ParaLR: Paper 03/10/2016

The key to achieve maximum parallelism

$$\text{minimize} \sum_{i=1}^{N} \sum_{e \in E} c_e x_{e,i} \quad (1)$$

$$\text{subject to} \sum_{i=1}^{N} x_{e,i} \le W, \quad \forall e \in E$$

$$N_i x_i = b_i, \quad i = 1, 2, \ldots N$$

$$x_{e,i} = 0 \text{ or } 1$$

⇓

Linear programming (LP)

$c_e \rightarrow$ timing delay associated with edge e

$x_{e,i} \rightarrow$ binary variable to indicate whether routing channel e is utilized by net i or not

$W \rightarrow$ No. of track in ~~the~~ routing channel

The inequality constraints is the channel width constraints where the number of nets utilizing a routing channel must be less that a constant W.

$x_i \rightarrow$ is the vector for all $x_{e,i}$ for net i and represents the route tree of net i

$N_i \rightarrow$ node-arch incidence matrix

$b_i \rightarrow$ node demand/supply vector

N_i and b_i in the equality constraints ensures that a valid route tree is formed for each net.

⇓ apply Lagrangian relation, decompose LP into sub-problem

⇓

solve using minm steiner tree Algo and satisfy $N_i x_i = b_i$

The channel width constraints introduce dependencies among the routing of the nets. Thus transforming the channel width constraints is the key to improve parallelization only

$$\min^m \sum_{i=1}^{N} \sum_{e \in E} c_e x_{e,i} + \sum_{e \in E} \lambda_e \left(\sum_{i=1}^{M} x_{e,i} - W \right)$$

net $i = 1$ to N —②

sub to $M_i x_i = b_i$, $i = 1 \dots N$

$x_{e,i} = 0$ or 1

$\lambda_e \geq 0$, Lagrange multipliers for every channel width constraints

$\Updownarrow$ modified LP

Significantly easier to solve because every net can now be routed indepedently. with $w_e = c_e + \lambda_e$

—③

$$\min^m \sum_{i=1}^{N} \sum_{e \in E} (c_e + \lambda_e) x_{e,i} - W \sum_{e \in E} \lambda_e$$

sub $M_i x_i = b_i$ and $x_{e,i} = 0$ or 1, $\lambda_e \geq 0$

$W \sum_{e \in E} \lambda_e \rightarrow$ constant with an iteration

$\boxed{\text{Weak duality theorem}} \rightarrow$ [1]

$\Rightarrow$ as $x_{e,i} \rightarrow$ discrete so the objective funⁿ is non-differentiable so gradient descent and Newton's method cannot be applied.

, subgradient method is used ?

non-differentiable convex $\rightarrow$ used for $\max^m / \min^m$

updating Lagrange multipliers – it is updated using sub-gradient method as

$$\lambda_e^{K+1} = \max(0, \lambda_e^K + \alpha_K(\sum_{i=1}^{N} x_{e,i} - W))$$

$\alpha_K \rightarrow$ step size at iteration K

$\sum_{i=1}^{N} x_{e,i} - W \rightarrow$ is the subgradient at iteration K. which is just partial differential of objective funⁿ in (3)

(with respect to λ_e) ?

?

To ensure convergece to the optimal value, the step size α_K is determined empirically to be $0.01/K$ → ? convergence issue ?

