DANTZIG-WOLFE DECOMPOSITION WITH GAMS

ERWIN KALVELAGEN

Abstract. This document illustrates the Dantzig-Wolfe decomposition algorithm using GAMS.

1. Introduction

Dantzig-Wolfe decomposition [2] is a classic solution approach for structured linear programming problems. In this document we will illustrate how Dantzig-Wolfe decomposition can be implemented in a GAMS environment. The GAMS language is rich enough to be able to implement fairly complex algorithms as is illustrated by GAMS implementations of Benders Decomposition [10], Cutting Stock Column Generation [11] and branch-and-bound algorithms [12].

Dantzig-Wolfe decomposition has been an important tool to solve large structured models that could not be solved using a standard Simplex algorithm as they exceeded the capacity of those solvers. With the current generation of simplex and interior point LP solvers and the enormous progress in standard hardware (both in terms of raw CPU speed and availability of large amounts of memory) the Dantzig-Wolfe algorithm has become less popular.

Implementations of the Dantzig-Wolfe algorithm have been described in [5, 6, 7]. Some renewed interest in decomposition algorithms was inspired by the availability of parallel computer architectures [8, 13]. A recent computational study is [16]. [9] discusses formulation issues when applying decomposition on multi-commodity network problems. Many textbooks on linear programming discuss the principles of the Dantzig-Wolfe decomposition [1, 14].

2. Block-angular models

Consider the LP:

$$\begin{aligned} \min c^T x& \\ Ax &= b \\ x &\ge 0 \end{aligned} \tag{1}$$

where A has a special structure:

$$Ax = \begin{pmatrix} B_0 & B_1 & B_2 & \dots & B_K \\ & A_1 & & & \\ & & A_2 & & \\ & & & \ddots & \\ & & & & A_K \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_K \end{pmatrix} \tag{2}$$

Date: May, 2003.

The constraints

$$\sum_{k=0}^{K} B_k x_k = b_0 \tag{3}$$

corresponding to the top row of sub-matrices are called the *coupling constraints*.

The idea of the Dantzig-Wolfe approach is to decompose this problem, such that never a problem has to be solved with all sub-problems $A_k x_k = b_k$ included. Instead a *master problem* is devised which only concentrates on the coupling constraints, and the sub-problems are solved individually. As a result only a series of smaller problems need to be solved.

3. Minkowski's Representation Theorem

Consider the feasible region of an LP problem:

$$P = \{x | Ax = b, x \geq 0\} \tag{4}$$

If P is bounded then we can characterize any point $x \in P$ as a linear combination of its extreme points $x^{(j)}$:

$$\begin{aligned} x &= \sum_j \lambda_j x^{(j)} \\ \sum_j \lambda_j &= 1 \\ \lambda_j &\geq 0 \end{aligned} \tag{5}$$

If the feasible region can not assumed to be bounded we need to introduce the following:

$$\begin{aligned} x &= \sum_j \lambda_j x^{(j)} + \sum_i \mu_i r^{(i)} \\ \sum_j \lambda_j &= 1 \\ \lambda_j &\geq 0 \\ \mu_i &\geq 0 \end{aligned} \tag{6}$$

where $r^{(i)}$ are the extreme rays of P. The above expression for x is sometimes called *Minkowski's Representation Theorem*[15]. The constraint $\sum_j \lambda_j = 1$ is also known as the *convexity constraint*.

A more compact formulation is sometimes used:

$$\begin{aligned} x &= \sum_j \lambda_j x^{(j)} \\ \sum_j \delta_j \lambda_j &= 1 \\ \lambda_j &\geq 0 \end{aligned} \tag{7}$$

where

$$\delta_j = \begin{cases} 1 & \text{if } x^{(j)} \text{ is an extreme point} \\ 0 & \text{if } x^{(j)} \text{ is an extreme ray} \end{cases} \tag{8}$$

I.e. we can describe the problem in terms of variables λ instead of the original variables x. In practice this reformulation can not be applied directly, as the number of variables λ_j becomes very large.

