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)

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. G=50; W={25,26,25}

OPT=(25;25),(26)

ALG=(25); (26); (25)

1. OPT>=1/G \* ∑(wi)

Algoritmul nostru lucreaza astfel incat 2 caminoane consecutive transporta cel putin greutate G.

numarul de perechi de camioane <=1/G \* ∑(wi)<=OPT

ALG/2<=OPT;

ALG<=2\*OPT

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.

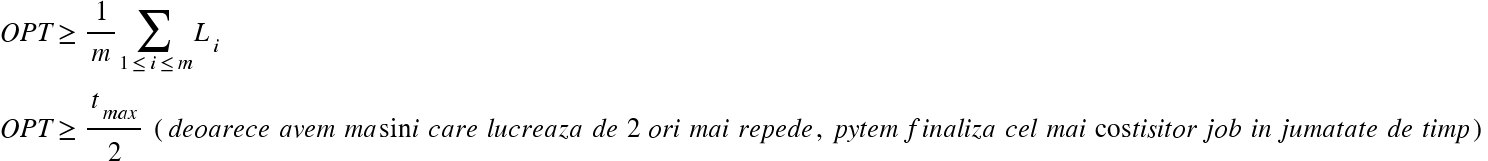
Rasouns: DA! Fie OPT o incarcatura optima. Daca job-urile vin asezate astfel incat atunci cand masini i, cu incarcarea cea mai mica, ii vine la rand exact jobul care ii este asociat in OPT.

Putem duce problema de load balance in problema de permutare!

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.

Raspuns: Consideram algoritmul de pe slide-ul 19, cu urmatoarele modificari

daca masina *i* (cea aleasa la linia 5 din algoritm) este una dintre cele *k* pasini, atunci la linia 7 vom avea Li+=tj/2. Altfel Li+=tj (daca masina *i* nu este printre cele *k* masini)



