Problème de la coupe minimale

Perrine Jouteur

1 L'algorithme de Karger

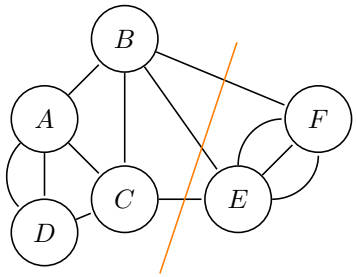
1.1 Position du problème

Définition 1 Soit $G = (V, E)$ un graphe non orienté. On appelle coupe du graphe G toute partition non triviale $\{V_1, V_2\}$ de V (c'est-à-dire que $V_1 \neq \emptyset$ et $V_2 \neq \emptyset$).
Le nombre d'arêtes entre V_1 et V_2 (c'est-à-dire les arêtes qui relient un sommet de V_1 et un sommet de V_2) est alors appelé cardinal de la coupe.

Définition 2 Le problème de la coupe minimale (MINCUT) est défini comme suit :

- **Entrée** : $G = (V, E)$ un graphe non orienté, sans boucle.
- **Sortie** : $\{V_1, V_2\}$ une coupe de G de cardinal minimal.

Exemple 1 On considère le graphe suivant :



Lorsqu'on découpe le graphe comme indiqué sur le schéma, c'est-à-dire que l'on partitionne V en $\{A, B, C, D\} \cup \{E, F\}$, on obtient une coupe de cardinal 3. On vérifie aisément qu'il n'existe pas de coupe de cardinal strictement inférieur à 3.
Cet exemple montre qu'il n'y a pas unicité de la coupe minimale en général. Ici, on aurait aussi pu choisir $V_1 = \{A, C, D\}$ ou $V_1 = \{D\}$ pour obtenir des coupes de cardinal 3.

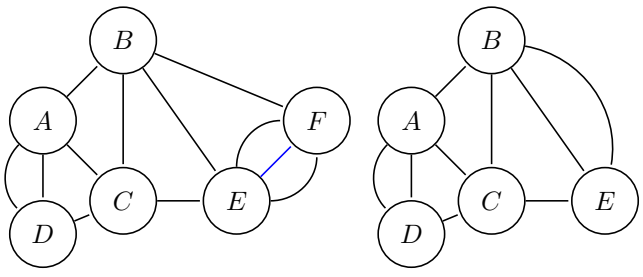
Remarque 1 Dans un graphe non connexe, le problème de la coupe minimale est trivial puisqu'il suffit de "couper" entre deux composantes connexes disjointes pour avoir une coupe de cardinal 0. Dans la suite, on ne considèrera donc que des graphes connexes.

1.2 Un algorithme probabiliste

Définition 3 L'algorithme de Karger utilise la notion de contraction d'arêtes : étant donné un graphe $G = (V, E)$, sans boucles, contenant au moins deux sommets, et une arête $e = \{u_1, u_2\} \in E$, on construit un graphe contracté $G_{>e<} = (V_{>e<}, E_{>e<})$ tel que

- $V_{>e<} = V \backslash \{u_2\}$
- $E_{>e<}$ contient les arêtes $\{v, w\}$, avec $v \neq u_1$, $w \neq u_1$ et $\{v, w\} \in E$ ou bien $v = u_1$ et $\{w, u_2\} \in E$.

Exemple 2 Contraction d'une arête $\{E, F\}$ dans le graphe de l'exemple 1 :



(a) Avant contraction (b) Après contraction

Voici le pseudo-code de l'algorithme de Karger :

Algorithm 1 RandMINCUT

Require: $G = (V, E)$ un graphe non orienté, sans boucles, connexe
Ensure: Le cardinal d'une coupe de G
  while $card(V) > 2$ **do**
    Choisir $e \in E$ de façon aléatoire uniforme
    $G \leftarrow G_{>e<}$
  end while
  Renvoyer $card(E)$

Propriété 1 Soit G un graphe à n sommets. L'algorithme de Karger termine, et renvoie une coupe minimale avec une probabilité d'au moins $\frac{2}{n(n-1)}$.

