LAGRANGIAN RELAXATION WITH GAMS

ERWIN KALVELAGEN

Abstract. This document describes an implementation of Lagrangian Relaxation using GAMS.

1. Introduction

Lagrangian Relaxation techniques [2, 3] form an important and popular tool in discrete optimization. We will show how Lagrangian Relaxation with subgradient optimization can be implemented in a GAMS environment.

2. Lagrangian Relaxation

We consider the Mixed Integer Programming model:

$$\boxed{\begin{array}{lll} \text{MIP} & \underset{x}{\text{minimize}} & z = c^T x \\ & & Ax \geq b \\ & & Bx \geq d \\ & & x \geq 0 \\ & & x_j \in \{0, 1, \ldots, n\} \text{ for } j \in J \end{array}}$$

There are two sets of linear constraints. We assume the set $Ax \geq b$ are the complicating constraints: if we relax the problem by removing these constraints, the remaining problem

$$\begin{aligned} \min\ & c^T x \\ & Bx \geq d \\ & x \geq 0 \\ & x_j \in \{0, 1, \ldots, n\} \text{ for } j \in J \end{aligned} \tag{1}$$

is relatively easy to solve.

We can form the *Lagrangian Dual*:

$$\begin{aligned} L(u) = \min\ & c^T x + u^T (b - Ax) \\ & Bx \geq d \\ & x \geq 0 \\ & x_j \in \{0, 1, \ldots, n\} \text{ for } j \in J \end{aligned} \tag{2}$$

For any $u \geq 0$, $L(u)$ forms a lower bound on problem MIP, as $u^T (b - Ax) \leq 0$. I.e. we have $L(u) \leq z$.

The task is to find

$$\max_{u \geq 0} L(u) \tag{3}$$

Date: 20 december 2002, revised may 2009.

which can provide a better bound than a linear programming relaxation. I.e.

$$z_{LP} \le \max_{u \ge 0} L(u) \le z \tag{4}$$

where z_{LP} is the optimal objective of the linear programming relaxation.

3. Subgradient Optimization

It can be shown that $L(u)$ is a piecewise linear function. Solving (3) is therefore a nondifferentiable optimization problem. A successful technique for this problem is *Subgradient Optimization*.

Following the notation in [5], the subgradient algorithm can be summarized as:

{Input}
An upper bound L^*
An initial value $u^0 \ge 0$
{Initialization}
$\theta_0 := 2$
{Subgradient iterations}
for $j := 0, 1, \dots$ **do**
  $\gamma^j := g(x^j)$ {gradient of $L(u^j)$}
  $t_j := \theta_j (L^* - L(u^j))/||\gamma^j||^2$ {step size}
  $u^{j+1} := \max\{0, u^j + t_j \gamma^j\}$
  if $||u^{j+1} - u^j|| < \varepsilon$ **then**
    Stop
  end if
  if no progress in more than K iterations **then**
    $\theta_{j+1} := \theta_j / 2$
  else
    $\theta_{j+1} := \theta_j$
  end if
  $j := j + 1$
end for

There are many variants possible with respect to the calculation of the step size and the updating of the parameter θ [1, 4].

4. Example

In the GAMS model below we illustrate the technique described above using a generalized assignment problem:

$$\begin{aligned} \min \ & \sum_i \sum_j c_{i,j} x_{i,j} \\ & \sum_j x_{i,j} = 1 \ \ \forall i \\ & \sum_i a_{i,j} x_{i,j} \le b_j \ \ \forall j \end{aligned} \tag{5}$$

The assignment constraint $\sum_j x_{i,j} = 1$ will be dualized. The resulting obtained bound of 12.5 is tighter than the LP relaxation bound of 6.4. To be complete, the optimal MIP solution is 18.

