PROJET
Les bicyclettes de Triville

Optimisation multi-objectif et circulation douce

Présentation

Le projet de ce semestre en théorie des graphes vous permettra d'utiliser certains des algorithmes classiques vus en cours, et de découvrir un domaine immense de l'informatique appliquée qui est celui de l'optimisation, et en particulier la prise en compte de plusieurs aspects des conséquences des choix effectués lors d'une recherche de solutions optimales. Ce sujet présentera rapidement les concepts utiles à la bonne compréhension des enjeux et des techniques que vous allez appliquer, suivi d'une illustration détaillée sur un cas simple. La 2ème partie décrira le travail à réaliser décomposé en grandes étapes, le barème de la progression, ainsi que quelques conseils et indications.

Concrètement le projet sera d'étudier des approches pour optimiser la mise en place d'arêtes sur un réseau de circulation à construire. Il s'agit donc d'une thématique proche de celle des arbres couvrants de poids minimal. Mais pour aller plus loin qu'un simple Prim ou Kruskal il sera demandé d'optimiser simultanément 2 coûts en partie contradictoires, ou un coût (financier) et un bénéfice (en terme d'usage). Le prétexte qui sert d'enrobage est la mise en place de pistes cyclables dans un plan de circulation urbain pré-existant. Les techniques d'optimisation multi-objectif étant assez difficiles à étudier à votre niveau, cette application concrète sera présentée initialement dans une version extrêmement simplifiée (pas de sens uniques etcetera). Il vous sera possible d'introduire des aspects plus réalistes dans le modèle à la partie « Extensions à la carte » sur 6 points.

Cette année le projet théorie des graphes est très orienté « recherche et développement » car au-delà de la seule programmation en C++ d'une base de code répondant aux questions, vous devrez développer des méthodes de travail proches de celles d'un chercheur : documenter le travail, les hypothèses, les tests, les résultats, organiser et archiver ces informations, et enfin en donner une synthèse lors de la soutenance. La bibliographie restera facultative. Vous aurez un délais supplémentaire après la « semaine piscine » pour mettre en forme ces éléments, mais vous n'aurez aucune matière à mettre en forme si vous n'avez pas fait ce travail d'archivage et d'organisation des données **pendant** la semaine de développement. **Organisez vous dès le départ pour tenir à jour un cahier de laboratoire**, utilisez un outil simple et robuste, par exemple un cahier papier physique ou un document partagé (google docs ou autre).

1) Théorie, concepts, outils

a) Optimisation

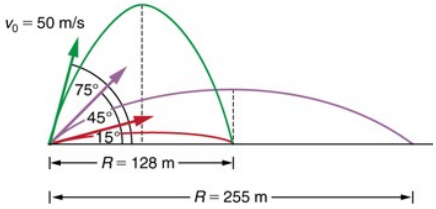
Dans ce module de théorie des graphes nous avons souvent vu l'objet mathématique « Graphe » avec son ensemble de sommets et son ensemble d'arêtes ou d'arcs comme un objet d'optimisation :

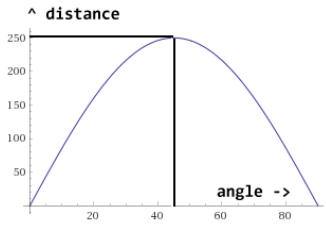
→ nous avons optimisé son utilisation : un graphe étant donné, comment minimiser le nombre de couleurs des sommets (coloration, Welsh et Powell), comment minimiser le parcours d'un sommet à un autre (BFS, Dijkstra), comment maximiser son flot (Ford-Fulkerson)

→ nous avons optimisé un graphe : sur la base d'un ensemble d'arêtes possibles, comment les choisir pour minimiser un coût total, arbre couvrant de poids minimal (Prim ou Kruskal)

Une recherche d'optimal implique d'être libre de certains choix, de pouvoir ajuster des variables. L'**ensemble des variables** sur lesquelles on peu agir pour améliorer une certaine mesure résultant des ces choix définit ce qu'on appelle l'**espace de décision**. Ces variables sur lesquelles on agit sont les **variables de décision**. Par ailleurs on dispose d'une **fonction objectif** qui a chaque choix de valeur(s) pour la ou les variables de décision va donner une valeur de coût ou de performance à minimiser ou maximiser (selon le problème). Cette valeur donnée par la fonction objectif en fonction des valeurs des variables de décision définit l'**espace des objectifs**.

Exemple : on nous donne un canon qui tire un projectile à 50 m/s (ceci est fixé, dans ce problème ce n'est pas une variable de décision). On peut régler l'angle du tir (c'est une variable de décision). On souhaite maximiser la distance parcourue par le projectile : la fonction objectif est la distance du projectile en fonction de l'angle, et on cherche à maximiser cette distance. Ci dessous sur la vignette à droite on voit la fonction objectif, l'axe des abscisses (angle) est l'espace de décision, l'axe des ordonnées (distance) est l'espace des objectifs. On voit clairement que la décision optimale qui maximise la distance est de choisir angle = 45°. Mais il n'est pas toujours possible de représenter graphiquement le problème...





Exemple : on nous donne à construire un barrage et une central hydroélectrique sur un cours d'eau. La production électrique à atteindre a été définie à 5MW. On cherche à minimiser le coût de construction : le coût constitue l'espace des objectifs. Les variables de décisions sont : type barrage-voûte ou à contreforts, l'épaisseur et la hauteur du barrage, les % de sable de ciment et de gravier dans le béton, le type de turbine (Francis, Kaplan, Pelton), la géométrie des conduites d'eau etc … Ces différents choix correspondent à l'espace de décision. On voit sur cet exemple que l'espace de décision est souvent multidimensionnel (de nombreux axes) et pas forcément continu (il peut y avoir des variables de décision à choix binaire, à choix ternaire…). Toutes les combinaisons n'ont pas forcément de sens : par exemple certains choix conduisent à un barrage trop fragile. Les variables de décisions doivent donc respecter un certain nombre de contraintes définies par des intervalles et des inégalités. On dit d'une solution qu'elle est admissible si elle respecte ces contraintes. L'ensemble des solutions admissibles de l'espace de décision définit l'espace de recherche (search space). Dans un cas complexe comme celui de cet exemple il n'est en général pas possible de donner une visualisation complète de cet espace, on devra se contenter de coupes ou de projections ou de listes partielles de solutions.