2 Couplage avec le problème du flot maximal

2.1 Réseaux de flots

Définition 4 Un réseau de flot $G = (V, E, c, s, d)$ est un graphe orienté sans boucle, muni d'une fonction de capacité $c : E \longmapsto \mathbb{N}^*$, et dans lequel on distingue un sommet source s (sans arc entrant) et un sommet destination d, (sans arc sortant).

Définition 5 Soit G un réseau de flot. On appelle flot de G toute fonction $f : V^2 \longmapsto \mathbb{R}^+$ telle que :

- $\forall (u, v) \in V^2, \begin{cases} f(u, v) \leqslant c(u, v) \text{ si } (u, v) \in A \\ f(u, v) = 0 \text{ sinon} \end{cases}$
- $\forall u \in S \backslash \{s, d\}, \sum_{v \in S} f(u, v) = \sum_{v \in S} f(v, u)$ (loi de Kirchhoff)

La valeur du flot est alors définie par : $|f| = \sum_{v \in S} f(s, v)$.

Définition 6 Le problème du flot maximal est défini comme suit :

- **Entrée** : un réseau de flot G
- **Sortie** : un flot f de G de valeur maximale.

2.2 Théorème du flot-max/coupe-min

Le théorème du flot-max/coupe-min énonce l'équivalence de ces deux problèmes.

Théoreme 1 Soit $G = (V, E, c, s, d)$ un réseau de flot. On note G' le graphe non orienté issu de G, où un arc $(u, v) \in E$ de capacité $c(u, v)$ devient $c(u, v)$ arêtes entre u et v.
Alors la valeur maximale d'un flot de G est égale au cardinal minimal d'une coupe de G' séparant s et d.

L'intérêt de ce théorème réside dans le fait que l'on connaisse un algorithme efficace pour calculer le flot maximal d'un réseau. Cet algorithme repose sur la construction d'un réseau de flot dit "résiduel".

Définition 7 Soit $G = (V, E, c, s, d)$ un réseau de flot, et f un flot de G. Le réseau de flot résiduel associé à G et f est $G_f = (V, E_f, c_f, s, d)$, avec c_f une nouvelle fonction de capacité telle que :

$$\forall (u, v) \in V^2, c_f(u, v) = \begin{cases} c(u, v) - f(u, v) \text{ si } (u, v) \in E \\ f(u, v) \text{ si } (v, u) \in E \\ 0 \text{ sinon} \end{cases}$$

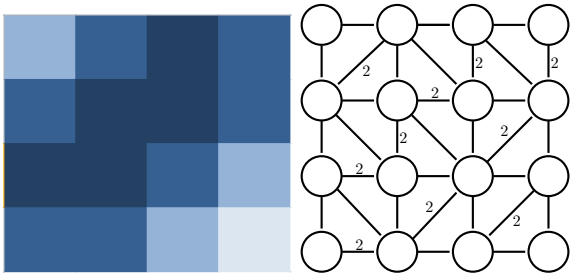
Et E_f est E auquel on a enlevé tous les arcs de capacité c_f nulle.

Algorithm 2 FordFulkerson

Require: Un réseau de flot $G = (V, E, c, s, d)$
Ensure: Un flot de G de valeur maximale
  $f \leftarrow$ flot nul
  while Il existe un chemin de s à d dans le réseau résiduel G_f **do**
    Choisir γ un tel chemin
    $c_{min} \leftarrow min\{c_f(u, v) \mid (u, v) \in \gamma\}$
    for $(u, v) \in \gamma$ **do**
      if $(u, v) \in E$ **then**
        $f(u, v) \leftarrow f(u, v) + c_{min}$
      else
        $f(u, v) = f(v, u) - c_{min}$
      end if
    end for
  end while
  renvoyer f

