

# Cours 9: Recherche locale

- Heuristiques de recherche locale
- Réseau de Hopfield ou verre de spin
- Max-Cut, NP-complétude, recherche locale.
- Classification par recherche locale

# Cours 9: Recherche locale

- Heuristiques de recherche locale
- Réseau de Hopfield ou verre de spin
- Max-Cut, NP-complétude, recherche locale.
- Classification par recherche locale

# Heuristiques de recherche locale: méthode du gradient

**Contexte:** un problème d'optimisation combinatoire,  $\min_X f(X)$ , avec une notion de voisinage entre configurations admissibles

**Méthode du gradient:** partir de  $X_0$  arbitraire

- itérer  $X_i =$  meilleur voisin de  $X_{i-1}$  tant que  $f(X_i) < f(X_{i-1})$
- renvoyer  $X_i$ .

# Heuristiques de recherche locale: méthode du gradient

**Contexte:** un problème d'optimisation combinatoire,  $\min_X f(X)$ , avec une notion de voisinage entre configurations admissibles

**Méthode du gradient:** partir de  $X_0$  arbitraire

- itérer  $X_i =$  meilleur voisin de  $X_{i-1}$  tant que  $f(X_i) < f(X_{i-1})$
- renvoyer  $X_i$ .

**Vertex cover:** étant donné un graphe  $G = (V, E)$ , trouver  $X \subset V$  couvrant toutes les arêtes et de cardinal minimal.

# Heuristiques de recherche locale: méthode du gradient

**Contexte:** un problème d'optimisation combinatoire,  $\min_X f(X)$ , avec une notion de voisinage entre configurations admissibles

**Méthode du gradient:** partir de  $X_0$  arbitraire

— itérer  $X_i =$  meilleur voisin de  $X_{i-1}$  tant que  $f(X_i) < f(X_{i-1})$

— renvoyer  $X_i$ .

**Vertex cover:** étant donné un graphe  $G = (V, E)$ , trouver  $X \subset V$  couvrant toutes les arêtes et de cardinal minimal.

**Voisinage d'une configuration:**  $X$  un couvrant

$\text{Voisins}(X) = \{\text{couvrants } Y \text{ obtenus en ajoutant ou supprimant un sommet}\}$

# Heuristiques de recherche locale: méthode du gradient

**Contexte:** un problème d'optimisation combinatoire,  $\min_X f(X)$ , avec une notion de voisinage entre configurations admissibles

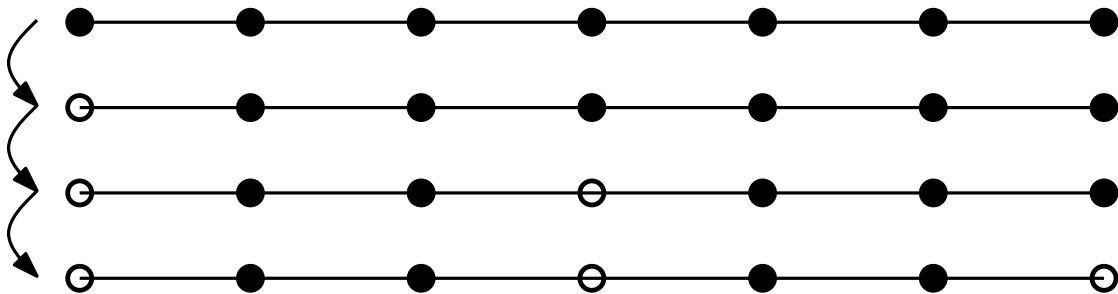
**Méthode du gradient:** partir de  $X_0$  arbitraire

- itérer  $X_i =$  meilleur voisin de  $X_{i-1}$  tant que  $f(X_i) < f(X_{i-1})$
- renvoyer  $X_i$ .

**Vertex cover:** étant donné un graphe  $G = (V, E)$ , trouver  $X \subset V$  couvrant toutes les arêtes et de cardinal minimal.

**Voisinage d'une configuration:**  $X$  un couvrant

$\text{Voisins}(X) = \{\text{couvrants } Y \text{ obtenus en ajoutant ou supprimant un sommet}\}$



# Heuristiques de recherche locale: méthode du gradient

**Contexte:** un problème d'optimisation combinatoire,  $\min_X f(X)$ , avec une notion de voisinage entre configurations admissibles

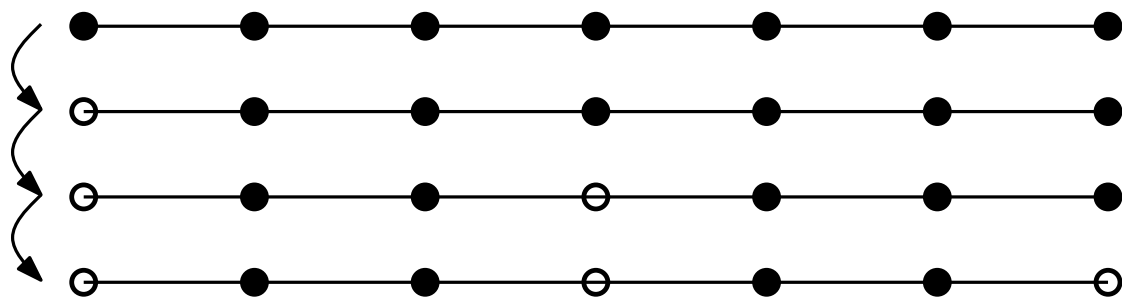
**Méthode du gradient:** partir de  $X_0$  arbitraire

- itérer  $X_i =$  meilleur voisin de  $X_{i-1}$  tant que  $f(X_i) < f(X_{i-1})$
- renvoyer  $X_i$ .

**Vertex cover:** étant donné un graphe  $G = (V, E)$ , trouver  $X \subset V$  couvrant toutes les arêtes et de cardinal minimal.

**Voisinage d'une configuration:**  $X$  un couvrant

$\text{Voisins}(X) = \{\text{couvrants } Y \text{ obtenus en ajoutant ou supprimant un sommet}\}$



Minimum local avec  $|X| = 4$ , or on peut couvrir avec 3 sommets.