<math xmlns="http://www.w3.org/1998/Math/MathML"><mi>f</mi><mi>i</mi><mi>e</mi><mo>&#xA0;</mo><mi>q</mi><mo>&#xA0;</mo><mo>-</mo><mo>&#xA0;</mo><mi>m</mi><mi>a</mi><mi>sin</mi><mi>a</mi><mo>&#xA0;</mo><mi>c</mi><mi>u</mi><mo>&#xA0;</mo><mi>i</mi><mi>n</mi><mi>c</mi><mi>a</mi><mi>r</mi><mi>c</mi><mi>a</mi><mi>t</mi><mi>u</mi><mi>r</mi><mi>a</mi><mo>&#xA0;</mo><mi>m</mi><mi>a</mi><mi>x</mi><mi>i</mi><mi>m</mi><mi>a</mi><mo>&#xA0;</mo><mi>l</mi><mi>a</mi><mo>&#xA0;</mo><mi>s</mi><mi>f</mi><mi>a</mi><mi>r</mi><mi>s</mi><mi>i</mi><mi>t</mi><mi>u</mi><mi>l</mi><mo>&#xA0;</mo><mi>a</mi><mi>l</mi><mi>g</mi><mi>o</mi><mi>r</mi><mi>i</mi><mi>t</mi><mi>m</mi><mi>u</mi><mi>l</mi><mi>u</mi><mi>i</mi><mo>&#xA0;</mo><mi>n</mi><mi>o</mi><mi>s</mi><mi>t</mi><mi>u</mi><mspace linebreak="newline"/><mi>d</mi><mi>e</mi><mi>c</mi><mi>i</mi><mo>&#xA0;</mo><mi>A</mi><mi>L</mi><mi>G</mi><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><mi>L</mi><mi>q</mi><mspace linebreak="newline"/><mi>F</mi><mi>i</mi><mi>e</mi><mo>&#xA0;</mo><mi>p</mi><mo>&#xA0;</mo><mo>-</mo><mo>&#xA0;</mo><mi>u</mi><mi>l</mi><mi>t</mi><mi>i</mi><mi>m</mi><mi>u</mi><mi>l</mi><mo>&#xA0;</mo><mi>j</mi><mi>o</mi><mi>b</mi><mo>&#xA0;</mo><mi>a</mi><mi>d</mi><mi>a</mi><mi>a</mi><mi>u</mi><mi>g</mi><mi>a</mi><mi>t</mi><mo>&#xA0;</mo><mi>l</mi><mi>a</mi><mo>&#xA0;</mo><mi>L</mi><mi>q</mi><mspace linebreak="newline"/><mi>f</mi><mi>i</mi><mi>e</mi><mo>&#xA0;</mo><mi>L</mi><mo>'</mo><mi>i</mi><mo>&#xA0;</mo><mo>-</mo><mo>&#xA0;</mo><mi>l</mi><mi>o</mi><mi>a</mi><mi>d</mi><mi>u</mi><mi>l</mi><mo>&#xA0;</mo><mi>m</mi><mi>a</mi><mi>sin</mi><mi>i</mi><mi>i</mi><mo>&#xA0;</mo><mo>'</mo><mi>i</mi><mo>'</mo><mo>&#xA0;</mo><mi>f</mi><mi>i</mi><mi>x</mi><mo>&#xA0;</mo><mi>i</mi><mi>n</mi><mi>a</mi><mi>i</mi><mi>n</mi><mi>t</mi><mi>e</mi><mo>&#xA0;</mo><mi>d</mi><mi>e</mi><mo>&#xA0;</mo><mi>a</mi><mo>&#xA0;</mo><mi>f</mi><mi>i</mi><mo>&#xA0;</mo><mi>a</mi><mi>d</mi><mi>a</mi><mi>u</mi><mi>g</mi><mi>a</mi><mi>t</mi><mi>a</mi><mo>&#xA0;</mo><mi>a</mi><mi>c</mi><mi>t</mi><mi>i</mi><mi>v</mi><mi>i</mi><mi>t</mi><mi>a</mi><mi>t</mi><mi>e</mi><mo>&#xA0;</mo><mi>p</mi><mo>&#xA0;</mo><mi>l</mi><mi>a</mi><mo>&#xA0;</mo><mi>m</mi><mi>a</mi><mi>sin</mi><mi>a</mi><mo>&#xA0;</mo><mi>q</mi><mo>.</mo><mspace linebreak="newline"/><mi>A</mi><mi>L</mi><mi>G</mi><mo>=</mo><mi>L</mi><mi>q</mi><mo>&#x2264;</mo><mi>L</mi><mo>'</mo><mi>q</mi><mo>+</mo><msub><mi>t</mi><mi>p</mi></msub><mo>&#xA0;</mo><mfenced><mrow><mi>d</mi><mi>e</mi><mi>o</mi><mi>a</mi><mi>r</mi><mi>e</mi><mi>c</mi><mi>e</mi><mo>&#xA0;</mo><mi>u</mi><mi>l</mi><mi>t</mi><mi>i</mi><mi>m</mi><mi>u</mi><mi>l</mi><mo>&#xA0;</mo><mi>t</mi><mi>e</mi><mi>r</mi><mi>m</mi><mi>e</mi><mi>n</mi><mo>&#xA0;</mo><mi>p</mi><mi>o</mi><mi>a</mi><mi>t</mi><mi>e</mi><mo>&#xA0;</mo><mi>f</mi><mi>i</mi><mo>&#xA0;</mo><mo>&#xA0;</mo><msub><mi>t</mi><mi>p</mi></msub><mo>&#xA0;</mo><mi>d</mi><mi>a</mi><mi>c</mi><mi>a</mi><mo>&#xA0;</mo><mi>q</mi><mo>&gt;</mo><mi>k</mi><mo>&#xA0;</mo><mi>s</mi><mi>a</mi><mi>u</mi><mo>&#xA0;</mo><msub><mi>t</mi><mi>p</mi></msub><mo>/</mo><mn>2</mn><mo>,</mo><mo>&#xA0;</mo><mi>a</mi><mi>l</mi><mi>t</mi><mi>f</mi><mi>e</mi><mi>l</mi></mrow></mfenced><mspace linebreak="newline"/><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#x2264;</mo><mfrac><mn>1</mn><mi>m</mi></mfrac><munder><mo>&#x2211;</mo><mrow><mn>1</mn><mo>&#x2264;</mo><mi>i</mi><mo>&#x2264;</mo><mi>m</mi></mrow></munder><mi>L</mi><mo>'</mo><mi>i</mi><mo>&#xA0;</mo><mo>+</mo><msub><mi>t</mi><mrow><mi>m</mi><mi>a</mi><mi>x</mi></mrow></msub><mo>&#xA0;</mo><mo>&#x2264;</mo><mfrac><mn>1</mn><mi>m</mi></mfrac><munder><mo>&#x2211;</mo><mrow><mn>1</mn><mo>&#x2264;</mo><mi>i</mi><mo>&#x2264;</mo><mi>m</mi></mrow></munder><mi>L</mi><mi>i</mi><mo>&#xA0;</mo><mo>+</mo><msub><mi>t</mi><mrow><mi>m</mi><mi>a</mi><mi>x</mi></mrow></msub><mo>&#x2264;</mo><mi>O</mi><mi>P</mi><mi>T</mi><mo>+</mo><mn>2</mn><mo>&#xB7;</mo><mi>O</mi><mi>P</mi><mi>T</mi><mo>=</mo><mn>3</mn><mo>&#xB7;</mo><mi>O</mi><mi>P</mi><mi>T</mi></math>

(2, 100)

1:100 - 50

2: 2 - 2

1: 2 - 1

2: 100 -100 XXXXX - nu e rezolvata!