Approx

7 septembre 2021 13:23

Soit P un pb de minimisation
↳ trouver une solution faisable (valide) qui minimise un certain critère

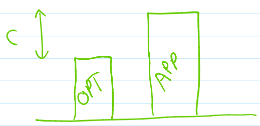
- Pour une instance X de P, on écrit $OPT(X)$ pour dénoter la valeur d'une sol. opt. pour X.
- Soit A un algo pour P, i.e. A retourne tjrs une sol. faisable ($\forall$ instance X de P)

On dénote par $APP(A, X)$ la valeur de la sol. retournée par A sur entrée X.

- On dit que A est une c-approximation à P si, $\forall$ instance X de P

① A s'exécute en temps polynomial, donc $O(|X|^k)$ k = constante

② $APP(A, X) \leq c \cdot OPT(X)$

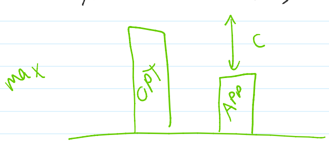


$c \geq 1$

- Si P est un pb de maximisation, A est une c-approx. si $\forall$ instance X de P

① A s'exécute en temps poly.

② $APP(A, X) \geq c \cdot OPT(X)$



$c \leq 1$

Note : si A et X sont claires, on écrit OPT et APP

Note : c peut être
- une constante
- une fonction de n $(n = |X|)$

ex : SET-COVER admet une $\log_2 n$-approx.

MAX-SAT (satisfaisabilité booléenne)

$C_1 = x_1$
$C_2 = x_2$

Clause = variables bool. liées par des $\vee$ (ou)

ex : $C = x_1 \vee \bar{x}_2 \vee x_3$

$C_3 = x_1 \vee (x_2 \wedge x_3) \equiv (x_1 \vee x_2) \wedge (x_1 \vee x_3)$

Une variable x_i peut être positive x_i, négative $\bar{x}_i$

$C_1 \wedge C_2 \wedge C_3$

$(x_1 \wedge x_1) \wedge (x_2 \wedge x_2) \wedge$
$(\bar{x}_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3)$

Une assignation attribue une valeur T ou F à chaque x_i

Elle satisfait une clause C si cette dernière évalue à true.

ex : $x_1 \vee \bar{x}_2 \vee x_3$ $x_1 = F$ $x_2 = F$ $x_3 = T$

$F \vee \bar{F} \vee T \equiv F \vee T \vee T \equiv T$

La seule façon de ne pas satisfaire C est $x_1 = F$ $x_2 = T$ $x_3 = F$

MAX-SAT :

Entrée : ensemble de clauses $C_1, C_2, \ldots, C_m$ sur variables $x_1, x_2, \ldots, x_n$

Sortie : ...

Entrée: ensemble de clauses $C_1, C_2, \dots, C_m$ sur variables $x_1, x_2, \dots, x_n$

Sortie: une assignation des x_i qui maximise le # de clauses satisfaites

maxsat$(C_1, C_2, \dots, C_m, x_1, x_2, \dots, x_n)$

    Soit A l'assign. t.q. $x_1 = T, x_2 = T, \dots, x_n = T$
    si A satisfait au moins $m/2$ clause
        return A

    soit $\bar{A}$ l'assign. t.q. $x_1 = F, x_2 = F, \dots, x_n = F$
    return $\bar{A}$

Théorème: maxsat est une $\frac{1}{2}$-approx.

Sketch: On a $OPT \leq m$ car il y a m clauses.

Pour APP, si l'algo retourne A, alors $APP \geq m/2$ (c'est vérifié explicitement).

Si l'algo retourne $\bar{A}$, c'est parce que A satisfait $< m/2$ clauses.
On remarque que tout ce que A ne sat. pas est sat. par $\bar{A}$

$$C_i = \bar{x}_a \vee \bar{x}_b \vee \bar{x}_c \vee \bar{x}_d$$

Donc, $\bar{A}$ satisfait $\geq m/2$ clauses.

Dans les 2 cas, on a $APP \geq m/2$

$OPT \leq m$

$APP \geq m/2$

$\Rightarrow APP \geq m/2 \geq \frac{OPT}{2} \geq \frac{1}{2} OPT$

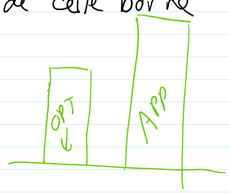
Technique fondamentale d'analyse d'approx.

1. Trouver une borne sur OPT
2. Montrer que APP est "proche" de cette borne

- Minimisation

  $OPT \geq k$

  $APP \leq c \cdot k$ $\quad c \geq 1$

  $\Rightarrow APP \leq c \cdot k \leq c \cdot OPT$

- Maximisation

  $OPT \leq k$

  $APP \geq c \cdot k$ $\quad c \leq 1$

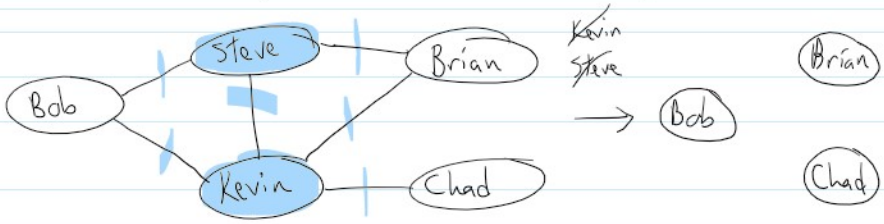
  $\Rightarrow APP \geq c \cdot k \geq c \cdot OPT$

VERTEX-COVER

Soit $G = (V, E)$ un graphe. Un ensemble $X \subseteq V$ est <u>couvrant</u> si $\forall uv \in E$, $u \in X$ ou $v \in X$ (ou les deux)

<u>Entrée</u> : un graphe $G=(V,E)$
<u>Sortie</u> : un ens. couvrant $X \subseteq V$ de taille min.

