Q0: De ce folosim algoritmi aproximativi?

* Problemele din NP-hard (cel putin la fel de dificile ca cele din NP-C) nu au algoritmi fezabili (in timp polinomial) pt determinarea optimului. Asa ca una dintre solutiile de compromis ar fi sa gasim o solutie “aproape” optima.

Q1: ce este factorul de aproximare pentru un algoritm?

Fie ALG - solutia noastra, OPT - solutia optima

In cazul unei probleme de minim, o constanta *c* (supraunitara) se numeste factor de aproximare daca:

OPT≤ALG≤c\*OPT

In cazul unei probleme de maxim, o constanta *c* (subunitara) se numeste factor de aproximare daca:

OPT≥ALG≥c\*OPT.

Q1.1 In cazul unei probleme de minim: Un algoritm 2-aproximativ poate fi numit 3-aproximativ?

Da - deoarece daca am un algoritm c-aproximativ si un c’>c vom avea

ALG≤c\*OPT ≤c’\*OPT - deci ALG≤c’\*OPT

Q1.2 Cum putem sa justidicam ca un *c* gasit este tight bound?

Tb sa arat ca daca, in cazul problemelor de minim, iau un c’<c (sau c’>c in cazul celor de maxim) atunci algoritmul meu nu are cum sa fie c’ aproximativ.

Mai simplu este sa gasesc o intrare I pentru care ALG(I) = c\*OPT(I)

Justificare:  
Sa zicem ca avem un algoritm (pt problema de minim) pe care l-am demonstrat ca ar fi 2 aproximativ. ALG(I)<=2\*OPT(I) pentru orice input I.

Ca sa arat ca acel factor este tight, tb sa gasesc o intrare I’ a.i. ALG(I’)=2\*OPT(I’)  
Astfel arat ca algoritmul nu ete 1,5 aproximativ, sau 1,75, sau 1,9999 aproximativ deoarece pt intrarea I’ ALG(I’)>1.999\*OPT(I’)

Probleme:

1. Avem următorul scenariu: Avem *n* colete de transportat, fiecare avand greutatea de *w1, w2,...,wn.* Pentru a le transporta, putem folosi un număr de camioane, fiecare avand capacitatea de transport *G*. Presupunem că *wi≤G*, pentru orice *i*. Ne dorim sa minimizăm numărul de camioane folosite. Considerăm următorul plan de încărcare a camioanelor:

Odată ce avem la dispoziție un camion pt a fi încărcat, iterăm prin mulțimea coletelor, incărcându-le in camion, până când dăm peste primul colet ce nu mai incape. În acel moment considerat că am terminat de încărcat camionul curent și trecem la următorul camion, prima dată încărcând coletul care nu a mai încăput în cel precedent.

1. Arătați, printr-un exemplu simplu, că metoda de mai sus nu furnizează soluția optimă.
2. Arătați totuși că soluția de mai sus este un algoritm 2-aproximativ pentru problema noastră.

Raspuns:

1. {1,4,1,4}; G=4  
   ALG:(1);(4);(1);(4)  
   OPT:(1,1); (4); (4)
2. Fie W - suma totala a greutatilor coletelor;   
   Fie OPT numarul minim de camioane folosite.

Lower bound pt OPT?

OPT>=W/G

Numaruld e camioane folosite poate fi par sau impar.  
numarul de camioane folosite de algoritmul nostra = 2\*c+1

W este transportata de 2\*c+1 camioane;

2\*c+1 camioane transporta > c\*G cantitate (eoarece 2 camioane transporta >G)  
W>c\*G

W/G>c

OPT>=W/G>c

Deci OPT>c. Deci in cazul optim voi avea nevoie de cel putin c+1 camioane.

OPT>=c+1  
2\*OPT>=2\*c+2>2c+1=ALG

**2\*OPT>ALG>=OPT** - deci algoritmul descris mai sus este 2 aproximativ.

2) Dat fiind algoritmul Load-Balance (Cursul 2, slide 19) să se stabilească dacă următoarea afirmație este adevărată sau falsă.

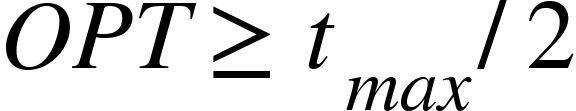
”Pentru orice instanța a problemei de Load-Balace, exista o anumită ordine a procesării activităților astfel încât algoritmul de tip greedy să dea o soluție optimă”

Dacă afirmația este adevărată, oferiți o demonstrație, altfel, găsiți un contraexemplu.

Da. Ne uitam la solutia optima, apoi “rulam” algoritmul nostru pas cu pas, vedem care este masina *i* care urmeaza sa primeasca o activitate si “avem grija” ca acea activitate sa fie “next in line”. Dar efectiv gasirea unei astfel e permutari “fericite” este ea o problema NP-hard.

3) Fie Problema Load Balance, dar cu următoarea modificare: Avem *n* joburi si *m* mașini, doar că pentru primele *k* mașini timpul de lucru al unei activități este înjumătățit. Să se găsească un algoritm bazat pe tehnica greedy care furnizeaza o soluție de cel mult 3xOPT.