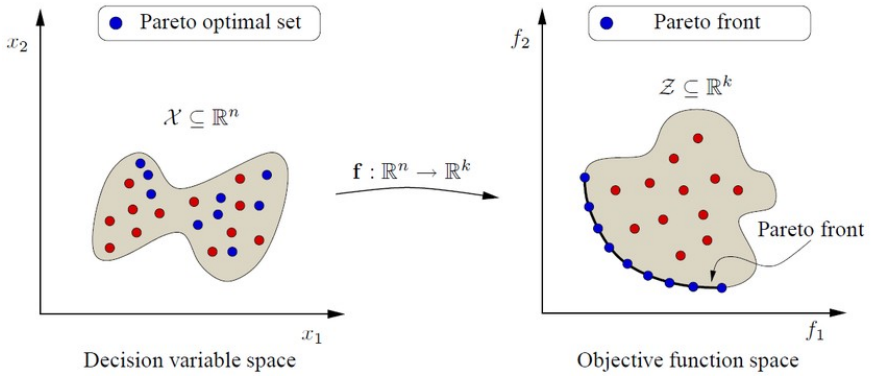
Dans certains cas on dispose d'une méthode (algorithme ou approche analytique) permettant de trouver directement une solution optimale. C'est par exemple ce qui se passe avec la minimisation de la somme des arêtes prises pour constituer un arbre couvrant de poids minimal : Kruskal ou Prim nous garantissent une solution optimale. Dans d'autres cas il existe des algorithmes approchés qui donnent une bonne solution en un temps raisonnable mais pas forcément la meilleure (par exemple Welsh et Powell, avec comme variables de décision la couleur donnée à chaque sommet). Une alternative pour trouver une solution optimale quand aucun algorithme efficace n'est connu consiste à étudier systématiquement toutes les combinaisons, mais ces approches « brute force » sont limitées à des problèmes de petite taille.

b) Optimisation multi-objectif

Pour reprendre l'exemple du barrage et de la centrale hydroélectrique, il peut sembler réducteur sur un gros chantier complexe de se focaliser uniquement sur un seul facteur à optimiser absolument (le coût de construction). On comprend qu'à coût de construction équivalent il peut exister des solution plus ou moins mauvaises au niveau environnemental (coût écologique) mais aussi au niveau de l'entretien (coût de fonctionnement) au niveau de l'efficacité (rendement d'utilisation de l'eau disponible). Il se pourrait par exemple que pour un coût de construction plus élevé de 10 % on a un coût de fonctionnement baissé de 20 % et en plus un coût écologique divisé par 2. Avec une optimisation mono-objectif on va passer à côté de ces bons compromis.

L'optimisation **multi-objectif** (on dit aussi **multi-critère**) consiste à prendre en compte plusieurs fonctions objectifs à optimiser « en même temps ». L'espace des objectifs devient alors un espace à 2, 3 … n dimensions. On pourrait se dire qu'il suffit de projeter l'ensemble de ces objectifs sur une seule valeur de synthèse, par exemple avec une combinaison linéaire, et d'utiliser cette valeur de synthèse pour se ramener au cas et aux méthodes de l'optimisation mono-objectif. Par exemple on aurait coûtGlobal = 0.33 coûtConstruction + 0.33 coûtEcologique + 0.33 coûtFonctionnement mais l'expérience montre que les solutions obtenues varient de façon discontinue selon le choix des coefficients (trouver les bons coefficients devient un problème supplémentaire) et que de façon générale il existe de bonnes solutions que cette approche par projection monodimensionnelle ne permet pas d'obtenir. Le formalisme qui permet d'aborder cette situation de façon satisfaisante est l'**optimum de Pareto**. https://fr.wikipedia.org/wiki/Optimum_de_Pareto

Dans la suite nous considérons des optimisations bi-objectif avec seulement 2 fonctions objectifs. L'espace des objectifs est alors un espace à 2 dimensions et on peut représenter les performances de l'ensemble des solutions admissibles comme un nuage de points avec la fonction objectif 1 en abscisses et la fonction objectif 2 en ordonnées.



La frontière de Pareto est l'ensemble des solutions qui sont des **optimums de Pareto** c'est à dire qui sont **non dominées** dans l'espace des objectifs. **Une solution est non dominée si elle est meilleure que chaque autre solution sur au moins un objectif (selon l'objectif 1 ou l'objectif 2).**

Il peut arriver dans certains cas qu'il n'existe qu'un seul optimum de Pareto (cas dégénéré) et alors les objectifs se confondent : par exemple la solution la plus économique est aussi la plus écologique. Mais souvent la nature intrinsèque du problème rend les 2 objectifs en partie antagonistes : la meilleurs solution économique ne sera pas la meilleure écologiquement, et réciproquement. Ceci correspondra aux 2 sommets extrêmes de part et d'autre de la frontière de Pareto (voir les sommets bleus respectivement le plus à gauche et le plus en bas figure précédente).

Ce qui va nous intéresser ce sont les compromis intermédiaires, si ils existent. Mais nous voulons évacuer les solutions dominées, celles qui sont plus mauvaises qu'au moins une autre solution sur les 2 critères à la fois. À la figure précédente, pour chaque chaque solution dominée (indiquée en rouge) il existe une solution en bleu qui est meilleure (plus faible) à la fois en abscisse et en ordonnée.

En dégageant les optimums de Pareto (la frontière de Pareto est constituée de ces optimums) on est alors en mesure de proposer aux décideurs (commités, élus, commanditaires, usagers…) l'ensemble des **bons compromis**, qui ne peuvent être amélioré sur l'un des aspects qu'au détriment de l'autre aspect, pas sur les 2 à la fois. Contrairement à l'optimisation mono-objectif qui se présente comme donnant **la** meilleure réponse unique, algorithmique, mécanique et non discutable à un problème, l'optimisation multi-objectif offre une **palette** de réponses permettant ensuite aux décideurs d'opérer un choix rationnel en comparant toutes les possibilités non dominées.

c) Recherche des solutions Pareto optimales

Si on est en mesure d'énumérer **toutes** les solutions admissibles (couvrir l'espace de décision) et de calculer les fonctions objectifs pour chacune alors il est simple de réaliser un filtrage pour ne garder que les solutions non dominées : par comparaison 2 à 2 entre toutes les solutions. Cette approche « brute force » est celle qui sera principalement retenue pour le travail demandé : elle est assez simple à implémenter, et nous garantit de trouver des optimums donc offre une visibilité parfaite sur ce qui se passe. Elle a le défaut d'être « combinatoire » donc peu efficace. Elle ne sera possible que pour des petits problèmes, pour des problèmes de taille plus grande les temps de calcul et les quantités de mémoire deviennent rédhibitoires.

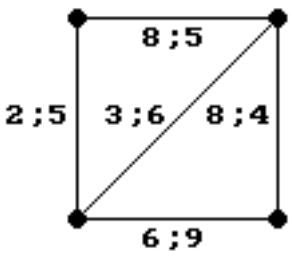
Dans la partie « Extensions à la carte » nous présenterons des alternatives permettant d'aller plus loin et d'aborder des problèmes de plus grande taille avec des approches efficaces mais généralement non optimales : ces approches ne nous permettront pas de trouver toutes les solutions Pareto optimales, ou alors elles trouveront des solutions approchées à la frontière de Pareto.

