In the name of God

Network Flows

6. Lagrangian Relaxation
6.2 Solving the Lagrangian Multiplier Problem

Fall 2010

Instructor: Dr. Masoud Yaghini

Outline

- Introduction
- Subgradient Optimisation Technique
- Subgradient Optimization and Inequality Constraints

Introduction

Solving the Lagrangian Multiplier Problem

- Consider the ***constrained shortest path problem***
- Suppose that now we have a time limitation of T = 14 instead of T = 10.
- When we relax the time constraint, the ***Lagrangian multiplier function L***(**μ**) becomes:

$$L(\mu) = \min\{c_P + \mu(t_P - T) : P \in \mathcal{P}\}.$$

  - where, $\mathcal{P}$ is the collection of all directed paths from the source node 1 to the sink node *n*.

Solving the Lagrangian Multiplier Problem

- For convenience, we refer to the quantity $c_p + \mu(t_P - T)$ as the ***composite cost*** of the path P.
- For a specific value of the Lagrangian multiplier μ, we can solve $L(\mu)$ by enumerating all the directed paths in $\mathcal{P}$ and choosing the path with the smallest composite cost.
- We can solve the Lagrangian multiplier problem by determining $L(\mu)$ for all nonnegative values of the Lagrangian multiplier μ and choosing the value that achieves $\mathbf{max}_{\mu \geq 0} \, L(\mu)$.

Solving the Lagrangian Multiplier Problem

- Path cost and time data with $T = 14$

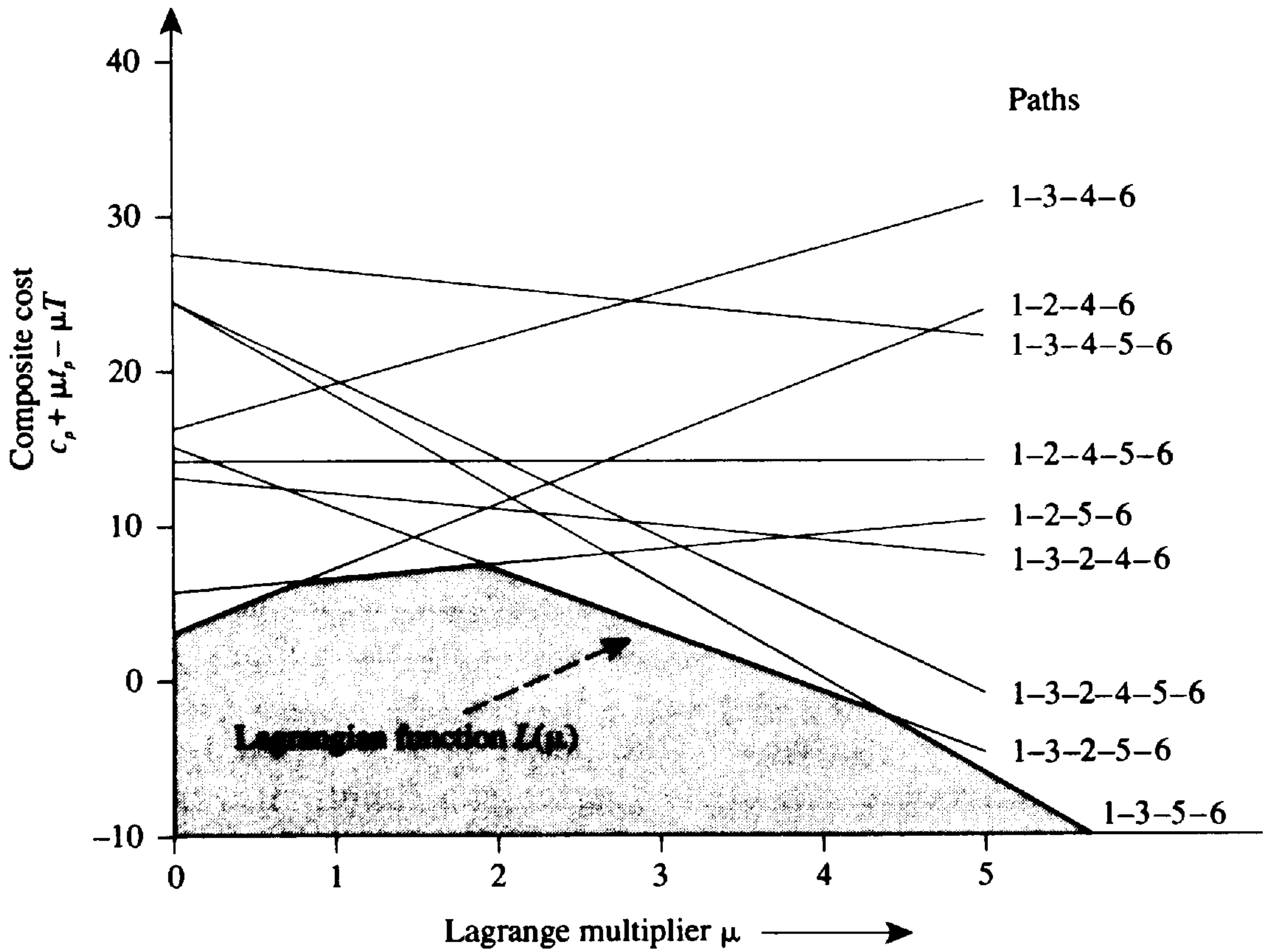
Path P	Path cost c_P	Path time t_P	Composite cost $c_P + \mu (t_P - T)$
1–2–4–6	3	18	$3 + 4\mu$
1–2–5–6	5	15	$5 + \mu$
1–2–4–5–6	14	14	14
1–3–2–4–6	13	13	$13 - \mu$
1–3–2–5–6	15	10	$15 - 4\mu$
1–3–2–4–5–6	24	9	$24 - 5\mu$
1–3–4–6	16	17	$16 + 3\mu$
1–3–4–5–6	27	13	$27 - \mu$
1–3–5–6	24	8	$24 - 6\mu$