07/10/2016

Subgradients by S. Boyd Stanford

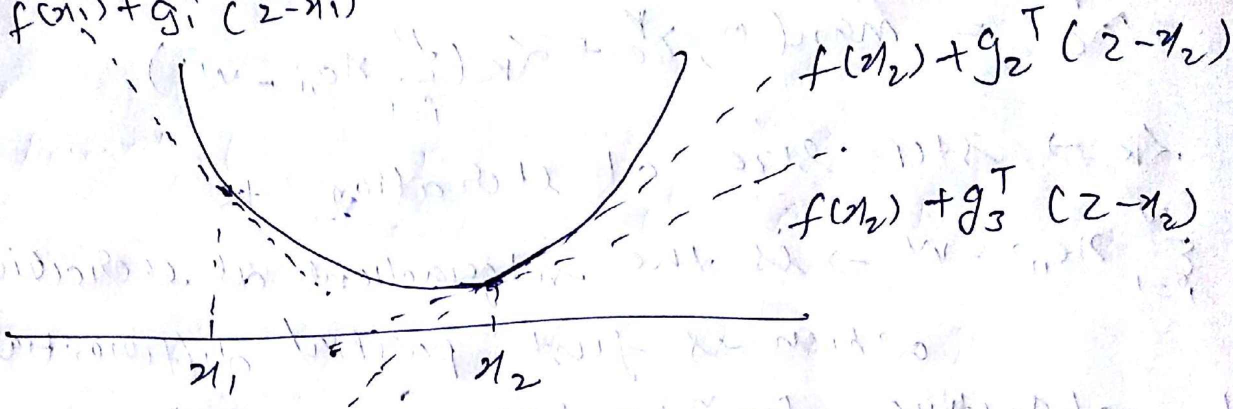
We say a vector $g \in R^n$ is a subgradient of $f: R^n \rightarrow R$ at $x \in$ dom f if for all $z \in$ dom f

$$f(z) \geq f(x) + g^T(z-x)$$

eg if f is convex and differentiable, then its gradient at x is a subgradient. But a subgradient can exist even when f is not differentiable at x.

→ A vector g is a subgradient of f at x if the affine function (of z) $f(x) + g^T(z-x)$ is a global underestimator of f

→ g is a subgradient of f at x if $(g, -1)$ supports epi f at $(x, f(x))$



At x_1, the convex fun^c is differentiable and g_1 (which is derivative of f at x_1) is the unique subgradient at x_1

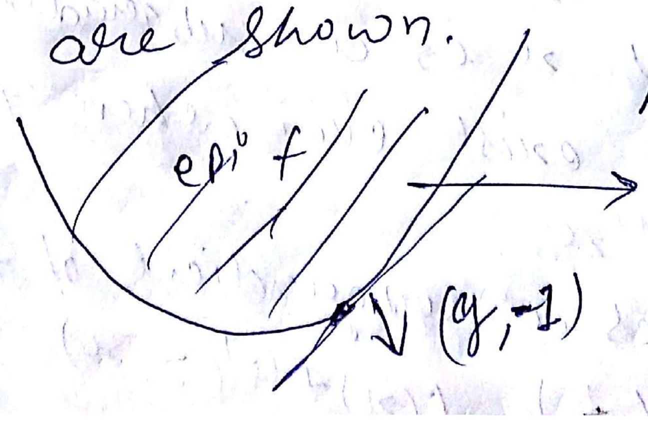
$(x_1 \quad f(x_1)) \qquad (z \quad f(z))$

$$g_1^T = \frac{f(z) - f(x_1)}{z - x_1}$$

or g_1^T

$$\Rightarrow \boxed{f(z) = f(x_1) + g_1^T (z - x_1)}$$

At x_2, f is not differentiable. At this p[oint] f has many subgradients, two subgradie[nts] g_2 and g_3 are shown.



A vector $(g, -1)$ defines a supporting hyperplane of epi of f at

$$f(z) = |z|$$

at $x<0$ $\quad \partial f(x) = \{-1\}$

at $x>0$ $\quad \partial f(x) = \{1\}$

At $x=0$ the subdifferential is defined by the inequality

$$f(z) \geq f(0) + g^T(z-0)$$

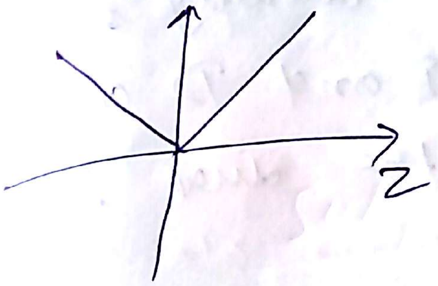
$$\boxed{|z| \geq gz} \text{ for all } z$$