2) Exemple concret d'optimisation bi-objectif sur un arbre couvrant

Les mécanismes présentés sur cet exemple correspondent grosso-modo à la moitié des points du projet : voir ci-après chapitres 1) et 2) de la partie « Travail à réaliser »

a) Un graphe à double pondération, les objectifs

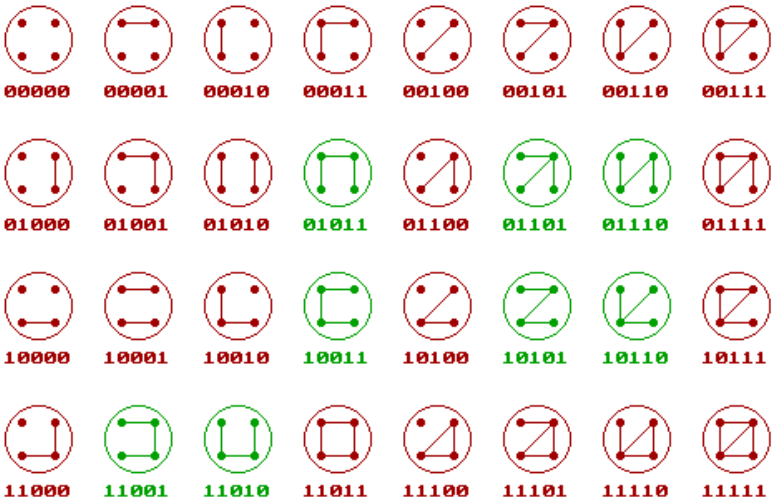
Nous allons partir d'un graphe de circulation existant, et nous cherchons à équiper une partie des rues (supposées non orientées) en pistes cyclables. Pour chaque rue nous avons un coût financier d'installation, et un coût environnemental (perte de fluidité pour les voitures, retards des bus, arbres à abattre …)
Par exemple sur l'arête de gauche le coût1 est de 2 et le coût2 est de 5.



Nous allons faire les hypothèses de travail suivantes (simplificatrices) : nous voulons un réseau cyclable connexe, et nous cherchons à minimiser le coûtTotal1 de la somme des coût1 des arêtes cyclables et en même temps minimiser le coûtTotal2 de la somme des coût2 de ces mêmes arêtes cyclables. Ces 2 problèmes pourraient être traités séparément à l'aide de 2 Prim ou 2 Kruskal qui nous donneraient l'arbre couvrant de poids minimal au sens du coût1 puis du coût2, mais ceci ne nous donnerait pas les (éventuels) bons compromis intermédiaires.

b) Recherche de toutes les solutions admissibles, approche brute force

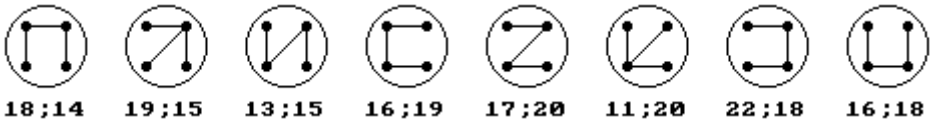
L'espace de décision est ici constitué de 5 variables booléennes : pour chacune des 5 arêtes du graphe, soit on la rend cyclable soit on ne la rend pas cyclable. C'est une combinatoire de $2^5 = 32$ cas à étudier. Vous voyez ici qu'il serait difficile de représenter cet espace de décision autrement que sous forme de liste :



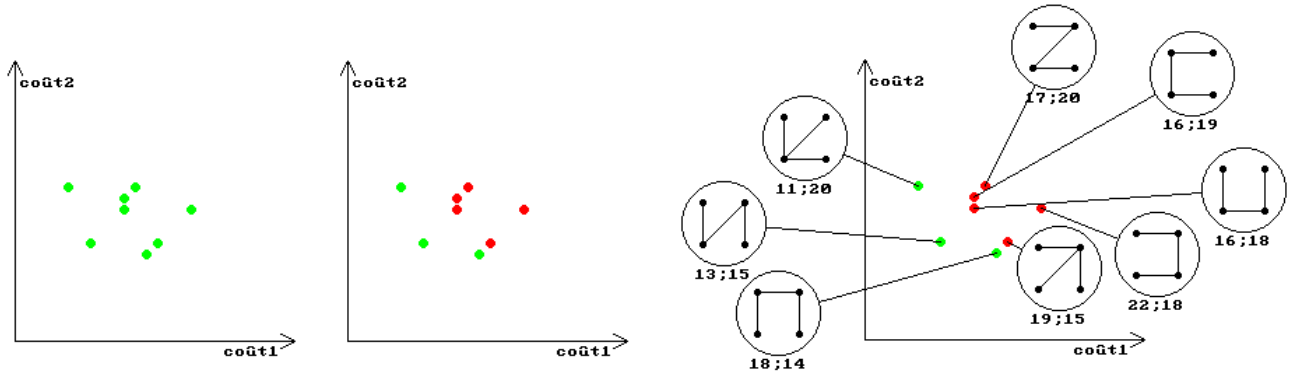
Il s'agit d'une énumération binaire. À partir de cette énumération des solutions on va pouvoir dégager les solutions admissibles (en vert sur la figure ci dessus) qui correspondent aux arbres couvrants. On peut procéder en 2 étapes, d'abord éliminer les combinaisons qui n'ont pas exactement (ordre-1) arêtes, ici il y en a 10, puis vérifier celles qui sont connexes (ce qui élimine 00111 et 11100). Ceci nous donne donc **l'intégralité de l'espace de recherche**.

c) Évaluation des objectifs et détermination de la frontière de Pareto

Pour chaque solution admissible de l'espace de recherche on calcule coûtTotal1 et coûtTotal2 (simplement en sommant les coûts des arêtes retenues, pas question ici de faire un Prim ou un Kruskal). Sans ordre particulier voici les résultats.

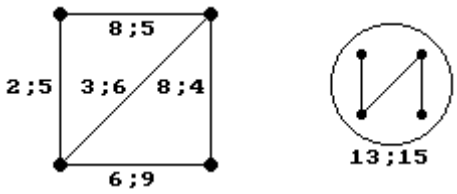


On retrouve bien l'arbre couvrant de poids minimal au sens de coût1 que nous donnerait Prim ou Kruskal uniquement avec les coût1 : (11;20). De même avec les coût2 : (18;14). On constate bien que les 2 objectifs ne peuvent pas être optimalement tenus simultanément : pas de (11;14). Représentons ces couples d'évaluations d'objectifs sur un diagramme 2D de l'espace des objectifs, nous y verrons plus clair. Voir page suivante.



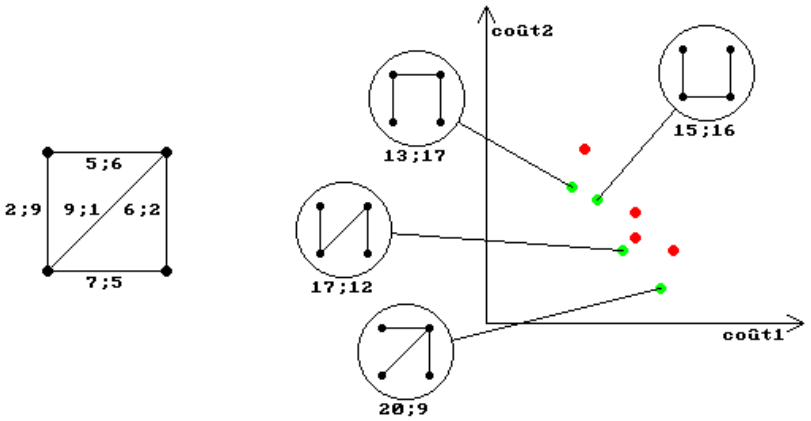
À gauche les évaluations pour les solutions admissibles. Au centre on a éliminé (marqué en rouge) les **solutions dominées** et on a gardé (marqué en vert) les **solutions non dominées** qui correspondent aux optimums de Pareto (frontière de Pareto). Par exemple la solution (19;15) est éliminée de l'ensemble Pareto optimal car elle est dominée : il existe une autre solution (18;14) qui est **aussi bonne en chaque objectif et strictement meilleure sur un objectif** (en l'occurrence elle est strictement meilleure sur tous).

