Tudorache Alexandru-Theodor

Grupa 242

Tema 1

**Load Balance**

1)

*a)*

Fie setul de date 80, 80, 40

Pentru acest exemplu, atat solutia optima, cat si cea propusa de student obtin incarcatura de 80 pe o masina si de 120 pe cealalta, astfel:

Masina 1: 80

Masina 2: 80, 40

Asadar, solutia studentului poate fi 1-aproximativa (adica optima), deci inclusiv 1.1 aproximativa.

*b)*

Cum una dintre masini are incarcatura de 120, pentru ca algoritmul sa fie 1.1 aproximativ, in solutia optima o masina ar trebui sa aiba incarcatura de minim 120/1.1=109.09109, asadar cealalta masina ar trebui sa aiba incarcatura de maxim 200-109=91.

Presupunem prin absurd ca exista OPT o solutie optima cu incarcatura 109 pe masina 1 si 91 pe masina 2. Cum o activitate are timpul de lucru cel mult 10, daca extragem o activitate de pe masina 1 si o punem pe masina 2, in noua solutie pe masina 1 va fi o incarcatura cuprinsa intre 99 si 108, iar pe masina 2 o incarcatura cuprinsa intre 92 si 101, astfel obtinandu-se o solutie mai buna decat OPT (108 < 109; 101<109). Absurd.

Daca presupunem ca exista o solutie optima cu o incarcatura mai mare decat 109 pe masina 1 si mai mica decat 91 pe masina 2, se va ajunge tot la ceva absurd deoarece diferenta dintre incarcaturile masinilor va creste, astfel putandu-se obtine solutii din ce in ce mai optime decat cea din presupunere.

Deci, solutia optima nu poate contine o masina cu incarcatura de minim 109, deci algoritmul propus de student nu poate fi 1.1 aproximativ.

3)

Fie k = indicele masinii cu load-ul maxim rezultat in urma executarii algoritmului

Deci solutia obtinuta de algoritm este ALG = load(k)

Fie j = ultima activitate adaugata pe masina k.

Fie load’(i) = load-ul masinii I fix inainte ca activitatea j sa fie pusa pe masina k, adica load-ul masinii I dup ace au fost distribuite primele j-1 activitati.

ALG = load(k) = load’(k) + tj

Cum algoritmul selecteaza masina cu load minim si pune pe ea urmatoarea activitate, load’(k) ≤ load’(i), ∀ i ∈ {1, …, m}

Deci load’(k) ≤

* Daca j ≤ m, atunci activitatea j va fi pusa pe o masina goala, deci ALG = tmax ≤ OPT, unde tmax este activitatea cu timpul de lucru maxim. In acest caz, algoritmul este optim.
* Daca j > m, atunci:

Cum activitatile sunt ordonate descrescator dupa timpul de lucru, iar j > m, atunci rezulta ca:

Din Lema 3(Curs 2, slide 43) avem

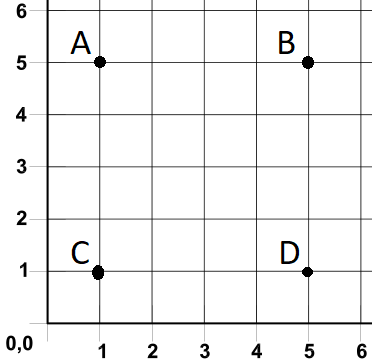
Deci, algoritmul dat este ( aproximativ.

