Cursul 3

Demonstrații și Justificări

Teorema 1, slide 7

Nu există nicio valoare *c* pentru care să existe un algoritm în timp polinomial și care să ofere o soluție cu un factor de aproximare *c* pentru TSP (în forma generală), presupunând că $P \neq NP$.

Justificare:

Teorema 1, slide 7

Nu există nicio valoare *c* pentru care să existe un algoritm în timp polinomial și care să ofere o soluție cu un factor de aproximare *c* pentru TSP (în forma generală), presupunând că $P \neq NP$.

Justificare:

Reductio ad absurdum

Presupunem că există un algoritm aproximativ astfel încât dacă există S – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult $c*S$.

Teorema 1, slide 7
Nu există nicio valoare *c* pentru care să existe un algoritm în timp polinomial și care să ofere o soluție cu un factor de aproximare *c* pentru TSP (în forma generală), presupunând că $P \neq NP$.

Justificare:

Reductio ad absurdum

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult $c*S$.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

Construim un <u>graf ponderat complet</u> *G'* pe baza lui *G* după cum urmează:

- $V(G')=V(G); \ |V(G')|=n;$
- Pentru fiecare $e \in E(G')$ pentru care $e \in E(G)$ avem $w(e, G') = 1$. (ponderea muchiei e în G' va fi 1)
- Pentru fiecare $e \in E(G')$ pentru care $e \notin E(G)$ avem $w(e, G') = c*n$.

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

Construim un <u>graf ponderat complet</u> *G'* pe baza lui *G* după cum urmează:

- *V(G')=V(G); | V(G')|=n;*
- *Pentru fiecare e* ∈ E(G') pentru care *e* ∈ E(G) avem w(e, G') = 1. (ponderea muchiei e în G' va fi 1)
- Pentru fiecare *e* ∈ E(G') pentru care *e* ∉ E(G) avem w(e, G') = c*n

* Dacă G are un ciclu hamiltonian, cum este soluția optimă pentru TSP(*G')?*

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

Construim un graf ponderat complet *G'* pe baza lui *G* după cum urmează:

- *V(G')=V(G); | V(G')|=n;*
- *Pentru fiecare e* ∈ E(G') pentru care *e* ∈ E(G) avem w(e, G') = 1. (ponderea muchiei e în G' va fi 1)
- Pentru fiecare *e* ∈ E(G') pentru care *e* ∉ E(G) avem w(e, G') = c*n

* Dacă G are un ciclu hamiltonian, cum este soluția optimă pentru TSP(*G')?*

Soluția optimă va fi de cost n, și va fi corespunzătoare ciclului hamiltonian din G.

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c**S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

Construim un graf ponderat complet *G'* pe baza lui *G* după cum urmează:

- *V(G')=V(G); | V(G')|=n;*
- *Pentru fiecare e* ∈ E(G') pentru care *e* ∈ E(G) avem w(e, G') = 1. (ponderea muchiei e în G' va fi 1)
- Pentru fiecare *e* ∈ E(G') pentru care *e* ∉ E(G) avem w(e, G') = c*n

* Dacă G are un ciclu hamiltonian, cum este soluția optimă pentru TSP(*G')?*

Soluția optimă va fi de cost n, și va fi corespunzătoare ciclului hamiltonian din G.

* Care este costul maxim al soluției date de algoritmul nostru aproximativ (cel care am presupus ca există)?

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

Construim un graf ponderat complet *G'* pe baza lui *G* după cum urmează:

- *V(G')=V(G); | V(G')|=n;*
- *Pentru fiecare e* ∈ E(G') pentru care *e* ∈ E(G) avem w(e, G') = 1. (ponderea muchiei e în G' va fi 1)
- Pentru fiecare *e* ∈ E(G') pentru care *e* ∉ E(G) avem w(e, G') = c*n

* Dacă G are un ciclu hamiltonian, cum este soluția optimă pentru TSP(*G')?*

Soluția optimă va fi de cost n, și va fi corespunzătoare ciclului hamiltonian din G.

* Care este costul maxim al soluției date de algoritmul nostru aproximativ (cel care am presupus ca există)?

