Feuille de travaux pratiques n° 3
Survie sur Mars

1 Contexte

Mars : ses fantastiques paysages, son Grand Canyon, ses sommets vertigineux de plus de 20 km de haut ! Vous avez toujours voulu y partir en vacances, et ce rêve devient enfin réalité. Toujours à l'affût des bons plans, vous avez bénéficié d'un aller simple vers cette magnifique planète pour un prix modique ! Il y a quelques petits détails que M. Musk (le vendeur) n'a pas soulignés explicitement avant l'embarquement. Ces quelques détails sont par contre parfaitement expliqués dans une vidéo en cours de vol. Tout d'abord, le voyage est long ! Cela implique qu'une grande partie de la nourriture et de l'oxygène disponibles seront consommés pendant le voyage. De plus, il n'y aura quasiment plus de carburant pour un vol retour. La partie touristique sur place s'annonce donc de courte durée avant une mort certaine. Pour survivre, il faudra renoncer au tourisme, et récupérer des ressources à transformer, pour produire des aliments, de l'oxygène et du carburant pour un futur retour. Pendant le vol, un passager a travaillé à partir d'images de Mars afin d'identifier le point d'atterrissage, la topologie des lieux, et des points d'intérêt environnants. On peut lui faire confiance, il est passé par le parcours VICO, et il semble certain du résultat de son algorithme de traitement d'images. Malheureusement, un problème linguistique vous empêche de prendre connaissance des données issues du traitement d'images : vous ne parlez pas la même langue ! Ce passager est en effet chinois ! Heureusement, un autre passager passionné par les langues peut vous aider. D'ailleurs, s'il a choisi de faire ce voyage, c'est justement pour apprendre de nouvelles langues, éventuellement inconnues sur Terre. Il s'agit d'un diplômé du parcours ATAL, il est parfaitement fiable sur ce sujet, et on fera l'hypothèse qu'il n'y a pas de perte d'information dans sa traduction. Il reste à faire des prévisions réalistes sur la consommation de ressources et les restrictions éventuelles pour la survie. En se basant sur la consommation de l'équipage en vol, un diplômé du parcours DS a proposé un modèle statistique. Les besoins en oxygène apparaissant comme étant critiques d'après ce modèle, un sous-ensemble de points d'intérêt sera donc considéré de manière prioritaire. Afin de se rendre dans les points d'intérêt et d'y récupérer les ressources, des drônes dont l'autonomie est limitée sont disponibles et leur programmation est assuré par un ancien diplômé du Master ARIA (ECN). Un SGBD a été mis en place par un diplômé du parcours ALMA, qui est prêt à travailler en support sur les autres tâches. Tout ce qu'il reste maintenant à faire, est de planifier les trajectoires des drônes vers les points d'intérêt, ainsi que leur retour à la base tout en minimisant le temps de trajet, car les drônes nécessitent également des ressources. Cette tâche demande des compétences en optimisation dont personne ne dispose à bord. Le diplômé du parcours ORO a pris la décision de ne pas embarquer. En effet, avant de valider son achat, il a pris connaissance de toutes les informations contractuelles, y compris celles qui étaient écrites en petit en bas de page. Il ne reste donc que vous, (futur) diplômé de licence, pour vous charger de la partie optimisation. Un dernier détail : les ressources étant limitées, il a été décidé de larguer dans l'espace tous les passagers inutiles. Vous n'avez donc pas droit à l'échec !

Il est temps de prendre connaissance en détail de la tâche qui vous revient. Ce projet est un sous-problème du problème de planning dynamique de mission pour une flotte de drônes d'exploration [1] . Ce problème complexe se situe à l'intersection de plusieurs domaines scientifiques : robotique, intelligence artificielle pour la décision, recherche opérationnelle.

À chaque drône est affecté un sous-ensemble de points d'intérêt. Chaque drône part d'une base (permettant de l'entretenir et de recharger sa batterie) et va ensuite explorer les points d'intérêt lui étant affectés avant de rentrer à la base tout en cherchant à minimiser le temps total/la consommation énergétique nécessaire pour cela.

