Discrete Optimization 2010
Lecture 8
Lagrangian Relaxation / $\mathcal{P}$, $\mathcal{NP}$ and co-$\mathcal{NP}$

Marc Uetz
University of Twente

m.uetz@utwente.nl

Outline

LP and Lagrangian Relaxation

$$\begin{aligned} \max\ & c^t x \\ \text{s.t. } & Ax \le b && (\leftarrow \text{complicating constraints}) \\ & Bx \le d && \text{(IP)} \\ & x \text{ integer} \end{aligned}$$

Lagrangian relaxation ($\lambda \geq 0$)

$$\begin{aligned} L(\lambda) &= \max_x\ c^t x - \lambda^t (Ax - b) \\ \text{or } L(\lambda) &= \lambda^t b + \max_x\ (c^t - \lambda^t A) x \\ \text{s.t. } & Bx \le d && (LR(\lambda)) \\ & x \text{ integer} \end{aligned}$$

Idea: penalize violation of constraints instead of enforcing them

Exercise: $L(\lambda)$ is an upper bound on optimum IP solution.

LP and Lagrangian Relaxation

Best possible Lagrangian upper bound is Langragian Dual:

$$LD = \min_{\lambda \geq 0} L(\lambda)$$ [a]

[a] note: λ is unrestricted in sign if we relax equality constraints $Ax = b$

LP upper bound:

$$LP = \max_x \{c^t x \mid Ax \leq b, Bx \leq d\}$$

denote by *ILP* the optimum solution value of the integer program

Theorem (for maximization problems)

$$ILP \leq LD \leq LP$$

(i.e., best Lagrangian bound is never worse than the LP relaxation)

Proof (of 2nd inequality)

Use (strong) linear programming duality twice:

$$
\begin{aligned}
LD &= \min_{\lambda\geq 0}\{b^t\lambda + \max_x\{(c^t-\lambda^t A)x \mid Bx\leq d, x \text{ integer}\}\}\\
&\leq \min_{\lambda\geq 0}\{b^t\lambda + \max_x\{(c^t-\lambda^t A)x \mid Bx\leq d\}\}\\
&\underbrace{=}_{\text{LP duality}} \min_{\lambda\geq 0}\{b^t\lambda + \min_y\{d^t y \mid B^t y = (c - A^t\lambda), y\geq 0\}\}\\
&= \min_{\lambda,y}\{b^t\lambda + d^t y \mid A^t\lambda + B^t y = c, y\geq 0, \lambda\geq 0\}\}\\
&\underbrace{=}_{\text{LP duality}} \max_x\{c^t x \mid Ax\leq b, Bx\leq d\} = LP \quad \square
\end{aligned}
$$

Note: '$=$' holds in 2nd step for example if B is totally unimodular

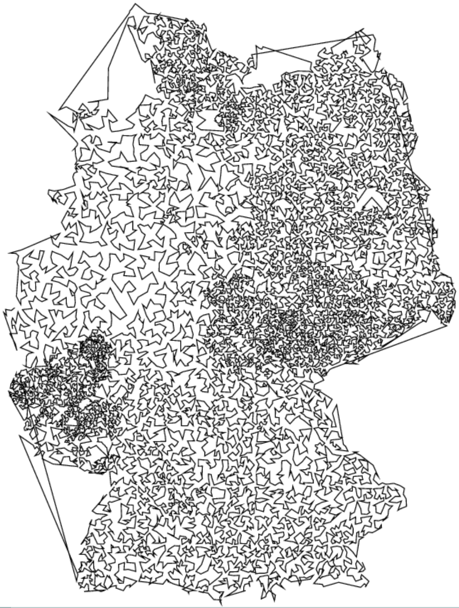
An Example: TSP

TSP

Given (w.l.o.g. complete) undirected graph $G = (V, E)$, $|V| = n$, edge lengths $c_e \geq 0$ $\forall\ e \in E$, find shortest Hamiltonian cycle T ($=$ tour).