# Heuristiques de recherche locale: méthode du gradient

**Contexte:** un problème d'optimisation combinatoire,  $\min_X f(X)$ , avec une notion de voisinage entre configurations admissibles

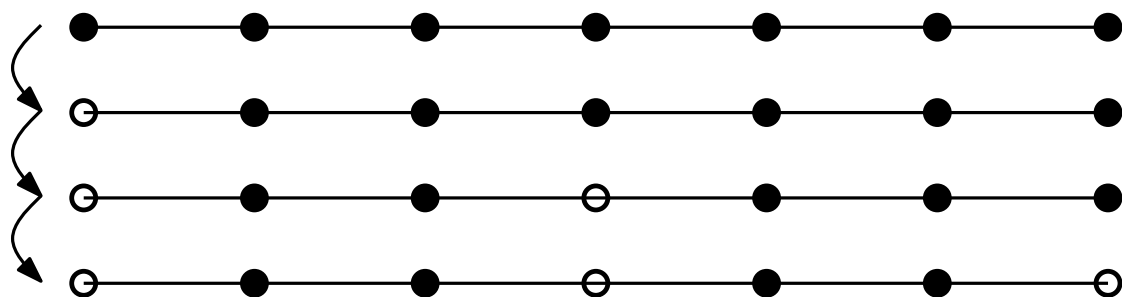
**Méthode du gradient:** partir de  $X_0$  arbitraire

- itérer  $X_i =$  meilleur voisin de  $X_{i-1}$  tant que  $f(X_i) < f(X_{i-1})$
- renvoyer  $X_i$ .

**Vertex cover:** étant donné un graphe  $G = (V, E)$ , trouver  $X \subset V$  couvrant toutes les arêtes et de cardinal minimal.

**Voisinage d'une configuration:**  $X$  un couvrant

$\text{Voisins}(X) = \{\text{couvrants } Y \text{ obtenus en ajoutant ou supprimant un sommet}\}$



Minimum local avec  $|X| = 4$ , or on peut couvrir avec 3 sommets.

Il s'agit essentiellement d'un algo glouton, sous-optimal ici

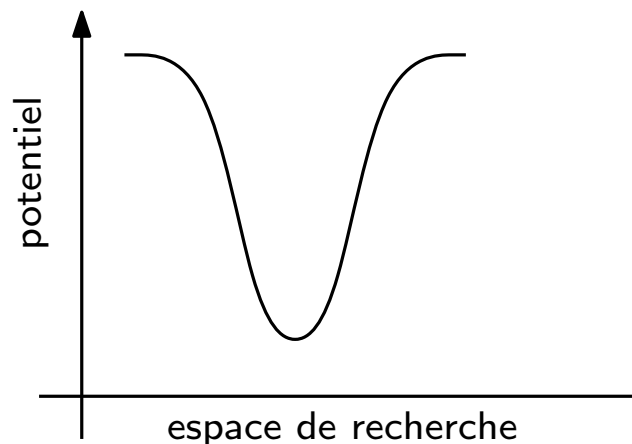
Dans d'autres cas ce peut être optimal: exemple de l'algo simplexe pour LP

# Heuristiques de recherche locale: puits et bosses

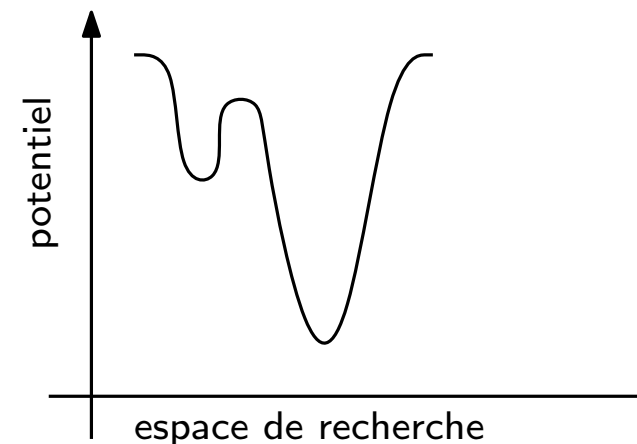
**Potentiel:** la méthode du gradient peut être vue comme une marche dans le graphe des configurations suivant la plus grande pente d'un potentiel.

**Puits et bosses:** par définition pour un problème d'espace fini, la méthode du gradient converge vers un minimum local, **un puits de potentiel**

La qualité du résultat dépend de la forme du potentiel et du pt de départ !



unique puits, convergence vers l'optimum global



existence de minimaux locaux sous-optimaux

# Heuristiques de recherche locale: métropolis

**Métropolis:** pour ne pas rester bloqué, introduire une part de hasard dans la marche sur les configurations et autoriser les "remontées" de potentiel

# Heuristiques de recherche locale: métropolis

**Métropolis:** pour ne pas rester bloqué, introduire une part de hasard dans la marche sur les configurations et autoriser les "remontées" de potentiel

On choisit le voisin au hasard avec une proba dépendant de la différence de potentiel:  $\Pr(X_i = Y) \propto e^{-(|Y|-|X|)/T}$ , où  $T$  est un paramètre

# Heuristiques de recherche locale: métropolis

**Métropolis:** pour ne pas rester bloqué, introduire une part de hasard dans la marche sur les configurations et autoriser les "remontées" de potentiel

On choisit le voisin au hasard avec une proba dépendant de la différence de potentiel:  $\Pr(X_i = Y) \propto e^{-(|Y|-|X|)/T}$ , où  $T$  est un paramètre

**Théorème:** si l'espace des configurations est connexe pour la notion de voisinage choisie, la probabilité de présence en  $X$  converge vers  $\propto e^{-|X|/T}$ .

# Heuristiques de recherche locale: métropolis

**Métropolis:** pour ne pas rester bloqué, introduire une part de hasard dans la marche sur les configurations et autoriser les "remontées" de potentiel

On choisit le voisin au hasard avec une proba dépendant de la différence de potentiel:  $\Pr(X_i = Y) \propto e^{-(|Y|-|X|)/T}$ , où  $T$  est un paramètre

**Théorème:** si l'espace des configurations est connexe pour la notion de voisinage choisie, la probabilité de présence en  $X$  converge vers  $\propto e^{-|X|/T}$ .

## Choix de la température ?

- élevée = marche au hasard non biaisée; distribution quasi uniforme
- basse = à la limite l'optimum global est favorisé mais évolution lente.

# Heuristiques de recherche locale: simulated annealing

Recuit simulé: faire diminuer la température par palier

Inspiré par des méthodes de physique des matériaux

# Heuristiques de recherche locale: simulated annealing

**Recuit simulé:** faire diminuer la température par palier

Inspiré par des méthodes de physique des matériaux

En pratique cela peut marcher raisonnablement pour éliminer des petits minima locaux sur la route d'un gros minimum global.

Exemple: ranger des billes en secouant le sac...