1. Ce problème est décrit dans les références suivantes :
Brummit, B. & Stentz, A. (1996). Dynamic mission planning for multiple robots. *Proceedings of the IEEE international conference on robotics and automation,* April 1996.
Brummit, B. & Stentz, A. (1998). GRAMMPS : a generalized mission planner for multiple robots. *Proceedings of the IEEE international conference on robotics and automation,* May 1998

Le planning initial (affectation de sous-ensembles de point d'intérêt à des drônes, puis définition pour chaque drône de l'ordre d'exploration de ces points d'intérêt) est réalisé à partir d'estimations des temps nécessaires pour qu'un drône se rende de la base vers un point d'intérêt, ou d'un point d'intérêt vers un autre point d'intérêt. Ces estimations ont été obtenues à l'aide de données partielles, et leur fiabilité pourrait donc être remise en cause. En effet, les informations concernant les zones à parcourir seront complétées pendant les déplacements des drônes. Les images de ces zones vues sous d'autres angles, peuvent permettre de découvrir un relief différent, des obstacles... Les données sont donc complétées en cours de mission. Il est possible de se rendre compte qu'un déplacement d'un point d'intérêt vers un autre, initialement prévu pour un drône, soit finalement beaucoup trop long. Il devient alors judicieux de revoir les affectations des points d'intérêt qu'il reste à explorer, aux drônes en cours de mission (et ensuite les ordres d'exploration des points d'intérêt). C'est pourquoi on parle de planning dynamique d'une mission, et c'est dans ces modifications de planning que la flotte complète est considérée.

Pour ce projet réalisé en trois séances, nous allons simplifier ce problème. Tout d'abord, nous travaillerons sans toutes les incertitudes sur les données. Les temps nécessaires pour se rendre d'un point d'intérêt à un autre seront donc connus de manière précise et définitive. Il n'y aura par conséquent pas de changement en ce qui concerne les affectations des points d'intérêt aux drônes. Chaque drône devra donc explorer un ensemble définitivement précisé de points d'intérêt, avant de rentrer dans sa base. Nous supposerons que l'affectation des points d'intérêt aux drônes est déjà réalisée et nous considérerons le sous-problème consistant à déterminer l'ordre de visite des points d'intérêt par un drône, de manière à minimiser le temps total d'exploration. Nous obtenons donc un problème de voyageur de commerce (souvent abrégé par TSP : Traveling salesman problem).

Le problème complet initialement posé est une variante du $mTSP$ (où le m signifie "multiple"), auquel on ajoute de l'incertitude sur les données. La différence essentielle entre le TSP et le $mTSP$ est qu'il y a plusieurs voyageurs partant de divers sites, et devant visiter de manière collaborative l'ensemble des villes.

2 Modélisation du problème et difficultés

Le nombre de lieux à visiter (base, points d'intérêt) est noté n et le temps nécessaire pour aller du lieu i au lieu j est notée c_{ij} (où $i, j \in \{1, \ldots, n\}$). Les variables de décision sont données par

$$x_{ij} = \begin{cases} 1 & \text{si le drône se rend directement du lieu } i \text{ au lieu } j \\ 0 & \text{sinon} \end{cases}$$

Le modèle s'écrit ensuite

$$\begin{array}{rlcll} \min\ z & = & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & & \\ & s.c. & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 & \forall i \in \{1,\ldots,n\} & (1) \\ & & \displaystyle\sum_{i=1}^{n} x_{ij} = 1 & \forall j \in \{1,\ldots,n\} & (2) \\ & & \displaystyle\sum_{i,j\in S} x_{ij} \leq |S|-1 & \forall S \text{ avec } |S| \leq n-1 & (3) \\ & & x_{ij} \in \{0,1\} & \forall i,j \in \{1,\ldots,n\} & \end{array}$$

Si on ignore l'ensemble de contraintes (3) (ayant pour but de casser les sous-tours), on obtient la modélisation du problème d'affectation linéaire. Ce dernier problème ne comptant que $2n$ contraintes, sa saisie ne saturera donc pas la mémoire de la machine utilisée pour le résoudre. Par contre, la saisie de la modélisation complète compte environ 2^n contraintes, elle devient donc inexploitable directement pour de larges valeurs de n.

Comme nous l'avons vu en cours, le problème d'affectation peut être vu comme un problème de permutation. Toute solution admissible de ce problème est décrit par une permutation (et inversement). Une permutation peut également être vue comme un produit de cycles disjoints. Par contre, une permutation ne décrit pas nécessairement une solution admissible du problème de voyageur de commerce. Il faut pour cela que le produit de cycles décrivant la permutation ne soit composé que d'un seul cycle.

Exemple : on considère le distancier suivant.