À droite on retrouve la description complète des solutions correspondantes. On constate qu'il existe un compromis intéressant que ni Prim ni Kruskal séparément sur coût1 ou coût2 n'auraient pu trouver, c'est la solution avec l'évaluation (13;15). Elle est à peine plus coûteuse que (18;14) sur le 2ème critère mais réalise une économie substantielle sur le 1er critère. Ce n'est pas **la** meilleure (il n'y a pas de meilleure parmi les 3 Pareto optimales) mais il se peut qu'elle soit au final retenue par les décideurs. L'algorithme présente un **ensemble** de bonnes solutions.

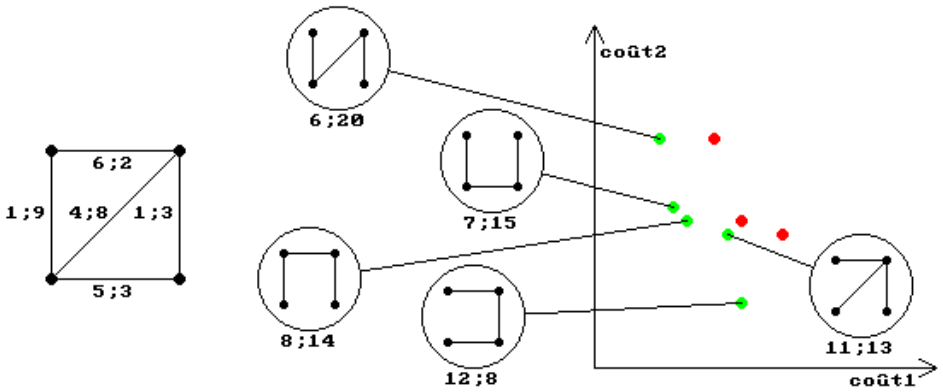


d) Frontière non convexe et optimums non accessibles linéairement

Nous n'allons pas entrer dans les détails mathématiques mais on peut faire la remarque que les solutions Pareto optimales qui sont sur l'enveloppe convexe de la frontière peuvent être obtenu par optimisation mono-objectif d'une combinaison linéaire « bien choisie » des 2 fonctions objectifs. On peut même ajouter que dans le cas très particulier d'optimiser un arbre couvrant sur 2 critères de poids (double pondération) comme illustré précédemment, on pourrait s'en sortir en transformant le problème en arbre couvrant de poids minimal avec comme poids unique de chaque arête une combinaison linéaire « bien choisie » des 2 poids du problème initial. Vous pouvez facilement vérifier sur l'exemple ci-dessus qu'en faisant pour chaque arête une moyenne des poids, et en faisant ensuite un seul Kruskal ou Prim, on retrouve l'arbre couvrant du compromis médian (13;15). En faisant « varier » les coefficients de la combinaison linéaire on trouverait tous les optimums de Pareto de l'enveloppe convexe. Mais toutes les solutions Pareto optimales ne sont pas automatiquement sur l'enveloppe convexe. Voir sur le graphe page suivante avec d'autres poids.

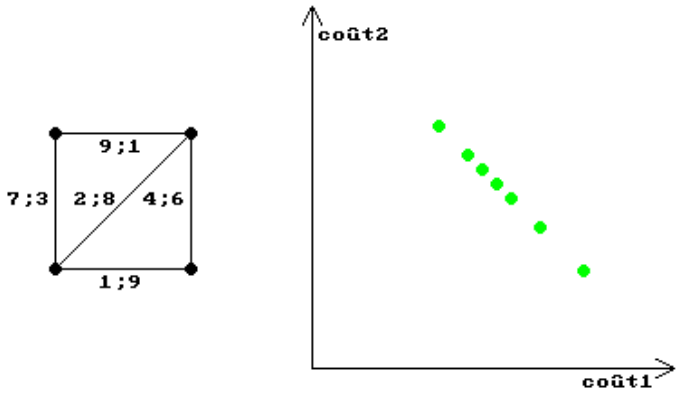


Ici, sur le même graphe mais avec d'autres poids, la solution Pareto optimale (15;16) ne peut pas être obtenue en rendant le problème mono-dimensionnel par combinaison linéaire des coûts car elle n'est pas sur l'enveloppe convexe de la frontière. Nous ne voulons pas passer à côté de ces solutions. Idem sur le cas étudié ci-dessous (encore des poids différents) pour l'optimum (11;13)...



e) Effets de corrélation des 2 pondérations

Si les 2 pondérations sont corrélées positivement (pour chaque arête quand coût1 est grand c'est aussi le cas pour coût2) alors le problème dégénère et se ramène effectivement à un problème mono-dimensionnel. La frontière de Pareto se réduit à un seul optimal qui domine tous les autres. À l'inverse si les poids sont anti-corrélés (pour chaque arête coût1+coût2 est à peu près le même) alors toutes les solutions sont sur la frontière de Pareto, les critères à optimiser étant parfaitement antagonistes, voir figure ci-dessous. On évitera d'étudier ces cas limites : les poids seront tirés aléatoirement indépendamment.



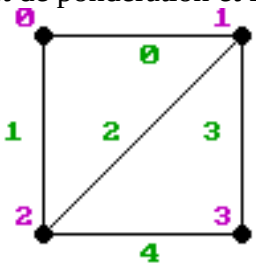
Travail à réaliser

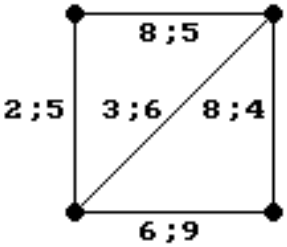
Le travail à réaliser correspond au développement d'une base de code C++ qui servira à répondre aux questions posées. Ce projet n'est pas constitué uniquement que de code mais aussi des résultats obtenus, archivés, organisés, et des remarques, calculs, analyses que vous pourrez faire en parallèle du développement logiciel proprement dit.