which is satisfy if and only if

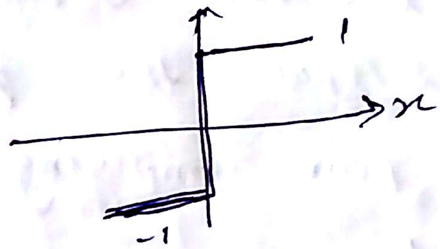
$g \in [-1, 1]$ $\Rightarrow$ $g \leq \frac{|z|}{z}$

so $\partial f(0) = [-1, 1]$

$f(z) = |z|$



$\partial f(x)$



Basic properties $\Rightarrow$

he subdifferential $\partial f(x)$ is always a closed
ivex set, even if f is not convex. this follows
m the fact that it is the intersection of an
nite set of halfspaces.

$$\partial f(x) = \bigcap_{z \in \mathrm{dom} f} \{ g \mid f(z) \geq f(x) + g^T(z-x) \}$$

→ Weak duality theorem ?

The Duality theorem

Primal (P) maximize $c^T x$

subject to $Ax \le b$, $0 \le x$

is the LP (Linear programming)

Dual (D) minimize $b^T y$

subject to $A^T y \ge c$, $0 \le y$

Since the problem D is a linear program it too has a dual. The duality terminology suggests that the problems P and D come as a pair implying that the dual to D should be P.

$$\begin{array}{lcl} \text{minimize } b^T y & & -\text{maximize } (-b)^T y \\ \text{subject to } A^T y \ge c & = & \text{subject to } (-A^T) y \le (-c) \\ 0 \le y & & 0 \le y \end{array}$$

$\xleftarrow{\text{dual}}$

$$\begin{array}{lcl} -\text{minimize } (-c)^T x & & \text{maximize } c^T x \\ \text{subject to } (-A^T)^T x \ge (-b) & = & \text{subject to } Ax \le b \\ 0 \le x & & 0 \le x \end{array}$$

The primal-dual pair of LPs P-D are related via the weak duality theorem.

Weak duality theorem :- If $x \in \mathbb{R}^n$ is feasible for P and $y \in \mathbb{R}^m$ is feasible for D then

Primal, Dual

$$c^T x \leq y^T A x \leq b^T y$$

Thus, if P is unbounded then D is necessarily infeasible and if D is unbounded, then P is necessarily infeasible, Moreover, if $c^T \bar{x} = b^T \bar{y}$ with $\bar{x}$ feasible for P and $\bar{y}$ feasible for D, then $\bar{x}$ must solve P and $\bar{y}$ must solve D

Strong Duality theorem :- if either P or D has a finite optimal value, then so does, the optimal values coincide, and optimal solutions to both P and D exist.

Subgradient method :-

Given convex $f: \mathbb{R}^n \to \mathbb{R}$, not necessarily differentiable.

just like descent, but replacing gradients with subgradients i.e., initialize $x^{(0)}$, then repeat

$$x^{(k)} = x^{(k-1)} - t_k \cdot g^{(k-1)}, \quad k = 1, 2, 3, \ldots$$

where $g^{(k-1)}$ is any subgradient of f at x^{k-1}

subgradient method is not necessarily a descent method, so we keep track of best iterate $x^{(k)}_{best}$

among $x^{(1)}, \ldots, x^{(k)}$ so far, i.e.

$$f(x^{(k)}_{best}) = \min_{i=1,\ldots k} f(x^{(i)})$$

Step size choices –

→ fixed step size, $t_k = t$ all $k = 1, 2, 3 \ldots$

→ Diminishing step size – choose step size t_k to satisfy

$$\sum_{k=1}^{\infty} t_k^2 < \infty, \quad \sum_{k=1}^{\infty} t_k = \infty$$

i.e. important that step sizes go to zero, but not ~~so f~~ too fast.

