de variables dans votre rapport en utilisant la référence mentionnée ci-dessus pour indiquer de quel groupe de variables il s'agit.
- Définissez chaque groupe de contraintes dans votre rapport en expliquant leur fonction et en utilisant la référence mentionnée ci-dessus pour indiquer où dans le code les contraintes sont situées.
- Définissez votre fonction objectif en mots en indiquant où elle se trouve avec la référence mentionnée ci-dessus.

Voici un exemple de code d'un modèle suivi d'une description correcte du modèle pour le problème MBI Corporation avec variables entières (du Tutoriel 8a) tel que modélisé ci-dessous:

```
## Paramètres
# P1
pProfit <- c(8000, 12000)
# P2
pOuvrage <- c(300, 500)
# P3
pCout <- c(10000, 15000)
# P4
pOuvrageMax <- 200000
# P5
pBudgetMax <- 8000000
# P6
pLowerBounds <- c(100, 200)

## Variables
# V1
```

```
vX <- CVXR::vstack(CVXR::Variable(name = 'X_1', integer = TRUE),
                   CVXR::Variable(name = 'X_2', integer = TRUE))

## Objectif
# O1
objective <- CVXR::Maximize(t(pProfit) %*% vX)

# Contraintes
constraints <- list(
  # C1
  t(pOuvrage) %*% vX <= pOuvrageMax,
  # C2
  t(pCout) %*% vX    <= pBudgetMax,
  # C3
  vX                 >= pLowerBounds
)

# Objet modèle
problem <- CVXR::Problem(objective,
                         constraints)
```

Les variables de décision, `vX` (V1), représentent le nombre de CC-7 et le CC-8 à produire le mois prochain.

Le vecteur de paramètres `pProfit` (P1) représente le profit unitaire en dollars pour la vente de CC-7 et de CC-8 (respectivement). Le vecteur de paramètres `pOuvrage` (P2) représente l'ouvrage, en nombre de jours de travail, requis pour produire un CC-7 et un CC-8 (respectivement). Le vecteur de paramètres `pCout` (P3) représente le coût du matériel requis (en dollars) pour produire un CC-7 et un CC-8 (respectivement). Le paramètre `pOuvrageMax` (P4) représente le budget total de l'usine en nombre de jours de travail pour le mois prochain. Le paramètre `pBudgetMax` (P5) représente le budget total de l'usine en dollars pour le mois prochain. Le vecteur de paramètres `pLowerBounds` (P6) représente respectivement le nombre minimal de CC-7 et de CC-8 à produire le mois prochain.

L'objectif est de maximiser le profit total (O1) sous les contraintes C1 à C3. La contrainte C1 borne le temps total requis (en nombre de jours) pour la production par le budget (en nombre de jours). La contrainte C2 borne le coût total en matériel (en dollars) pour la production par le budget de matériel (en dollars). Les contraintes du groupe C3 représentent les bornes inférieures sur la production de CC-7 et de CC-8.