# Heuristiques de recherche locale: simulated annealing

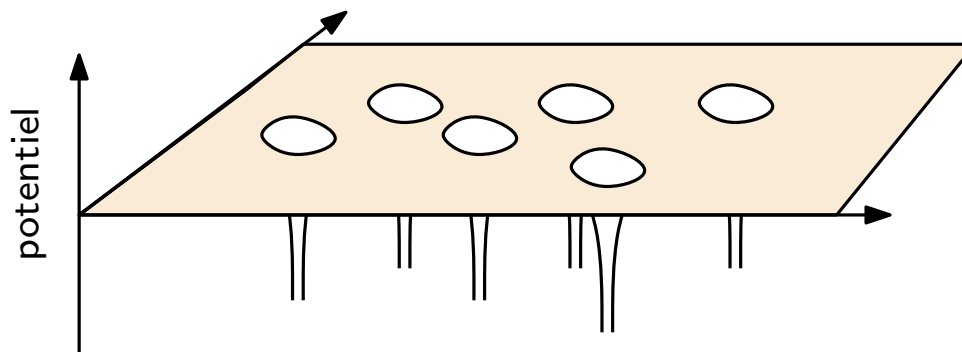
**Recuit simulé:** faire diminuer la température par palier

Inspiré par des méthodes de physique des matériaux

En pratique cela peut marcher raisonnablement pour éliminer des petits minima locaux sur la route d'un gros minimum global.

Exemple: ranger des billes en secouant le sac...

Pas efficace pour des paysages lunaires: lorsqu'il existe plusieurs bonnes solutions de structures très différentes.



# Changer de voisinage pour changer le relief !

Les problèmes viennent des minimaux locaux...

en redéfinissant la notion de voisins on change de *localité*

# Changer de voisinage pour changer le relief !

Les problèmes viennent des minimaux locaux...

en redéfinissant la notion de voisins on change de *localité*

Si on prend Voisins = {toutes les configurations}, local = global...  
mais on ne sait plus trouver un (le) meilleur voisin efficacement !

# Changer de voisinage pour changer le relief !

Les problèmes viennent des minimaux locaux...

en redéfinissant la notion de voisins on change de *localité*

Si on prend Voisins = {toutes les configurations}, local = global...  
mais on ne sait plus trouver un (le) meilleur voisin efficacement !

**Exemple:** recherche de flot maximal dans un graphe

- **voisinage simple:** flots obtenus par saturation le long d'un chemin de  $G$   
⇒ on reste bloqué dans des maximaux locaux (flots saturés)

# Changer de voisinage pour changer le relief !

Les problèmes viennent des minimaux locaux...

en redéfinissant la notion de voisins on change de *localité*

Si on prend Voisins = {toutes les configurations}, local = global...  
mais on ne sait plus trouver un (le) meilleur voisin efficacement !

**Exemple:** recherche de flot maximal dans un graphe

- **voisinage simple:** flots obtenus par saturation le long d'un chemin de  $G$   
⇒ on reste bloqué dans des maximaux locaux (flots saturés)
- **voisinage plus malin:** flots obtenus par saturation le long d'un chemin du graphe d'augmentation  $G_\phi$ .

# Changer de voisinage pour changer le relief !

Les problèmes viennent des minimaux locaux...

en redéfinissant la notion de voisins on change de *localité*

Si on prend Voisins = {toutes les configurations}, local = global...  
mais on ne sait plus trouver un (le) meilleur voisin efficacement !

**Exemple:** recherche de flot maximal dans un graphe

- **voisinage simple:** flots obtenus par saturation le long d'un chemin de  $G$   
⇒ on reste bloqué dans des maximaux locaux (flots saturés)
- **voisinage plus malin:** flots obtenus par saturation le long d'un chemin du graphe d'augmentation  $G_\phi$ .

**Remarque:** on a pas besoin de trouver le meilleur chemin d'augmentation, il suffit de toujours en trouver un qui permet d'augmenter.

# Cours 9: Recherche locale

- Heuristiques de recherche locale
- Réseau de Hopfield ou verre de spin
- Max-Cut, NP-complétude, recherche locale.
- Classification par recherche locale

# Réseaux neuronaux de Hopfield

Un modèle simple de "mémoire associative": la valeur d'un "neurone" est corrélée à celle de ses voisins et ces neurones s'auto-organisent (modèle équivalent au modèle des verres de spin en physique statistique)

**Données:** Un graphe  $G = (V, E)$  avec des poids  $p_a \in \mathbb{R}$  sur les arêtes.

Une **configuration**  $\sigma$  est une assignation  $V \rightarrow \{-1, +1\}$ .

Une arête  $xy$  est **contente** si  $p_{xy}\sigma_x\sigma_y < 0$ , **frustrée** sinon.

Un sommet est **satisfait** si  $\sum_{y \sim x} p_{xy}\sigma_x\sigma_y \leq 0$ : le poids des arêtes contentes dépasse celui des arêtes frustrées.

Une configuration est **stable** si tous les sommets sont satisfaits.

# Réseaux neuronaux de Hopfield

Un modèle simple de "mémoire associative": la valeur d'un "neurone" est corrélée à celle de ses voisins et ces neurones s'auto-organisent (modèle équivalent au modèle des verres de spin en physique statistique)

**Données:** Un graphe  $G = (V, E)$  avec des poids  $p_a \in \mathbb{R}$  sur les arêtes.

Une **configuration**  $\sigma$  est une assignation  $V \rightarrow \{-1, +1\}$ .

Une arête  $xy$  est **contente** si  $p_{xy}\sigma_x\sigma_y < 0$ , **frustrée** sinon.

Un sommet est **satisfait** si  $\sum_{y \sim x} p_{xy}\sigma_x\sigma_y \leq 0$ : le poids des arêtes contentes dépasse celui des arêtes frustrées.

Une configuration est **stable** si tous les sommets sont satisfaits.

**Problème:** existe-t-il une configuration stable pour tout réseau de Hopfield ?

# Réseaux neuronaux de Hopfield

Un modèle simple de "mémoire associative": la valeur d'un "neurone" est corrélée à celle de ses voisins et ces neurones s'auto-organisent (modèle équivalent au modèle des verres de spin en physique statistique)

**Données:** Un graphe  $G = (V, E)$  avec des poids  $p_a \in \mathbb{R}$  sur les arêtes.

Une **configuration**  $\sigma$  est une assignation  $V \rightarrow \{-1, +1\}$ .

Une arête  $xy$  est **contente** si  $p_{xy}\sigma_x\sigma_y < 0$ , **frustrée** sinon.