↳ All step sizes options are pre-specified, not adaptively computed

Convergence Analysis

Assume that $f: \mathbb{R}^n \to \mathbb{R}$ is convex, also:

- f is Lipschitz continuous with constant $G > 0$, ?

$$|f(x) - f(y)| \leq G\|x - y\| \quad \text{for all } x, y$$

equivalently: $\|g\| \leq G$ for any subgradient of f ~~f~~ at any x

- $\|x^{(1)} - x^*\| \leq R$, (equivalently, $\|x^{(0)} - x^*\|$ is bounded) ?

Theorem – for a fixed step size t, subgradient method satisfies

$$\lim_{k \to \infty} f(x^{(k)}_{best}) \leq f(x^*) + G^2 t/2$$

theorem : for a diminishing step size, subgradient method statisfics

$$\lim_{k \to \infty} f(x^{(k)}_{best}) = f(x^*)$$

Convergence rate

After k iterations, what is complexity of error $f(x^{(k)}_{best}) - f(x^*)$?

Consider taking $t_i = R/(G\sqrt{k})$, all $i = 1, 2, \dots k$ then the basic bound is

$$\frac{R^2 + G^2 \sum_{i=1}^{k} t_i^2}{2\sum_{i=1}^{k} t_i} = RG/\sqrt{k}$$

$$f(x^{(k)}_{best}) - f(x^*) \leq \frac{R^2 + G^2 \sum_{i=1}^{k} t_i^2}{2\sum_{i=1}^{k} t_i}$$

i.e. subgradient method has convergence rate $\boxed{O(1/\sqrt{k})}$ i.e. to get $f(x^{(k)}_{best}) - f(x^*) \leq \epsilon$ need $O(1/\epsilon^2)$ iterations.

Subgradient optimization Technique for Lagrange relaxations →

$$\min \; Cx$$
$$\text{sub to } Ax = b$$

Lagrange fun

$$L(u) = \min (Cx + u(Ax - b) : x \in X)$$

has a unique soln $\bar{x}$

since $L(u) = cx + u(Ax - b)$

and the sol^n $\bar{x}$ remains optimal for small changes in the value of u, the gradient at this point is

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

so a gradient method would change the value of u as follow

$$u \leftarrow u + \theta(A\bar{x} - b)$$

$\theta \rightarrow$ step size (a scalar) that specifies how far we move in the gradient direction.

$u^0 \leftarrow$ initial choice of Lagrange multiplier

$$u^{k+1} = u^k + \theta_k(Ax^k - b)$$

care must be take to choose θ

$\theta \rightarrow$? how to choose

case 1

if $\theta \rightarrow$ too small, the Algorithm would stuck at the current point and not converge

case 2

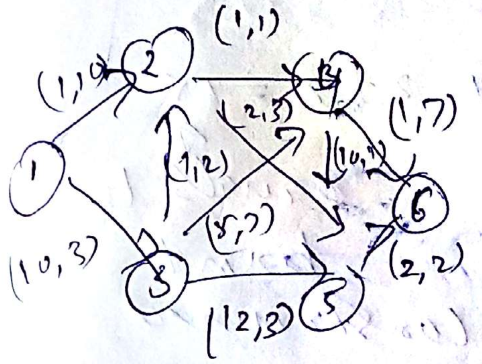
if $\theta \rightarrow$ too large, the iterates u^k might overshoot the optimal sol^n perhaps oscillate b/w two non optimal sol^n and

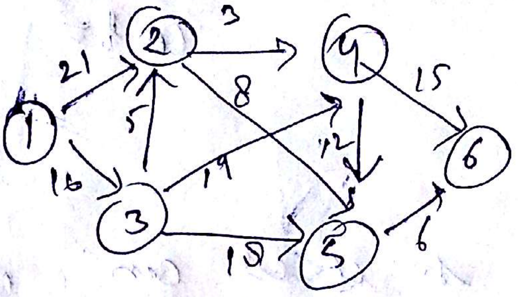
$\theta \rightarrow 0$ and $\sum_{j=1}^{k} \theta_j \rightarrow \infty$

Choose $\vartheta_k = \frac{1}{k}$ satisfies these conditions $\Rightarrow$?