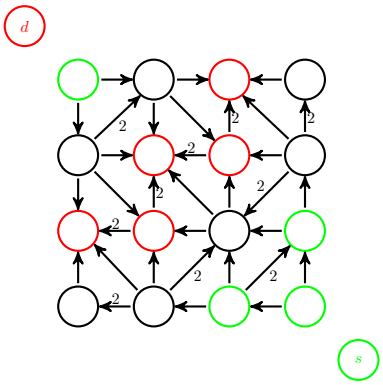
3 Application

Le théorème flot-max/coupe-min est notamment utilisé dans le traitement d'images bicolores, pour éliminer du bruit. L'idée générale est de construire un graphe à partir de l'image à traiter, un pixel correspondant à un noeud du graphe. Deux pixels sont reliés par une arête dont le poids est d'autant plus fort que les pixels sont proches (en termes d'espace et de couleur). En pratique, cette construction du graphe associé se fait au moyen d'outils probabilistes et d'inférences bayésiennes.



(c) Image à traiter (d) Graphe associé

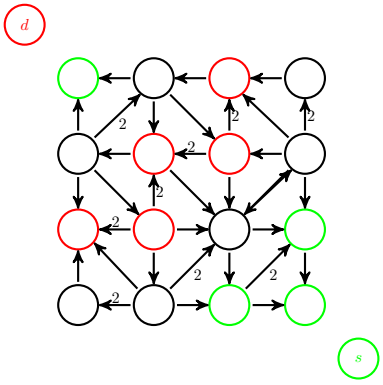
Pour rendre l'image plus nette, on veut classer les pixels en deux catégories, les foncés et les clairs. On ajoute donc au graphe deux sommets, s et d, qui représentent ces deux catégories, et on relie les pixels très foncés à s et les pixels très clairs à t, en conférant à chacune de ces arêtes une capacité maximale. Enfin, on oriente les arêtes (qui deviennent donc des arcs) de la source s vers la destination s.
Pour des raisons de lisibilité, on représente le fait qu'un sommet soit relié à s (resp. à d) en le coloriant en vert (resp. en rouge).



(e) Graphe orienté, muni des sommets source et destination

Il suffit alors d'appliquer l'algorithme de Ford-Fulkerson et le théorème flot-max/coupe-min pour déterminer une coupe minimale du graphe, qui distinguera les sommets clairs des sommets foncés. Minimiser le cardinal de la coupe permet de préserver une certaine cohérence dans l'image, en s'assurant que des sommets très proches soient dans la même catégorie.

Lorsque l'on applique cette méthode au graphe ci-dessus (il y a en annexe les détails des étapes de l'algorithme), on obtient une valeur maximale de flot égale à 6. Le dernier graphe résiduel donne la coupe à réaliser pour obtenir une coupe de cardinal 6 (donc une coupe minimale). En effet, il suffit de prendre pour V_1 l'ensemble des sommets qui sont accessibles à partir du sommet source (c'est-à-dire les sommets u tels qu'il existe un chemin de s à u) et de poser $V_2 = V_1^c$.

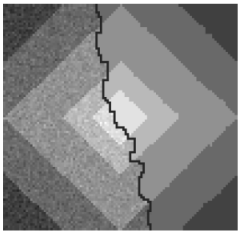


(f) Dernier graphe résiduel, sans chemin de s à d

Ici, les sommets accessibles depuis s sont exactement les sommets reliés à s, donc les sommets verts. L'image traitée aura donc cette allure :

(g) Image traitée

Ainsi, le couplage entre le problème de la coupe minimale et celui du flot maximal permet de développer des techniques d'amélioration d'images qui trouvent leur application dans de nombreux domaines : imagerie médicale, restauration d'images d'archives, publicité...