Solving the Lagrangian Multiplier Problem

- The composite cost $c_P + \mu(t_P - T)$ for any path P is a linear function of μ with an intercept of c_P and a slope of $(t_P - T)$

Solving the Lagrangian Multiplier Problem

- Lagrangian function, $T = 14$



Solving the Lagrangian Multiplier Problem

- To find the ***optimal multiplier value*** μ^* of the Lagrangian multiplier problem, we need to find ***the highest point*** of the ***Lagrangian multiplier function*** $L(\mu)$.
- Suppose that we consider the polyhedron defined by those points that lie on or below the function $L(\mu)$.
- These are the shaded points in the Figure.
- Then geometrically, we are finding the highest point in a polyhedron defined by the function $L(\mu)$, which is a linear program.

Solving the Lagrangian Multiplier Problem

- Consider the generic optimization model (P), defined

$$\min\{cx : \mathcal{A}x = b, x \in X\}$$

- and suppose that the set X is finite.

$$X = \{x^1, x^2, \ldots, x^K\}$$

- By relaxing the constraints $Ax = b$, we obtain the Lagrangian multiplier function:

$$L(\mu) = \min\{cx + \mu(\mathcal{A}x - b) : x \in X\}.$$

- By definition:

$$L(\mu) \leq cx^k + \mu(\mathcal{A}x^k - b) \qquad \text{for all } k = 1, 2, \ldots, K.$$

Solving the Lagrangian Multiplier Problem

- In the space of composite costs and Lagrange multipliers μ, each function $cx^k + \mu(Ax^k - b)$ is a multidimensional "line" called a hyperplane (if μ is two-dimensional, it is a plane).
- The Lagrangian multiplier function $L(\mu)$ is the lower envelope of the hyperplanes $cx^k + \mu(Ax^k - b)$ for $k = 1, 2, \ldots, K$.
- In the Lagrangian multiplier problem, we wish to determine the highest point on this envelope

Solving the Lagrangian Multiplier Problem

- We can find this the highest point by solving the optimization problem:

$$\text{Maximize} \quad w$$

subject to

$$w \le cx^k + \mu(\mathcal{A}x^k - b) \qquad \text{for all } k = 1, 2, \ldots, K,$$

$$\mu \text{ unrestricted},$$

Solving the Lagrangian Multiplier Problem

- Theorem.
  - The Lagrangian multiplier problem $L^* = \max_{\mu} L(\mu)$ with

$$L(\mu) = \min\{cx^k + \mu(\mathcal{A}x - b) : x \in X\}$$

  - is equivalent to the linear programming problem

$$L^* = \max\{w : w \leq cx^k + \mu(\mathcal{A}x^k - b) \text{ for } k = 1, 2, \ldots, K\}.$$

Subgradient Optimisation Technique

Subgradient Optimisation Technique

- In solving optimization problems with the nonlinear objective function $\boldsymbol{f}(\boldsymbol{x})$ of an n-dimensional vector $\boldsymbol{x}$, researchers and practitioners often use ***gradient methods***.

