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:

\[
\begin{align*}
\text{MIP} \quad & \text{minimize} & & z = c^T x \\
& & & Ax \geq b \\
& & & Bx \geq d \\
& & & x \geq 0 \\
& & & x_j \in \{0, 1, \ldots, n\} \text{ for } j \in J
\end{align*}
\]

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

\[
\begin{align*}
\min & & c^T x \\
& & Bx \geq d \\
& & x \geq 0 \\
& & x_j \in \{0, 1, \ldots, n\} \text{ for } j \in J
\end{align*}
\]

is relatively easy to solve.

We can form the \textit{Lagrangian Dual}:

\[
L(u) = \min \ c^T x + u^T (b - Ax)
\]

\[
\begin{align*}
& & Bx \geq d \\
& & x \geq 0 \\
& & x_j \in \{0, 1, \ldots, n\} \text{ for } j \in J
\end{align*}
\]

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

\[
\max_{u \geq 0} L(u)
\]

Date: 20 december 2002, revised may 2009.
which can provide a better bound than a linear programming relaxation. I.e.

\[ z_{LP} \leq \max_{u \geq 0} L(u) \leq z \]

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:

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

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

4. Example

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

\[
\begin{align*}
\min & \quad \sum_i \sum_j c_{i,j} x_{i,j} \\
\text{subject to} & \quad \sum_j x_{i,j} = 1 \ \forall i \\
& \quad \sum_i a_{i,j} x_{i,j} \leq b_j \ \forall j
\end{align*}
\]

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.

$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;

http://www.amsterdamoptimization.com/models/lagrel.gms
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/;

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


AMSTERDAM OPTIMIZATION MODELING GROUP., WASHINGTON D.C./THE HAGUE
E-mail address: erwin@amsterdamoptimization.com