CS-258

Solving Hard Problems in Combinatorial Optimization

Pascal Van Hentenryck

Course Overview

Study of combinatorial optimization
from a practitioner standpoint

- the main techniques and tools
- several representative problems

What is expected from you:

- 7 team assignments (2 persons team)
- Short write-up on the assignments
- Final presentation of the results

What you should not do in this class?

- read papers/books/web/... on combinatorial optimization
- Read the next slide during class

Course Overview

January 31: Knapsack
 * Assignment: Coloring
February 7: LS + CP + LP + IP
February 14: Linear Programming
 * Assignment: Supply Chain
February 28: Integer Programming
 * Assignment: Green Zone
March 7: Travelling Salesman Problem
 * Assignment: TSP
March 14: Advanced Mathematical Programming
March 21: Constraint Programming
 * Assignment: Nuclear Reaction
April 4: Constraint-Based Scheduling
April 11: Approximation Algorithms (Claire)
 * Assignment: Operation Hurricane
April 19: Vehicle Routing
April 25: Advanced Constraint Programming
 * Assignment: Special Ops.

Knapsack Problem

$$\max \quad \sum_{i=1}^{n} c_i x_i$$

subject to:

$$\sum_{i=1}^{n} w_i x_i \leq l$$

$$x_i \in [0, 1]$$

NP-Complete Problems

Decision problems

Polynomial-time verifiability

- We can verify if a guess is a solution in polynomial time

Hardest problem in NP (NP-complete)

- If I solve one NP-complete problem, I solve them all

Approaches

Exponential Algorithms (sacrificing time)
- dynamic programming
- branch and bound
- branch and cut
- constraint programming

Approximation Algorithms (sacrificing quality)
- polynomial time
- guarantee of certain performance

Local Search Algorithms (sacrificing quality)
- iterative improvement of a solution
- not always a performance guarantee

Dynamic Programming

$$\max \sum_{i=1}^{n} c_i x_i$$

subject to:

$$\sum_{i=1}^{n} w_i x_i \le l \qquad x_i \in [0, 1]$$

Divide and Conquer:

P(k, d) is defined as: $\max \sum_{i=1}^{k} c_i x_i$

subject to $\sum_{i=1}^{k} w_i x_i \le d$

variable **1 … k**

capacity is **d**

Dynamic Programming

Solve: P(k, d)

- take variable k ($x_k = 1$)

$$\textbf{max} \quad \sum_{i=1}^{k-1} c_i x_i + c_k$$

$$\textbf{subject to} \quad \sum_{i=1}^{k-1} w_i x_i \leq d - w_k$$

- drop variable k ($x_k = 0$)

$$\textbf{max} \quad \sum_{i=1}^{k-1} c_i x_i$$

$$\textbf{subject to} \quad \sum_{i=1}^{k-1} w_i x_i \leq d$$

Dynamic Programming

Original Problem:

p(n, l)

Basic Divide and Conquer step for p(k, d)

- take variable k if $w_k \le d$

  $$c_k + p((k-1),(d-w_k))$$

- drop variable k

  $$p(k-1, d)$$

p(k, d) = max(c_k + p(k-1, d-w_k), p(k-1, d))

p(0, d) = 0

Dynamic Programming

Recursive formulation

```
long  p(k, d)   {
   if k = 0 then
      return 0;
   else if w_k > d then
      return p(k-1, d);
   else
      return max(p(k-1, d), ck + p(k-1, d-w_k))
   }
```

Is this algorithm any good?

Dynamic Programming

```
long  fib(n)  {
   if n = 0 then return 1;
   else if n = 1 then return 1;
   else return fib(n-1) + fib(n-2);
}
```

Is this any good?

Dynamic Programming

of objects

	0	1		3	4				n
0									
1									
2									
...									
...									
l									

```
for i := 1 to n  do
  for d := 0 to l  do
     if w_i > d then
        p[d, i] = p[d, i-1]
     else
        p[d, i] = max(c_i + p[d-w_i, i-1],p[d, i-1])
```

Complexity ??

Dynamic Programming

Complexity: **O(ln)**

Is this polynomial contradicting $P \neq NP$?

- How do you represent **l** on a computer?

