Q0: In ce context 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 dar in timp polinomial.

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* se numeste factor de aproximare daca:

OPT≤ALG≤c\*OPT.

in cazul unei probleme de maxim

OPT≥ALG≥c\*OPT.

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

DA

Consecinta: pt un ALG sa gasim *c* - cat mai aproape de 1 (tight bound)

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

ar tb sa gasim o intrare *I* astfel incat:

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.

Rasounsuri:  
a) G=3; Colete {1,3,1}

Alg: (1);(3);(1);

OPT: (1,1);(3)

b) Fie ALG numarul de camioane folosite de catre algoritm. ALG poate fi de forma (2c) sau (2c+1). Ce cantitate de marfa transporta fiecare dintre cele ‘c’ perechi de camioane? Fiecare pereche de camioane va transporta o cantitate de marfa >G;

Ce cantitate vor transporta toate cele ‘c’ perechi? >c\*G;

Fie W - greutatea intregii cantitati de marfa.

si OPT - numarul minim de camioane ce poate fi folosit pt a transporta W.

OPT>=W/G

Toate cele ALG camioane transporta W;

Toate cele ALG camioane transpora >c\*G

W>c\*G;

OPT>=W/G>c

OPT>c; OPT>=c+1

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

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.

Raspuns:

Da - deoarce daca am avea acces la o solutie optima la dispozitie, putem genera o permutare a activitatilor astfel incat pseudocodul de la slide-ul 19 sa obtina fix acea solutie optima.Raspuns:

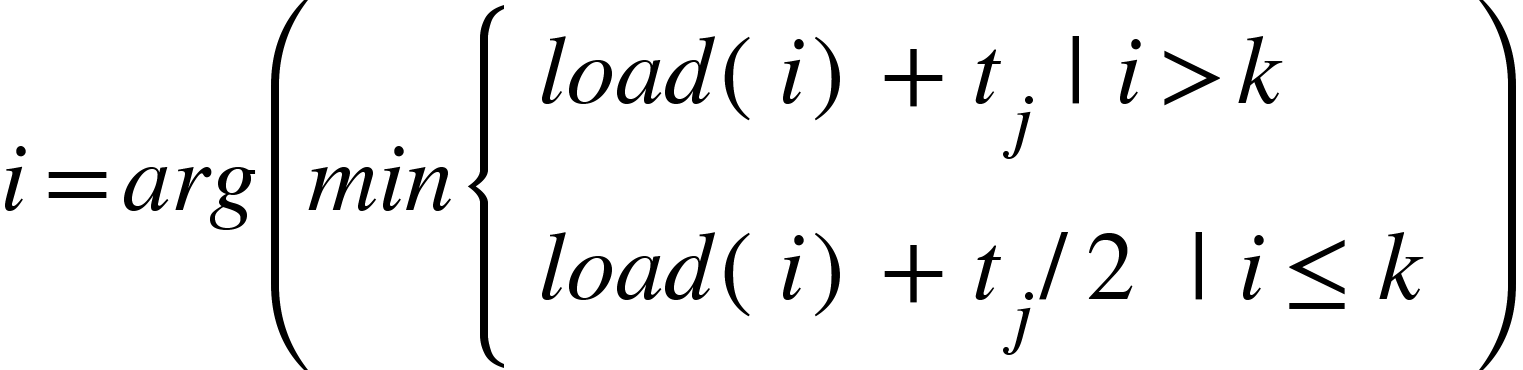
Observatie: Problema incarcarii optime (load balance) este de fapt o problema de permutare!

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:  
 Pentru fiecare job j aleg o masina i astfel incat incarcatura finala a ei sa fie cat mai mica.

Adica, j sa se termine cat mai repede.

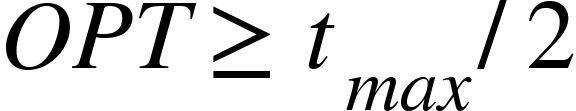
tb sa gasesc



Acel job *j* il adaug peste i-ul gasit.