Subgradient Optimisation Technique

- Suppose that in solving the ***Lagrangian multiplier problem***, we are at a point where the Lagrangian function has a unique solution $\bar{x}$.

$$L(\mu) = \min\{cx + \mu(\mathcal{A}x - b) : x \in X\}$$

- Since

$$L(\mu) = c\bar{x} + \mu(\mathcal{A}\bar{x} - b)$$

- The solution $\bar{x}$ remains optimal for small changes in the value of **μ**, the ***gradient*** at this point is

$$\mathcal{A}\bar{x} - b$$

Subgradient Optimisation Technique

- A gradient method would change the value of $\boldsymbol{\mu}$ as follows: $$\mu \leftarrow \mu + \theta(\mathcal{A}\bar{x} - b).$$
  - $\boldsymbol{\theta}$: is a step size (a scalar) that specifies how far we move in the gradient direction.
  - If $(\mathcal{A}\boldsymbol{x} - \boldsymbol{b})_i = \mathbf{0}$, the solution $\boldsymbol{x}$ uses up exactly the required units of the $\boldsymbol{i}$th resource, and we hold the Lagrange multiplier (the toll) $\boldsymbol{\mu}_i$ of that resource at its current value;
  - If $(\mathcal{A}\boldsymbol{x} - \boldsymbol{b})_i < \mathbf{0}$, the solution $\boldsymbol{x}$ uses up less than the available units of the $\boldsymbol{i}$th resource and we decrease the Lagrange multiplier $\boldsymbol{\mu}_i$ on that resource;
  - If $(\mathcal{A}\boldsymbol{x} - \boldsymbol{b})_i > \mathbf{0}$, the solution $\boldsymbol{x}$ uses up more than the available units of the $\boldsymbol{i}$th resource and we increase the Lagrange multiplier $\boldsymbol{\mu}_i$ on that resource.

Subgradient Optimisation Technique

- To solve the Lagrangian multiplier problem, let $\boldsymbol{\mu}^0$ be any initial choice of the Lagrange multiplier;
- we determine the subsequent values $\boldsymbol{\mu}^k$ for $\boldsymbol{k} = 1, 2, \ldots,$ of the Lagrange multipliers as follows:

$$\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b).$$

- where $\boldsymbol{x}^k$: is any solution to the Lagrangian subproblem when $\boldsymbol{\mu} = \boldsymbol{\mu}^k$ and $\boldsymbol{\theta}_k$ is the step length at the $\boldsymbol{k}$th iteration.

Subgradient Optimisation Technique

- To ensure that this method solves the Lagrangian multiplier problem, we need to exercise some care in the choice of the step sizes $\boldsymbol{\theta}$.
  - If we choose them too small, the algorithm would become stuck at the current point and not converge;
  - If we choose the step sizes too large, the iterates $\boldsymbol{\mu}^k$ might overshoot the optimal solution and perhaps even oscillate between two nonoptimal solutions.

Subgradient Optimisation Technique

- ***Newton's method***
  - It is an important variant of the subgradient optimization procedure
  - Suppose that

$$L(\mu^k) = cx^k + \mu^k(\mathcal{A}x^k - b)$$

  - that is, x^k solves the Lagrangian subproblem when $\boldsymbol{\mu} = \boldsymbol{\mu}^k$
  - We assume that x^k continues to solve the Lagrangian subproblem as we vary $\boldsymbol{\mu}$; or, stated in another way, we make a linear approximation