4. The decomposition

The K subproblems are dealing with the constraints

$$\begin{aligned} A_k x_k &= b_k \\ x_k &\geq 0 \end{aligned} \tag{9}$$

while the *Master Problem* is characterized by the equations:

$$\begin{aligned} &\min \sum_k c_k^T x_k \\ &\sum_k B_k x_k = b_0 \\ &x_0 \geq 0 \end{aligned} \tag{10}$$

We can substitute equation 7 into 10, resulting in:

$$\begin{aligned} &\min c_0^T x_0 + \sum_{k=1}^{K} \sum_{j=1}^{p_k} (c_k^T x_k^{(j)}) \lambda_{k,j} \\ &B_0 x_0 + \sum_{k=1}^{K} \sum_{j=1}^{p_k} (B_k x_k^{(j)}) \lambda_{k,j} = b_0 \\ &\sum_{j=1}^{p_k} \delta_{k,j} \lambda_{k,j} = 1 \text{ for } k = 1, \dots, K \\ &x_0 \geq 0 \\ &\lambda_{k,j} \geq 0 \end{aligned} \tag{11}$$

This is a huge LP. Although the number of rows is reduced, the number of extreme points and rays $x_k^{(j)}$ of each subproblem is very large, resulting in an enormous number of variables $\lambda_{k,j}$. However many of these variables will be non-basic at zero, and need not be part of the problem. The idea is that only variables with a promising reduced cost will be considered in what is also known as a *delayed column generation* algorithm.

The model with only a small number of the λ variables, compactly written as:

$$\begin{aligned} &\min c_0^T x_0 + c^T \lambda' \\ &B_0 x_0 + B\lambda' = b_0 \\ &\Delta \lambda' = 1 \\ &x_0 \geq 0 \\ &\lambda' \geq 0 \end{aligned} \tag{12}$$

is called the *restricted master problem.* The missing variables are fixed at zero. The restricted master problem is not fixed in size: variables will be added to this problem during execution of the algorithm.

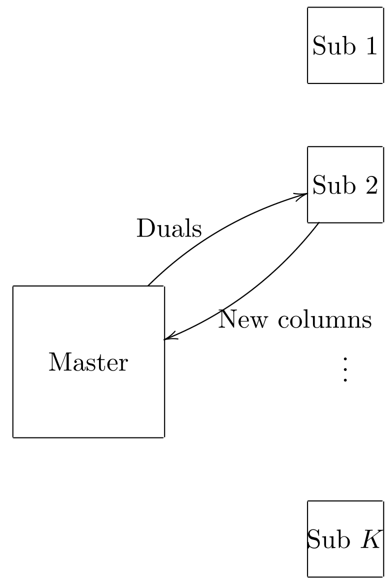

FIGURE 1. Communication between restricted master and sub-problems

The attractiveness of a variable $\lambda_{k,j}$ can be measured by its reduced cost[1]. If we denote the dual variables for constraint $B_0x_0 + B\lambda' = b_0$ by π_1 and those for the convexity constraints $\sum_j \delta_{k,j}\lambda'_{k,j} = 1$ by $\pi_2^{(k)}$, then reduced cost for the master problem look like:

$$\sigma_{k,j} = (c_k^T x_k^{(j)}) - \pi^T \begin{pmatrix} B_k x_k^{(j)} \\ \delta_{k,j} \end{pmatrix} = (c_k^T - \pi_1^T B_k) x_k^{(j)} - \pi_2^{(k)} \delta_{k,j} \tag{14}$$

Assuming the sub-problem to be bounded, the most attractive *bfs* (basic feasible solution) x_k to enter the master problem is found by maximizing the reduced cost giving the following LP:

$$\begin{aligned} \min_{x_k} \ & \sigma_k = (c_k^T - \pi_1^T B_k) x_k - \pi_2^{(k)} \\ & B_k x_x = b_k \\ & x_k \geq 0 \end{aligned} \tag{15}$$

The operation to find these reduced costs is often called *Pricing*. If $\sigma_k^* < 0$ we can introduce the a new column $\lambda_{k,j}$ to the master, with a cost coefficient of $c_k^T x_k^*$.