Example

a short but non-optimal tour for all 15,112 cities in D

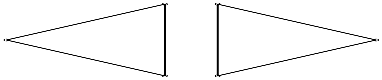


IP Formulation for TSP I

variables $x_e = 1$ if $e \in T$, 0 otherwise

$$\begin{aligned}
\min \quad & \sum_{e\in E} c_e x_e \\
& \sum_{e\in\delta(v)} x_e = 2 && v \in V \\
& \sum_{e\in S} x_e \leq |S| - 1 && \emptyset \subset S \subset V \\
& x \in \{0,1\}^n
\end{aligned}$$

Subtour elimination constraints (SEC), need to forbid solution:



IP Formulation for TSP II

variables $x_e = 1$ if $e \in T$, 0 otherwise

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in E} x_e = n \\ & \sum_{e \in \delta(S)} x_e \geq 2 \qquad \qquad \emptyset \subset S \subset V \\ & x \in \{0,1\}^n \end{aligned}$$

IP Formulation for TSP III (why valid?)

variables $x_e = 1$ if $e \in T$, 0 otherwise

$$\begin{aligned} \min \quad & \sum_{e\in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e\in E} x_e = n \\ & \sum_{e\in\delta(v_1)} x_e = 2, \quad \sum_{e\in\delta(v)} x_e = 2 && v \in V \setminus \{v_1\} \\ & \sum_{e\in S} x_e \le |S| - 1 && \emptyset \subset S \subset V \setminus \{v_1\} \\ & x \in \{0,1\}^n \end{aligned}$$

[Held & Karp 1970] Lagrange-relax all but one of the constraints $\sum_{e\in\delta(v)} x_e = 2$ (all but for node v_1)

TSP & One-Tree Relaxation

Claim

Lagrange-relaxing the degree constraints yields a spanning tree on $G \setminus \{v_1\}$ + two edges incident with v_1

Proof.

$\sum_{e \in E} x_e = n$, so we have $n-2$ edges on $G \setminus \{v_1\}$ (with $n-1$ nodes), and $\sum_{e \in \delta(S)} x_e \geq 1$ for all $\emptyset \subset S \subset V \setminus \{v_1\}$, so the solution on $G \setminus \{v_1\}$ is a connected subgraph □

- But we know how to compute a minimum spanning tree in polynomial time

The One-Tree Relaxation (given λ)

$$\begin{aligned}
&\min \sum_{e\in E} c_e x_e - \sum_{v\in V} \lambda_v \Big(\sum_{e\in\delta(v)} x_e - 2\Big) \quad (\text{let } \lambda_{v_1} = 0) \\
&= \min\ 2\sum_{v\in V} \lambda_v + \sum_{e=\{u,v\}\in E} (c_e - \lambda_v - \lambda_u) x_e \quad (\text{let } \lambda_{v_1} = 0) \\
&\text{s.t.} \ \sum_{e\in E} x_e = n, \quad \sum_{e\in\delta(v_1)} x_e = 2, \\
&\qquad \sum_{e\in S} x_e \le |S| - 1, \quad \emptyset \subset S \subset V \setminus \{v_1\} \\
&\qquad x \in \{0,1\}^n
\end{aligned}$$

Solution

Pick the two cheapest edges leaving v_1, and compute a minimum spanning tree (edge weights $c_e - \lambda_u - \lambda_v$) on the graph $G \setminus v_1$.

Held & Karp Bound for TSP

To get a good lower bound for the TSP need values for λ...

Procedure for Computing λ

- Start with $\lambda_v = 0 \; \forall \; v \in V$ in iteration 1
- in iteration k, compute optimal one-tree x^* and update λ:
  - if $k := \sum_{e \in \delta(v)} x_e^* = 1$, let $\lambda_v = \lambda_v + \Delta_k(2-k)$
    [making edges incident with v cheaper, i.e., more 'attractive']
  - if $k := \sum_{e \in \delta(v)} x_e^* \geq 3$, let $\lambda_v = \lambda_v + \Delta_k(2-k)$
    [making edges incident with v less 'attractive']

  for some "step size" $\Delta_k \geq 0$
- iterate until improvement of bound becomes marginal

By "right" choice of step size Δ_k ($\Delta_k \downarrow 0, \sum_k \Delta_k = \infty$) this converges to the optimal solution of the Lagrangian dual, LD.

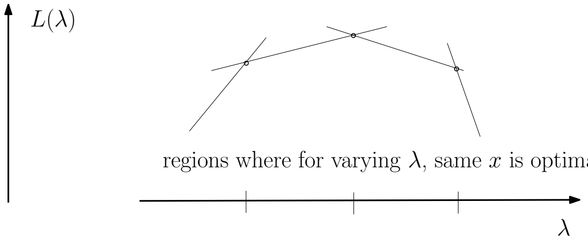
Subgradient Optimization for LD

For the TSP (minimization problem) we have $LD = \max_\lambda L(\lambda)$

- $L(\lambda)$ is linear in λ for given one-tree x^*

$$L(\lambda, x^*) = \sum_{e \in E} c_e x^*_e - \sum_{v \in V} \lambda_v \Big(\sum_{e \in \delta(v)} x^*_e - 2 \Big)$$

hence $L(\lambda) = \min_x L(\lambda, x)$ is a piecewise linear, concave function