1) Entrées/sorties, représentations, arbres couvrants mono-objectif 5 points

Après avoir attentivement lu l'intégralité de l'énoncé et choisi le modèle de structures de données qui vous semble le plus adapté (pas juste pour répondre aux 1ères questions mais aussi pour la suite), vous mettrez en place le chargements de fichiers descripteurs de graphes (topologie) et de poids (pondération). Les formats de fichiers sont **imposés**, votre code devra pouvoir lire des fichiers qui seront spécifiquement conçus pour la soutenance (que vous n'aurez pas encore vus). Un ensemble de fichiers exemples est fourni, naturellement vous pouvez développer les vôtres selon vos besoins. Les fichiers sont donnés séparément pour les topologies (sommets et arêtes) et pour les poids : sur une même topologie on pourra étudier plusieurs pondérations (voir précédemment dans le présentation). **Lien vers les fichiers de test : http://files.ece.fr/~fercoq/graphes/files.zip**

Exemple de graphe et de pondération et fichiers correspondants :





Fichier « **broadway.txt** »

```
4
0    100  100
1    200  100
2    100  200
3    200  200
5
0    0    1
1    0    2
2    1    2
3    1    3
4    2    3
```

Fichier « **broadway_weights_0.txt** »

```
5    2
0    8.0    5.0
1    2.0    5.0
2    3.0    6.0
3    8.0    4.0
4    6.0    9.0
```

Explication du format de fichier de topologie :

Les sommets sont repérés par un indice à partir de 0 (voir en **violet** sur la figure)
Les arêtes sont repérées par un indice à partir de 0 (voir en **vert** sur la figure)

- Le fichier commence par le nombre de sommets (sur cet exemple 4 sommets)
- Suivent pour chaque sommet, sur chaque ligne :
  - l'indice du sommet (cette information est redondante, on les aura toujours dans l'ordre)
  - l'abscisse du sommet
  - l'ordonnée du sommet
    Ces coordonnées sont indicatives, elles peuvent servir pour dessiner les graphes mais ne jouent aucun rôle dans les déterminations des algorithmes d'optimisation

- Ensuite le nombre d'arêtes (sur cet exemple 5 arêtes)
- Suivent pour chaque arête, sur chaque ligne :
  - l'indice d'arête (cette information est redondante, on les aura toujours dans l'ordre)
  - l'indice du premier sommet relié par l'arête
  - l'indice du deuxième sommet relié par l'arête

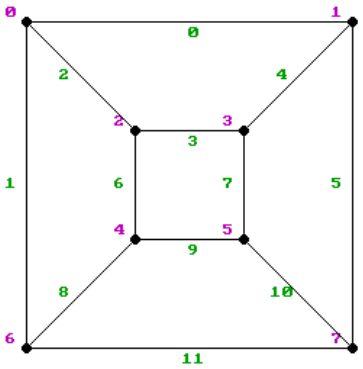
Le format correspond à la description de graphes non orientés, l'ordre de liaison des sommets n'est pas important (dans les fichiers fournis le sommet d'indice le plus faible vient en 1er). *Le format pourrait facilement être étendu (voir Extensions à la carte) pour représenter des graphes orientés, dans ce cas l'ordre des sommets est important, et une liaison réciproque nécessite 2 arcs distincts.*

Explication du format de fichier de poids :

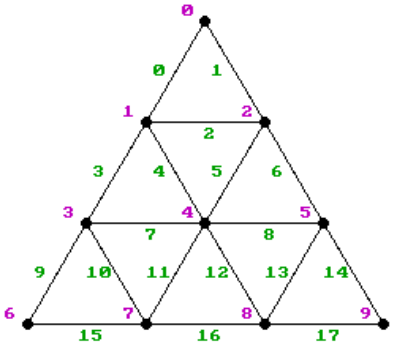
Les arêtes sont repérées par un indice à partir de 0 (voir en **vert** sur la figure)
Même si pour des raison de lisibilité les exemples n'utilisent que des entiers, **les poids sont des nombres flottants et votre code doit faire en sorte de gérer des nombres flottants** (il n'est sans doute pas nécessaire d'utiliser des double, des float suffisent).

- Le fichier commence par le nombre d'arêtes (sur cet exemple 5 arêtes)
- Sur la même ligne on a le nombre de pondération (sur cet exemple 2 poids par arêtes)
  Pouvoir gérer d'autres valeurs que 2 n'est pas exigé *(voir Extensions à la carte)*
- Suivent pour chaque sommet, sur chaque ligne :
  - l'indice d'arête (cette information est redondante, on les aura toujours dans l'ordre)
  - les 2 poids (ou plus) de l'arête

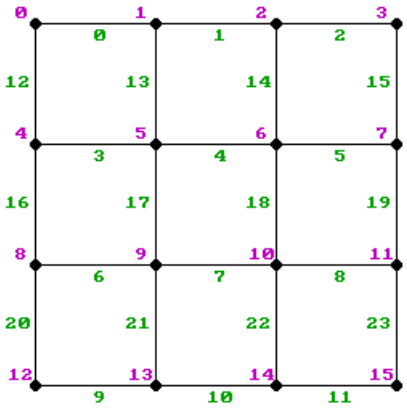
Voici les autres topologies proposées, par ordre croissant :



cubetown



triville



manhattan

Sur la base des ces fichiers vous coderez l'obtention des arbres couvrants de poids minimal selon les 2 objectifs séparément (optimiser pour coût1 ou pour coût2) avec Prim ou Kruskal. Les arbres résultats seront donnés

→ Soit à la console, faites comme vous pouvez pour que ce soit lisible
→ Soit sous forme de fichier vectoriel (voir les codes du 1er semestre : svgfile... est utilisable)
→ Soit sous forme d'affichage bitmap avec Allegro
→ Soit sous forme d'affichage graphviz en passant par le format de fichier dot intermédiaire
Voir http://www.webgraphviz.com/ ou https://dreampuf.github.io/GraphvizOnline/
https://cyberzoide.developpez.com/graphviz/ positionner etc... faites vos recherches

Votre programme indiquera également, si possible sur la même représentation que l'arbre, les 2 coûts totaux pour chaque arbre. Avec broadway.txt et broadway_weights_0.txt on doit retrouver les 2 solutions (11;20) et (18;14) de la figure en haut de la page 6.

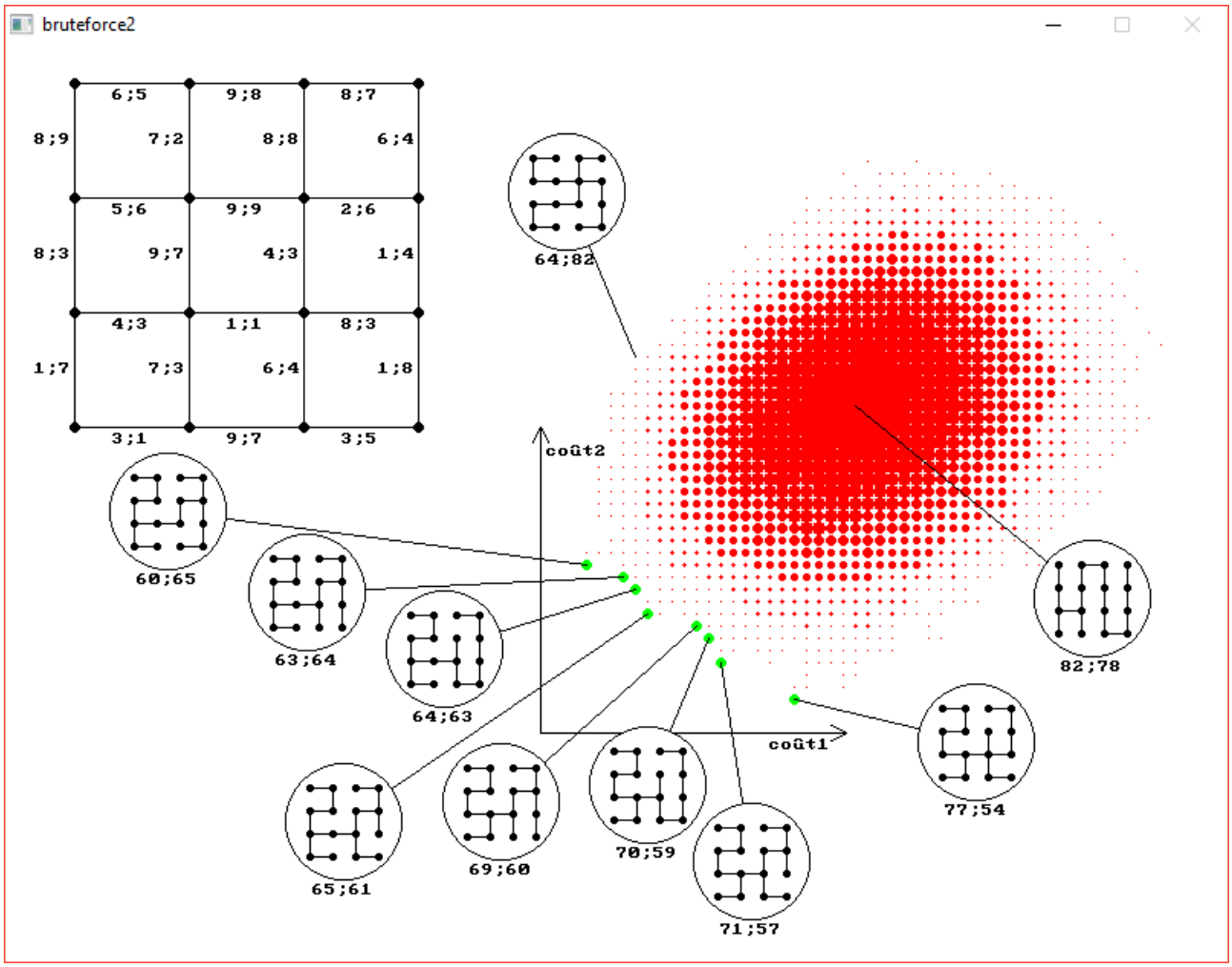
2) Double pondération, optimisation arbre couvrant bi-objectif 5 points