Un sommet est **satisfait** si  $\sum_{y \sim x} p_{xy}\sigma_x\sigma_y \leq 0$ : le poids des arêtes contentes dépasse celui des arêtes frustrées.

Une configuration est **stable** si tous les sommets sont satisfaits.

**Problème:** existe-t-il une configuration stable pour tout réseau de Hopfield ?

Si oui comment en trouver une efficacement ?

# Réseaux neuronaux de Hopfield: recherche locale

Recherche locale par flip de sommet: les voisins d'une configuration sont obtenus par changement de signe sur 1 sommet non satisfait.

# Réseaux neuronaux de Hopfield: recherche locale

**Recherche locale par flip de sommet:** les voisins d'une configuration sont obtenus par changement de signe sur 1 sommet non satisfait.

**Remarque:** Le changement de spin sur un sommet échange les arêtes contentes et frustrées autour de ce sommet.

⇒ si  $v$  n'était pas satisfait dans  $\sigma$  c'est-à-dire si  $\sum_{y \sim x} p_{xy} \sigma_x \sigma_y \geq 0$ , il le sera dans  $\sigma'$  avec  $\sigma'_x = -\sigma_x$ ,  $\sigma'_y = \sigma_y$  pour tout  $y \neq x$ .

# Réseaux neuronaux de Hopfield: recherche locale

**Recherche locale par flip de sommet:** les voisins d'une configuration sont obtenus par changement de signe sur 1 sommet non satisfait.

**Remarque:** Le changement de spin sur un sommet échange les arêtes contentes et frustrées autour de ce sommet.

$\Rightarrow$  si  $v$  n'était pas satisfait dans  $\sigma$  c'est-à-dire si  $\sum_{y \sim x} p_{xy} \sigma_x \sigma_y \geq 0$ , il le sera dans  $\sigma'$  avec  $\sigma'_x = -\sigma_x$ ,  $\sigma'_y = \sigma_y$  pour tout  $y \neq x$ .

Cependant le nombre de sommets satisfait peut diminuer après un flip !  
Convergence ?

# Réseaux neuronaux de Hopfield: recherche locale

**Recherche locale par flip de sommet:** les voisins d'une configuration sont obtenus par changement de signe sur 1 sommet non satisfait.

**Remarque:** Le changement de spin sur un sommet échange les arêtes contentes et frustrées autour de ce sommet.

⇒ si  $v$  n'était pas satisfait dans  $\sigma$  c'est-à-dire si  $\sum_{y \sim x} p_{xy} \sigma_x \sigma_y \geq 0$ , il le sera dans  $\sigma'$  avec  $\sigma'_x = -\sigma_x$ ,  $\sigma'_y = \sigma_y$  pour tout  $y \neq x$ .

Cependant le nombre de sommets satisfait peut diminuer après un flip !  
Convergence ?

**Proposition:** si on pose  $\Phi(\sigma) = \sum_{a \text{ contente}} |p_a|$ , alors tout flip augmente  $\Phi$

# Réseaux neuronaux de Hopfield: recherche locale

**Recherche locale par flip de sommet:** les voisins d'une configuration sont obtenus par changement de signe sur 1 sommet non satisfait.

**Remarque:** Le changement de spin sur un sommet échange les arêtes contentes et frustrées autour de ce sommet.

$\Rightarrow$  si  $v$  n'était pas satisfait dans  $\sigma$  c'est-à-dire si  $\sum_{y \sim x} p_{xy} \sigma_x \sigma_y \geq 0$ , il le sera dans  $\sigma'$  avec  $\sigma'_x = -\sigma_x$ ,  $\sigma'_y = \sigma_y$  pour tout  $y \neq x$ .

Cependant le nombre de sommets satisfait peut diminuer après un flip !  
Convergence ?

**Proposition:** si on pose  $\Phi(\sigma) = \sum_{a \text{ contente}} |p_a|$ , alors tout flip augmente  $\Phi$

**Corollaire:** la recherche s'arrête sur un maximum local de  $\Phi$  et la configuration  $\sigma$  obtenue est stable: complexité  $n\Phi$ .

# Réseaux neuronaux de Hopfield: recherche locale

**Recherche locale par flip de sommet:** les voisins d'une configuration sont obtenus par changement de signe sur 1 sommet non satisfait.

**Remarque:** Le changement de spin sur un sommet échange les arêtes contentes et frustrées autour de ce sommet.

$\Rightarrow$  si  $v$  n'était pas satisfait dans  $\sigma$  c'est-à-dire si  $\sum_{y \sim x} p_{xy} \sigma_x \sigma_y \geq 0$ , il le sera dans  $\sigma'$  avec  $\sigma'_x = -\sigma_x$ ,  $\sigma'_y = \sigma_y$  pour tout  $y \neq x$ .

Cependant le nombre de sommets satisfait peut diminuer après un flip !  
Convergence ?

**Proposition:** si on pose  $\Phi(\sigma) = \sum_{a \text{ contente}} |p_a|$ , alors tout flip augmente  $\Phi$

**Corollaire:** la recherche s'arrête sur un maximum local de  $\Phi$  et la configuration  $\sigma$  obtenue est stable: complexité  $n\Phi$ .

$\Rightarrow$  il existe une configuration stable ! (plus vrai si on oriente les arêtes)

# Cours 9: Recherche locale

- Heuristiques de recherche locale
- Réseau de Hopfield ou verre de spin
- Max-Cut, NP-complétude, recherche locale.
- Classification par recherche locale

# Maximum-cut et optimal stable Hopfield network

**Max-cut:** dans un graphe  $G$  valué trouver la partition  $A, B$  des sommets qui maximise

$$\sum_{(x,y); x \in A, y \in B} p_{xy}$$

Si les poids sont tous négatifs, il s'agit d'un pb de coupe minimale  
 $\Rightarrow$  min-cut = max-flow: on trouve la coupe par Ford Fulkerson.

# Maximum-cut et optimal stable Hopfield network

**Max-cut:** dans un graphe  $G$  valué trouver la partition  $A, B$  des sommets qui maximise

$$\sum_{(x,y); x \in A, y \in B} p_{xy}$$

Si les poids sont tous négatifs, il s'agit d'un pb de coupe minimale

⇒ min-cut = max-flow: on trouve la coupe par Ford Fulkerson.

Lorsqu'il y a des poids positifs, le problème devient difficile: si tous les poids sont positifs, on va voir qu'il est NP-complet.