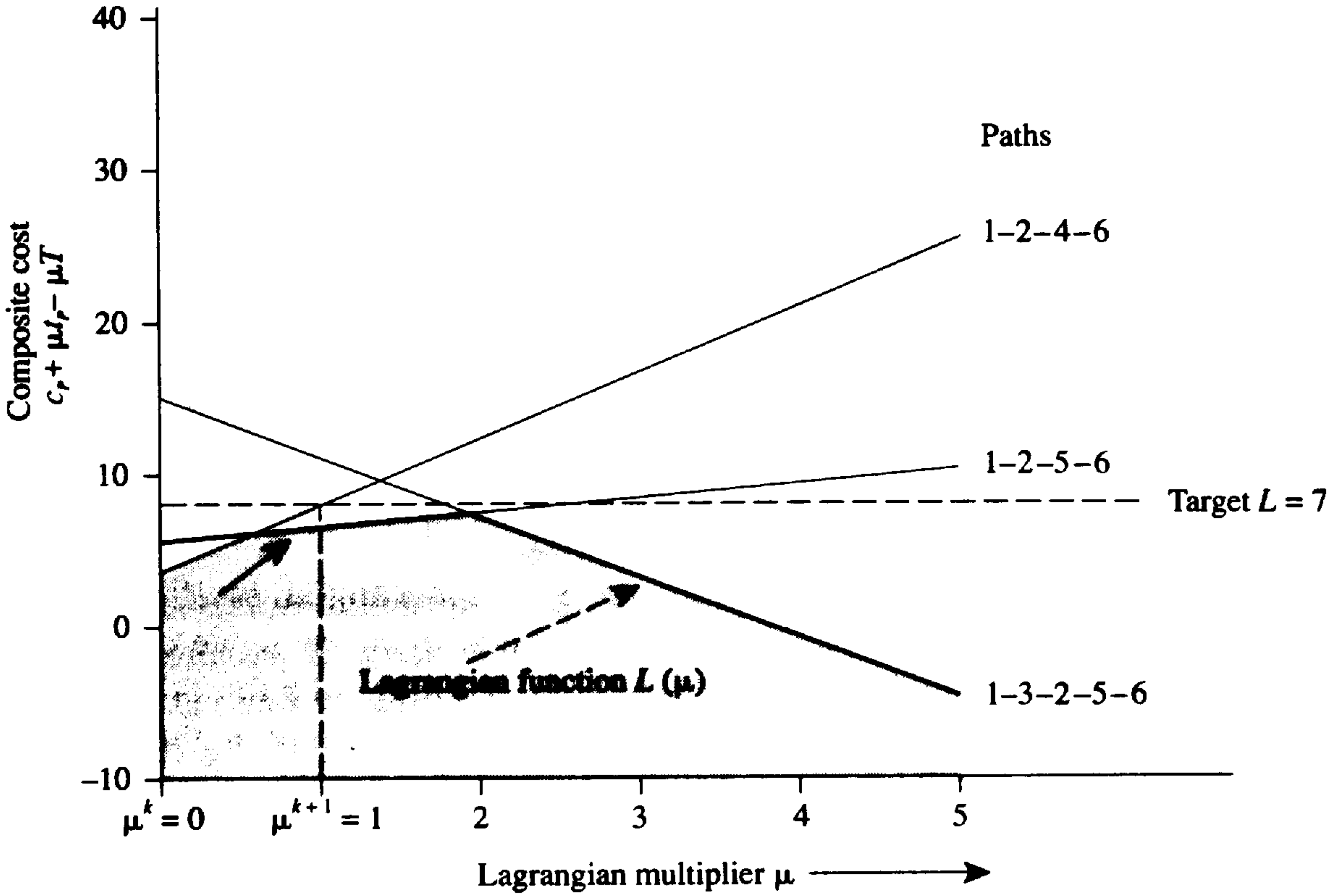
$$r(\mu) = cx^k + \mu(\mathcal{A}x^k - b) \text{ to } L(\mu)$$

Subgradient Optimisation Technique

- ***Newton's method (cont.)***
  - Suppose that we know the optimal value ***L**** of the Lagrangian multiplier problem (which we do not).
  - Then we might move in the subgradient direction until the value of the linear approximation exactly equals ***L****.