Subgradient Optimization for LD (cont'd)

Now for any given λ and the corresponding optimal one-tree x^*, consider

$$d_v := \frac{dL(\lambda, x^*)}{d\lambda_v} = \left(2 - \sum\nolimits_{e\in\delta(v)} x_e^*\right)$$

Then d_v coincides with our update step for λ, as we defined

$$\lambda_v = \lambda_v + \Delta_k \left(2 - \sum\nolimits_{e\in\delta(v)} x_e^*\right) = \lambda_v + \Delta_k d_v$$

Exercise: vector $d = \frac{dL(\lambda,x^*)}{d\lambda} \in \mathbb{R}^n$ is a subgradient of $L(\,\cdot\,)$ at λ, i.e., $L(\lambda') \leq L(\lambda) + d^t(\lambda' - \lambda)\ \forall \lambda' \in \mathbb{R}^n$

Note: Previous "intuitive" update of λs is just a special case of what is known as subgradient optimization

More Generally

Given λ, let x^* be the optimal solution to a Lagrangian $L(\lambda)$ with

$$L(\lambda) = \min_x c^t x - \lambda^t (Ax - b)$$

then $b - Ax^*$ is a subgradient of $L(\,\cdot\,)$ at λ, that is,

$$L(\lambda') \leq L(\lambda) + (b - Ax^*)^t(\lambda' - \lambda) \quad \forall \lambda'$$

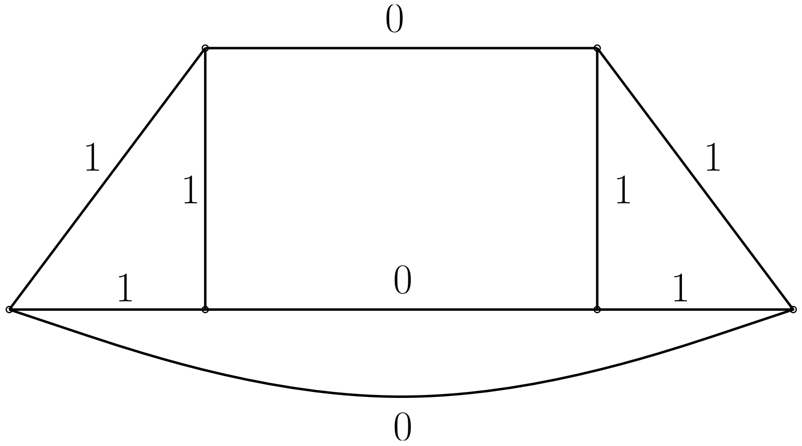
Proof: Exercise

Remark: For any $L : \mathbb{R}^n \to \mathbb{R}$ concave, the set $dL(\lambda) := \{d \in \mathbb{R}^n \mid L(\lambda') \leq L(\lambda) + d^t(\lambda' - \lambda) \quad \forall \lambda'\}$ is called subdifferential of L at λ. If L is differentiable at λ, then $dL(\lambda) = \{\nabla L(\lambda)\}$. If $0 \in dL(\lambda)$, L attains its minimum in λ

Remarks on the Held & Karp bound

- Very strong lower bound in practical implementations
- Reasonably fast to compute, so feasible within B & B
- Gives same lower bound as the LP relaxation, $LP = LD$
  [fractional 1-tree relaxation always has integer optimal solution, follows from a theorem by Edmonds 1971]
- There are examples where $LD = \max_\lambda L(\lambda) < IP$
  [to really solve TSP to optimality, can use Lagrangian relaxation only as a lower bound, e.g. within B & B]

Example for Suboptimality of 1-Tree Bound



Exercise: Optimal tour costs 4, but there is always a 1-tree with cost ≤ 3, no matter what the values of the λ_v's are

One-Trees: Trees With a Cycle!

Outline

Decision Problems

Definition

A decision (recognition) problem P is a set of instances I that can be partitioned into 'Yes' and 'No' instances I_Y, I_N, such that

- $I = I_Y \cup I_N$
- $I_Y \cap I_N = \emptyset$

Examples

- All graphs $G = (V, E)$ with edge lengths c_e, and $k \in \mathbb{N}$. Graphs with TSP tour of length $\leq k$, and those without.
- All graphs $G = (V, E)$, and $k \in \mathbb{N}$. Graphs with a matching $M \subseteq E$ of cardinality $\geq k$, and those without.