**TSP**

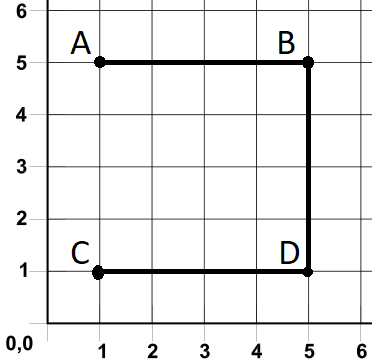
2)

*a)*

Fie multimea de puncte P = {A, B, C, D} din figura de mai jos

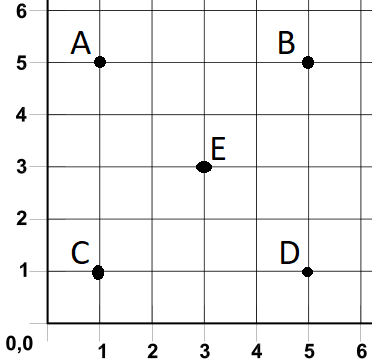


Fie T = MST-ul pentru punctele A, B, C, D astfel:

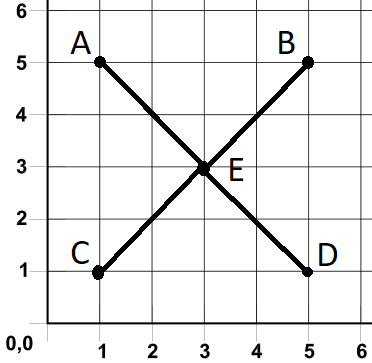


Costul lui T este 4 + 4 + 4 = 12

Fie punctul q=E in plan astfel:



Fie T’=MST pentru punctele A, B, C, D, E astfel:



Cum ABDC este patrat, iar E este centrul patratului de latura 4, atunci AE = BE = CE = DE = =

Costul noului MST este .

Asadar, exista cazuri in care alegand un punct q ∉ P, MST-ul pentru multimea de puncte P ∪{q} are un cost mai mic decat T.

**Vertex Cover**

a)

Algoritmul descris poate face maxim m iteratii prin multimea C, unde m este lungimea multii C. Acest caz se intampla atunci cand se alege aleator din Cj un xi care nu se mai afla in niciun alt predicat din C.

Algoritmul optim poate face cel putin 1 iteratie prin multimea C. Acest caz se intampla atunci cand se alege un xi din multimea X pentru a fi true, acest xi facand parte din toate predicatele din C.

Worst-case-ul factorului de aproximare al algoritmului este dat de cazul in care in multimea X exista un singur xi care se afla in toare predicatele din C, iar algoritmul Greedy-3CNF(C, X) nu-l va alege la nicio iteratie prin multimea C pe xi. In acest caz, algoritmul Greedy va face numarul maxim de m iteratii, iar cel optim o singura iteratie.

Astfel, factorul de aproximare al algoritmului este m, deci algoritmul este m-aproximativ.

b)

1: C = {C1, … Cm} mulțimea de predicate, X = {x1, … xn} - mulțime de variabile

2: cât timp C ≠ ∅ execută

3: Alegem aleator Cj ∈ C.

4: Fie a1, a2, a3 variabilele din Cj.

5: a1 ← true; a2 ← true; a3 ← true.

6: Eliminăm din C toate predicatele care conțin una dintre variabilele a1, a2, a3.

7: return X

Fie T = {xi | xi = true} dupa executarea algoritmului propus mai sus

Trebuie sa aratam ca |T| <= 3OPT, unde OPT este numarul de elemente din X care care au valorea *true*, rezultat in urma executarii algoritmului optim.

Fie C\* multimea de predicate selectate la pasul 3 al algoritmului (*Alegem aleator Cj ∈ C*). Cum la pasul 6 eliminam din C toate predicatele ce contin pe oricare dintre variabilele a1, a2, a3 extrase la pasul 5, in multimea C\* vor fi numai predicate ce contin variabile diferite.

OPT >= |C\*| =

OPT >=

3OPT >= |T|

Deci algoritmul propus mai sus este 3-aproximativ.

c)

f(xi) =

Trebuie sa minimizam

Constrangeri:

f(xj) + f(xk) + f(xl) ≥ 1, pentru oricare Ci (xj, xk, xl) ∈ C;

0 ≤ f(xi) ≤ 1, pentru orice i ∈ {1, 2, …, n}