Ce chapitre de votre projet correspond à la démarche abondamment détaillée et illustrée à l'exemple concret donné à la partie présentation (pages 4 à 7) : sur la base d'un graphe arbitraire et de pondérations arbitraires correspondants à des fichiers en entrée, vous coderez une approche brute force avec énumération des solutions, filtrage des solutions admissibles, évaluations des solution selon les 2 objectifs, détermination des solutions non dominées (frontière de Pareto) et représentation de toutes les solutions dans l'espace des objectifs, avec une indication d'appartenance à la frontière (couleur et/ou forme différente).

Pour la partie représentation vous pouvez utiliser une des solutions ci-dessus (console, format svg, Allegro, graphviz) ou éventuellement un logiciel type tableur (excel, libreoffice calc) qui permet de réaliser des nuages de points XY (*scatter plot*). Il n'est pas demandé d'avoir une représentation avec vignettes illustrant les arbres couvrant associé aux solutions directement sur le même diagramme, **il n'est pas nécessaire ni demandé ni exigé de reproduire les schémas des exemples pages 6 et 7**. Mais vous devrez pouvoir présenter les arbres couvrants associés aux solutions Pareto optimales d'une façon ou d'une autre, par exemple avec un système d'indices : sur le diagramme de l'espace des objectifs vos points sont numérotés et on retrouve une liste numérotée des graphes à coté ou sur un document séparé.

Au niveau des choix d'implémentation, mesurez la quantité de traitements à faire. Le graphe « manhattan » (voir figure en bas à droite page 9) peut tourner en moins d'une seconde entre le chargement du fichier et la livraison des résultats, sur un PC modeste.

→ Quelle est la taille de l'espace de décision de ce graphe ?
→ Peut-on facilement trouver un meilleur majorant au nombre de solutions admissibles en utilisant une formule bien connue en combinatoire ?
→ Faites une estimation de la quantité de mémoire liée au stockage d'autant de solutions en fonction du type de représentation de votre graphe...
→ Suggestion : ne dupliquez pas un objet graphe pour chaque solution, prévoyez de stocker les solutions « à côté » du graphe, sous forme de **vecteurs de booléens** (1 bool = 1 bit)



manhattan.txt avec manhattan_weights_0.txt

Cette représentation graphique n'est pas contractuelle, vous n'êtes pas obligé de faire pareil. Mais votre programme doit bien retrouver les mêmes 8 solutions Pareto optimales. Ici on a donné aux solutions dominées (en rouge) une taille correspondant à leur densité : énormément de solutions admissibles atterrissent au centre d'une zone de mauvaises performances. Les bonnes solutions, proches de la frontière de Pareto, sont rares.

Conseil sur l'implémentation de la détermination efficace de la frontière de Pareto : la méthode suggérée dans la présentation, comparer 2 à 2 les solutions et éliminer les solution dominées pourrait faire croire qu'on a un algorithme quadratique. Dans le cas particulier de poids anti-corrélés (page 7 en bas) toutes les solutions sont Pareto optimales et donc il n'y en a aucune d'éliminée : dans ce cas l'algorithme est bien quadratique. *Questions : avec le graphe « manhattan » combien de passes doit effectuer un algorithme quadratique ? Quel est la borne inférieure du temps d'exécution sur une machine à 3GHz ? Est-ce supérieur ou inférieur au temps d'une soutenance ?*

Mais dans un cas de poids aléatoires non corrélés, comme vous pouvez le voir sur le nuage de points ci dessus, de très nombreuses solutions sont nettement dominées. Passez en revue chaque sommet et pour chacun de ces sommets passez en revue tous les autres et éliminez directement ceux qui sont dominés. Dès qu'un bon sommet est trouvé l'ensemble est décimé, plus on avance plus il devient facile d'avancer : la boucle externe se raccourci et la boucle interne aussi. Au final on se retrouve typiquement avec une complexité linéaire ou log-linéaire (à déterminer… voir Extensions)

Or il se trouve que les conteneurs ensemblistes et associatifs du C++ peuvent effacer efficacement des éléments d'un ensemble en train d'être parcouru sans avoir à interrompre ou reprendre le parcours. **Mais ceci nécessite quelques précautions et une tournure de code spéciale**. Compte tenu du caractère centrale et critique de la double boucle d'élimination des solutions dominées (détermination de la frontière de Pareto) je vous recommande chaudement de vous plonger dans les exemples de code donnés ci après. Regardez très attentivement ce qui est fait ou pas fait à l'itérateur, en particulier dans la boucle for (idem pour les ordered : std::map et std::set)

https://en.cppreference.com/w/cpp/container/unordered_map/erase
https://en.cppreference.com/w/cpp/container/unordered_set/erase

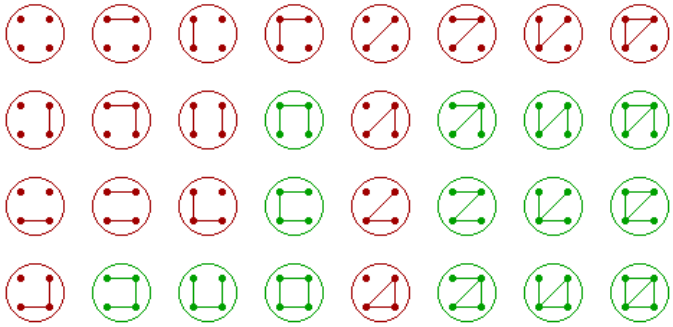
3) Optimisation bi-objectif coût / distances 4 points

Vous avez maintenant la maîtrise du processus d'optimisation à 2 objectifs mais avec un seul type d'aspect du graphe utilisé 2 fois. Notre frontière de Pareto s'étendait d'un résultat d'arbre couvrant de poids minimal à un autre arbre couvrant de poids minimal. Nous allons maintenant essayer d'avoir 2 objectifs de natures différentes : le 1er restera un coût total à minimiser, le second sera le temps de parcours pour joindre un sommet (quelconque) à un autre (quelconque).