# Maximum-cut et optimal stable Hopfield network

**Max-cut:** dans un graphe  $G$  valué trouver la partition  $A, B$  des sommets qui maximise

$$\sum_{(x,y); x \in A, y \in B} p_{xy}$$

Si les poids sont tous négatifs, il s'agit d'un pb de coupe minimale

$\Rightarrow$  min-cut = max-flow: on trouve la coupe par Ford Fulkerson.

Lorsqu'il y a des poids positifs, le problème devient difficile: si tous les poids sont positifs, on va voir qu'il est NP-complet.

**Lien avec Hopfield:** Si les  $p_{xy}$  sont positifs,  $p_{xy}\sigma_x\sigma_y < 0 \Leftrightarrow \sigma_x\sigma_y < 0$ : en posant  $\sigma_x = 1$  pour  $x \in A$  et  $\sigma_y = -1$  pour  $y \in B$  on voit que le potentiel de Hopfield correspond à la capacité  $\Phi(\sigma)$  de la coupe.

# Maximum-cut et optimal stable Hopfield network

**Max-cut:** dans un graphe  $G$  valué trouver la partition  $A, B$  des sommets qui maximise

$$\sum_{(x,y); x \in A, y \in B} p_{xy}$$

Si les poids sont tous négatifs, il s'agit d'un pb de coupe minimale

$\Rightarrow$  min-cut = max-flow: on trouve la coupe par Ford Fulkerson.

Lorsqu'il y a des poids positifs, le problème devient difficile: si tous les poids sont positifs, on va voir qu'il est NP-complet.

**Lien avec Hopfield:** Si les  $p_{xy}$  sont positifs,  $p_{xy}\sigma_x\sigma_y < 0 \Leftrightarrow \sigma_x\sigma_y < 0$ : en posant  $\sigma_x = 1$  pour  $x \in A$  et  $\sigma_y = -1$  pour  $y \in B$  on voit que le potentiel de Hopfield correspond à la capacité  $\Phi(\sigma)$  de la coupe.

La recherche locale nous permet de trouver une configuration stable, maximum local de  $\Phi$ . On ne s'attend pas à obtenir ainsi l'optimum global.

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-SAT se réduit à 3-NAESAT:** soit  $(C_i = x_i \vee y_i \vee z_i)_{i=1,\dots,m}$  une instance de 3-SAT; on pose  $C'_i = x_i \vee y_i \vee t_i$  et  $C''_i = z_i \vee \bar{t}_i \vee u$  où  $u$  et les  $t_i$  sont de nouvelles variables.

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-SAT se réduit à 3-NAESAT:** soit  $(C_i = x_i \vee y_i \vee z_i)_{i=1,\dots,m}$  une instance de 3-SAT; on pose  $C'_i = x_i \vee y_i \vee t_i$  et  $C''_i = z_i \vee \bar{t}_i \vee u$  où  $u$  et les  $t_i$  sont de nouvelles variables.

Il faut voir que  $(C_i)$  admet une affectation ssi  $(C'_i, C''_i)$  admet une bi-affectation (ie chaque clause a un littéral VRAI et un FAUX):

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-SAT se réduit à 3-NAESAT:** soit  $(C_i = x_i \vee y_i \vee z_i)_{i=1,\dots,m}$  une instance de 3-SAT; on pose  $C'_i = x_i \vee y_i \vee t_i$  et  $C''_i = z_i \vee \bar{t}_i \vee u$  où  $u$  et les  $t_i$  sont de nouvelles variables.

Il faut voir que  $(C_i)$  admet une affectation ssi  $(C'_i, C''_i)$  admet une bi-affectation (ie chaque clause a un littéral VRAI et un FAUX):

- s'il existe une affectation telle que tous les  $C_i$  soient vraies, on la complète par  $t_i = \overline{x_i \vee y_i}$  et  $u = \text{FAUX}$ : c'est une bi-affectation.

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-SAT se réduit à 3-NAESAT:** soit  $(C_i = x_i \vee y_i \vee z_i)_{i=1,\dots,m}$  une instance de 3-SAT; on pose  $C'_i = x_i \vee y_i \vee t_i$  et  $C''_i = z_i \vee \bar{t}_i \vee u$  où  $u$  et les  $t_i$  sont de nouvelles variables.

Il faut voir que  $(C_i)$  admet une affectation ssi  $(C'_i, C''_i)$  admet une bi-affectation (ie chaque clause a un littéral VRAI et un FAUX):

- s'il existe une affectation telle que tous les  $C_i$  soient vraies, on la complète par  $t_i = \overline{x_i \vee y_i}$  et  $u = \text{FAUX}$ : c'est une bi-affectation.
- s'il existe une bi-affectation de  $(C'_i, C''_i)$ :
  - soit  $u = \text{FAUX}$ , alors  $z_i \vee \bar{t}$  est vraie donc  $x_i \vee y_i \vee z_i$  est vraie.
  - soit  $u = \text{VRAI}$ , alors nier toutes les valeurs des variables...

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-NAESAT se réduit à MAXCUT:** à  $m$  clauses  $C_i$  sur  $n$  variables, on associe le graphe à  $2n$  sommets (un par littéral), avec un triangle pour chaque clause et les arêtes  $x, \bar{x}$  (soit  $3m + n$  arêtes).

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-NAESAT se réduit à MAXCUT:** à  $m$  clauses  $C_i$  sur  $n$  variables, on associe le graphe à  $2n$  sommets (un par littéral), avec un triangle pour chaque clause et les arêtes  $x, \bar{x}$  (soit  $3m + n$  arêtes).

Il faut voir que  $(C_i)$  admet une bi-affectation ssi le graphe associé admet une coupe avec  $2m + n$  arêtes.

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-NAESAT se réduit à MAXCUT:** à  $m$  clauses  $C_i$  sur  $n$  variables, on associe le graphe à  $2n$  sommets (un par littéral), avec un triangle pour chaque clause et les arêtes  $x, \bar{x}$  (soit  $3m + n$  arêtes).

Il faut voir que  $(C_i)$  admet une bi-affectation ssi le graphe associé admet une coupe avec  $2m + n$  arêtes.

- Si on a une bi-affectation, on prend  $X_+$  les sommets associés aux littéraux VRAI: dans la coupe, 2 arêtes par triangle et les  $(x, \bar{x})$ .

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-NAESAT se réduit à MAXCUT:** à  $m$  clauses  $C_i$  sur  $n$  variables, on associe le graphe à  $2n$  sommets (un par littéral), avec un triangle pour chaque clause et les arêtes  $x, \bar{x}$  (soit  $3m + n$  arêtes).

Il faut voir que  $(C_i)$  admet une bi-affectation ssi le graphe associé admet une coupe avec  $2m + n$  arêtes.