Subgradient Optimisation Technique

- The constrained shortest path example, T=14



Subgradient Optimisation Technique

- **The constrained shortest path example**
  - The figure shows an example of Newton's method when applied to the constrained shortest path example, starting with $\boldsymbol{\mu}^k = \mathbf{0}$.
  - At this point, the path $\boldsymbol{P} = \mathbf{1\text{-}2\text{-}4\text{-}6}$ solves the Lagrangian subproblem and $\mathcal{A}\boldsymbol{x} - \mathbf{b}$ equals $\boldsymbol{t_p}$ - $\mathbf{T = 18 - 14 = 4}$.
  - Since $\boldsymbol{L^*} = \mathbf{7}$ and the path $\boldsymbol{P}$ has a cost $\boldsymbol{c_p} = \mathbf{3}$, in accordance with this linear approximation, or Newton's method, we would approximate

$$\boldsymbol{L}(\boldsymbol{\mu}) \text{ by } \boldsymbol{r}(\boldsymbol{\mu}) = \mathbf{3 + 4}\boldsymbol{\mu}, \text{ set } \mathbf{3 + 4}\boldsymbol{\mu} = \mathbf{7}$$

  - and define the new value of $\boldsymbol{\mu}$ as

$$\boldsymbol{\mu}^{k+1} = \mathbf{(7 - 3)/4 = 1}$$

Subgradient Optimisation Technique

- In general, we set the step length θ_k so that

$$r(\mu^{k+1}) = cx^k + \mu^{k+1}(\mathcal{A}x^k - b) = L^*,$$

- since,

$$\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b),$$

- then,

$$r(\mu^{k+1}) = cx^k + [\mu^k + \theta_k(\mathcal{A}x^k - b)](\mathcal{A}x^k - b) = L^*.$$

Subgradient Optimisation Technique

- recalling that

$$L(\mu^k) = cx^k + \mu(\mathcal{A}x^k - b)$$

- and letting the Euclidean norm of the vector ***y***:

$$\| y \| = (\textstyle\sum_j y_j^2)^{1/2}$$

- we can solve for the ***step length*** and find that

$$\theta_k = \frac{L^* - L(\mu^k)}{\| \mathcal{A}x^k - b \|^2}.$$

Subgradient Optimisation Technique

- Since we do not know the optimal objective function value ***L**** of the Lagrangian multiplier problem,
  - practitioners of Lagrangian relaxation often use the following popular heuristic for selecting the ***step length***:

$$\theta_k = \frac{\lambda_k[\mathrm{UB} - L(\mu^k)]}{\| \mathcal{A}x^k - b \|^2}.$$

◆ **UB** : is an upper bound on the optimal objective function value *z** of the problem (**P**), and so an upper bound on ***L**** as well

◆ λ_k : is a scalar chosen (strictly) between 0 and 2.

Subgradient Optimisation Technique

- ***The heuristic procedure:***
  - Initially, the upper bound is the objective function value of any known feasible solution to the problem (P).
  - As the algorithm proceeds, if it generates a better (i.e., lower cost) feasible solution, it uses the objective function value of this solution in place of the upper bound **UB**.
  - Usually, practitioners choose the scalars $\boldsymbol{\lambda_k}$ by starting with $\boldsymbol{\lambda_k = 2}$ and then reducing $\boldsymbol{\lambda_k}$ whenever the best Lagrangian objective function value found so far has failed to increase in a specified number of iterations.
  - Since this version of the algorithm has no convenient stopping criteria, practitioners usually terminate it after it has performed a specified number of iterations.

Subgradient Optimisation Technique

- we might note that the subgradient optimization procedure is not the only way to solve the Lagrangian multiplier problem:
  - practitioners have used a number of other heuristics, including methods known as ***multiplier ascent methods***.

Subgradient Optimization and Inequality Constraints

Subgradient Optimization and Inequality Constraints

- If we apply Lagrangian relaxation to a problem with constraints $\mathcal{A}x - b$ stated in inequality form instead of the equality constraints, the Lagrange multipliers $\boldsymbol{\mu}$ are constrained to be nonnegative.
- The update formula

$$\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b)$$

- might cause one or more of the components $\boldsymbol{\mu}_i$ of $\boldsymbol{\mu}$ to become negative.

Subgradient Optimization and Inequality Constraints

- To avoid this possibility, we modify the update formula as follows:

$$\mu^{k+1} = [\mu^k + \theta_k(\mathcal{A}x^k - b)]^+$$

- where, the notation $[\mathbf{y}]^+$ denotes the "positive part" of the vector y; that is, the *i*th component of $[\mathbf{y}]^+$ equals the maximum of **0** and y_i.
- Stated in another way, if the update formula

$$\mu^{k+1} = \mu^k + \theta_k(\mathcal{A}x^k - b)$$

- would cause the *i*th component of $\boldsymbol{\mu}_i$ to be negative, then we simply set the value of this component to be zero.