* pentru fiecare jpb *j* cautam acea masina *i* astfel incat jobul *j* sa se termine cat mai devreme. Daca acea masina este din primele k, atunci se adauga la load-ul ei cantitatea tj/2. Altfel, se adauga tj



<math xmlns="http://www.w3.org/1998/Math/MathML"><mi>O</mi><mi>P</mi><mi>T</mi><mo>&#x2265;</mo><mfrac><mn>1</mn><mrow><mi>m</mi><mo>-</mo><mi>k</mi><mo>+</mo><mn>2</mn><mi>k</mi></mrow></mfrac><munder><mo>&#x2211;</mo><mrow><mn>1</mn><mo>&#x2264;</mo><mi>j</mi><mo>&#x2264;</mo><mi>n</mi></mrow></munder><msub><mi>t</mi><mi>j</mi></msub><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><mi>t</mi><mspace linebreak="newline"/><mi>D</mi><mi>e</mi><mi>m</mi><mi>o</mi><mi>n</mi><mi>s</mi><mi>t</mi><mi>r</mi><mi>a</mi><mi>t</mi><mi>i</mi><mi>e</mi><mo>:</mo><mspace linebreak="newline"/><mi>p</mi><mi>r</mi><mi>e</mi><mi>s</mi><mi>u</mi><mi>p</mi><mi>u</mi><mi>n</mi><mi>e</mi><mi>m</mi><mo>&#xA0;</mo><mi>c</mi><mi>a</mi><mo>&#xA0;</mo><mi>f</mi><mi>i</mi><mi>e</mi><mi>c</mi><mi>a</mi><mi>r</mi><mi>e</mi><mo>&#xA0;</mo><mi>m</mi><mi>a</mi><mi>sin</mi><mi>a</mi><mo>&#xA0;</mo><mi>l</mi><mi>u</mi><mi>c</mi><mi>r</mi><mi>e</mi><mi>a</mi><mi>z</mi><mi>a</mi><mo>&#xA0;</mo><mi>m</mi><mi>a</mi><mi>i</mi><mo>&#xA0;</mo><mi>p</mi><mi>u</mi><mi>t</mi><mi>i</mi><mi>n</mi><mo>&#xA0;</mo><mi>d</mi><mi>e</mi><mi>c</mi><mi>a</mi><mi>t</mi><mo>&#xA0;</mo><mi>t</mi><mspace linebreak="newline"/><munder><mo>&#x2211;</mo><mrow><mn>1</mn><mo>&#x2264;</mo><mi>j</mi><mo>&#x2264;</mo><mi>n</mi></mrow></munder><msub><mi>t</mi><mi>j</mi></msub><mo>=</mo><mi>t</mi><mfenced><mrow><mi>m</mi><mo>-</mo><mi>k</mi><mo>+</mo><mn>2</mn><mi>k</mi></mrow></mfenced><mo>=</mo><mfenced><mrow><mi>m</mi><mo>-</mo><mi>k</mi></mrow></mfenced><mo>&#xB7;</mo><mi>t</mi><mo>+</mo><mn>2</mn><mi>k</mi><mo>&#xB7;</mo><mi>t</mi><mo>&gt;</mo><munder><mo>&#x2211;</mo><mrow><mn>1</mn><mo>&#x2264;</mo><mi>j</mi><mo>&#x2264;</mo><mi>n</mi></mrow></munder><msub><mi>t</mi><mi>j</mi></msub><mo>&#xA0;</mo><mo>-</mo><mo>&#xA0;</mo><mi>t</mi><mi>i</mi><mi>m</mi><mi>p</mi><mi>u</mi><mi>l</mi><mo>&#xA0;</mo><mi>t</mi><mi>o</mi><mi>t</mi><mi>a</mi><mi>l</mi><mo>&#xA0;</mo><mi>d</mi><mi>e</mi><mo>&#xA0;</mo><mi>l</mi><mi>u</mi><mi>c</mi><mi>r</mi><mi>u</mi><mo>&#xA0;</mo><mi>a</mi><mi>l</mi><mo>&#xA0;</mo><mi>t</mi><mi>u</mi><mi>t</mi><mi>u</mi><mi>r</mi><mi>o</mi><mi>r</mi><mo>&#xA0;</mo><mi>a</mi><mi>c</mi><mi>t</mi><mi>v</mi><mi>i</mi><mi>t</mi><mi>a</mi><mi>t</mi><mi>i</mi><mi>l</mi><mi>o</mi><mi>r</mi><mspace linebreak="newline"/><mi>C</mi><mi>o</mi><mi>n</mi><mi>t</mi><mi>r</mi><mi>a</mi><mi>d</mi><mi>i</mi><mi>c</mi><mi>t</mi><mi>i</mi><mi>e</mi><mo>!</mo><mspace linebreak="newline"/><mi>D</mi><mi>e</mi><mi>c</mi><mi>i</mi><mo>&#xA0;</mo><mi>n</mi><mi>u</mi><mo>&#xA0;</mo><mi>s</mi><mi>e</mi><mo>&#xA0;</mo><mi>p</mi><mi>o</mi><mi>a</mi><mi>t</mi><mi>e</mi><mo>&#xA0;</mo><mi>c</mi><mi>a</mi><mo>&#xA0;</mo><mi>f</mi><mi>i</mi><mi>e</mi><mi>c</mi><mi>a</mi><mi>r</mi><mi>e</mi><mo>&#xA0;</mo><mi>m</mi><mi>a</mi><mi>sin</mi><mi>a</mi><mo>&#xA0;</mo><mi>s</mi><mi>a</mi><mo>&#xA0;</mo><mi>l</mi><mi>u</mi><mi>c</mi><mi>r</mi><mi>e</mi><mi>z</mi><mi>e</mi><mo>&#xA0;</mo><mo>&lt;</mo><mo>&#xA0;</mo><mi>t</mi><mo>.</mo><mo>&#xA0;</mo><mi>D</mi><mi>e</mi><mi>c</mi><mi>i</mi><mo>&#xA0;</mo><mi>O</mi><mi>P</mi><mi>T</mi><mo>&#x2265;</mo><mi>t</mi></math>