	1	2	3	4	5	6	7
1	0	786	549	657	331	559	250
2	786	0	668	979	593	224	905
3	549	668	0	316	607	472	467
4	657	979	316	0	890	769	400
5	331	593	607	890	0	386	559
6	559	224	472	769	386	0	681
7	250	905	467	400	559	681	0

On peut remarquer que ce distancier est symétrique (il ne s'agit pas bien sûr pas d'une obligation en général, car il pourrait y avoir du relief ou des obstacles). Si nous ignorons l'ensemble de contraintes (3) et résolvons l'instance correspondante du problème d'affectation, nous obtenons nécessairement la solution donnée par $x_{11} = x_{22} = x_{33} = x_{44} = x_{55} = x_{66} = x_{77} = 1$ avec $z = 0$ (tous les autres x_{ij} sont nuls). Si nous voyons cette solution comme une permutation, nous obtenons

$$\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

et comme un produit de cycles disjoints, nous obtenons

$$(1)(2)(3)(4)(5)(6)(7).$$

Sans surprise, cette solution est composée de 7 cycles et n'est donc pas admissible pour le problème de voyageur de commerce. Ce résultat s'obtiendra quelque soit l'instance considérée (car la distance de i vers i est nécessairement de 0). Nous pouvons donc immédiatement imposer que $x_{ii} = 0$ pour tout i. Cela ne nécessite en réalité pas l'ajout d'une contrainte additionnelle, mais une simple modification des ensembles de contraintes (1) et (2). On obtient alors

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1, \ \ \forall i \in \{1, \ldots, n\} \ \ (1')$$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1, \ \ \forall j \in \{1, \ldots, n\} \ \ (2')$$

L'ensemble de contraintes (3) peut être modifié en conséquence pour ne contenir que les contraintes cassant les sous-tours de taille ≥ 2, on obtient

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \ \ \forall S \text{ avec } 2 \leq |S| \leq n - 1 \ \ (3')$$

Les ensembles de contraintes $(1')$, $(2')$ et $(3')$ sont équivalents aux ensembles de contraintes (1), (2) et (3). En ignorant l'ensemble de contraintes $(3')$ et en résolvant le problème correspondant, la solution obtenue ne contiendra plus de cycle de taille 1. Si nous somme chanceux, nous obtiendrons une permutation qui se décompose en un unique cycle, soit une solution admissible pour le problème de voyageur de commerce. **Affirmation :** Cette solution est alors également optimale pour ce problème.

La résolution avec l'instance de l'exemple en ignorant l'ensemble de contraintes $(3')$ nous donne $x_{17} = x_{26} = x_{34} = x_{43} = x_{51} = x_{62} = x_{75} = 1$ avec $z = 2220$ (tous les autres x_{ij} sont nuls). Si nous voyons cette solution comme une permutation, nous obtenons

$$\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 7 & 6 & 4 & 3 & 1 & 2 & 5 \end{array}$$

et comme un produit de cycles disjoints, nous obtenons

$$(175)(26)(34).$$

Nous n'avons pas la chance d'obtenir une solution composée d'un seul cycle. Nous aurons donc besoin de considérer les contraintes incluses dans $(3')$.

3 Résolution exacte : Introduction progressive des contraintes cassant les sous-tours

Il est bien sûr impensable d'ajouter toutes les contraintes de l'ensemble $(3')$. Nous allons donc à chaque fois ajouter une seule contrainte de manière à casser un sous-tour et résoudre le problème obtenu. Nous espérons ainsi avoir la chance d'obtenir une solution composée d'un seul cycle. En général, on préfère casser le plus petit cycle du produit.

Si on souhaite casser le cycle (26) du produit, nous devons uniquement ajouter la contrainte

$$x_{26} + x_{62} \leq 1.$$

En résolvant le problème avec cette contrainte additionnelle, on obtient la solution donnée par $x_{17} = x_{25} = x_{34} = x_{43} = x_{56} = x_{62} = x_{71} = 1$ avec $z = 2335$ (tous les autres x_{ij} sont nuls). Si nous voyons cette solution comme une permutation, nous obtenons

$$\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 7 & 5 & 4 & 3 & 6 & 2 & 1 \end{array}$$

et comme un produit de cycles disjoints, nous obtenons

$$(17)(256)(34).$$

Nous n'avons pas encore une solution composée d'un seul cycle. Nous cassons alors le cycle (17) en ajoutant la contrainte