Tb sa aratam ca acest algoritm este 3-aproximativ:

Fie OPT - solutia optima pt problema

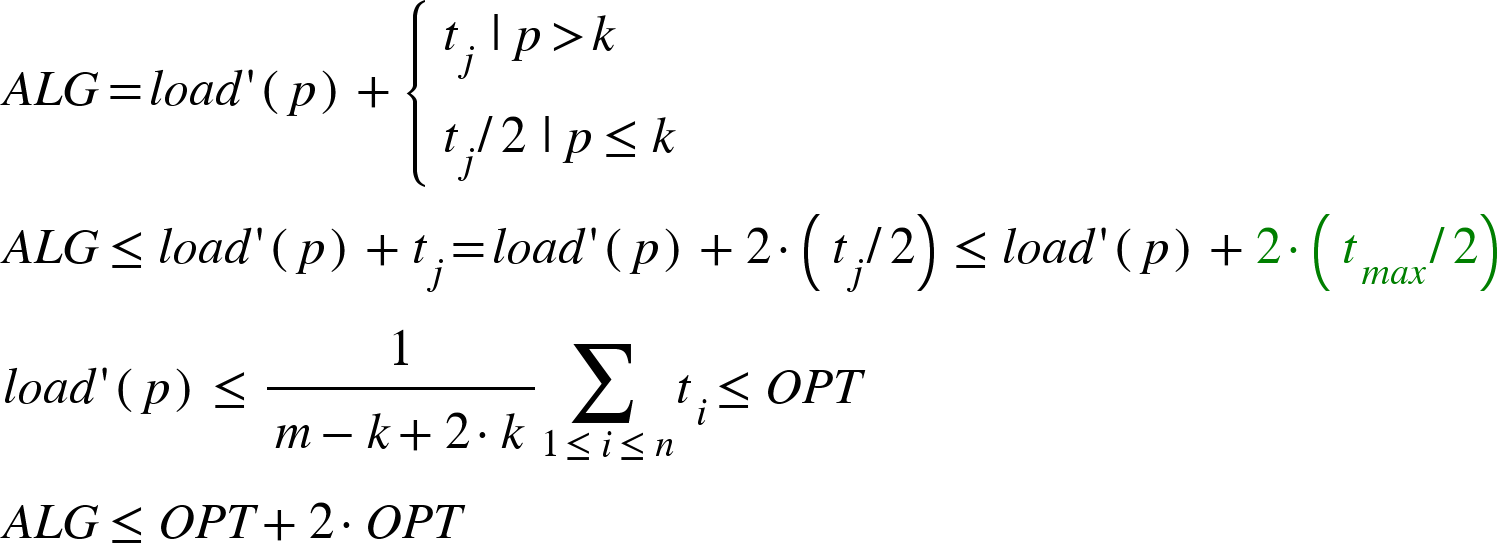


<math xmlns="http://www.w3.org/1998/Math/MathML"><mi>O</mi><mi>P</mi><mi>T</mi><mo>&#x2265;</mo><mfrac><mn>1</mn><mrow><mi>m</mi><mo>-</mo><mi>k</mi><mo>+</mo><mn>2</mn><mi>k</mi></mrow></mfrac><munder><mo>&#x2211;</mo><mrow><mn>1</mn><mo>&#x2264;</mo><mi>i</mi><mo>&#x2264;</mo><mi>n</mi></mrow></munder><msub><mi>t</mi><mi>i</mi></msub><mo>=</mo><mi>T</mi><mspace linebreak="newline"/><mi>P</mi><mi>r</mi><mi>e</mi><mi>s</mi><mi>p</mi><mi>u</mi><mi>p</mi><mi>u</mi><mi>n</mi><mi>e</mi><mi>m</mi><mo>&#xA0;</mo><mi>O</mi><mi>P</mi><mi>T</mi><mo>&lt;</mo><mi>T</mi><mspace linebreak="newline"/><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>d</mi><mi>a</mi><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><mo>&#xA0;</mo><mi>T</mi><mo>&#xA0;</mo><mi>u</mi><mi>n</mi><mi>i</mi><mi>t</mi><mi>a</mi><mi>t</mi><mi>i</mi><mo>&#xA0;</mo><mi>d</mi><mi>e</mi><mo>&#xA0;</mo><mi>t</mi><mi>i</mi><mi>m</mi><mi>p</mi><mo>,</mo><mo>&#xA0;</mo><mi>t</mi><mi>o</mi><mi>t</mi><mi>u</mi><mi>s</mi><mi>i</mi><mo>&#xA0;</mo><mi>i</mi><mi>n</mi><mi>t</mi><mi>r</mi><mi>e</mi><mi>g</mi><mi>u</mi><mi>l</mi><mo>&#xA0;</mo><mi>p</mi><mi>r</mi><mi>o</mi><mi>i</mi><mi>e</mi><mi>c</mi><mi>t</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><mi>f</mi><mi>i</mi><mi>n</mi><mi>a</mi><mi>l</mi><mi>i</mi><mi>z</mi><mi>a</mi><mi>t</mi><mspace linebreak="newline"/><munder mathcolor="#FF0000"><mo>&#x2211;</mo><mrow><mn>1</mn><mo>&#x2264;</mo><mi>i</mi><mo>&#x2264;</mo><mi>n</mi></mrow></munder><msub><mi mathcolor="#FF0000">t</mi><mi mathcolor="#FF0000">i</mi></msub><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><mfenced><mrow><mi>m</mi><mo>-</mo><mi>k</mi><mo>+</mo><mn>2</mn><mi>k</mi></mrow></mfenced><mi>T</mi><mo>=</mo><mfenced><mrow><mi>m</mi><mo>-</mo><mi>k</mi></mrow></mfenced><mi>T</mi><mo>+</mo><mn>2</mn><mi>k</mi><mi>T</mi><mo