

DM872 – Spring 2021  
Math Optimization at Work

## Lagrangian Relaxation

Marco Chiarandini

Department of Mathematics & Computer Science  
University of Southern Denmark

*[Partly based on slides by David Pisinger, DIKU (now DTU)]*

# Outline

# Relaxation

In branch and bound we find upper bounds by relaxing the problem

## Relaxation

$$\max_{s \in P} g(s) \geq \left\{ \begin{array}{l} \max_{s \in P} f(s) \\ \max_{s \in S} g(s) \end{array} \right\} \geq \max_{s \in S} f(s)$$

- $P$ : candidate solutions;
- $S \subseteq P$  feasible solutions;
- $g(x) \geq f(x)$

Which constraints should be relaxed?

- Quality of bound (tightness of relaxation)
- Remaining problem can be solved efficiently
- Proper multipliers can be found efficiently
- Constraints difficult to formulate mathematically
- Constraints which are too expensive to write up

# Relevant Relaxations

Different relaxations

- LP-relaxation
- Deleting constraint
- Lagrange relaxation
- Surrogate relaxation
- Semidefinite relaxation

Tighter



Best surrogate  
relaxation

Best Lagrangian  
relaxation

LP relaxation

Relaxations are often used in combination.

# Surrogate Relaxation

Integer Programming Problem:  $\max\{cx \mid Ax \leq b, Dx \leq d, x \in \mathbb{Z}_+^n\}$

Relax complicating constraints  $Dx \leq d$ .

Surrogate Relax  $Dx \leq d$  using multipliers  $\lambda \geq 0$ , i.e., add together constraints using weights  $\lambda$

$$\begin{aligned} z_{SR}(\lambda) = \max \quad & cx \\ \text{s.t.} \quad & Ax \leq b \\ & \lambda Dx \leq \lambda d \\ & x \in \mathbb{Z}_+^n \end{aligned}$$

**Proposition:** Optimal Solution to relaxed problem gives an upper bound on original problem

**Proof:** show that it is a relaxation

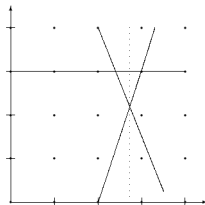
Each multiplier  $\lambda_i$  is a **weighting** of the corresponding constraint

If  $\lambda_i$  large  $\implies$  constraint satisfied (at expenses of other constraints)

If  $\lambda_i = 0 \implies$  drop the constraint

### Surrogate relaxation, example

$$\begin{array}{ll}
 \text{maximize} & 4x_1 + x_2 \\
 \text{subject to} & 3x_1 - x_2 \leq 6 \\
 & x_2 \leq 3 \\
 & 5x_1 + 2x_2 \leq 18 \\
 & x_1, x_2 \geq 0, \text{ integer}
 \end{array}$$



IP solution  $(x_1, x_2) = (2, 3)$  with  $z_{IP} = 11$

LP solution  $(x_1, x_2) = (\frac{30}{11}, \frac{24}{11})$  with  $z_{LP} = \frac{144}{11} = 13.1$

First and third constraint complicating, surrogate relax using multipliers  $\lambda_1 = 2$ , and  $\lambda_3 = 1$

$$\begin{array}{ll}
 \text{maximize} & 4x_1 + x_2 \\
 \text{subject to} & x_2 \leq 3 \\
 & 11x_1 \leq 30 \\
 & x_1, x_2 \geq 0, \text{ integer}
 \end{array}$$

Solution  $(x_1, x_2) = (2, 3)$  with  $z_{SR} = 4 \cdot 2 + 3 = 11$

Upper bound

# Tightness of Relaxations (1/2)

Integer Linear Programming problem

$$\begin{aligned} z &= \max cx \\ \text{s.t. } Ax &\leq b \\ Dx &\leq d \\ x &\in \mathbb{Z}_+^n \end{aligned}$$

Lagrangian Relaxation,  $\lambda \geq 0$ :

$$\begin{aligned} z_{LR}(\lambda) &= \max cx - \lambda(Dx - d) \\ \text{s.t. } Ax &\leq b \\ x &\in \mathbb{Z}_+^n \end{aligned}$$

with best multipliers  $\lambda$  it corresponds to:

$$z_{LD} = \max \{ cx : Dx \leq d, x \in \text{conv}(Ax \leq b, x \in \mathbb{Z}_+^n) \}$$

It corresponds to:

$$z = \max \{ cx : x \in \text{conv}(Ax \leq b, Dx \leq d, x \in \mathbb{Z}_+^n) \}$$