$$x_{17} + x_{71} \leq 1.$$

La résolution de ce nouveau problème nous donne alors $x_{15} = x_{23} = x_{34} = x_{47} = x_{56} = x_{62} = x_{71} = 1$ avec $z = 2575$ (tous les autres x_{ij} sont nuls). Si nous voyons cette solution comme une permutation, nous obtenons

$$\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 5 & 3 & 4 & 7 & 6 & 2 & 1 \end{array}$$

soit le cycle (1562347) ! **Affirmation :** Nous avons donc ici obtenu une solution admissible et optimale pour cette instance du problème de voyageur de commerce en ne considérant que deux contraintes de l'ensemble $(3')$.

Nous résumons l'approche utilisée.
- Résoudre le problème en ne considérant que les ensembles de contraintes $(1')$ et $(2')$
- Répéter
  - Décomposer la solution obtenue en un produit de cycles disjoints
  - Si la solution obtenue est composée de plusieurs cycles alors ajouter au problème la contrainte cassant le plus petit sous-tour, puis résoudre le problème obtenu

  Jusqu'à l'obtention d'une solution composée d'un seul cycle

4 Résolution approchée

Comme une résolution exacte peut prendre du temps, on peut envisager suivant le contexte, d'avoir recours à une méthode de résolution approchée. Pour cette partie du projet, nous ferons l'hypothèse que le distancier est symétrique. Définir une solution admissible pour le problème de voyageur de commerce est particulièrement simple, il suffit de définir un ordre de visite des lieux. Bien entendu, nous souhaitons de plus que la longueur du cycle soit aussi courte que possible. L'heuristique la plus simple pour cela est appelée l'heuristique de plus proche voisin. En partant d'un point d'intérêt arbitraire, on choisit de manière répétée le lieu le plus proche parmi les lieux qui n'ont pas encore été explorés. Une fois tous les lieux explorés, on revient au lieu initial.

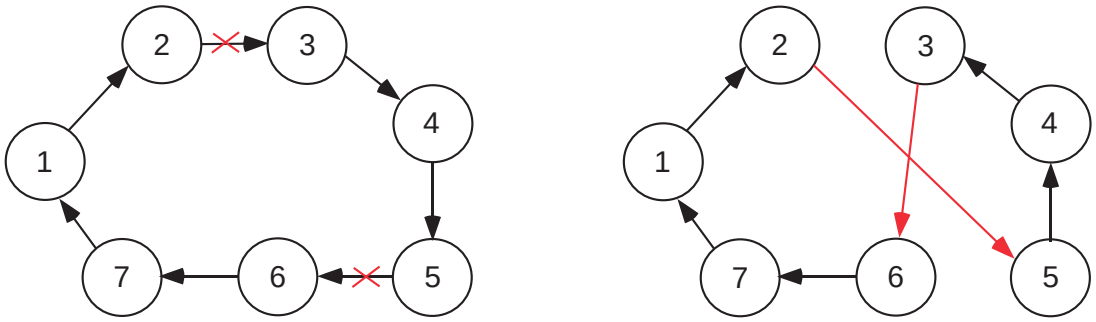
Dans l'exemple, en partant du lieu d'indice 1, on se dirige séquentiellement vers les lieux d'indices 7, 4, 3, 6, 2, 5 avant de revenir au lieu 1. On obtient donc le cycle (1743625) avec $z = 2586$.

Il est intéressant de remarquer que la solution obtenue dépend du choix du lieu de départ.

Après avoir obtenu une solution admissible, on peut avoir recours à une heuristique qui définit d'autres solutions admissibles (dites voisines), avec l'espoir de trouver une solution admissible qui soit meilleure que la solution de départ. Une des heuristiques les plus connues pour cela est appelée l'heuristique 2-opt. Son principe se décrit plus simplement en voyant un cycle comme un ensemble d'arêtes (en théorie des graphes, on préfère alors utiliser le terme circuit, mais on tolèrera cette approximation). Par exemple, le cycle (1234567) peut être vu comme la suite d'arêtes $\{(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,1)\}$. En prenant deux arêtes non consécutives (i,j) et (i',j') qu'on supprime du cycle, et en ajoutant les arêtes (i,i') et (j,j'), on obtient un nouveau cycle. Nous n'avons pas besoin de construire ce nouveau cycle pour connaître son coût total, il suffit de mesurer la différence Δ avec le cycle de départ, où $\Delta = c_{ij'} + c_{jj'} - c_{ij} - c_{i'j'}$. Si on a $\Delta < 0$, on a trouvé une solution améliorante. Ensuite, il y a plusieurs manières d'implémenter l'heuristique 2-opt, deux sont proposées ici : soit on énumère tous les voisins possibles pour trouver le meilleur voisin (en espérant qu'il soit améliorant), soit on arrête l'énumération dès qu'on a un voisin améliorant.

