1055-2017

Assigning agents to districts under multiple constraints using PROC CLP

Kevin K. Gillette and Stephen B. Sloan, Accenture

ABSTRACT

The Challenge: assigning outbound calling agents in a telemarketing campaign to geographic districts. The districts have a variable number of leads and each agent needs to be assigned entire districts with the total number of leads being as close as possible to a specified number for each of the agents (usually, but not always, an equal number). In addition, there are constraints concerning the distribution of assigned districts across time zones, in order to maximize productivity and availability.

Our Solution: uses the SAS/OR ® procedure PROC CLP to formulate the challenge as a constraint satisfaction problem (CSP), since the objective is not necessarily to minimize a cost function, but rather to find a feasible solution to the constraint set. The input consists of the number of agents, the number of districts, the number of leads in each district, the desired number of leads per agent, the amount by which the actual number of leads can differ from the desired number, and the time zone for each district.

INTRODUCTION

This paper concerns an application of the SAS ® procedure PROC CLP, an experimental component of SAS/OR ®. The problem involves the assigning of outbound calling agents in a telemarketing campaign to geographic districts throughout the country. The solution will work for any organization that needs to assign agents or salespeople to districts. The districts have varying numbers of leads, the agents need to be assigned entire districts, and the number of leads per agent needs to be kept as consistent as possible. In addition, agents need to be assigned districts in every time zone in order to maximize productivity and availability. Our solution also allows organizations to specify different numbers of leads for different agents and provide a specified level of time zones for each agent.

The problem looks like an assignment problem, but formulates a bit more naturally as a constraint satisfaction problem (CSP), since the objective is not necessarily to minimize a cost function *per se*, but rather to simply find a *feasible* solution to the constraint set.

SOLUTION

Our solution takes information about the following:

- The number of agents
- The number of districts
- The number of leads in each district
- The time zones containing the districts
- The desired number of leads per agent
- The amount by which the actual number of leads can differ from the desired number

Assume that there are N agents, numbered 1, 2, …, N, whom we'll refer to as agents A_1, A_2, …, A_N. Each agent has a target number of calls they are expected to make overall; let's label these as AL_1, AL_2, …, AL_N, where the AL_i are all integers (this isn't a requirement, but it makes solving the problem easier from an algorithmic perspective). The sum of all of the AL_i should equal the overall number of calls to be made in the marketing campaign; so if the total number of calls to be made is L_{total}, the total number of leads, then

$$\sum_{i=1}^{N} AL_i = L_{total}$$

There are M districts, which we'll label D_1, D_2, …, D_M, each of which has its own number of leads, which we'll label DL_1, DL_2, …, DL_M. Note that, as above, the sum of all of these should equal L_{total}, i.e.,

$$\sum_{i=1}^{M} DL_i = L_{total}$$

Since we don't know beforehand which agents will be assigned to which districts, we need to provide for the possibility that any agent could be assigned to any district; therefore we'll need to have binary decision variables that reflect all combinations. Let's let AD_{ij} be the binary (0/1) variable that indicates whether or not agent A_i is assigned to district D_j. This means we have N x M total decision variables.

Calling agents are typically shift workers; they report for work at various times. In this problem, since they are calling all U.S. time zones (including Alaska and Hawaii), it's important that they always have calls to make that correspond to normal business hours (e.g., 8am – 5pm) wherever the call is bound. To ensure that, for example, calling agents who arrive early for work don't have to wait for their first call to open for business, another set of constraints is designed to make sure that the assignment of districts to agents includes a "spread" of time zones; more concretely, we'd like each agent to be assigned to *at least one* district from each time zone. Assume we have Z time zones, labeled T_1, T_2, …, T_Z.
We have relatively few agents, and most likely a great many leads to call; however, it's very unlikely that it's possible to exactly match leads (districts) to agents to precisely match the target AL_i for each agent A_i. To ensure that we at least have a good chance of finding a feasible assignment, we can allow for some sort of threshold interval around the target number of leads for the agent, AL_i, typically expressed as a percent, e.g., 5%, 10%, etc. We'll call this threshold P, expressed as a decimal ($P = 0.05 \Rightarrow 5\%$, etc.). Each agent will be assigned a number of leads that differs from the target by less than the specified threshold.