(a) *Diamond* restoration

(b) Original *Bell Quad*

(c) "Restored" *Bell Quad*

(h) Exemples de restauration d'images

4 Bibliographie

[1] Nathalie BERTRAND. *Cours magistral 4, algorithmes probabilistes*, 2020

[2] François SCHWARTZENTRUBER. *Notes de cours - Flots*, 2019

[3] D.M. GREIG, B.T. PORTEOUS et A.H. SEHEULT. *Exact Maximum A Posteriori Estimation for Binary Images*, 1989

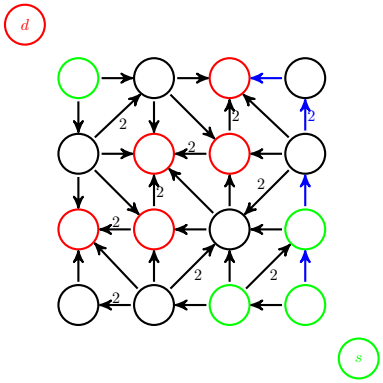
[4] Yuri BOYKOV et Vladimir KOLMOGOROV. *An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision*, 2004

[5] Robert CORI, Guillaume HANROT, Claire KENYON et Jean-Marc STEYAERT. *Conception et analyse d'algorithmes*, 2013

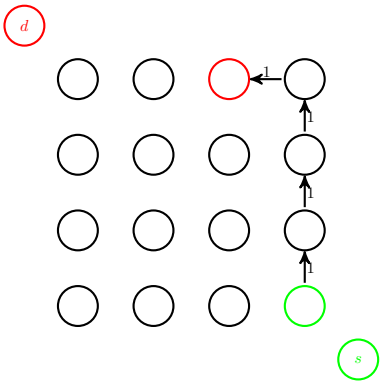
[6] Wikipédia. *Graph cuts in computer vision ; Théorème flot-max/coupe-min ; Image segmentation*

5 Annexe

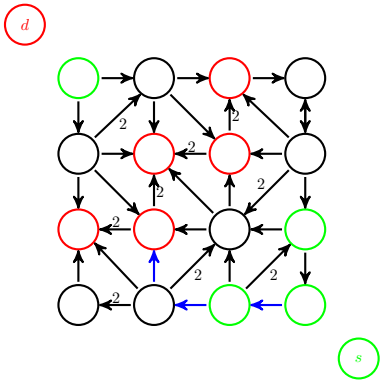
Exécution de l'algorithme de Ford-Fulkerson sur un exemple



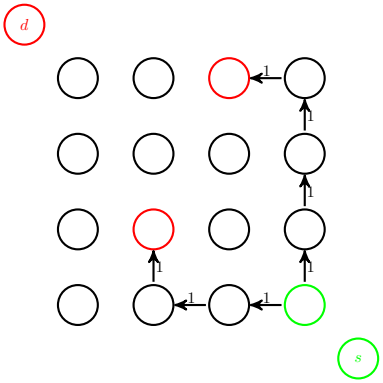
(i) Graphe originel et un chemin de s à d (en bleu)



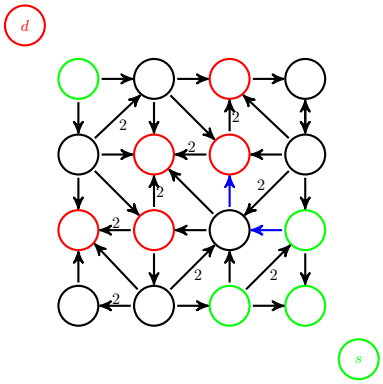
(j) Flot à l'étape 1



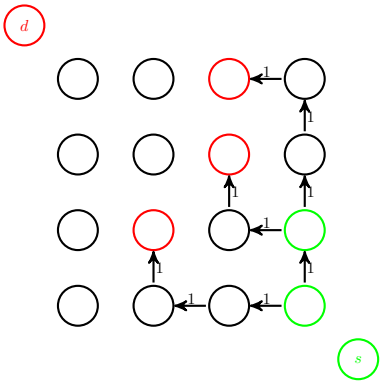
(k) Graphe résiduel et un chemin de s à d (en bleu)