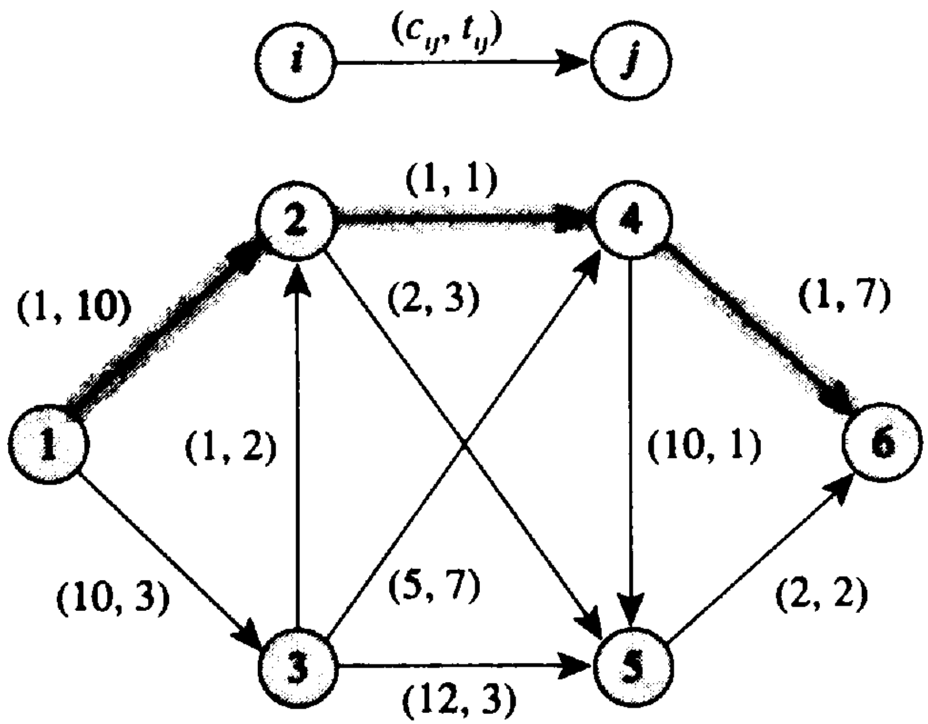
Subgradient Optimization and Inequality Constraints

- We then implement all the other steps of the subgradient procedure exactly the same as for problems with equality constraints.
  - i.e., the choice of the step size 9 at each step and
  - the solution of the Lagrangian subproblems
- For problems with both equality and inequality constraints, we use a mixture of the equality and inequality versions of the algorithm

Subgradient Optimization and Inequality Constraints

- **The constrained shortest path example:**
  - We start to solve our constrained shortest path problem at $\boldsymbol{\mu^0 = 0}$ with $\boldsymbol{\lambda^0 = 0.8}$ and with **UB = 24**, the cost corresponding to the shortest path 1-3-5-6.
  - Suppose that we choose to reduce the scalar $\boldsymbol{\lambda_k}$ by a factor of **2** whenever three successive iterations at a given value of $\boldsymbol{\lambda_k}$ have not improved on the best Lagrangian objective function value $\boldsymbol{L(\mu)}$.
  - The solution $\boldsymbol{x^0}$ to the Lagrangian subproblem with $\boldsymbol{\mu = 0}$ corresponds to the path $\boldsymbol{P}$ **= 1-2-4-6**, the Lagrangian subproblem has an objective function value of $\boldsymbol{L(0) = 3}$, and the subgradient $\boldsymbol{\mathcal{A}x^0 - b}$ at $\boldsymbol{\mu = 0}$ is $\boldsymbol{(t_p - 14) = 18 - 14 = 4}$.