A basic Dantzig-Wolfe decomposition algorithm can now be formulated:

Dantzig-Wolfe decomposition algorithm.

{initialization}
Choose initial subsets of variables.
while true **do**

[1]The reduced cost of a variable x_j is

$$\sigma_j = c_j - \pi^T A_j \tag{13}$$

where A_j is the column of A corresponding to variable x_j, and π are the duals.

{Master problem}
Solve the restricted master problem.
π_1 := duals of coupling constraints
$\pi_2^{(k)}$:= duals of the k^{th} convexity constraint
{Sub problems}
for k=1,...,K **do**
Plug π_1 and $\pi_2^{(k)}$ into sub-problem k
Solve sub-problem k
if $\sigma_k^* < 0$ **then**
Add proposal x_k^* to the restricted master
end if
end for
if No proposals generated **then**
Stop: optimal
end if
end while

5. Initialization

We did not pay attention to the initialization of the decomposition.

The first thing we can do is solve each sub-problem:

$$\begin{aligned} \min c_k^T x_k & \\ A_k x_k &= b_k \\ x_k &\geq 0 \end{aligned} \tag{16}$$

If any of the subproblems is infeasible, the original problem is infeasible. Otherwise, we can use the optimal values x_k^* (or the unbounded rays) to generate an initial set of proposals.

6. Phase I/II algorithm

The initial proposals may violate the coupling constraints. We can formulate a Phase I problem by introducing artificial variables and minimizing those. The use of artificial variables is explained in any textbook on Linear Programming (e.g. [1, 14]). It is noted that the reduced cost for a Phase I problem are slightly different from the Phase II problem.

As an example consider that the coupling constraints are

$$\sum_j x_j \leq b \tag{17}$$

We can add an artificial variable $x_a \geq 0$ as follows:

$$\sum_j x_j - x_a \leq b \tag{18}$$

The phase I objective will be:

$$\min x_a \tag{19}$$

The reduced cost of a variable x_j is now as in equation (14) but with $c_k^T = 0$.

It is noted that it is important to remove artificials once a phase II starts. We do this in the example code by fixing the artificial variables to zero.

7. Example: multi-commodity network flow

The multi-commodity network flow (MCNF) problem can be stated as:

$$\begin{aligned}
&\min \sum_{k\in K} \sum_{(i,j)\in A} c^k_{i,j} x^k_{i,j} \\
&\sum_{(i,j)\in A} x^k_{i,j} - \sum_{(j,i)\in A} x^k_{j,i} = b^k_j \\
&\sum_{k\in K} x^k_{i,j} \le u_{i,j} \\
&x^k_{i,j} \ge 0
\end{aligned} \tag{20}$$

This is sometimes called the *node-arc* formulation.

Dantzig-Wolfe decomposition is a well-known solution strategy for this type of problems. For each commodity a subproblem is created.

We consider here a multi-commodity transportation problem:

$$\begin{aligned}
&\min \sum_{k\in K} \sum_{(i,j)} c^k_{i,j} x^k_{i,j} \\
&\sum_{j} x^k_{i,j} = \mathit{supply}^k_i \\
&\sum_{i} x^k_{i,j} = \mathit{demand}^k_j \\
&\sum_{k\in K} x^k_{i,j} \le u_{i,j} \\
&x^k_{i,j} \ge 0
\end{aligned} \tag{21}$$

with data from [4]. A similar Dantzig-Wolfe decomposition algorithm written in AMPL can be found in [3].