(c_{ij}, t_{ij})

$(i) \xrightarrow[t]{c_{ij} \to \text{cost}} (j)$





$c_{ij} \to$ cost node i to j

$t_{ij} \Rightarrow$ traversal time

$(1) \to (2)$

$(1, 10)$

$\downarrow \quad \downarrow$

$c_{12} \quad t_{12}$

Shortest path for $\mu = 0$

$(1) \to (2) \to (4) \to (6)$

$(i) \xrightarrow{c_{ij} + \mu t_{ij}} (j)$

$\downarrow$

Lagrange multiplier

Let $\mu = 2$

$(1) \longrightarrow (2)$

$1 + 10 \times 2 = 21$

Shortest path

$(1) \to (3) \to (2) \to (5) \to (6)$

require 10 unit traversal time

$(1) \to (3) \quad t_{13} = 3$

$(3) \to (2) \quad t_{32} = 2$

$\boxed{T = 10}$

$\boxed{C + \mu t = 35}$

$\rightarrow$ Cost $c_P = \sum_{(i,j) \in P} c_{ij}$

$\rightarrow$ traversal time $t_P = \sum_{(i,j) \in P} t_{ij}$

Since the path P is feasible for the constrained shortest path problem, t_P is at most $T = 10$

$c_P + \mu t_P \rightarrow$ path's true cost plus $\mu t_P \leq \mu T$ limits with respect to modified cost $c_{ij} + \mu t_{ij}$

therefore $c_P + \mu t_P - \mu T = c_P + \mu(t_P - T) \leq c_P \rightarrow$ lower bound

$35 - 2(T) = 35 - 2 \times 10 = 15$ is a lower bound for given graph.

path $1 \to 3 \to 2 \to 5 \to 6$

cost = $\underset{\uparrow\, c_{13}}{10} + \underset{\uparrow\, c_{32}}{1} + 2 + 2 = \underset{\uparrow}{15}$ equals to lower bound.

so it is an optimal constrained shortest path.

⇒ Consider the integer programming model

Minimize cx

s.t. $x \in F$

$F \to$ the set of feasible soln to an integer program that is $x = (x_1, x_2, \ldots, x_J)$

$Ax = b$

$x_j = 0$ or 1 for $j = 1, 2, \ldots J$

for $J = 100$ (100 decision variable)

2^{100} solution

10^{-9} second / solution

$10^{-9} \times 2^{100}$ second is very large.

Let $F = \underset{\substack{\downarrow \\ \text{for } x = 0}}{F^1} \cup \underset{\substack{\downarrow \\ \text{for } x = 1}}{F^2}$

In a general $\boxed{\text{Branch and Bound}}$ procedure

$F \rightarrow$ subregion $F^1, F^2, \ldots F^K$

Let $\bar{x} \rightarrow$ best feasible soln

→ suppose for $k = 1, 2, \ldots K$ either F^K is empty or x^K is a soln of set F^K and $c\bar{x} \leq c x^K$

then no point in any of region $F^1, F^2 \ldots F^K$ could have a better objective funⁿ value than $\bar{x}$

→ If $c\bar{x} > cK^K$, subdivide the region by branching on some of the variable i.e. by setting $x_j = 0$ or $x_j = 1$ → into 2 subregion

→ In practice, in implementing the branch and bound procedure, we need to make many design decision concerning the order for choosing the subregions.

* Solving the Lagrangian multiplier problem

suppose we have time limitations of $T = 14$ instead of $T = 10$.

$$L(\mu) = \min\{c_p + \mu(t_p - T) : p \in \mathcal{P}\}$$

$\mathcal{P} \rightarrow$ collection of all directed path from the source node 1 to sink node n.