At this point we are ready to formulate the problem. As mentioned at the beginning of the paper, we aren't really interested in *optimizing* anything; we're really concerned with finding a *feasible* solution to the set of constraints about to be described below. This amounts to simply running Phase I of the Simplex Method (with integer constraints). However, to keep the problem formulation in line with standard mathematical programming practice, we will formulate it as follows:

$$\text{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{M} AD_{ij} \qquad \text{where all } AD_{ij} \in \{0, 1\}$$

subject to

$\sum_{i=1}^{N} AD_{ij} = 1$ for all j = 1, …, M (each district must be assigned to some agent)

$\sum_{j=1}^{M} AD_{ij} \geq 1$ for all i = 1, …, N (each agent must have at least one district assigned to them)

$\sum_{j=1}^{M} DL_j * AD_{ij} \geq MAX(FLOOR((1 - P) * AL_i), 0)$ for all i = 1, …, N (lower threshold on leads for each agent)

$\sum_{j=1}^{M} DL_j * AD_{ij} \leq CEILING((1 + P) * AL_i)$ for all i = 1, …, N (upper threshold on leads for each agent)

$\sum_{j=1}^{M} \{AD_{ij} : DT_j = k\} \geq 1$ for all i = 1, …, N and k = 1, …, Z (each agent must have at least one district from each time zone assigned to them)

[NOTE: In the last set of constraints, the condition "$\{AD_{ij} : DT_j = k\}$" is meant to be read as "AD_{ij} such that $DT_j = k$", or the agent-district assignment is in time zone k]

The function calls are described below. They match the SAS functions of the same name.

- MAX(x,y) – Returns the maximum numeric value of x or y. In this example, it keeps the minimum number of leads from becoming negative.
- FLOOR(x) – Returns the floor function of x, which is the largest integer $\leq$ x. For instance, FLOOR(2.5) = 2, and FLOOR(-2.5) = -3. Note that if x is already an integer, then FLOOR(x) $\equiv$ x.
- CEILING(x) – Returns the ceiling function of x, which is the smallest integer $\geq$ x. For instance, CEILING(2.5) = 3, and CEILING(-2.5) = -2. Note that if x is already an integer, then CEILING(x) $\equiv$ x.

Below we have included some sample SAS ® code illustrating an example problem with 5 agents, 13 districts, and 2 time zones. Because of the way the numbers can vary, this code 'spoofs' DATA ® and PROC ® statements extensively with SAS macro logic. To see how this logic is actually working, it is best to run the code and read the SAS ® log via the MPRINT option (which is set in this example). The output produced by our use of PROC CLP is below, followed by the SAS ® code.

The result of the program shows the district assignments from the sample data, including both the desired number of leads and the actual leads assigned. The results place the actual leads need within 10% of the target number, as specified in the program statement "%LET p_threshold = 0.10;". For simplicity's sake we only included 2 time zones, but the program will handle as many time zones as desired.

Agent	Target # of leads	District	Time zone	# of leads	Total leads
1	400	9	2	255	
		11	1	176	**431**
2	350	4	2	200	
		6	2	110	
		8	1	54	**364**
3	250	7	1	150	
		13	2	101	**251**
4	500	3	1	145	
		10	2	314	**459**
5	580	1	1	55	
		2	2	90	
		5	1	196	
		12	1	234	**575**