Model lagrel.gms. [1]

```
$ontext


  Lagrangian Relaxation
  using a Generalized Assignment Problem

  LP Relaxation  : 6.4286
  Lagrangian Relaxation : 12.5
  Optimal Integer Solution : 18

  Erwin Kalvelagen, Amsterdam Optimization


  Reference:
    Richard Kipp Martin, Large Scale Linear and Integer Optimization,
    Kluwer, 1999


$offtext

set i 'tasks' /i1*i3/;
set j 'servers' /j1*j2/;

parameter b(j) 'available resources' /
   j1 13
   j2 11
/;

table c(i,j) 'cost coefficients'
       j1    j2
i1      9     2
i2      1     2
i3      3     8
;

table a(i,j) 'resource usage'
       j1    j2
i1      6     8
i2      7     5
i3      9     6
;

*-----------------------------------------------------------------------
* standard MIP problem formulation
* solve as RMIP to get initial values for the duals
*-----------------------------------------------------------------------

variables
   cost   'objective variable'
   x(i,j) 'assignments'
;
binary variable x;

equations
   obj         'objective'
   assign(i)   'assignment constraint'
   resource(j) 'resource limitation constraint'
;

obj..             cost =e= sum((i,j), c(i,j)*x(i,j));
assign(i)..       sum(j, x(i,j)) =e= 1;
resource(j)..     sum(i, a(i,j)*x(i,j)) =l= b(j);

option optcr=0;
model genassign /obj,assign,resource/;
solve genassign minimizing cost using rmip;
```

[1] http://www.amsterdamoptimization.com/models/lagrel.gms

```
*-------------------------------------------------------------------
* Lagrangian dual
* Let assign be the complicating constraint
*-------------------------------------------------------------------


parameter u(i);
variable bound;

equation LR 'lagrangian relaxation';
LR.. bound =e= sum((i,j), c(i,j)*x(i,j))
              + sum(i, u(i)*[1-sum(j,x(i,j))]);

model ldual /LR,resource/;


*-------------------------------------------------------------------
* subgradient iterations
*-------------------------------------------------------------------


set iter /iter1*iter50/;
scalar continue /1/;
parameter stepsize;
scalar theta /2/;
scalar noimprovement /0/;
scalar bestbound /-INF/;
parameter gamma(i);
scalar norm;
scalar upperbound;
parameter uprevious(i);
scalar deltau;
parameter results(iter,*);


*
* initialize u with relaxed duals
*
u(i) = assign.m(i);
display u;


*
* an upperbound on L
*
parameter initx(i,j) / i1.j1 1, i2.j2 1, i3.j2 1 /;
upperbound = sum[(i,j), c(i,j)*initx(i,j)];
display upperbound;

loop(iter$continue,

*
* solve the lagrangian dual problem
*
    option optcr=0;
    option limrow = 0;
    option limcol = 0;
    ldual.solprint = 0;
    solve ldual minimizing bound using mip;
    results(iter,'dual obj') = bound.l;
    if (bound.l > bestbound,
       bestbound = bound.l;
       display bestbound;
       noimprovement = 0;
    else
       noimprovement = noimprovement + 1;
       if (noimprovement > 1,
          theta = theta/2;
          noimprovement = 0;
       );
    );
    results(iter,'noimprov') = noimprovement;
```

```
    results(iter,'theta') = theta;

*
* calculate step size
*

    gamma(i) = 1-sum(j,x.l(i,j));
    norm = sum(i,sqr(gamma(i)));
    stepsize = theta*(upperbound-bound.l)/norm;
    results(iter,'norm') = norm;
    results(iter,'step') = stepsize;



*
* update duals u
*
    uprevious(i) = u(i);
    u(i) = max(0, u(i)+stepsize*gamma(i));
    display u;

*
* converged ?
*
   deltau = smax(i,abs(uprevious(i)-u(i)));
   results(iter,'deltau') = deltau;
    if( deltau < 0.01,
      display "Converged";
      continue = 0;
   );



);

display results;
```

References

1. M. S. Bazaraa and H. D. Sherali, *On the choice of step sizes in subgradient optimization*, Journal of Operational Research **7** (1981), 380–388.
2. Marshall L. Fisher, *An applications oriented guide to lagrangian relaxation*, Interfaces **15** (1985), no. 2, 10–21.
3. A. M. Geoffrion, *Lagrangian relaxation and its uses in integer programming*, Mathematical Programming Study **2** (1974), 82–114.
4. M. H. Held, P. Wolfe, and H. D. Crowder, *Validation of subgradient optimization*, Mathematical Programming **6** (1974), no. 1, 62–88.
5. Richard Kipp Martin, *Large scale linear and integer optimization; a unified approach*, Kluwer, 1999.

Amsterdam Optimization Modeling Group., Washington D.C./The Hague
E-mail address: erwin@amsterdamoptimization.com