LP-relaxation:

$$z_{LP} = \max \{ cx : x \in Ax \leq b, Dx \leq d, x \in \mathbb{R}_+^n \}$$

Lagrange Dual Problem

$$z_{LD} = \min_{\lambda \geq 0} z_{LR}(\lambda)$$

## Tightness of Relaxations (2/2)

Surrogate Relaxation,  $\lambda \geq 0$

$$\begin{aligned} z_{SR}(\lambda) = \max \quad & cx \\ \text{s.t.} \quad & Ax \leq b \\ & \lambda Dx \leq \lambda d \\ & x \in \mathbb{Z}_+^n \end{aligned}$$

Surrogate Dual Problem

$$z_{SD} = \min_{\lambda \geq 0} z_{SR}(\lambda)$$

with best multipliers  $\lambda$ :

$$z_{SD} = \max \{ cx : x \in \text{conv}(Ax \leq b, \lambda Dx \leq \lambda d, x \in \mathbb{Z}_+^n) \}$$

↪ Best surrogate relaxation (i.e., best  $\lambda$  multipliers) is tighter than best Lagrangian relaxation.



# Relaxation strategies

Which constraints should be relaxed

- "the complicating ones"
- remaining problem is polynomially solvable  
(e.g. min spanning tree, assignment problem, linear programming)
- remaining problem is totally unimodular  
(e.g. network problems)
- remaining problem is NP-hard but good techniques exist  
(e.g. knapsack)
- constraints which cannot be expressed in MIP terms  
(e.g. cutting)
- constraints which are too extensive to express  
(e.g. subtour elimination in TSP)

# Subgradient optimization Lagrange multipliers

$$\begin{aligned} z &= \max cx \\ \text{s. t. } Ax &\leq b \\ Dx &\leq d \\ x &\in \mathbb{Z}_+^n \end{aligned}$$

Lagrange Relaxation, multipliers  $\lambda \geq 0$

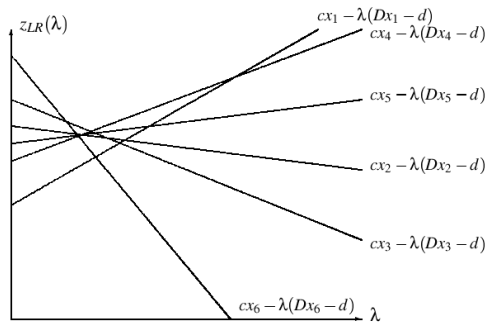
$$\begin{aligned} z_{LR}(\lambda) &= \max cx - \lambda(Dx - d) \\ \text{s. t. } Ax &\leq b \\ x &\in \mathbb{Z}_+^n \end{aligned}$$

Lagrange Dual Problem

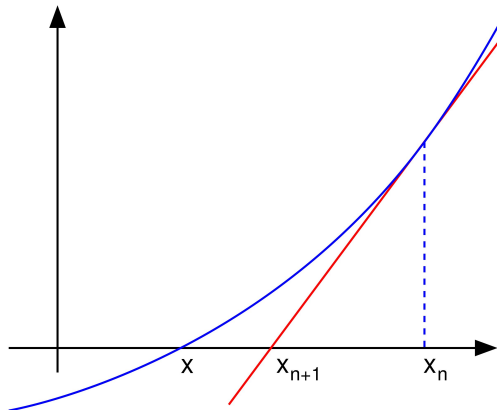
$$z_{LD} = \min_{\lambda \geq 0} z_{LR}(\lambda)$$

- We do not need best multipliers in B&B algorithm
- Subgradient optimization fast method
- Works well due to convexity
- Roots in nonlinear programming, Held and Karp (1971)

# Subgradient optimization, motivation



Lagrange function  $z_{LR}(\lambda)$  is piecewise linear and convex



Netwon-like method to minimize a function in one variable

# Digression: Gradient methods

Gradient methods are iterative approaches:

- find a descent direction with respect to the objective function  $f$
- move  $x$  in that direction by a step size

The descent direction can be computed by various methods, such as gradient descent, Newton-Raphson method and others. The step size can be computed either exactly or loosely by solving a line search problem.

Example: gradient descent

1. Set iteration counter  $t = 0$ , and make an initial guess  $x_0$  for the minimum
2. Repeat:
3.   Compute a descent direction  $\Delta_t = \nabla(f(x_t))$
4.   Choose  $\alpha_t$  to minimize  $f(x_t - \alpha\Delta_t)$  over  $\alpha \in \mathbb{R}_+$
5.   Update  $x_{t+1} = x_t - \alpha_t\Delta_t$ , and  $t = t + 1$
6. Until  $\|\nabla f(x_k)\| < tolerance$

Step 4 can be solved 'loosely' by taking a fixed small enough value  $\alpha > 0$

# Newton-Raphson method

[from Wikipedia]

Find zeros of a real-valued derivable function

$$x : f(x) = 0.$$

- Start with a guess  $x_0$
- Repeat:  
Move to a better approximation

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

until a sufficiently accurate value is reached.

