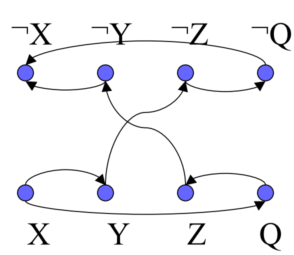
Approximate algorithms

EECS 477

Lecture 23, 12/05/2002

SAT-2-CNF

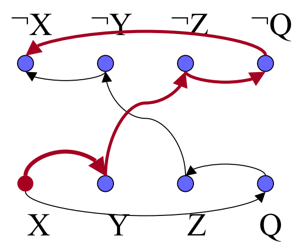
- Is in P
  - Formula (¬X+Y)(¬Z+¬Y)(¬Q+Z)(Q+¬X)
    - Construct directed graph: two verts for each variable, edge from X to Y if there is a clause equiv to (¬X+Y) = ¬(¬Y) + (¬X)



SAT-2-CNF

P

- Claim
  - If there are paths from some N to ¬N and from ¬N to N then the formula cannot be satisfied
  - Otherwise it is satisfiable (example)



$X \Rightarrow Y$
$Y \Rightarrow \neg Z$
$\neg Z \Rightarrow \neg Q$
$\neg Q \Rightarrow \neg X$
Hence
$X \Rightarrow \neg X$
Hmm…

NP-completeness

- Decision problem X is NP-complete
  1. X is in NP
  2. $Y \leq^p X$ for every problem Y in NP
- X is polynomially harder than any other NP problem
- If we know that X is NP-complete and $X \leq^p Z$ then Z is NP-complete
- If we could only find one such X

Some NP-complete problems

- SAT
- 3SAT: clauses have three variables
- 3DM: 3D matching
- HAMD: hamiltonian circuit
- PARTITION: set A and $s: A \to Z^+$
  - Partition A into two equally sized parts
- CLIQUE: clique of size J or more
- VERTEX COVER: of size K or less
- K-COL: graph colorability with K colors or less

NP-hard

- X is NP-hard
  - if there is an NP-complete problem Y that can be polynomially reduced to X
    - $Y \leq^p X$
  - Does not have to be a decision problem
  - Decision problem can be NP-hard but not in NP, for instance exact K-colorability
    - Any K-coloring is a certificate for K-COL but not for K-COLE(exact: can color with K but no less)

What about two colors?

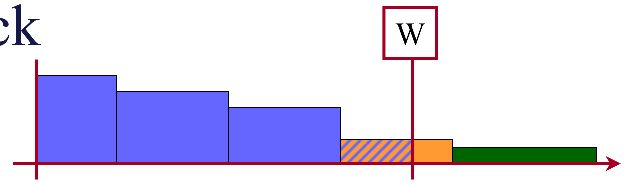
- Determine whether a graph is two-colorable
  - A polynomial algorithm?

Metric TSP

- Undirected graph (complete)
- Distance matrix satisfies
  - Triangle inequality

    d(x,z) <= d(x,y) + d(y,z)
    - length(Hamiltonian cycle) >= length(Hamiltonian path) >= length(MST)
  - Construct an MST, tour around it will cost no more than 2*length(MST)
  - tour with shortcuts <= 2 length(MST)

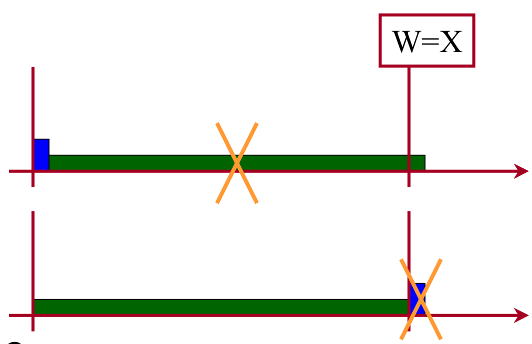
Knapsack



- Greedy
  - Optimal when objects are breakable
  - If not breakable:

```
float greedy-knapsack(vector& w, vector& v, W){
    sort decreasing v[j]/w[j]
    weight = 0; value = 0;
    for(j=0; j<w.size(); ++j) {
        if(weight+w[j]<=W) { // add j-th object
            value += v[j];  weight += w[j];
        }
    }
    return value;
}
```

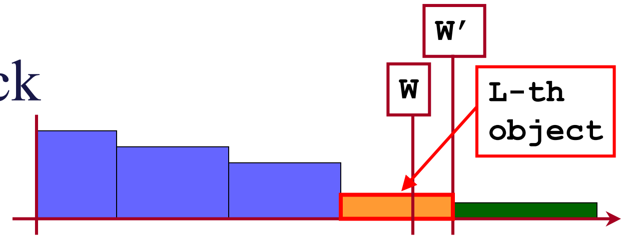
Knapsack



- Bad example
  - Ratio between optimal and greedy values

    w[0]=1, v[0]=2, w[1]=X, v[1]=X
  - Greedy value is 2
  - Optimal value is X
    - As X grows Optimal/Greedy goes to infinity

Knapsack



- Fix it

```
float approx-knapsack(vector& w, vector& v, W){
    biggest_value = max(v);
    return max(biggest, greedy-knapsack(w,v,W);
}
```

- Opt <= Opt' = sum(v, j=0..L)
- Greedy = sum(v, j=0..L-1)
- Approx = max(biggest_value, Greedy)
- Claim: Approx >= Opt/2

Max(a,b) >= (a+b)/2

Approximations

- Maximization
- Different guarantees
  - Absolute
    - Absolute error is less than C

      Optimal – C <= Abs-Approx <= Optimal
  - Relative
    - Relative error less than ε

      (1- ε)Optimal <= Rel-Approx <= Optimal

MTSP

- MTSP $\leq^p$ C-Abs-MTSP for any C>0
- Make new instance M'
  – multiplying the original distance matrix M by K=floor(C)+1
- New approximate satisfies

  K*Opt(M)<=Apprx(M')<=Opt(M')+C=K*Opt(M)+C

  Opt(M)<=Apprx(M')/K<=Opt(M)+C/K < Opt(M)+1

  Everything is integer so Apprx(M')/K = Opt(M)

- EVALUATION FORMS!
- Tuesday: final exam review