mathcolor="#FF0000">&gt;</mo><munder mathcolor="#FF0000"><mo>&#x2211;</mo><mrow><mn>1</mn><mo>&#x2264;</mo><mi>i</mi><mo>&#x2264;</mo><mi>n</mi></mrow></munder><msub><mi mathcolor="#FF0000">t</mi><mi mathcolor="#FF0000">i</mi></msub><mo mathcolor="#FF0000">&#xA0;</mo><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></math>

Fie p - masina cea mai solicitata la finalul algoritmului (incarcatura ei ne da costul total)

Fie j - ultimul job care este pus peste masina p

load’(i) reprezinta load-ul masiniii i dupa ce s-au asignat primele j-1 joburi



4) O firma pentru care faceți consultanta folosește 10 masini pentru a procesa *n* activități cu timpii de procesare t1, t2, …, tn. Pentru alocarea de activități pt fiecare mașină ei folosesc algoritmul de tip Load-Balance descris în curs. Motivul pentru care v-au angajat sa faceti consultanță este acela că au auzit că acest algoritm nu este tocmai eficient, și că găsește soluții de până la de 2 ori mai costisitoare ca timp decât soluția optimă. Ei sunt îngrijorați de faptul că un recent studiu de piață a tras concluzia că orice firmă de tipul lor, care se abate cu mai mult de 20% de la procesarea optimă a activităților, riscă faliment. După o investigație mai amănunțită observați că toate activitățile ce sosesc spre procesare au un timp de execuție de maxim 50, iar suma timpului de procesare al tuturor activităților este de minim 3000.

În aceste condiții va putea ca această firmă să își continue activitatea ca până acum, sau este condamnată să dea faliment?

Fie S - timpul de lucru al tuturor activităților S>=300

OPT>=S/10

ALG<=50+S/10

OPT>=300

ALG<=350

Daca OPT =300

ALG = 350 - abaterea este de ⅙<20%