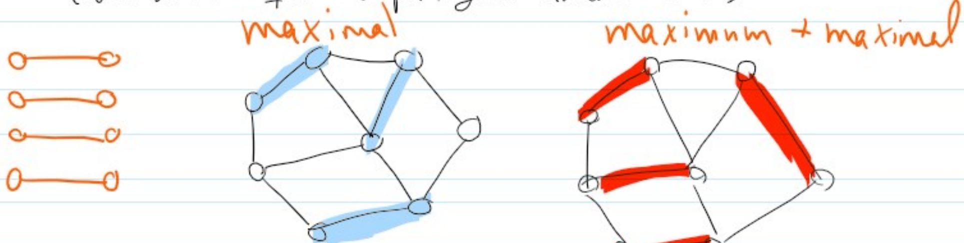
ex : vous avez un bar, où certaines paires de clients qui se battent si présent en même.
Vous voulez bloquer l'entrée a un # min pour 0 bataille.



① Trouver une borne sur OPT $(OPT \geq k)$

La taille d'un <u>matching maximal</u> est une borne sur OPT.

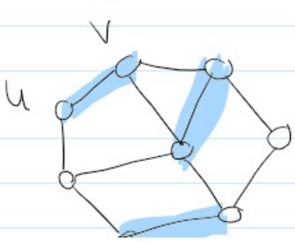
<u>Déf.</u> : sur un graphe $G=(V,E)$, un matching est un ensemble d'arêtes $M \subseteq E$ tel que $\forall uv, xy \in M$, $u \neq x, u \neq y, v \neq x, v \neq y$
(des arêtes qui ne partagent aucun bout)



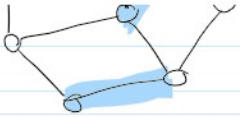
Un matching est <u>maximal</u> si on ne peut plus lui ajouter d'arêtes.
M est maximal si $\forall uv \in E \setminus M$, $M \cup \{uv\}$ n'est pas un matching.

<u>Lemme</u> : soit M un matching maximal, et soit X un ens. couvrant minimum.
Alors $|X| \geq |M|$. $(OPT \geq |M|)$

<u>Sketch</u> : X doit couvrir chaque arête de M.



Puisque chaque arête de M ne partage pas de sommet, X doit contenir au moins un sommet de chaque arête de M. ∎



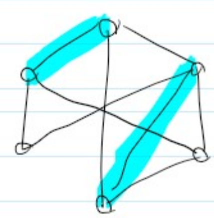
Voici une 2-approx.

vc-matching$(G=(V,E))$
$X = \{\}$
tant que $|E| > 0$
soit $uv \in E$
Ajouter u à X, ajouter v à X
Enlever u et v de G (avec leurs arêtes)
return X

À prouver : le X retourné est faisable (ens. couvrant)
$APP \leq 2 \cdot OPT$

<u>Théorème</u> : l'algo vc-matching est une 2-approx. au pb VERTEX-COVER

<u>Preuve</u> : Soit X l'ens. retourné par l'algo.
On a que X contient les sommets d'un matching maximal M.
On sait que $OPT \geq |M|$.
D'un autre côté, $APP = |X| = 2|M|$.
$\Rightarrow APP = 2|M| \leq 2\,OPT$.
Reste à prouver : X est un ens. couvrant. ▢



<u>Analyse "serrée"</u> (tight analysis)

Est-ce notre analyse sur un algo peut être améliorée ?

ex : est-ce que vc-matching est une 1.5-approx ?
Rep : non, vc-matching n'est pas mieux qu'une 2-approx

$OPT = 2$ $\Rightarrow \exists$ instances telles que

$OPT = 2$
$APP = 4$ $\Rightarrow$ $\exists$ instances telles que $APP = 2 \cdot OPT$

Revenons à 3-SET-COVER

algo(U, S)

pour chaque $u_i \in U$
  si u_i n'est pas couvert
    ajouter un $S' \in S$ qui couvre u_i

$OPT \geq n/3$
$APP \leq n-2$ $\Rightarrow APP < 3\,OPT$

Q: exemple(s) montrant que $APP \leq c \cdot OPT$, $c < 3$ est impossible

$U = \{u_1, \ldots, u_n\}$

$S_1 = u_1, u_2, u_3$
$S_2 = u_1, u_2, u_4$
$S_3 = u_1, u_2, u_5$
$\ldots$
$S_{n-2} = u_1, u_2, u_n$
$S_1^* = u_1, u_2, u_3$
$S_2^* = u_4, u_5, u_6$
$\ldots$
$S_{n/3}^* = u_{n-2}, u_{n-1}, u_n$

Puisque l'algo d'approx fait des choix arbitraires, il pourrait retourner
$\{S_1, S_2, \ldots, S_{n-2}\}$

Par contre, OPT est
$\{S_1^*, S_2^*, \ldots, S_{n/3}^*\}$

$APP = n-2$

$OPT = n/3$

$APP = x \cdot OPT$
$n - 2 = x \cdot n/3 \Rightarrow x = \frac{n-2}{n/3} = \frac{n}{n/3} - \frac{2}{n/3} = 3 - \frac{6}{n}$

Donc $APP = \left(3 - \frac{6}{n}\right) \cdot OPT$

- On dit qu'une analyse de c-approx. d'un algo A est "serrée" si $\forall \varepsilon > 0$, il existe une instance

X telle que

$$APP(A, X) > (c - \varepsilon) \cdot OPT(X)$$

(pb de minimisation)