Pseudo Polynomial Algorithm

- Polynomial algorithm if the numbers are small

Knapsack: Example

max $16x_1 + 19x_2 + 23x_3 + 28x_4$

subject to $2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7$

$x_i \in [0, 1]$

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	**35**	35
6	0	16	35	39	39
7	0	16	35	42	44

Dynamic Programming

How do we get the optimal solution?

$x^*_n = 0$ if p(n, l) = p(n-1, l)

$x^*_n = 1$ otherwise

Denote **d(k) = l -** $\sum_{i=k+1}^{n} w_i x_i^*$

We have

x^*_k = 0 if p(n, d(k)) = p(n-1, d(k))

x^*_k = 1 otherwise.

```
d := l
for k := n down to 1 do
  if p[d, k] = p[d, k-1] then
     x*k = 0;
  else
     x*k = 1;
     d := d - wk;
```

Generalizations

Q: Can you generalize it if the variables are not bounded?

Q: Can we solve the problem in linear space in l?

Knapsack Generalization

$$\textbf{max} \quad \sum_{i=1}^{n} c_i x_i \quad \textbf{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq L$$

Define **p(d)** as

$$\textbf{max} \quad \sum_{i=1}^{n} c_i x_i \quad \textbf{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq d$$

Now

$$p(d) = \max\{c_j + p(d-w_j) \mid w_j \leq d \ \&\ 1 \leq j \leq n\}$$

Complexity

O(nL)

Branch and Bound

One of the most successful techniques for solving optimization problems optimally

- Divide and Conquer

Two Steps

- **Bounding:** getting an optimistic estimation of the optimum for a subproblem (key step)
- **Branching:** split the problem into subproblems (key step)

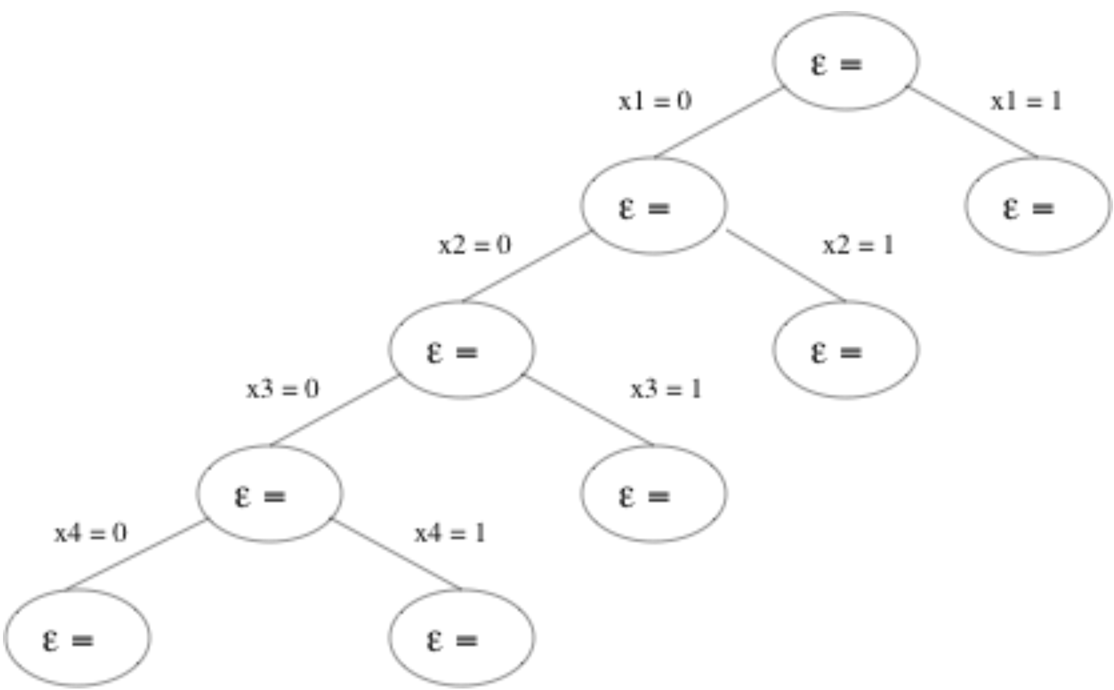
Why is bounding so important?

Branch and Bound

max $16x_1 + 19x_2 + 23x_3 + 28x_4$

subject to $2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7$

$x_i \in [0, 1]$



Branch and Bound

Best-first:

- explore the node with the best evaluation function
- when does pruning occur?
- what is the main limitation?