- Si on a une bi-affectation, on prend  $X_+$  les sommets associés aux littéraux VRAI: dans la coupe, 2 arêtes par triangle et les  $(x, \bar{x})$ .
- Une coupe de taille au moins  $2m + n$  ne peut contenir plus de 2 arêtes de chaque triangle, donc elle en contient exactement 2.

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-NAESAT se réduit à MAXCUT:** à  $m$  clauses  $C_i$  sur  $n$  variables, on associe le graphe à  $2n$  sommets (un par littéral), avec un triangle pour chaque clause et les arêtes  $x, \bar{x}$  (soit  $3m + n$  arêtes).

Il faut voir que  $(C_i)$  admet une bi-affectation ssi le graphe associé admet une coupe avec  $2m + n$  arêtes.

- Si on a une bi-affectation, on prend  $X_+$  les sommets associés aux littéraux VRAI: dans la coupe, 2 arêtes par triangle et les  $(x, \bar{x})$ .
- Une coupe de taille au moins  $2m + n$  ne peut contenir plus de 2 arêtes de chaque triangle, donc elle en contient exactement 2.

**CQFD**

# NP-complétude de MAXCUT

On réduit 3-SAT à MAXCUT via le problème 3-NAESAT: étant donné un ensemble de clauses à 3 littéraux, trouver une affectation où chaque clause contient au moins un littéral vrai et un littéral faux.

**3-NAESAT se réduit à MAXCUT:** à  $m$  clauses  $C_i$  sur  $n$  variables, on associe le graphe à  $2n$  sommets (un par littéral), avec un triangle pour chaque clause et les arêtes  $x, \bar{x}$  (soit  $3m + n$  arêtes).

Il faut voir que  $(C_i)$  admet une bi-affectation ssi le graphe associé admet une coupe avec  $2m + n$  arêtes.

- Si on a une bi-affectation, on prend  $X_+$  les sommets associés aux littéraux VRAI: dans la coupe, 2 arêtes par triangle et les  $(x, \bar{x})$ .
- Une coupe de taille au moins  $2m + n$  ne peut contenir plus de 2 arêtes de chaque triangle, donc elle en contient exactement 2. **CQFD**

**Remarque:** on a utilisé la version de Max-Cut avec arêtes multiples (ou de manière équivalente avec arêtes de poids entier  $i \geq 2$ ).

# Maximum-cut, approximation par recherche locale

Que dire d'une configuration stable obtenue par flips ?

Posons  $A = \{v \mid \sigma_v = +1\}$ ,  $B = \{v \mid \sigma_v = -1\}$

Pour  $x \in A$ , la condition  $x$  satisfait s'écrit:  $\sum_{y \in A} p_{xy} \leq \sum_{y \in B} p_{xy}$

En sommant sur les  $x \in A$  il vient que la capacité de la coupe est supérieure à 2 fois le poids des arêtes internes à  $A$ :  $2 \sum_{xy \subset A} p_{xy} \leq \sum_{xy \in A \times B} p_{xy}$ .

En symétrisant puis en ajoutant la capacité de la coupe des 2 côtés on obtient:

$$\sum_{xy} p_{xy} \leq 2 \sum_{xy \in A \times B} p_{xy}.$$

On en déduit que  $\sum_{xy \in A \times B} p_{xy} \geq \frac{1}{2} C^*$ : approximation à un facteur 2.

# Cours 9: Recherche locale

- Heuristiques de recherche locale
- Réseau de Hopfield ou verre de spin
- Max-Cut, NP-complétude, recherche locale.
- Classification par recherche locale

# Classification par recherche locale

**Données:** Un graphe  $G$ , un entier  $k$ , des poids  $p_{xy}$  sur les arêtes et des vecteurs de poids  $v_x = [v_x^1, \dots, v_x^k]$ .

**Problème:** trouver un étiquetage  $\ell : V \rightarrow \{1, \dots, k\}$  qui minimise

$$\Phi(\ell) = \sum_{x \in V} v_x^{\ell(x)} + \sum_{xy \in E} \delta_{f(x) \neq f(y)} p_{xy}.$$

**Interprétation:** on paye un coût spécifique au sommet pour la couleur choisie et un coût par arête ayant des extrémités de couleur différente.

$\Rightarrow$  cela permet de modéliser des pbs de classification ;  
par exemple segmentation d'image (sommets = pixels, graphe = grille)

# Classification par recherche locale

**Données:** Un graphe  $G$ , un entier  $k$ , des poids  $p_{xy}$  sur les arêtes et des vecteurs de poids  $v_x = [v_x^1, \dots, v_x^k]$ .

**Problème:** trouver un étiquetage  $\ell : V \rightarrow \{1, \dots, k\}$  qui minimise

$$\Phi(\ell) = \sum_{x \in V} v_x^{\ell(x)} + \sum_{xy \in E} \delta_{f(x) \neq f(y)} p_{xy}.$$

**Interprétation:** on paye un coût spécifique au sommet pour la couleur choisie et un coût par arête ayant des extrémités de couleur différente.

$\Rightarrow$  cela permet de modéliser des pbs de classification ;  
par exemple segmentation d'image (sommets = pixels, graphe = grille)

Pour  $k = 2$ , le problème se modélise par min-cut  $\Rightarrow$  algo polynomial

# Classification par recherche locale

**Données:** Un graphe  $G$ , un entier  $k$ , des poids  $p_{xy}$  sur les arêtes et des vecteurs de poids  $v_x = [v_x^1, \dots, v_x^k]$ .

**Problème:** trouver un étiquetage  $\ell : V \rightarrow \{1, \dots, k\}$  qui minimise

$$\Phi(\ell) = \sum_{x \in V} v_x^{\ell(x)} + \sum_{xy \in E} \delta_{f(x) \neq f(y)} p_{xy}.$$

**Interprétation:** on paye un coût spécifique au sommet pour la couleur choisie et un coût par arête ayant des extrémités de couleur différente.

$\Rightarrow$  cela permet de modéliser des pbs de classification ;  
par exemple segmentation d'image (sommets = pixels, graphe = grille)

Pour  $k = 2$ , le problème se modélise par min-cut  $\Rightarrow$  algo polynomial  
(à faire...)

# Classification par recherche locale

**Données:** Un graphe  $G$ , un entier  $k$ , des poids  $p_{xy}$  sur les arêtes et des vecteurs de poids  $v_x = [v_x^1, \dots, v_x^k]$ .