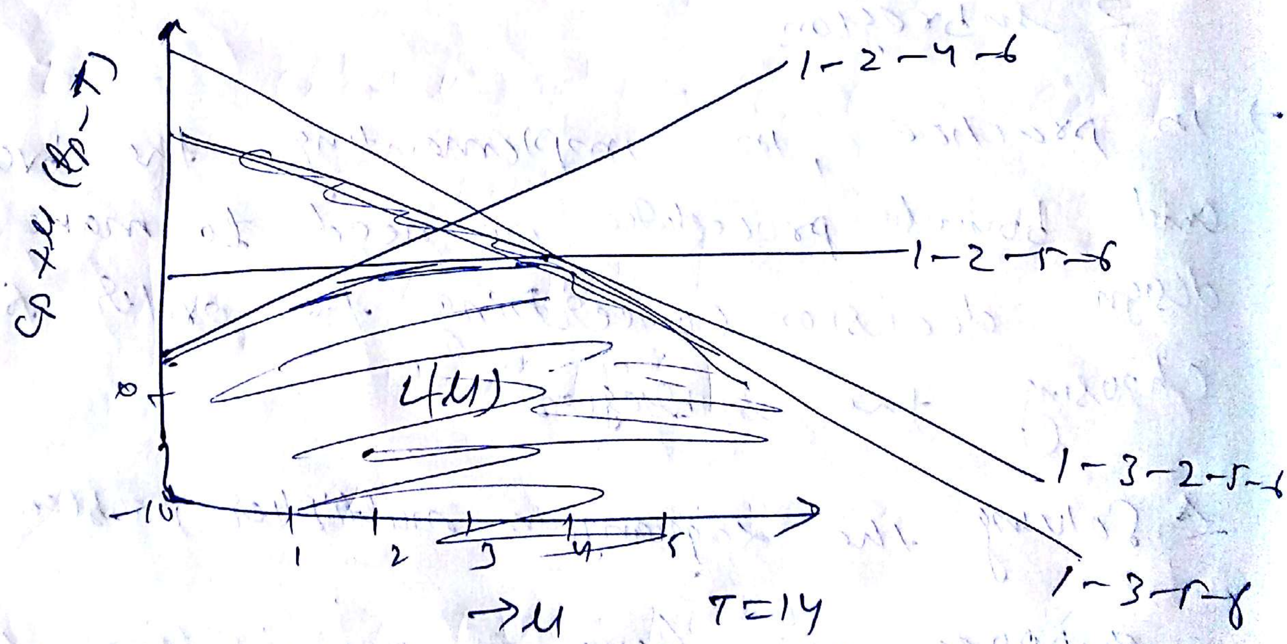
solution $= \max_{\mu \geq 0} L(\mu)$

→

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	$5+\mu$
⋮		⋮	
1-3-5-6	24	8	$24-6\mu$

this process require exponential complexity.

→ Geometric Aproach



Problem becomes

maximize w

subject to $w \le cx^k + \mu(Ax^k - b)$; for all $k=1, 2, \ldots K$

$\mu \rightarrow$ unrestricted

Scanned by CamScanner

Theorem — The Lagrangian multiplier problem $L^* = \max_u L(u)$ with $L(u) = \min(cx^k + u(Ax-b) ; x \in X)$ is equivalent to the linear programming problem $L^* = \max(w ; w \le cx^k + u(Ax^k - b)$ for $k = 1, 2, \ldots K\}$

↓

Linear programming

↓

→ Dantzig-Wolfe decomposition position or generalized linear programming Algorithm

↓ other approach
→ gradient method to Lagrangian fun^n $L(u)$ (for non differentiable fun^n)

at $u = 0$ differentiable

→ subgradient optimization technique

⇒ variant of subgradient optimization procedure could be an adaption of "Newton's method" for solving nonlinear equations

i.e. we make a linear approximation $r(u) = cx^k + u(Ax^k - b)$ to $L(u)$

suppose we know the optimal value L^* of Lagrange multiplies (when we do not) then we might move in the subgradient direction untill the value of linear approximations exactly equals L^*

the path $P = 1 \to 2 \to 4 \to 6$

$c_p = 3 \quad t_p = 18 \quad T = 14$

$Ax^k - b$ equal $t_p - T = 4$. ~~$L^* = 7$~~

since $L^* = 7$.