Geometrically,  $(x_n, 0)$  is the intersection with the  $x$ -axis of a line tangent to  $f$  at  $(x_n, f(x_n))$ .

$$f'(x_n) = \frac{\Delta y}{\Delta x} = \frac{f(x_n) - 0}{x_n - x_{n+1}}.$$

## Subgradient

Generalization of gradients to non-differentiable functions.

### Definition

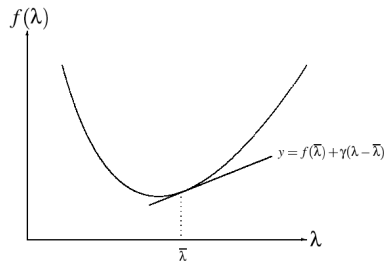
An  $m$ -vector  $\gamma$  is subgradient of  $f(\lambda)$  at  $\bar{\lambda}$  if

$$f(\lambda) \geq f(\bar{\lambda}) + \gamma(\lambda - \bar{\lambda})$$

The inequality says that the hyperplane

$$y = f(\bar{\lambda}) + \gamma(\lambda - \bar{\lambda})$$

is tangent to  $y = f(\lambda)$  at  $\lambda = \bar{\lambda}$  and supports  $f(\lambda)$  from below



**Proposition** Given a choice of nonnegative multipliers  $\bar{\lambda}$ . If  $x'$  is an optimal solution to  $z_{LR}(\lambda)$  then

$$\gamma = d - Dx'$$

is a subgradient of  $z_{LR}(\lambda)$  at  $\lambda = \bar{\lambda}$ .

**Proof** We wish to prove that from the subgradient definition:

$$\max_{Ax \leq b} (cx - \lambda(Dx - d)) \geq \max_{Ax \leq b} (cx - \bar{\lambda}(Dx - d)) + \gamma(\lambda - \bar{\lambda})$$

Using:

- an opt. solution to  $f(\bar{\lambda}) = \max_{Ax \leq b} (cx - \bar{\lambda}(Dx - d))$  is  $x'$
- the definition of  $\gamma$

$$\begin{aligned} \max_{Ax \leq b} (cx - \lambda(Dx - d)) &\geq (cx' - \bar{\lambda}(Dx' - d)) + (d - Dx')(\lambda - \bar{\lambda}) \\ &= cx' - \lambda(Dx' - d) \end{aligned}$$

## Intuition

Lagrange dual:

$$\min z_{LR}(\lambda) = cx - \lambda(Dx - d)$$

$$\text{s.t. } Ax \leq b$$

$$x \in \mathbb{Z}_+^n$$

Gradient in  $x'$  is

$$\gamma = d - Dx'$$

## Subgradient Iteration

Recursion

$$\lambda^{k+1} = \max \{ \lambda^k - \theta \gamma^k, 0 \}$$

where  $\theta > 0$  is step-size

If  $\gamma > 0$  and  $\theta$  is sufficiently small  $z_{LR}(\lambda)$  will decrease.

- Small  $\theta$  slow convergence
- Large  $\theta$  unstable



# Held and Karp procedure (gradient descent)

Initially

$$\lambda^{(0)} = \{0, \dots, 0\}$$

compute the new multipliers by recursion

$$\lambda_i^{(k+1)} := \begin{cases} \lambda_i^{(k)} & \text{if } |\gamma_i| \leq \epsilon \\ \max(\lambda_i^{(k)} - \theta \gamma_i, 0) & \text{if } |\gamma_i| > \epsilon \end{cases}$$

where  $\gamma$  is subgradient.

The step  $\theta$  is defined by

$$\theta = \mu \frac{z_{LR}(\lambda^k) - \underline{z}}{\sum_i \lambda_i^2}$$

where  $\mu$  is an appropriate constant and  $\underline{z}$  a heuristic lower bound for the original ILP problem.

E.g.  $\mu = 1$  and halved if upper bound not decreased in 20 iterations.

## **Lagrange relaxation and LP**

For an LP-problem where we Lagrange relax all constraints

- Dual variables are best choice of Lagrange multipliers
- Lagrange relaxation and LP "relaxation" give same bound

Gives a clue to solve LP-problems without Simplex

- Iterative algorithms
- Polynomial algorithms