Model dw.gms. [2]

```
$ontext

  Dantzig-Wolfe Decomposition with GAMS

  Reference:
      http://www.gams.com/~erwin/dw/dw.pdf

  Erwin Kalvelagen, April 2003

$offtext


sets
  i  'origins'       /GARY, CLEV, PITT /
  j  'destinations'  /FRA, DET, LAN, WIN, STL, FRE, LAF /
  p  'products'      /bands, coils, plate/
;

table supply(p,i)
           GARY   CLEV   PITT
   bands    400    700    800
   coils    800   1600   1800
   plate    200    300    300
;
```

[2] http://www.amsterdamoptimization.com/models/dw/dw.gms

```
table demand(p,j)
           FRA   DET   LAN   WIN   STL   FRE   LAF
   bands   300   300   100    75   650   225   250
   coils   500   750   400   250   950   850   500
   plate   100   100     0    50   200   100   250
;

parameter limit(i,j);
limit(i,j) = 625;

table cost(p,i,j) 'unit cost'
               FRA  DET  LAN  WIN  STL  FRE  LAF
  BANDS.GARY    30   10    8   10   11   71    6
  BANDS.CLEV    22    7   10    7   21   82   13
  BANDS.PITT    19   11   12   10   25   83   15

  COILS.GARY    39   14   11   14   16   82    8
  COILS.CLEV    27    9   12    9   26   95   17
  COILS.PITT    24   14   17   13   28   99   20

  PLATE.GARY    41   15   12   16   17   86    8
  PLATE.CLEV    29    9   13    9   28   99   18
  PLATE.PITT    26   14   17   13   31  104   20
;


*-----------------------------------------------------------------------
* direct LP formulation
*-----------------------------------------------------------------------


positive variable
   x(i,j,p)     'shipments'
;
variable
   z     'objective variable'
;


equations
   obj
   supplyc(i,p)
   demandc(j,p)
   limitc(i,j)
;

obj..  z =e= sum((i,j,p), cost(p,i,j)*x(i,j,p));

supplyc(i,p).. sum(j, x(i,j,p)) =e= supply(p,i);

demandc(j,p).. sum(i, x(i,j,p)) =e= demand(p,j);

limitc(i,j).. sum(p, x(i,j,p)) =l= limit(i,j);


model m/all/;
solve m minimizing z using lp;


*-----------------------------------------------------------------------
* subproblems
*-----------------------------------------------------------------------

positive variables xsub(i,j);
variables zsub;

parameters
   s(i)     'supply'
   d(j)     'demand'
   c(i,j)   'cost coefficients'
   pi1(i,j) 'dual of limit'
   pi2(p)   'dual of convexity constraint'
```

```
   pi2p
;

equations
   supply_sub(i)    'supply equation for single product'
   demand_sub(j)    'demand equation for single product'
   rc1_sub          'phase 1 objective'
   rc2_sub          'phase 2 objective'
;

supply_sub(i).. sum(j, xsub(i,j)) =e= s(i);
demand_sub(j).. sum(i, xsub(i,j)) =e= d(j);
rc1_sub..       zsub =e= sum((i,j), -pi1(i,j)*xsub(i,j)) - pi2p;
rc2_sub..       zsub =e= sum((i,j), (c(i,j)-pi1(i,j))*xsub(i,j)) - pi2p;


model sub1 'phase 1 subproblem' /supply_sub, demand_sub, rc1_sub/;
model sub2 'phase 2 subproblem' /supply_sub, demand_sub, rc2_sub/;



*-----------------------------------------------------------------------
* master problem
*-----------------------------------------------------------------------

set k 'proposal count' /proposal1*proposal1000/;
set pk(p,k);
pk(p,k) = no;

parameter proposal(i,j,p,k);
parameter proposalcost(p,k);
proposal(i,j,p,k) = 0;
proposalcost(p,k) = 0;


positive variables
   lambda(p,k)
   excess   'artificial variable'
;
variable zmaster;

equations
    obj1_master    'phase 1 objective'
    obj2_master    'phase 2 objective'
    limit_master(i,j)
    convex_master
;

obj1_master..  zmaster =e= excess;
obj2_master..  zmaster =e= sum(pk, proposalcost(pk)*lambda(pk));

limit_master(i,j)..
   sum(pk, proposal(i,j,pk)*lambda(pk)) =l= limit(i,j) + excess;

convex_master(p).. sum(pk(p,k), lambda(p,k)) =e= 1;

model master1 'phase 1 master' /obj1_master, limit_master, convex_master/;
model master2 'phase 2 master' /obj2_master, limit_master, convex_master/;



*-----------------------------------------------------------------------
* options to reduce solver output
*-----------------------------------------------------------------------

option limrow=0;
option limcol=0;

master1.solprint = 2;
master2.solprint = 2;

sub1.solprint = 2;
