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 in timp fezabil.

Q1: ce este factorul de aproximare pentru un algoritm?

Se refera la probleme de optim!

Fie ALG solutia data de algoritmul nostru si OPT solutia optima pt algoritmul nostru.  
Numim *c* - factor de aproximare daca

in cazul problemelor de minim avem:  
OPT<= ALG <=c\*OPT |c>1

in cazul problemelor de maxim avem:

OPT>= ALG >= c\*OPT|c<1

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

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?

In cazul problemelor de minim, 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.

Raspuns:

1. G=100

W={10, 20, 80, 90}

ALG=(10,20);(80);(90)

OPT=(10,90);(20,80)

1. hint: ne uitam la perechile de camioane incarcate de algoritmul nostru.

observam ca o pereche de camioane va avea o incarcatura >G

Fie cazul cel mai nefavorbil in care algoritmul nostru foloseste 2q+1 camioane;

W - suma greutatilor tuturor coletelor.

deoarece fiecare pereche de camioane (dintre primele q) transporta cel putin G greutate, avem ca

W>q\*G; W/G >q; OPT>q; OPT>=q+1; 2\*OPT>=2q+2>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.

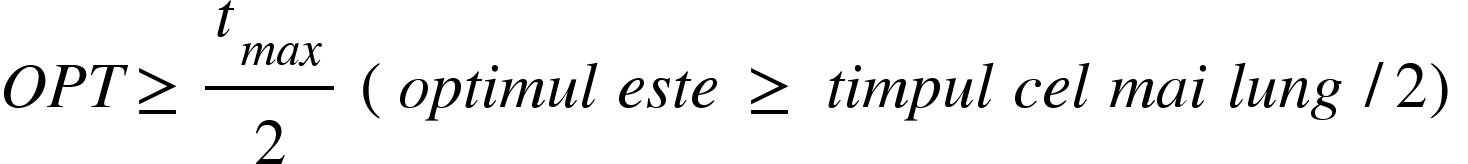
Raspuns:

Fie π - permutarea astfel incat Algoritmul din curs sa dea solutia optima. Daca la rand in algorit se afla masina *i*, atunci π(j) trebuie sa fie unul dintre joburile asociate masinii *i* in solutia optima.

Consecinta: problema gasirii unei “ordini” pentru cele n activitati, a.i. pseudocodul sa ne dea rezultat optim este 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.

Raspuns:  
Parcurgem multimea activitatilor:  
 pt fiecare activitate j alegem acea masina *i* pentru care dupa ce adaugam timpul de lucrul al lui j (care, dupa caz poate fi tj sau tj/2)peste incarcatura lui *i* se obtine o incarcatura cat mai mica.



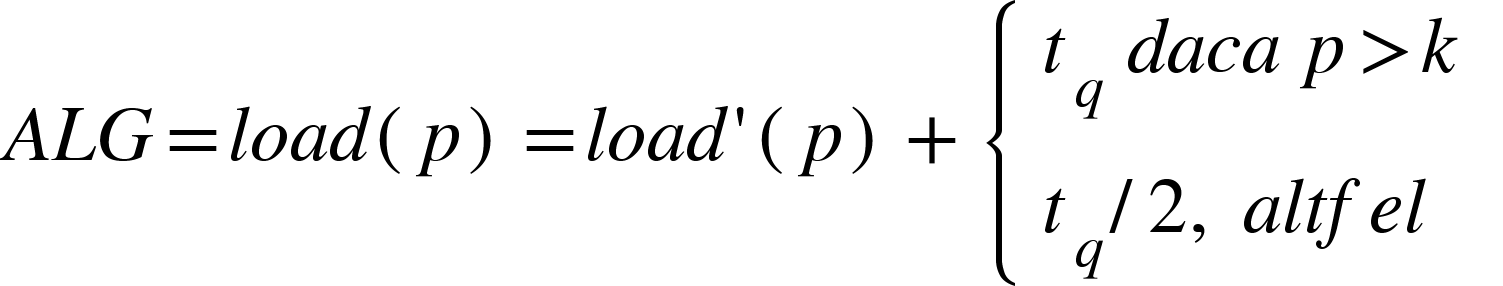
<math xmlns="http://www.w3.org/1998/Math/MathML"><mi>O</mi><mi>P</mi><mi>T</mi><mo>&#x2265;</mo><mfrac><mn>1</mn><mrow><mi>m</mi><mo>-</mo><mi>k</mi><mo>+</mo><mn>2</mn><mo>&#xB7;</mo><mi>k</mi></mrow></mfrac><munder><mo>&#x2211;</mo><mrow><mn>1</mn><mo>&#x2264;</mo><mi>j</mi><mo>&#x2264;</mo><mi>n</mi>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

sa ne gandim la masina care lucreaza cel mai mult

fie p masina care la final lucreaza cel mai mult;

fie q ultima activitatea pusa pe masina p

fie load’(p) loadul masinii p fix inainte de ultima activitate q.



ALG<=load’(p)+tmax ...