Subgradient Optimization and Inequality Constraints



- bold lines denote the shortest path with ***Lagrange multiplier*** $\mu = 0$

Subgradient Optimization and Inequality Constraints

– So at the first step, we choose

$$\theta_k = \frac{\lambda_k[\text{UB} - L(\mu^k)]}{\| \mathcal{A}x^k - b \|^2}.$$

$$\theta_0 = 0.8(24 - 3)/16 = 1.05,$$

$$\mu^{k+1} = [\mu^k + \theta_k(\mathcal{A}x^k - b)]^+$$

$$\mu^1 = [0 + 1.05(4)]^+ = 4.2.$$

Subgradient Optimization and Inequality Constraints

- For **μ^1 = 4.2**, the path ***P* = 1-3-2-5-6** solves the agrangian subproblem;
- Therefore,

$$L(\mu) = \min\{c_P + \mu(t_P - T) : P \in \mathcal{P}\}.$$

$$L(4.2) = 15 + 4.2(10) - 4.2(14) = 15 - 16.8 = -1.8,$$

- And $\mathcal{A}\mathbf{x}^l - \mathbf{b}$ equals $(t_p - 14) = 10 - 14 = -4.$
- Since the path 1-3-2-5-6 is feasible, and its cost of 15 is less than **UB**, we change **UB** to value **15**.
- Therefore,

$$\theta_1 = 0.8(15 + 1.8)/16 = 0.84,$$

$$\mu^2 = [4.2 + 0.84(-4)]^+ = 0.84.$$

Subgradient Optimization and Inequality Constraints

k	μ^k	$t_p - T$	$L(\mu^k)$	λ_k	θ_k
0	0.0000	4	3.0000	0.80000	1.0500
1	4.2000	−4	−1.8000	0.80000	0.8400
2	0.8400	4	6.3600	0.80000	0.4320
3	2.5680	−4	4.7280	0.80000	0.5136
4	0.5136	4	5.0544	0.80000	0.4973
5	2.5027	−4	4.9891	0.40000	0.2503
6	1.5016	1	6.5016	0.40000	3.3993
7	4.9010	−6	−5.4059	0.40000	0.2267
8	3.5406	−4	0.8376	0.40000	0.3541
9	2.1244	−4	6.5026	0.40000	0.2124
10	1.2746	1	6.2746	0.40000	3.4902
11	4.7648	−6	−4.5886	0.40000	0.2177
12	3.4589	−4	1.1646	0.20000	0.1729
13	2.7671	−4	3.9316	0.20000	0.1384
14	2.2137	−4	6.1453	0.20000	0.1107
15	1.7709	1	6.7709	0.20000	1.6458

Subgradient Optimization and Inequality Constraints

16	3.4167	−4	1.3330	0.20000	0.1708
17	2.7334	−4	4.0664	0.20000	0.1367
18	2.1867	−4	6.2531	0.10000	0.0547
19	1.9680	1	6.9680	0.10000	0.8032
20	2.7712	−4	3.9150	0.10000	0.0693
21	2.4941	−4	5.0235	0.10000	0.0624
22	2.2447	−4	6.0212	0.05000	0.0281
23	2.1325	−4	6.4701	0.05000	0.0267
24	2.0258	−4	6.8966	0.05000	0.0253
25	1.9246	1	6.9246	0.00250	0.0202
26	1.9447	1	6.9447	0.00250	0.0201
27	1.9649	1	6.9649	0.00250	0.0201
28	1.9850	1	6.9850	0.00250	0.0200
29	2.0050	−4	6.9800	0.00250	0.0013
30	2.0000	−4	7.0000	0.00250	0.0012
31	1.9950	1	6.9950	0.00250	0.0200
32	2.0150	−4	6.9400	0.00250	0.0013
33	2.0100	−4	6.9601	0.00125	0.0006

Subgradient Optimization and Inequality Constraints

- From iterations 2 through 5, the shortest paths alternate between the paths 1-2-4-6 and 1-3-2-5-6.
- At the end of the fifth iteration, the algorithm has not improved upon (increased) the best Lagrangian objective function value of **6.36** for three iterations, so we reduce $\boldsymbol{\lambda}_k$ by a factor of 2.
- In the next 7 iterations the shortest paths are the paths 1-2-5-6, 1-3-5-6, 1-3-2-5-6, 1-3-2-5-6, 1-2-5-6, 1-3-5-6, and 1-3-2-5-6.

Subgradient Optimization and Inequality Constraints

- Once again for three consecutive iterations, the algorithm has not improved the best Lagrangian objective function value, so we decrease $\boldsymbol{\lambda}_k$ by a factor of 2 to value 0.2.
- From this point on, the algorithm chooses either path 1-3-2-5-6 or path 1-2-5-6 as the shortest path at each step.
- As we see, the Lagrangian objective function value is converging to the optimal value $L^* = 7$ and the Lagrange multiplier is converging to its optimal value of $\boldsymbol{\mu}^* = 2$.

The End