then approximate $z(u)$ by $g(u) = c_p + u(Ax - b) = 3 + 4u$

$7 = 3 + 4u \Rightarrow$

u as $u^{k+1} = (7-3)/4 = 1$

in general

$$g(u^{k+1}) = cx^k + u^{k+1}(Ax^k - b) = L^*$$

$$u^{k+1} = u^k + \theta_k (Ax^k - b)$$

so

$$g(u^{k+1}) = cx^k + \left(u^k + \theta_k (Ax^k - b)\right)(Ax^k - b) = L^*$$

$$L^* = \underbrace{cx^k + u^k(Ax^k - b)}_{L(u^k)} + \theta_k \underbrace{(Ax^k - b)(Ax^k - b)}_{\|Ax^k - b\|_2^2}$$

$$\boxed{\theta_k = \frac{L^k - L(u^k)}{\|Ax^k - b\|^2}}$$

since L^* is unknown

so $$\theta_k = \frac{\lambda_k (UB - L(u^k))}{\|Ax^k - b\|^2}$$

$UB \to$ upper bound on the optimal objective funⁿ value z^* of problem (P)

$\lambda_k \to$ scalar chosen (strictly) b/w 0 and 2 ?

Usually choose $\lambda_k = 2$ (initially) and then reducing λ_k by factor of 2 whenever the best Lagrange objective func value found so far has failed to increase in a specific number of iteration.

?

→ To avoid -ve vity of u^{k+1}

$$u^{k+1} = [u^k + \theta_k (Ax^k - b)]^+$$

$[y]^+$ → denotes +ve part of vector y; that is, the i^{th} component of $[y]^+$ equals maximum of 0 and y_i — i.e.

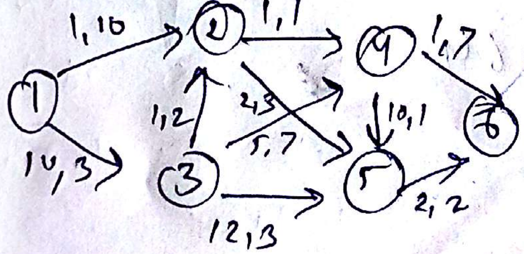
If the update formula

$$u^{k+1} = u^k + \theta_k (Ax^k - b)$$ would

cause i^{th} component $u_i \to$ -ve

then $u_i \to 0$

Eg



$u^0 = 0$

$\lambda^0 = 0.8$

Reduced λ by factor 2 whenever 3 consecutive iteration at a given value of λ_k have not improved the best Lagrange —

P = 1 – 2 – 4 – 6 $\quad$ $t_P = 18$

T = 14

U.B = 24

$L(0) = 3$ $\quad$ $Ax^k - b$ at $u = 0$ $\quad$ $t_P - T = 4$

$\theta_0 = 0.8(24 - 3)/4^2 = 1.05$

$$\mu^1 = [\mu^0 + \theta^0 (Ax^k - b)]^+$$

$$= (0 + 1.05(4))^+ = 4.2$$

for this value of ~~lagrange~~ lagrange ($\mu^1 = 4.2$) solve subproblem

Path $P = 1-3-2-5-6$

$$L(4.2) = c_p + \mu^1 (t_p - T)$$

now $24 = c_p + 4.2(4)$ (L^+ or UB, c_p, μ', $Ax^k - b$)

so $c_p = 7.2$ $1-3-2-5-6$ only

$$= 15 + 4.2(10 - 14)$$

(t_p of $1-3-2-5-6$)

$$= -1.8$$

and $Ax^1 - b = t_p - T = -4$

since ~~L(4.2)~~ path $1-3-2-5-6$ feasible

and

$c_p < U.B$

$15 \quad 24.$

therefore now $U.B. = 15$ (c_p of $1-3-2-5-6$)

$$\theta^1 = 0.8(15 - (-1.8)) / (-4)^2$$

$$= 0.84$$

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

$P = 1-3-2-5-6$ same path

$$L(\mu^2) = c_p + \mu^2 (t_p - T)$$

now for μ^2: $15 = c_p + 0.84 \times (-4)$, $c_p = 18.36$

$$= 15 + .84(-4)$$

$$= 11.64$$