L'heuristique 2-opt peut ensuite être appliquée de manière répétée tant qu'on trouve une solution améliorante. On parle de méthode de descente.

Une illustration de la création d'un voisin à l'aide de l'heuristique 2-opt est donnée ci-dessous. Si on supprime les arêtes (arcs sur la figure) $(2,3)$ et $(5,6)$, on ajoute alors les arêtes (arcs sur la figure) $(2,5)$ et $(3,6)$. Dans le but de créer un cycle (un séquence de lieux), l'ordre de visite entre les lieux 3 et 5 est inversé. En partant du cycle (1234567), on obtient alors le cycle (1254367).



5 Travail à effectuer

Une implémentation des méthodes de résolution (exacte et approchée) est attendue en utilisant Julia/JuMP/-GLPK.

Plusieurs fichiers sont disponibles sur madoc dans l'archive `ProjetRO.zip` :
- Le fichier `Projet_NOM1_NOM2.jl` contenant un squelette à compléter,
- Un dossier `plat` contenant des instances numériques dont le distancier est symétrique,
- Un dossier `relief` contenant des instances numériques dont le distancier est asymétrique.

Les instances numériques sont des fichiers textes dont le format est donné par :
- La première ligne indique la taille du problème,
- Les lignes suivantes indiquent le distancier.

Le fichier des données de l'exemple est indiqué ci-dessous.

```
7
0 786 549 657 331 559 250
786 0 668 979 593 224 905
549 668 0 316 607 472 467
657 979 316 0 890 769 400
331 593 607 890 0 386 559
559 224 472 769 386 0 681
250 905 467 400 559 681 0
```

Un parseur pour ces fichiers de données est fourni dans le squelette de code. Un script de résolution de la totalité des instances numériques est également fourni. Les fonctions suivantes seront utiles pour la méthode exacte :

– `AffExpr()` : Cette fonction sans argument retourne une expression affine (expression linéaire des variables) vide.
– `push!(aff::AffExpr, new_coeff::Float64, new_var::Variable)` : Cette fonction ajoute un terme dans l'expression affine passée en premier paramètre. Par exemple, l'appel à `push!(expr,1.0,x[2,6]` ajoute `1.0x[2,6]` dans l'expression `expr`.
– `@constraint(m::Model,con)` qui est la macro déjà utilisée dans les TPs précédents sans donner de nom à la contrainte, doit absolument être utilisée avec une variable qui récupère la valeur de retour. Par exemple `xxx = @constraint(m,expr <= 1)` ajoute la contrainte `expr <= 1` où `expr` est une expression affine.

La date limite de remise des projets est fixée au samedi 7 avril à 12h. Le code source et le rapport devront être remis sur madoc dans une archive `.tar.gz` ou `.zip`. Le contenu du rapport devra impérativement décrire :

– La preuve de l'affirmation qui apparaît deux fois en gras dans le texte. Plus précisément, en résolvant un problème ne contenant qu'un sous-ensemble des contraintes (3'), si on obtient une solution composée d'un seul cycle alors cette solution est optimale.
– La récupération de la solution optimale des problèmes résolus par GLPK et sa traduction en permutations, puis en produit de cycles disjoints,
– Une analyse expérimentale de la méthode de résolution exacte, effectuée à partir des instances fournies (temps CPU, nombre de contraintes ajoutées pour résoudre le problème...). On considérera séparément les instances de la catégorie `plat` et `relief`. Lesquelles sont les plus difficiles à résoudre avec l'algorithme proposé, pourquoi ?
– Une analyse expérimentale de la méthode de résolution approchée (temps CPU, qualité de solution obtenue). Pour ce type de méthode, on ne considèrera que les instances de la catégorie `plat`.
– Éventuellement, des améliorations pourront être proposées suite à ces analyses expérimentales. Si des étudiants finissent vite ce projet, les enseignants pourront faire des propositions.