**Problème:** trouver un étiquetage  $\ell : V \rightarrow \{1, \dots, k\}$  qui minimise

$$\Phi(\ell) = \sum_{x \in V} v_x^{\ell(x)} + \sum_{xy \in E} \delta_{f(x) \neq f(y)} p_{xy}.$$

**Interprétation:** on paye un coût spécifique au sommet pour la couleur choisie et un coût par arête ayant des extrémités de couleur différente.

$\Rightarrow$  cela permet de modéliser des pbs de classification ;  
par exemple segmentation d'image (sommets = pixels, graphe = grille)

Pour  $k = 2$ , le problème se modélise par min-cut  $\Rightarrow$  algo polynomial  
(à faire...)

Pour  $k \geq 3$ , le problème est NP-complet.

# Classification par recherche locale

Une première idée: voisins obtenus en changeant la couleur d'un sommet  
⇒ minimum locaux non optimaux même pour  $k = 2$ ,  
arbitrairement mauvais en général...

# Classification par recherche locale

**Une première idée:** voisins obtenus en changeant la couleur d'un sommet  
⇒ minimum locaux non optimaux même pour  $k = 2$ ,  
arbitrairement mauvais en général...

**Amélioration:** voisins obtenus en faisant basculer simultanément un ensemble arbitraire de sommets vers la couleur  $i$

Le nombre de voisins d'une configuration est alors  $k \cdot 2^n$ : exponentiel !

# Classification par recherche locale

**Une première idée:** voisins obtenus en changeant la couleur d'un sommet  
⇒ minimum locaux non optimaux même pour  $k = 2$ ,  
arbitrairement mauvais en général...

**Amélioration:** voisins obtenus en faisant basculer simultanément un ensemble arbitraire de sommets vers la couleur  $i$

Le nombre de voisins d'une configuration est alors  $k \cdot 2^n$ : exponentiel !

## Résultats:

- on peut trouver un voisin améliorant en temps polynomial s'il existe
- le minimum local atteint approche l'optimum à un facteur 2

# Classification par recherche locale

Pour trouver un voisin améliorant de l'étiquetage  $\ell$  du graphe  $G$  on utilise le problème Min-Cut suivant:

# Classification par recherche locale

Pour trouver un voisin améliorant de l'étiquetage  $\ell$  du graphe  $G$  on utilise le problème Min-Cut suivant:

On construit un graphe  $G'$  de sommets  $V' = V \cup \{s, t\}$  (source et puits)

# Classification par recherche locale

Pour trouver un voisin améliorant de l'étiquetage  $\ell$  du graphe  $G$  on utilise le problème Min-Cut suivant:

On construit un graphe  $G'$  de sommets  $V' = V \cup \{s, t\}$  (source et puits)  $(s, x)$  de capacité  $v_x^{\ell(x)}$  si  $\ell(x) \neq i$ ,  $\infty$  sinon: si  $x$  avec  $t$  il garde sa couleur

# Classification par recherche locale

Pour trouver un voisin améliorant de l'étiquetage  $\ell$  du graphe  $G$  on utilise le problème Min-Cut suivant:

On construit un graphe  $G'$  de sommets  $V' = V \cup \{s, t\}$  (source et puits)  
 $(s, x)$  de capacité  $v_x^{\ell(x)}$  si  $\ell(x) \neq i$ ,  $\infty$  sinon: si  $x$  avec  $t$  il garde sa couleur  
 $(x, t)$  de capacité  $v_x^i$ : si  $x$  est avec  $s$  il prend la couleur  $i$  et paye  $v_x^i$

# Classification par recherche locale

Pour trouver un voisin améliorant de l'étiquetage  $\ell$  du graphe  $G$  on utilise le problème Min-Cut suivant:

On construit un graphe  $G'$  de sommets  $V' = V \cup \{s, t\}$  (source et puits)

- $(s, x)$  de capacité  $v_x^{\ell(x)}$  si  $\ell(x) \neq i$ ,  $\infty$  sinon: si  $x$  avec  $t$  il garde sa couleur
- $(x, t)$  de capacité  $v_x^i$ : si  $x$  est avec  $s$  il prend la couleur  $i$  et paye  $v_x^i$
- $(x, y)$  de capacité  $p_{xy}$  si  $\ell(x) = \ell(y)$  ou  $\ell(x) = i$  ou  $\ell(y) = i$ : on payera  $p_{xy}$  si et seulement si  $x$  et  $y$  sont séparés par la coupure.

# Classification par recherche locale

Pour trouver un voisin améliorant de l'étiquetage  $\ell$  du graphe  $G$  on utilise le problème Min-Cut suivant:

On construit un graphe  $G'$  de sommets  $V' = V \cup \{s, t\}$  (source et puits)  
 $(s, x)$  de capacité  $v_x^{\ell(x)}$  si  $\ell(x) \neq i$ ,  $\infty$  sinon: si  $x$  avec  $t$  il garde sa couleur  
 $(x, t)$  de capacité  $v_x^i$ : si  $x$  est avec  $s$  il prend la couleur  $i$  et paye  $v_x^i$   
 $(x, y)$  de capacité  $p_{xy}$  si  $\ell(x) = \ell(y)$  ou  $\ell(x) = i$  ou  $\ell(y) = i$ : on payera  $p_{xy}$  si et seulement si  $x$  et  $y$  sont séparés par la coupure.

sinon à chaque arête  $xy$  de  $G$  tq  $\ell(x) \neq \ell(y) \neq i$ , on associe un nouveau sommet  $a$  et des arêtes  $(x, a)$ ,  $(a, y)$  et  $(a, s)$  de capacités  $p_{xy}$ :

dans la coupe min on pourra toujours placer  $a$  de façon à payer  $p_{xy}$  au plus une fois: on payera sauf si  $x$ ,  $y$  et  $s$  sont ensembles, *i.e.* sauf si les 2 sommets prennent la couleur  $i$ .

# Classification par recherche locale

Pour trouver un voisin améliorant de l'étiquetage  $\ell$  du graphe  $G$  on utilise le problème Min-Cut suivant:

On construit un graphe  $G'$  de sommets  $V' = V \cup \{s, t\}$  (source et puits)  
 $(s, x)$  de capacité  $v_x^{\ell(x)}$  si  $\ell(x) \neq i$ ,  $\infty$  sinon: si  $x$  avec  $t$  il garde sa couleur  
 $(x, t)$  de capacité  $v_x^i$ : si  $x$  est avec  $s$  il prend la couleur  $i$  et paye  $v_x^i$   
 $(x, y)$  de capacité  $p_{xy}$  si  $\ell(x) = \ell(y)$  ou  $\ell(x) = i$  ou  $\ell(y) = i$ : on payera  $p_{xy}$  si et seulement si  $x$  et  $y$  sont séparés par la coupure.

sinon à chaque arête  $xy$  de  $G$  tq  $\ell(x) \neq \ell(y) \neq i$ , on associe un nouveau sommet  $a$  et des arêtes  $(x, a)$ ,  $(a, y)$  et  $(a, s)$  de capacités  $p_{xy}$ :

dans la coupe min on pourra toujours placer  $a$  de façon à payer  $p_{xy}$  au plus une fois: on payera sauf si  $x$ ,  $y$  et  $s$  sont ensembles, *i.e.* sauf si les 2 sommets prennent la couleur  $i$ .