Optimization and Decision Problems

Optimization (OPT)

Given an instance $I = (S, f)$ of discrete optimization problem P, find a solution $s^* \in S$ minimizing / maximizing $f(s)$.

Decision (DEC)

Given an instance $I = (S, f)$ of discrete optimization problem P, and integer k, decide if $\exists$ solution s with $f(s) \leq / \geq k$.

Clearly, if we can solve OPT, we can also solve DEC.

What about the other direction?

Decision Problems are often General Enough

Proposition

(For quite many discrete optimization problems) if we can solve DEC in poly-time, we can also solve OPT in poly-time.

There may be cases where OPT is provably more difficult (an open research question), but in many cases it isn't.

How to solve OPT using an algorithm for DEC?

1. First compute optimal solution value ξ for OPT, by binary search on possible optimal values
2. Given ξ, construct a corresponding optimal solution

Example: Maximum Matching

Given $G = (V, E)$, find a matching of maximal cardinality.

Assume algorithm $A(k)$ that correctly tells us:
Is there a matching of size $\geq k$ in a graph G?

```
1  Binary Search for optimal value $\xi$ (note, $\xi \in [0, m]$, integer)
     • Let $[a, b] = [0, m]$
     • While $(b > a)$
         • $k = a + \lceil \frac{b-a}{2} \rceil$
         • If $A(k) =$ 'Yes' on $G$, then $a = k$
         • Otherwise $b = k - 1$
     • return $\xi := a (= b)$                         $O(\log m) \times A(k)$
2  Finding the solution
     • For all edges $(e \in E)$
         • if $A(\xi) =$ 'Yes' on graph $(G - e)$ then delete $e$ from $G$
     • return remaining edges $E(G)$                   $O(m) \times A(k)$
```

Problem class $\mathcal{P}$: Deterministic Polynomial Time

Given: Decision problem P, $|I|$ encoding length of instance $I \in P$

Definition

Algorithm A is a deterministic polynomial time algorithm for P if there exists a polynomial p such that,

- $A(I) =$'Yes' $\Leftrightarrow I$ is 'Yes' instance
- $t_A(I) \leq p(|I|)\ \forall$ instances I of P (A is a poly-time algorithm)

Definition

Problem $P \in \mathcal{P} :\Leftrightarrow P$ has deterministic poly-time algorithm

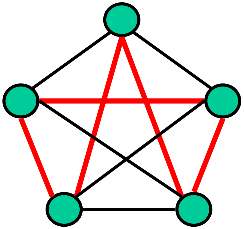
Examples: decision problems for MST, ShortestPath, MaxFlow, MinCostFlow, Matching $\in \mathcal{P}$

Certificates

Given: TSP instance, a complete undirected graph with integer edge lengths $c_e \geq 0$, $e \in E$, integer k: $\exists$ tour $\leq k$?

Can we give a short certificate, that is, a short proof that I is a 'Yes' instance?

This is simple: The tour $T \subseteq E$ itself (n edges, $|T| \in O(n)$)



Verifying the certificate

- check that T is indeed a tour
- check its length $\sum_{e \in T} c_e \leq k$?

Possible in polynomial time, $O(n^2)$

Note: Finding the certificate maybe hard, but verifying its correctness is easy (in polynomial time)

Problem class $\mathcal{NP}$: Nondeterministic Polynomial Time

Given: Decision problem P, $|I|$ encoding length of instance $I \in P$

Definition

Algorithm A is nondeterministic polynomial time algorithm for P if

- For all 'Yes' instances I, $\exists$ poly-length certificate $z(I)$ (i.e., $|z(I)| \leq p(|I|)$ for some polyn. p), and $A(I, z(I)) =$'Yes'
- For all 'No' instances I, and any poly-length input z ('fake certificate'), $A(I, z) =$'No'
- $A(I, z)$ is a poly-time algorithm for any poly-length z

Definition

$P \in \mathcal{NP} :\Leftrightarrow$ P has nondeterministic poly-time algorithm
$\Leftrightarrow$ 'Yes' instances of P have a polynomial certificate that can be verified in polynomial time

Example: We have just argued that TSP$\in \mathcal{NP}$

Why 'nondeterministic'?

Definition uses existence of a polynomial length certificate $z(I)$, but not how to obtain it!