Depth first:

- when does pruning occur?

There are other strategies as well

- Limited Discrepancy Search

Branch and Bound

How to bound?

- Relaxation

How to relax?

- Remove some constraints

Knapsack Problem

$$\textbf{max} \quad \sum_{i=1}^{n} c_i x_i$$

subject to:

$$\sum_{i=1}^{n} w_i x_i \le l$$

$$x_i \in [0, 1]$$

Branch and Bound

Naïve evaluation

- remove the capacity constraint

$$\max c_1 v_1 + \ldots + c_m v_m + \sum_{i=m+1}^{n} c_i x_i$$

$$\text{subject to} \quad x_i \in [0, 1]$$

Branch and Bound

Linear Programming Relaxation

- remove the integrality requirement

$$\textbf{max} \quad \sum_{i=1}^{n} c_i x_i$$

$$\textbf{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq l \qquad 0 \leq x \leq 1$$

What is the optimal solution assuming that

$$\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \ldots \geq \frac{c_n}{w_n} \quad ?$$

Branch and Bound

$$\textbf{max} \quad \sum_{i=1}^{n} c_i x_i$$

$$\textbf{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq l \qquad 0 \leq x \leq 1$$

Rewrite $x_i = \dfrac{y_i}{c_i}$ to obtain

$$\textbf{max} \quad \sum_{i=1}^{n} y_i$$

$$\textbf{subject to} \quad \sum_{i=1}^{n} \frac{w_i}{c_i} y_i \leq l \qquad \boldsymbol{y_i \in [0 \ldots 1]}$$

We have that $\dfrac{w_1}{c_1} \leq \dfrac{w_2}{c_2} \leq \ldots \leq \dfrac{w_n}{c_n}$

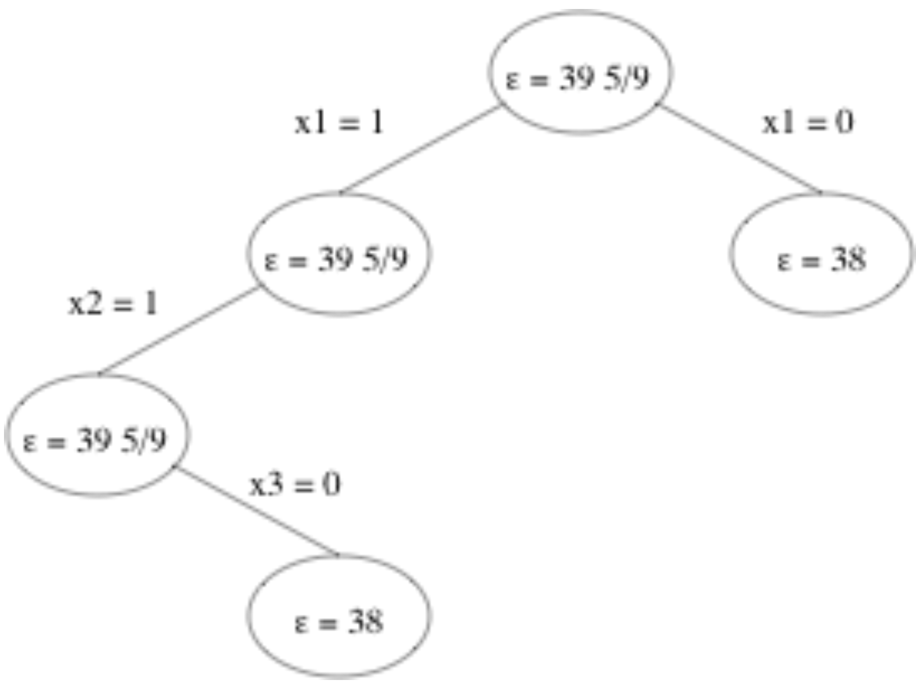
What is the optimal solution?

Branch and Bound

max $8x_1 + 16x_2 + 20x_3 + 12x_4 + 6x_5 + 10x_6 + 4x_7$

subject to

$$3x_1 + 7x_2 + 9x_3 + 6x_4 + 3x_5 + 5x_6 + 2x_7 \leq 17$$



Time to Pack

Q: Can you think of a way to combine both algorithms?