**Proposition:** La coupe minimum de  $G'$  donne le coût du meilleur réétiquetage de sommets de  $(G, \ell)$  par l'étiquette  $i$ .

# Classification par recherche locale

Il reste à voir que l'optimum local obtenu par réétiquetages successifs est à un facteur 2 de l'optimum.

# Classification par recherche locale

Il reste à voir que l'optimum local obtenu par réétiquetages successifs est à un facteur 2 de l'optimum.

On considère un optimum global  $\ell^*$  et un optimum local  $\ell$ .

# Classification par recherche locale

Il reste à voir que l'optimum local obtenu par réétiquetages successifs est à un facteur 2 de l'optimum.

On considère un optimum global  $l^*$  et un optimum local  $l$ .

Pour chaque  $i$ , on peut réétiqueter  $l$  en  $l_i$  avec  $l_i(x) = i$  si  $x \in V_i^*$  où  $V_i^*$  est l'ensemble des sommets d'étiquette  $i$  dans l'optimum.

# Classification par recherche locale

Il reste à voir que l'optimum local obtenu par réétiquetages successifs est à un facteur 2 de l'optimum.

On considère un optimum global  $\ell^*$  et un optimum local  $\ell$ .

Pour chaque  $i$ , on peut réétiqueter  $\ell$  en  $\ell_i$  avec  $\ell_i(x) = i$  si  $x \in V_i^*$  où  $V_i^*$  est l'ensemble des sommets d'étiquette  $i$  dans l'optimum.

Comme  $\ell$  est localement optimal,  $\Phi(\ell) \leq \Phi(\ell_i)$ , ce qui donne:

$$0 \leq \Phi(\ell_i) - \Phi(\ell) \leq \sum_{x \in V_i^*} (v_x^i - v_x^{\ell(x)}) + \sum_{x \in V_i^*, y \notin V_i^*} \overset{\text{contribution des arêtes avec un pied dans } V_i^*}{\text{à } \Phi(\ell_i)} p_{xy} - \sum_{x \in V_i^*, \ell(x) \neq \ell(y)} \overset{\text{à } \Phi(\ell)}{p_{xy}}$$

# Classification par recherche locale

Il reste à voir que l'optimum local obtenu par réétiquetages successifs est à un facteur 2 de l'optimum.

On considère un optimum global  $\ell^*$  et un optimum local  $\ell$ .

Pour chaque  $i$ , on peut réétiqueter  $\ell$  en  $\ell_i$  avec  $\ell_i(x) = i$  si  $x \in V_i^*$  où  $V_i^*$  est l'ensemble des sommets d'étiquette  $i$  dans l'optimum.

Comme  $\ell$  est localement optimal,  $\Phi(\ell) \leq \Phi(\ell_i)$ , ce qui donne:

$$0 \leq \Phi(\ell_i) - \Phi(\ell) \leq \sum_{x \in V_i^*} (v_x^i - v_x^{\ell(x)}) + \sum_{x \in V_i^*, y \notin V_i^*} p_{xy} - \sum_{x \in V_i^*, \ell(x) \neq \ell(y)} p_{xy}$$

à  $\Phi(\ell_i)$ 
à  $\Phi(\ell)$

contribution des arêtes avec un pied dans  $V_i^*$

On somme ces inégalités pour tout  $i$ :

$$\sum_i \left[ \sum_{x \in V_i^*} v_x^i + \sum_{x \in V_i^*, y \notin V_i^*} p_{xy} \right] \geq \sum_i \left[ \sum_{x \in V_i^*} v_x^{\ell(x)} + \sum_{x \in V_i^*, \ell(x) \neq \ell(y)} p_{xy} \right]$$

# Classification par recherche locale

Il reste à voir que l'optimum local obtenu par réétiquetages successifs est à un facteur 2 de l'optimum.

On considère un optimum global  $\ell^*$  et un optimum local  $\ell$ .

Pour chaque  $i$ , on peut réétiqueter  $\ell$  en  $\ell_i$  avec  $\ell_i(x) = i$  si  $x \in V_i^*$  où  $V_i^*$  est l'ensemble des sommets d'étiquette  $i$  dans l'optimum.

Comme  $\ell$  est localement optimal,  $\Phi(\ell) \leq \Phi(\ell_i)$ , ce qui donne:

$$0 \leq \Phi(\ell_i) - \Phi(\ell) \leq \sum_{x \in V_i^*} (v_x^i - v_x^{\ell(x)}) + \sum_{x \in V_i^*, y \notin V_i^*} p_{xy} - \sum_{x \in V_i^*, \ell(x) \neq \ell(y)} p_{xy}$$

à  $\Phi(\ell_i)$ 
à  $\Phi(\ell)$ 
contribution des arêtes avec un pied dans  $V_i^*$

On somme ces inégalités pour tout  $i$ :

$$\sum_i \left[ \sum_{x \in V_i^*} v_x^i + \sum_{x \in V_i^*, y \notin V_i^*} p_{xy} \right] \geq \sum_i \left[ \sum_{x \in V_i^*} v_x^{\ell(x)} + \sum_{x \in V_i^*, \ell(x) \neq \ell(y)} p_{xy} \right]$$

**Proposition:**  $\Phi(\ell^*) \geq \frac{1}{2} \Phi(\ell)$ , la recherche locale est 2-approchée.

# Cours 9: Recherche locale

- Heuristiques de recherche locale
- Réseau de Hopfield ou verre de spin
- Max-Cut, NP-complétude, recherche locale.
- Classification par recherche locale

**Conclusion:** Heuristique facile à mettre en place  
mais difficile de dire quand ça marche bien (rarement) !

Recuit simulé ok quand il s'agit juste d'aplanir  
un relief un peu rugueux

Sinon le secret est dans le choix du voisinage...  
ne pas avoir peur de bcp travailler à chaque pas !