```

```
sub2.solprint = 2;

*-----------------------------------------------------------------------
* options to speed up solver execution
*-----------------------------------------------------------------------

master1.solvelink = 2;
master2.solvelink = 2;
sub1.solvelink = 2;
sub2.solvelink = 2;

*-----------------------------------------------------------------------
* DANTZIG-WOLFE INITIALIZATION PHASE
*    test subproblems for feasibility
*    create initial set of proposals
*-----------------------------------------------------------------------

display "-----------------------------------------------------------------",
        "INITIALIZATION PHASE",
        "-----------------------------------------------------------------";

set kk(k) 'current proposal';
kk('proposal1') = yes;

loop(p,

*
* solve subproblem, check feasibility
*
    c(i,j) = cost(p,i,j);
    s(i) = supply(p,i);
    d(j) = demand(p,j);
    pi1(i,j) = 0;
    pi2p = 0;
    solve sub2 using lp minimizing zsub;
    abort$(sub2.modelstat = 4) "SUBPROBLEM IS INFEASIBLE: ORIGINAL MODEL IS INFEASIBLE";
    abort$(sub2.modelstat <> 1) "SUBPROBLEM NOT SOLVED TO OPTIMALITY";

*
* proposal generation
*
    proposal(i,j,p,kk) = xsub.l(i,j);
    proposalcost(p,kk) = sum((i,j), c(i,j)*xsub.l(i,j));
    pk(p,kk) = yes;
    kk(k) = kk(k-1);

);

option proposal:2:2:2;
display proposal;

*-----------------------------------------------------------------------
* DANTZIG-WOLFE ALGORITHM
*   while (true) do
*      solve restricted master
*      solve subproblems
*   until no more proposals
*-----------------------------------------------------------------------


set iter 'maximum iterations' /iter1*iter15/;
scalar done /0/;
scalar count /0/;
scalar phase /1/;
scalar iteration;

loop(iter$(not done),

    iteration = ord(iter);
    display "-----------------------------------------------------------------",
            iteration,
            "-----------------------------------------------------------------";
```

```
*
* solve master problem to get duals
*
    if (phase=1,
        solve master1 minimizing zmaster using lp;
        abort$(master1.modelstat <> 1) "MASTERPROBLEM NOT SOLVED TO OPTIMALITY";
        if (excess.l < 0.0001,
           display "Switching to phase 2";
           phase = 2;
           excess.fx = 0;
        );

    );

    if (phase=2,
        solve master2 minimizing zmaster using lp;
        abort$(master2.modelstat <> 1) "MASTERPROBLEM NOT SOLVED TO OPTIMALITY";
    );

    pi1(i,j) = limit_master.m(i,j);
    pi2(p) = convex_master.m(p);

    count = 0;
    loop(p$(not done),

*
* solve each subproblem
*
        c(i,j) = cost(p,i,j);
        s(i) = supply(p,i);
        d(j) = demand(p,j);
        pi2p = pi2(p);

        if (phase=1,
           solve sub1 using lp minimizing zsub;
           abort$(sub1.modelstat = 4) "SUBPROBLEM IS INFEASIBLE: ORIGINAL MODEL IS INFEASIBLE";
           abort$(sub1.modelstat <> 1) "SUBPROBLEM NOT SOLVED TO OPTIMALITY";
        else
           solve sub2 using lp minimizing zsub;
           abort$(sub2.modelstat = 4) "SUBPROBLEM IS INFEASIBLE: ORIGINAL MODEL IS INFEASIBLE";
           abort$(sub2.modelstat <> 1) "SUBPROBLEM NOT SOLVED TO OPTIMALITY";
        );


*
* proposal
*
        if (zsub.l < -0.0001,
           count = count + 1;
           display "new proposal", count,xsub.l;
           proposal(i,j,p,kk) = xsub.l(i,j);
           proposalcost(p,kk) = sum((i,j), c(i,j)*xsub.l(i,j));
           pk(p,kk) = yes;
           kk(k) = kk(k-1);
       );

    );

*
* no new proposals?
*
   abort$(count = 0 and phase = 1) "PROBLEM IS INFEASIBLE";
   done$(count = 0 and phase = 2) = 1;
);

abort$(not done) "Out of iterations";


*-----------------------------------------------------------------------
* recover solution
*-----------------------------------------------------------------------
```

```
parameter xsol(i,j,p);
xsol(i,j,p) = sum(pk(p,k), proposal(i,j,pk)*lambda.l(pk));
display xsol;

parameter totalcost;
totalcost = sum((i,j,p), cost(p,i,j)*xsol(i,j,p));
display totalcost;
```

