DM872
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

Outline

1. Relaxations and Bounds

2. Subgradient Optimization

Relaxation

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

Relaxation

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

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

Which constraints should be relaxed?

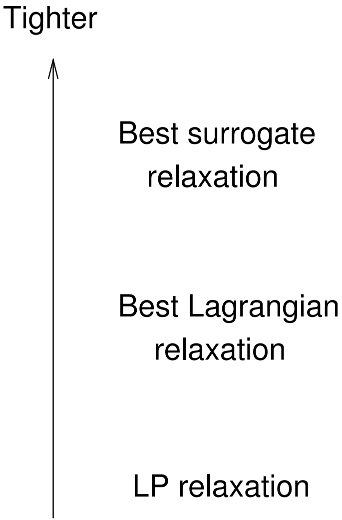
- Quality of bound (tightness of relaxation)
- Remaining problem can be solved efficiently
- Proper multipliers can be found efficiently
- Constraints difficult to formulate mathematically
- Constraints which are too expensive to write up

Relevant Relaxations

Different relaxations

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

Relaxations are often used in combination.



Surrogate Relaxation

Integer Programming Problem: $\max\{cx \mid Ax \leq b, Dx \leq d, x \in \mathbb{Z}^n_+\}$[1]
Relax complicating constraints $Dx \leq d$.
Surrogate Relax $Dx \leq d$ using multipliers $\lambda \geq 0$, i.e., add together constraints using weights λ

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

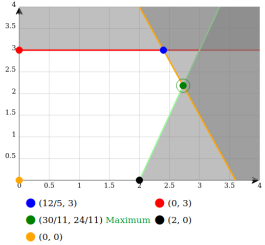
Proposition: Optimal Solution to relaxed problem gives an upper bound on original problem
Proof: show that it is a relaxation

Each multiplier λ_i is a weighting of the corresponding constraint
If λ_i large $\Longrightarrow$ constraint satisfied (at expenses of other constraints)
If $\lambda_i = 0 \Longrightarrow$ drop the constraint

[1] Notation: in this set of slides vectors are not in bold

Surrogate Relaxation: Example

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



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 constraints complicating, surrogate relax using multipliers $\lambda_1 = 2$ and $\lambda_3 = 1$:

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

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

Lagrangian Relaxation

Integer Linear Programming problem

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

We relax the $Dx \leq d$ constraints:

Lagrangian Relaxation, $\lambda \geq 0$:

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

optimizes over the x variables with λ fixed

Lagrange Dual Problem

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

optimizes over the λ variables with x fixed

Tightness of Relaxations (1/2)

Integer Linear Programming problem

$$z = \max\ cx$$
$$\text{s.t.}\ \ Ax \leq b$$
$$Dx \leq d$$
$$x \in \mathbb{Z}^n_+$$

It corresponds to:

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

LP-relaxation:

$$\boxed{z_{LP} = \max\left\{cx\ :\ x \in Ax \leq b, Dx \leq d, x \in \mathbb{R}^n_+\right\}}$$

Lagrangian Relaxation, $\lambda \geq 0$:

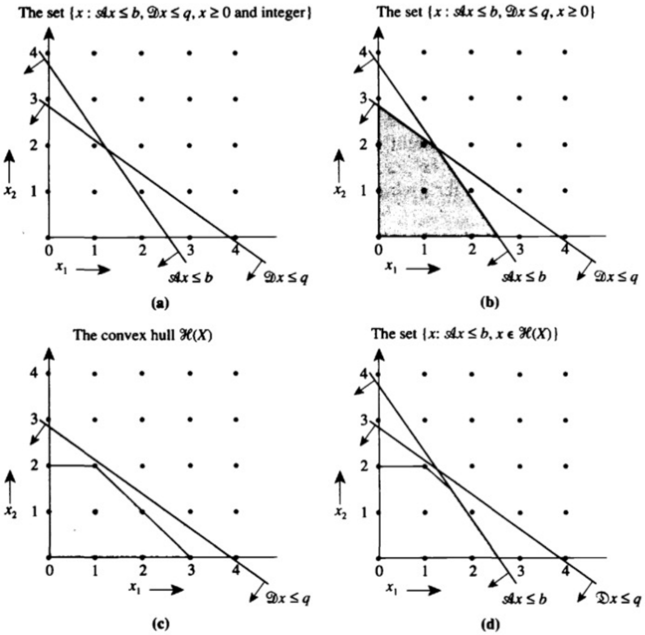
$$z_{LR}(\lambda) = \max\ cx - \lambda(Dx - d)$$
$$\text{s.t.}\ \ Ax \leq b$$
$$x \in \mathbb{Z}^n_+$$