Algoritmul oferă o soluție de cost cel mult *n*c.* **(1)**

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult $c*S$.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

Construim un graf ponderat complet *G'* pe baza lui *G* după cum urmează:

- $V(G')=V(G); \; |V(G')|=n;$
- *Pentru fiecare e* $\in E(G')$ pentru care *e* $\in E(G)$ avem $w(e, G') = 1$. (ponderea muchiei e în G' va fi 1)
- Pentru fiecare *e* $\in E(G')$ pentru care *e* $\notin E(G)$ avem $w(e, G') = c*n$

* Dacă G **nu** are un ciclu hamiltonian, cum este soluția optimă pentru TSP(*G')?*

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

Construim un <u>graf ponderat complet</u> *G'* pe baza lui *G* după cum urmează:

- *V(G')=V(G); | V(G')|=n;*
- *Pentru fiecare e* ∈ E(G') pentru care *e* ∈ E(G) avem w(e, G') = 1. (ponderea muchiei e în G' va fi 1)
- Pentru fiecare *e* ∈ E(G') pentru care *e* ∉ E(G) avem w(e, G') = c*n

* Dacă G **<u>nu</u>** are un ciclu hamiltonian, cum este soluția optimă pentru TSP(*G')?*

Soluția optimă va fi de cost minim (n-1)+n*c, deoarece trebuie să se folosească cel puțin o muchie ∉ E(G) .

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

Construim un <u>graf ponderat complet</u> *G'* pe baza lui *G* după cum urmează:

- *V(G')=V(G); | V(G')|=n;*
- *Pentru fiecare e* ∈ E(G') pentru care *e* ∈ E(G) avem w(e, G') = 1. (ponderea muchiei e în G' va fi 1)
- Pentru fiecare *e* ∈ E(G') pentru care *e* ∉ E(G) avem w(e, G') = c*n

* Dacă G **<u>nu</u>** are un ciclu hamiltonian, cum este soluția optimă pentru TSP(*G')?*

Soluția optimă va fi de cost minim (n-1)+n*c, deoarece trebuie să se folosească cel puțin o muchie ∉ E(G).

* Care este costul **<u>minim</u>** al soluției date de algoritmul nostru aproximativ (cel care am presupus ca există)?

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

Construim un <u>graf ponderat complet</u> *G'* pe baza lui *G* după cum urmează:

- *V(G')=V(G); | V(G')|=n;*
- *Pentru fiecare e* ∈ E(G') pentru care *e* ∈ E(G) avem w(e, G') = 1. (ponderea muchiei e în G' va fi 1)
- Pentru fiecare *e* ∈ E(G') pentru care *e* ∉ E(G) avem w(e, G') = c*n

* Dacă G **<u>nu</u>** are un ciclu hamiltonian, cum este soluția optimă pentru TSP(*G')?*

Soluția optimă va fi de cost minim (n-1)+n*c, deoarece trebuie să se folosească cel puțin o muchie ∉ E(G) .

* Care este costul **<u>minim</u>** al soluției date de algoritmul nostru aproximativ (cel care am presupus ca există)?

Algoritmul oferă o soluție de cost > *n*c.* **(2)**

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

- Dacă G are un ciclu Hamiltonian, Algoritmul oferă o soluție de **cost cel mult** *n*c*. **(1)**
- Dacă G **nu** are un ciclu Hamiltonian**,** Algoritmul oferă o soluție de cost > *n*c*. **(2)**

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

- Dacă G are un ciclu Hamiltonian, Algoritmul oferă o soluție de **cost cel mult** *n*c*. **(1)**
- Dacă G **nu** are un ciclu Hamiltonian, Algoritmul oferă o soluție de cost > *n*c*. **(2)**

- Graful *G'* – se obține în timp polinomial

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

- Dacă G are un ciclu Hamiltonian, Algoritmul oferă o soluție de **cost cel mult** *n*c*. **(1)**
- Dacă G **nu** are un ciclu Hamiltonian**,** Algoritmul oferă o soluție de cost > *n*c*. **(2)**

- Graful *G'* – se obține în timp polinomial
- Algoritmul rulează in timp polinomial

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c*S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

- Dacă G are un ciclu Hamiltonian, Algoritmul oferă o soluție de **cost cel mult** *n*c*. **(1)**
- Dacă G **nu** are un ciclu Hamiltonian**,** Algoritmul oferă o soluție de cost > *n*c*. **(2)**

- Graful *G'* – se obține în timp polinomial
- Algoritmul rulează in timp polinomial
- Din **(1)&(2)** rezultă faptul ca putem decide HCP in timp polinomial in funcție de outputul algoritmului nostru pentru problema de TSP (daca outputul este cel mult *n*c*, atunci graful G este Hamiltonian, altfel nu este Hamiltonian).