Ce 2ème critère correspond à l'optimisation d'une valeur d'usage du graphe partiel qu'on met en place. Pour déterminer à partir d'une solution admissible la valeur de cet objectif on devra calculer la somme des distances entre chaque couple de sommets. Il est clair que, du point de vue de ce seul critère et en faisant totalement abstraction de l'autre objectif (minimiser le coût de construction) une solution optimale serait de prendre toutes les arêtes du graphe de départ. Ceci correspondrait évidemment à la solution la plus coûteuse. On voit bien sur cet exemple qu'on a deux solutions antagonistes et donc probablement des solution intermédiaires pouvant correspondre à de bons compromis à présenter aux décideurs.

Questions : est il possible d'avoir une situation dans laquelle prendre toutes les arêtes du graphe de départ ne constitue pas une des extrémités de la frontière de Pareto ? Si oui exhiber un exemple, si non dire pourquoi.

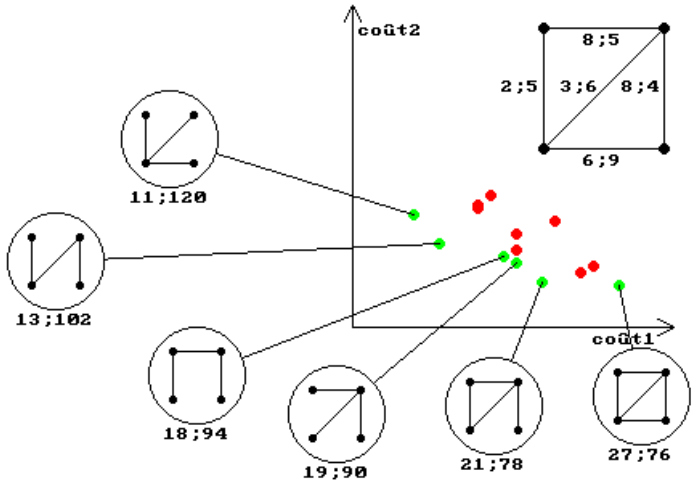
Nous voulons inclure comme solutions admissibles des graphes partiels couvrants connexes qui ne sont pas tous des arbres : en effet l'introduction de cycles dans le réseau cyclable, outre qu'elle semble naturelle, permet certainement d'optimiser les temps de trajets, au détriment du coût total mais nous sommes là pour en discuter (étudier tous les compromis non dominés). Donc nous sommes amenés à compter des temps de parcours, ce qui est équivalent à évaluer des distances, sur un graphe avec cycles : vous n'avez pas d'autre choix que d'utiliser un algorithme bien connu qui détermine les plus courts chemins sur un graphe pondéré, et ce pour chaque couple de sommets.



Est-ce bien raisonnable ? Pouvez vous donner une évaluation très approximative du nombre d'opérations élémentaires à réaliser pour chacuns des 4 fichiers de topologies proposés ? En supposant une implémentation naïve quadratique de l'algorithme bien connu ? Et en supposant une implémentation log-quadratique de l'algorithme bien connu (avec priority_queue) ? Une conclusion est-elle possible à ce stade ? Va-t-il falloir dégainer A ? Quel en serait l'heuristique ?*

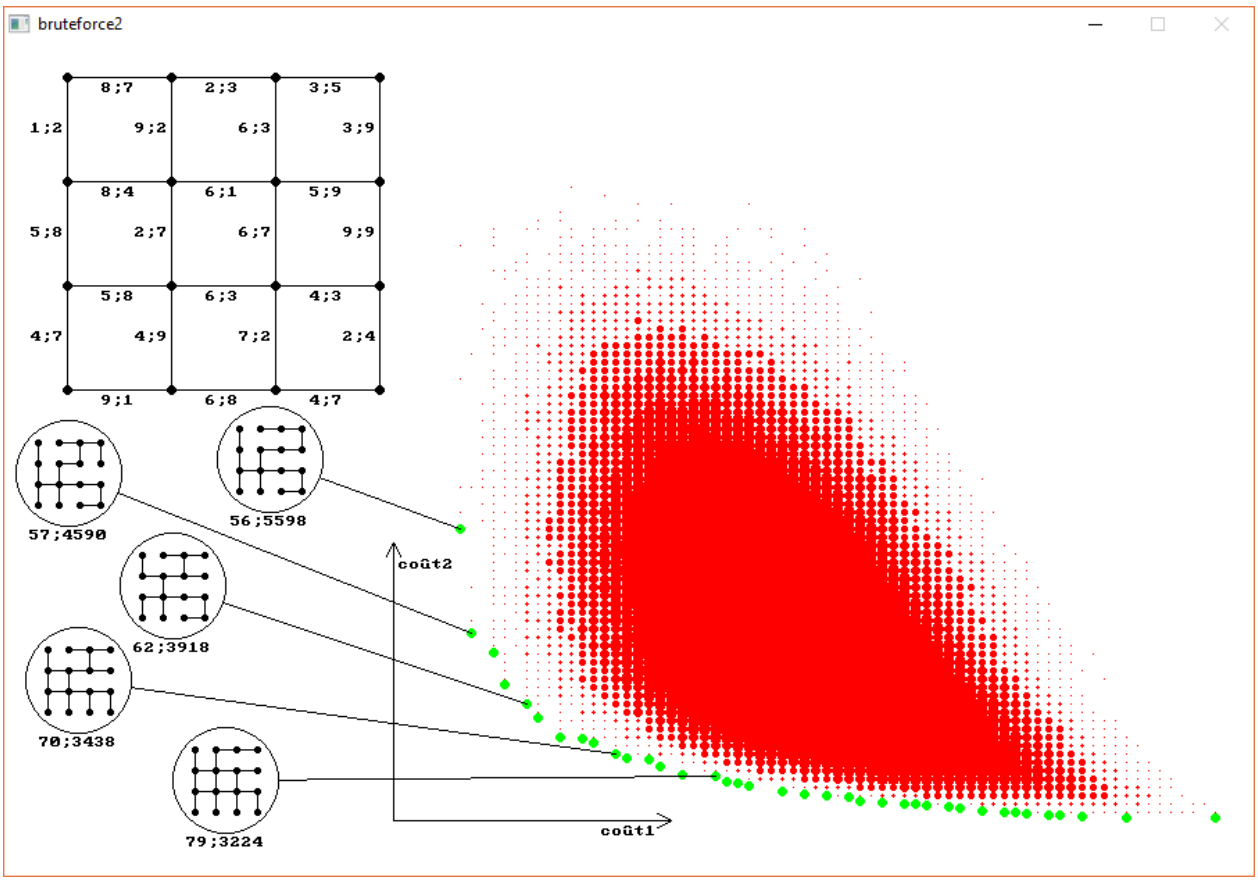
On vous fait remarquer qu'il n'est pas nécessaire de relancer un nouveau Dijkstra pour chaque couple de sommet mais « seulement » pour chaque sommet, puisqu'un Dijkstra déroulé complètement donne automatiquement un arbre couvrant des plus courts chemins partant de ce sommet de départ, et donc donne déjà n distances (les distances partielles à la fin de l'algo quand tout le monde est marqué). Cela change-t-il les conclusions précédentes ?