The reported solution is:

```
----    317 PARAMETER xsol

              bands       coils       plate

GARY.STL    400.000      64.099     160.901
GARY.FRE                625.000
GARY.LAF                110.901      39.099
CLEV.FRA    264.099                  10.901
CLEV.DET     67.906     457.094     100.000
CLEV.LAN                400.000
CLEV.WIN     43.972     169.025      50.000
CLEV.STL    250.000     260.901      39.099
CLEV.FRE                162.003     100.000
CLEV.LAF     74.024     150.976
PITT.FRA     35.901     500.000      89.099
PITT.DET    232.094     292.906
PITT.LAN    100.000
PITT.WIN     31.028      80.975
PITT.STL                625.000
PITT.FRE    225.000      62.997
PITT.LAF    175.976     238.123     210.901


----    321 PARAMETER totalcost            =   199500.000
```

References

1. V. Chvatal, *Linear programming*, Freeman, 1983.
2. G. B. Dantzig and P. Wolfe, *Decomposition principle for linear programs*, Operations Research **8** (1960), 101–111.
3. R. Fourer and D. Gay, *Looping with ampl*, `http://www.netlib.org/ampl/looping/`, 1995.
4. R. Fourer, D. Gay, and B. Kernighan, *AMPL: A modelling language for mathematical programming*, Boyd & Fraser, 1993.
5. J. K. Ho and E. Loute, *An advanced implementation of the Dantzig-Wolfe decomposition algorithm for linear programming*, Mathematical Programming **20** (1981), 303–326.
6. ———, *Computational experience with advanced implementation of decomposition algorithms for linear programming*, Mathematical Programming **27** (1983), 283–290.
7. James K. Ho and R. P. Sundarraj, *DECOMP: an implementation of Dantzig-Wolfe decomposition for linear programming*, Lecture Notes in Economics and Mathematical Systems, vol. 338, Springer-Verlag, 1989.
8. ———, *Distributed nested decomposition of staircase linear programs*, ACM Transactions on Mathematical Software (TOMS) **23** (1997), no. 2, 148–173.
9. K. L. Jones, I. J. Lustig, J. M. Farvolden, and W. B. Powell, *Multicommodity network flows: The impact of formulation on decomposition*, Mathematical Programming **62** (1993), 95–117.
10. Erwin Kalvelagen, *Benders decomposition with GAMS*, `http://www.amsterdamoptimization.com/pdf/benders.pdf`, December 2002.
11. ———, *Column generation with GAMS*, `http://www.amsterdamoptimization.com/pdf/colgen.pdf`, December 2002.
12. ———, *Branch-and-bound methods for an MINLP model with semi-continuous variables*, `http://www.amsterdamoptimization/pdf/bb.pdf`, April 2003.
13. Deepankar Medhi, *Decomposition of structured large-scale optimization problems and parallel optimization*, Tech. Report 718, Computer Sciences Department, University Of Wisconsin, September 1987.

14. Katta G. Murty, *Linear programming*, Wiley, 1983.
15. George L. Nemhauser and Laurence A. Wolsey, *Integer and combinatorial optimization*, Interscience Series in Discrete Mathematics and Optimization, Wiley, 1988.
16. James Richard Tebboth, *A computational study of Dantzig-Wolfe decomposition*, Ph.D. thesis, University of Buckingham, December 2001.

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