Lagrange Dual Problem

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

with best multipliers λ it corresponds to:

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



(NB: role of $Ax \leq b$ and $Dx \leq d$ inverted wrt previous slide)

Fig 16.6 from [AMO]

Tightness of Relaxations (2/2)

Surrogate Relaxation, $\lambda \geq 0$

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

Surrogate Dual Problem

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

with best multipliers λ:

$$\boxed{z_{SD} = \max\left\{cx \ : \ x \in \operatorname{conv}(Ax \leq b, \lambda Dx \leq \lambda d, x \in \mathbb{Z}^n_+)\right\}}$$

$\rightsquigarrow$ Best surrogate relaxation (i.e., best λ 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)

Outline

1. Relaxations and Bounds

2. Subgradient Optimization

Subgradient Optimization of Lagrangian 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 $\lambda \geq 0$

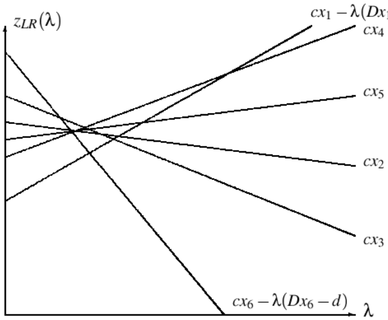
$$\begin{aligned} z_{LR}(\lambda) = \max\ & cx - \lambda(Dx - d) \\ \text{s. t.}\ & Ax \leq b \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

Lagrange Dual Problem

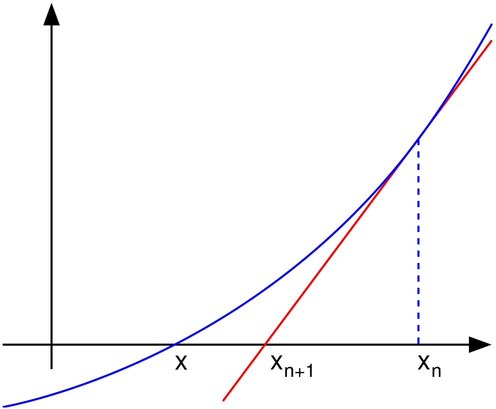
$$z_{LD} = \min_{\lambda \geq 0} z_{LR}(\lambda)$$

- 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.

Gradient descent algorithm:

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

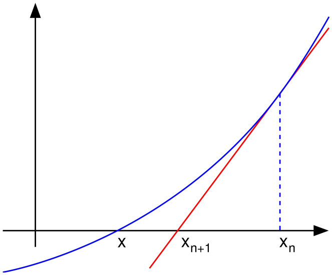
We will set α_t 'loosely' by taking small enough values $\alpha_t > 0$

Newton-Raphson method

Example of gradient algorithm:

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, $(x_{n+1}, 0)$ 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

Subgradient: Generalization of gradients to non-differentiable functions.

Definition

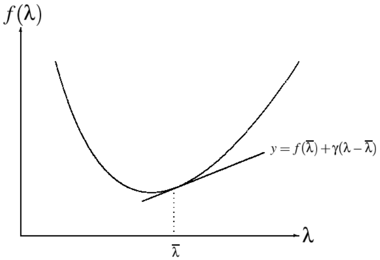
An m-vector γ is subgradient of $f(\lambda)$ at $\bar{\lambda}$ if

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

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 $z_{LR}(\bar{\lambda})$ then

$$\gamma = d - Dx'$$

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

Proof Note that for us in the LD problem: $f(\lambda) = \max_{Ax \leq b} (cx - \lambda(Dx - d))$.
We wish to prove that the inequality from the subgradient definition holds:

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

Indeed:

- We note that in the LHS: $\max_{Ax \leq b} \left(cx - \bar{\lambda}(Dx - d)\right) = (cx' - \bar{\lambda}(Dx' - d))$ because x' is by hyothesis the optimal solution of $f(\bar{\lambda})$.
- Rewriting the inequality using the hypothesis on γ we have:

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

  The right most part is the evaluation of the left most problem at a single feasible solution. Hence, it can be at most $\leq$.

Intuition

Lagrange dual:

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

Gradient in x' is

$$\gamma = d - Dx'$$

Subgradient Iteration

Recursion

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

where $\theta > 0$ is step-size

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

- Small θ slow convergence
- Large θ unstable

Held and Karp procedure (gradient descent)

Initially

$$\lambda^0 = [0, \ldots, 0]$$

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 γ is subgradient.
The step θ is defined by

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

where μ is an appropriate constant and $\underline{z}$ a heuristic lower bound for the orginal 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