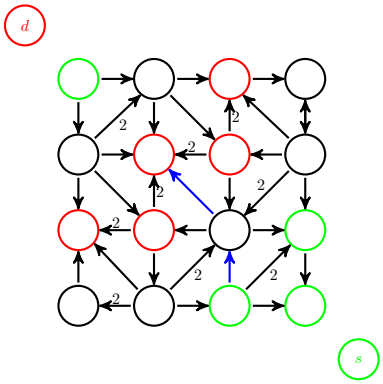
(l) Flot à l'étape 2



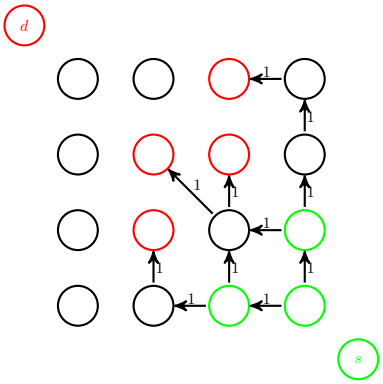
(m) Graphe résiduel et un chemin de s à d (en bleu)



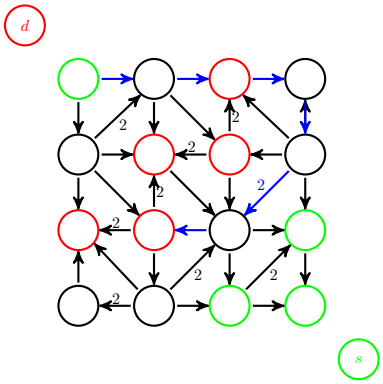
(n) Flot à l'étape 3



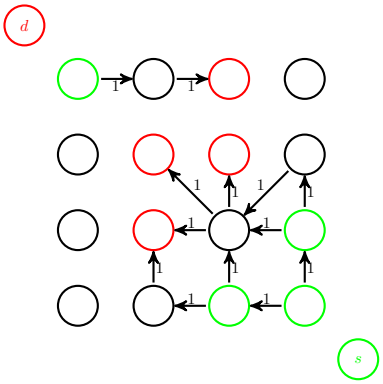
(o) Graphe résiduel et un chemin de s à d (en bleu)



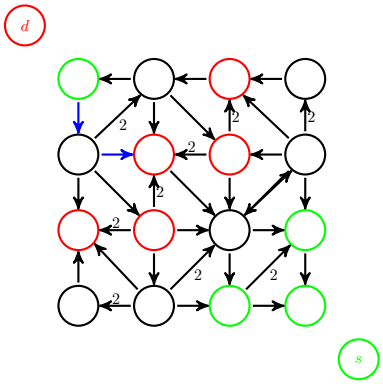
(p) Flot à l'étape 4



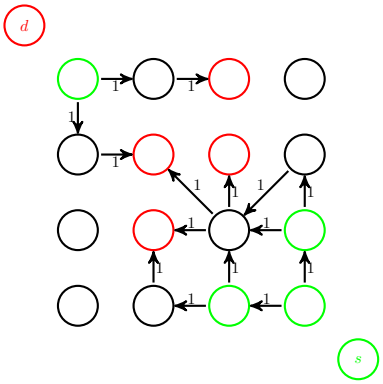
(q) Graphe résiduel et un chemin de s à d (en bleu)



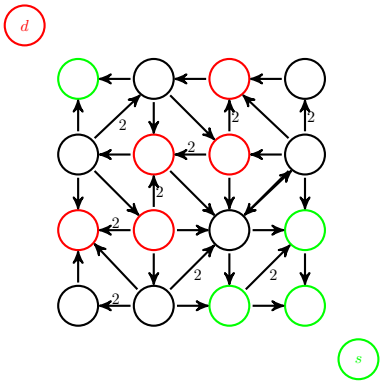
(r) Flot à l'étape 5



(s) Graphe résiduel et un chemin de s à d (en bleu)



(t) Flot à l'étape 6



(u) Graphe résiduel sans chemin de s à d