We can interpret this equivalently as:

Consider all possible certificates z of polynomial length,

If the instance $I =$'Yes', at least one yields the algorithm to terminate with 'Yes'

Could we 'guess' the right certificate $z(I)$ for any given 'Yes' instance I, we could solve the problem

$\uparrow$

guessing = nondeterminism

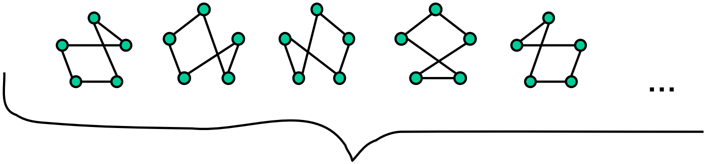
What about ‘No’ Instances

Given: TSP instance I, so a complete undirected graph with integer edge lengths $c_e \geq 0$, $e \in E$, integer k.

Can we give a short certificate that I is a ‘No’ instance?

???

All proofs seem to require exponential length...



check all $\frac{(n-1)!}{2}$ tours, see they all have length $\geq k+1$

'No' Instances with Certificates

Given: An integer matrix $A \in \mathbb{Z}^{m \times n}$: is A totally unimodular?

Short certificate for 'No' instances: square submatrix B of A

- encoding length $|B| \leq |A|$, so certainly polynomial
- verifying the certificate: verify that $\det(B) \notin \{0, \pm 1\}$

Is certificate verification possible in polynomial time?

- Laplace formula $\det(B) = \sum_{j=1}^{k} b_{ij} (-1)^{i+j} det(B_{-i,-j})$ requires $\Omega(n!)$ time
- but $O(n^3)$ is possible, computing LU decomposition of B ($B = LU$, with $L =$ lower triangular, $U =$ upper triangular)

Problem class co-$\mathcal{NP}$: Complement-$\mathcal{NP}$

Given: Decision problem P, $|I|$ encoding length of instance $I \in P$

Definition

Algorithm A is nondeterministic polynomial time algorithm for the complement of P if

- For all 'No' instances I, $\exists$ poly-length certificate $z(I)$ (i.e., $|z(I)| \leq p(|I|)$ for some polyn. p), and $A(I, z(I)) =$'Yes'
- For all 'Yes' instances I, and any poly-length input z ('fake certificate'), $A(I, z) =$'No'
- $A(I, z)$ is a poly-time algorithm for any poly-length z

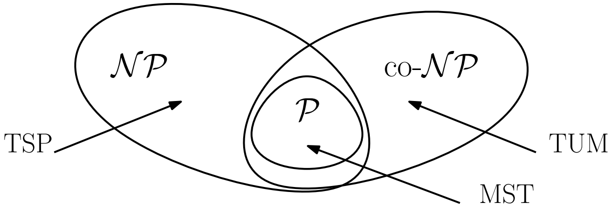
Definition

$P \in$ co-$\mathcal{NP}$:$\Leftrightarrow$ complement of P has nondet. poly-time algorithm
$\Leftrightarrow$ 'No' instances of P have a polynomial certificate that can be verified in polynomial time

$\mathcal{P}$, $\mathcal{NP}$, and co-$\mathcal{NP}$

Theorem

$\mathcal{P} \subseteq \mathcal{NP}$ and $\mathcal{P} \subseteq$ co-$\mathcal{NP}$



Open conjectures

- $\mathcal{P} \neq \mathcal{NP}$, worth 1.000.000$ claymath.org/millennium/P_vs_NP/
- $\mathcal{P} = \mathcal{NP} \cap$co-$\mathcal{NP}$, worth world-fame and recognition as scientist

Proof $\mathcal{P} \subseteq \mathcal{NP}$,co-$\mathcal{NP}$

Given Problem $Q \in \mathcal{P}$.

To show: $Q \in \mathcal{NP}$ i.e., for all '<u>Yes</u>' instances $I \in Q$ exists polynomial length certificate $z(I)$, and an algorithm A that verifies the certificate in poly-time, so $A(I, z(I)) =$ 'Yes' $\Leftrightarrow I =$ '<u>Yes</u>'.

Proof: We know $Q \in \mathcal{P}$, so Q has a poly-time algorithm A_Q with $A_Q(I) =$'<u>Yes</u>' $\Leftrightarrow I =$'<u>Yes</u>'

Now use $z(I) = \emptyset$, and define $A(I, \emptyset) := A_Q(I)$ $\square$

To see $\mathcal{P} \subseteq$co-$\mathcal{NP}$, replace all underlined '<u>Yes</u>' by 'No', and let $A(I, \emptyset) := \neg A_Q(I)$ $\square$