Implémenter tout ça et comparer avec vos prédiction, compter le nombre de passages dans les boucles les plus internes.



Ci contre à gauche :
broadway et broadway_weights_0
en abscisses = somme des coût1 des arêtes
en ordonnées = somme des temps de parcours de sommet à sommet (coût2).

Ci dessous :
manhattan et manhattan_weights_1



4) Extensions à la carte
6 points max

Il peut sembler frustrant de devoir s'arrêter à des graphes de petites dimensions, mais l'approche brute-force a ses limites. Son intérêt repose sur sa capacité à explorer complètement un espace de recherche et à pouvoir vérifier et étalonner des algorithmes efficaces mais approximatifs qui pourraient prendre la relève sur des graphes de plus grande taille.

Si vous explorez la littérature sur le sujet vous verrez que l'optimisation multi-objectifs et en particulier sur les graphes est un thème de recherche récent et encore actif. Si vous trouvez un algorithme qui vous semble abordable vous pouvez essayer de l'implémenter, ou de montrer de façon convaincante au jury que vous avez compris un aspect important. Attention la lecture d'articles de recherche peut être un gouffre temporel dont on ne sort pas facilement.

Plus classiquement, seront retenus comme « extensions » et valorisés en fonction du niveau de difficultés estimé et de pertinence par rapport au sujet :

- Des améliorations en termes de présentation/représentation/graphismes/interactivité. Ceci n'est pas obligatoire et il est possible de rendre un très bon sujet sans utiliser Allegro. Une musique de fond n'est pas considérée comme pertinente vis à vis du sujet.

- Reprendre tout ou parti des chapitre 1) 2) 3) avec des graphes orientés pour modéliser des sens uniques. Ceci implique de développer un algorithme de détermination de forte-connexité pour filtrer les solutions admissibles.

- Mettre en place un/des objectif(s) d'autre nature : par exemple à la place de la distance à parcourir de point à point, l'objectif à optimiser serait de diminuer le temps de trajet du pire trajet (plutôt facile). Ou bien d'optimiser le flot admissible entre des sommets sources (résidentiels) et des sommets puits (lieux de travail).

- Etendre la méthode à de l'optimisation tri-objectif. Utiliser un logiciel de visualisation de nuage de points 3d (XYZ scatter plot) pour montrer la frontière de Pareto en 3d.

- Développer une analyse sophistiquée (expérimentale et/ou théorique) sur un aspect obscure de l'énoncé, par exemple en bas de la page 11 l'auteur se demande si la détermination de la frontière de Pareto avec une double boucle imbriquée et effacement des dominés au fur et à mesure est linéaire ou log-linéaire dans un « cas moyen » (à définir).

- Sur le même thème de la détermination de la frontière de Pareto, il est connu qu'un algorithme de détermination de frontière de Pareto (célèbre, à chercher) commence par trier les solutions par combinaison linéaire des objectifs pour permettre aux meilleures de décimer plus rapidement les mauvaises.

- Essayer d'« interpoler » entre les vecteurs solutions extrêmes qui sont connus (aux extrémités de la frontière de Pareto). Combien de chemins existent entre des vecteurs binaires ? Représenter les vecteurs intermédiaires, par exemple le vecteur X ci dessous est « intermédiaire » entre les vecteurs A et B. Combien existe-t-il de tels vecteurs ? Ce nombre est-il économique par rapport aux approches brutes-forces précédentes ? Y a t il beaucoup de solutions admissibles dans ces cas intermédiaires ? Sont elles proches ou sur la frontière de Pareto ? Les représenter. Des conjectures intéressantes peuvent être émises...

  `A : 10110111010101010110111001`
  `X : 10110x11x10101011011xxxx` x représente une position avec choix 0 ou 1
  `B : 10110011110101010110110110`

- Essayez une approche « heuristique » que vous imaginez. Il n'est pas nécessaire qu'elle réussisse pour valoir des points : elle vaudra si vous pouvez l'implémenter, expliquer la motivation, présenter des résultats expérimentaux, et en tirer des conclusions de manière rationnelle.

- Rendre le problème traité plus réaliste. Par exemple seule une partie de la ville sera cyclable. Un cycliste rejoint (au plus court) une piste cyclable, et quitte (au plus court) une piste cyclable. Pour aller d'un endroit à un autre de la ville on passe forcément par des rues non cyclables. Comment bien poser ce problème ? Comment le traiter ?

- Enfin on peut chercher à s'affranchir de l'approche brute force pour l'obtention de toutes les solutions en adoptant une approche différente pour découvrir les bonne solutions (pas forcément les meilleures). Un tel développement sera illustré par l'utilisation de graphes de « grande taille » (plus de 32 arêtes). Par ordre de difficulté croissante :
  - Echantillonage aléatoire : étudier une fraction de l'espace de décisions
  - Suivi de gradient, hill-climbing, recuit simulé (simulated annealing), faites vos recherches en partant de là…
  - Algorithmes génétiques, version simple (reproduction à la frontière de Pareto) ou modèle plus sophistiqué : http://oklahomaanalytics.com/data-science-techniques/nsga-ii-explained/

Soyez réaliste, vous ne pourrez pas faire tout, et ça n'est pas demandé !

Consignes générales

En dehors des exemples du cours et des codes officiellement fournis, toute détection de copie massive de code tiers sera considéré comme plagiat et très sévèrement sanctionné, avec circonstances aggravantes si
- l'emprunt n'est pas cité
- l'emprunt vient du travail d'une autre équipe d'étudiants de l'ECE pour ce même projet
- il y a tentative de dissimulation (maquillage de code...)
- vous avez déjà été repéré dans les « cas suspects » des dépôts de TP

Il reste possible d'utiliser un code publique (publié sur Internet) avec ou sans modifications mais l'emprunt doit être **très explicitement et très clairement cité** en tant que tel dans le code (source, début et fin de l'emprunt) et également **explicitement mentionné lors de la soutenance**.

Le code source doit être présentable : bien indenté, raisonnablement aéré et commenté (chapitré). Les identifiants doivent être explicites, on doit comprendre de quoi on parle en lisant le nom des classes, des attributs, des méthodes. Le projet est structuré en .cpp et .h par classe et si possible hiérarchisé en arborescence de répertoire avec des thématiques.

Le projet est à faire en trinôme ou binôme, à l'intérieur d'un même groupe de TP. Les consignes de constitution des équipes de projet ont été communiquées de même que la date de soutenance et les modalités de rendu.

Robin Fercoq et toute l'équipe encadrante,
15/04/2019
Version 1, sujette à ajustements mineurs en cas d'urgence