Presupunem că există un algoritm aproximativ astfel încât dacă există *S* – costul optim al unui traseu închis pentru TSP, atunci algoritmul nostru va oferi un traseu închis de cost cel mult *c**S*.

Fie *G* – un graf simplu, neponderat. Problema determinării unui *HC* în *G* este NPC.

- Dacă G are un ciclu Hamiltonian, Algoritmul oferă o soluție de **cost cel mult** $n*c$. **(1)**
- Dacă G **nu** are un ciclu Hamiltonian, Algoritmul oferă o soluție de cost > $n*c$. **(2)**

- Graful *G'* – se obține în timp polinomial
- Algoritmul rulează in timp polinomial
- Din **(1)&(2)** rezultă faptul ca putem decide HCP in timp polinomial in funcție de outputul algoritmului nostru pentru problema de TSP (daca outputul este cel mult $n*c$, atunci graful G este Hamiltonian, altfel nu este Hamiltonian).
- **Dar HCP este NPC. ※**

Lema 2, slide 12

Fie G un graf complet, ponderat, care respectă regula triunghiului. Și fie v1, v2, v3, …., vk un lanț în graful G. Atunci avem len((v1,vk))≤len(v1, v2, v3, …., vk)

Justificare:

Inducție

- presupunem ca len((v1,v_{k-1}))≤len(v1, v2, v3, …., v_{k-1})
- din regula triunghiului avem ca
- len((v1,vk))≤len((v1,v_{k-1}))+len((v_{k-1},v_k))≤len(v1, v2, v3, …., v_{k-1})+len((v_{k-1},v_k))

  =len(v1, v2, v3, …., vk) ∴

Lema 2, slide 12

Fie G un graf complet, ponderat, care respectă regula triunghiului. Și fie v1, v2, v3, …., vk un lanț în graful G. Atunci avem len((v1,vk))≤len(v1, v2, v3, …., vk)

Justificare:

Inducție

- presupunem ca len((v1,v_{k-1}))≤len(v1, v2, v3, …., v_{k-1})
- din regula triunghiului avem ca
- len((v1,vk))≤len((v1,v_{k-1}))+len((v_{k-1},v_k))≤len(v1, v2, v3, …., v_{k-1})+len((v_{k-1},v_k))

Lema 2, slide 12

Fie G un graf complet, ponderat, care respectă regula triunghiului. Și fie v1, v2, v3, …., vk un lanț în graful G. Atunci avem len((v1,vk))≤len(v1, v2, v3, …., vk)

Justificare:

Inducție

- presupunem ca len((v1,v_{k-1}))≤len(v1, v2, v3, …., v_{k-1})
- din regula triunghiului avem ca
- len((v1,vk))≤len((v1,v_{k-1}))+len((v_{k-1},v_k))≤len(v1, v2, v3, …., v_{k-1})+len((v_{k-1},v_k))

  =len(v1, v2, v3, …., vk) ∴

Lema 3 slide 17:
OPT ≥ MST

Justificare:

Reducere la absurd

<u>presupunem ca MST>OPT</u>

Din ciclul Hamiltonian din care rezultă OPT putem scoate orice muchie și obținem un lanț!

Lema 3 slide 17:
OPT ≥ MST

Justificare:

Reducere la absurd

presupunem ca MST>OPT

Din ciclul Hamiltonian din care rezultă OPT putem scoate orice muchie și obținem **un lanț**!

Cum un lanțul obținut are un cost total mai mic decât aș ciclului inițial, dar lanțul este un arbore, deci are costul cel puțin egal cu MST, obținem contradicția că MST<OPT

Teorema 4 slide 20:
Algoritmul descris in slideurile 18-19 este un algoritm 2-aproximativ pentru TSP.

Justificare:

Observam ca ciclul nostru contine atat muchii/lanturi din MST - parcurse o singura data (ex: v3-v1-v8) dar si muchii care nu sunt in MST (ex: v8-v7), dar costul unei muchii de forma (xy) - care nu este in MST - va fi mai mic decat costul unicului lant din MST care uneste x de y (ex costul lui v8-v7 va fi < costul lantului v8-v1-v3-v7) conform Lemei 2

ALG va contine muchii care sunt in MST si muchii care nu sunt in MST dar au costul < decat lanturile echivalente din MST

ALG<=2*MST<=2*OPT