SAS CODE

```
OPTIONS ERRORS=0 MPRINT SYMBOLGEN DQUOTE;

  /* This program is a proof=of-concept for an agent/district assignment routine that can be adapted for use across
a broad spectrum of problems. */


%LET p_threshold = 0.10;      /* We will permit a 10% leeway in the assignment of leads to agents relative to their
                                 target leads counts */
%LET N=;                     /* This will eventually hold the number of agents */
%LET M=;                     /* This will eventually hold the number of districts */
%LET Z=;                     /* This will eventually hold the number of time zones */

  /* The first input dataset is the Agents data - this links agents with the target number of leads to assign to them */
DATA agents;
   INPUT agent tgt_leads;
   IF _N_ = 1 THEN ag_cnt = 0;
   RETAIN ag_cnt;
   ag_cnt + 1;
   CALL SYMPUT('N',LEFT(ag_cnt));
   DROP ag_cnt;
   DATALINES;
1 400
2 350
3 250
4 500
5 580
;
RUN;


PROC SORT DATA=agents;

   BY agent;
RUN;


  /* Put the agent information into arrays for later convenience */
DATA ag_arrays;
   SET agents END=eof;
   ARRAY ag_nums{*} ag_num1 - ag_num&N.;
   ARRAY tgt_lds{*} tgt_ld1 - tgt_ld&N.;
   IF _N_ = 1 THEN n = 0;
   RETAIN n ag_num1-ag_num&N. tgt_ld1-tgt_ld&N.;
   n + 1;
   ag_nums{n} = agent;
   tgt_lds{n} = tgt_leads;
```

```
   IF eof THEN OUTPUT;
   KEEP ag_num1-ag_num&N. tgt_ld1-tgt_ld&N. n;
RUN;


  /* The next input dataset is the Districts data - this links districts with the number of leads they possess and the
     time zone in which they exist */
DATA districts;
   INPUT district dst_leads tz;
   IF _N_ = 1 THEN dst_cnt = 0;
   RETAIN dst_cnt;
   dst_cnt + 1;
   CALL SYMPUT('M',LEFT(dst_cnt));
   DROP dst_cnt;
   DATALINES;
1   55 1
2   90 2
3  145 1
4  200 2
5  196 1
6  110 2
7  150 1
8   54 1
9  255 2
10 314 2
11 176 1
12 234 1
13 101 2
;
RUN;

PROC SORT DATA=districts;
   BY district;
RUN;

  /* Put the district information into arrays for later convenience */
DATA ds_arrays;
   SET districts END=eof;
   ARRAY ds_nums{*} ds_num1 - ds_num&M.;
   ARRAY dst_lds{*} dst_ld1 - dst_ld&M.;
   ARRAY dst_tzs{*} dst_tz1 - dst_tz&M.;
   IF _N_ = 1 THEN m = 0;
   RETAIN m ds_num1-ds_num&M. dst_ld1-dst_ld&M. dst_tz1-dst_tz&M.;
   m + 1;
   ds_nums{m} = district;
   dst_lds{m} = dst_leads;
   dst_tzs{m} = tz;
   IF eof THEN OUTPUT;
   KEEP ds_num1-ds_num&M. dst_ld1-dst_ld&M. dst_tz1-dst_tz&M. m;
```

```
RUN;


  /* For the macro below that sets up the time zone constraints, a Time-Zone dataset is produced here */
PROC SORT DATA=districts OUT=timezones (KEEP=tz) NODUPKEY;
   BY tz;
RUN;

DATA timezones;
   SET timezones;
   IF _N_ = 1 THEN tz_cnt = 0;
   RETAIN tz_cnt;
   tz_cnt + 1;
   CALL SYMPUT('Z',LEFT(tz_cnt));
   DROP tz_cnt;
RUN;




  /* Calculate some useful constants and bounds, and put them into macro variables as well */
DATA _NULL_;
   LENGTH constr $ 10000;
   mn = &M. * &N.;
   CALL SYMPUT('obj_ub',mn);
     /* This set of loops sets up macro variables to place lower and upper bounds on the decision variables */
   DO agt = 1 TO &N.;
      DO dst = 1 TO &M.;
         CALL SYMPUT('var_dcl_'||TRIM(LEFT(agt))||'_'||TRIM(LEFT(dst)),'VAR
assign_'||PUT(agt,Z2.)||'_'||PUT(dst,Z2.)||' = [0,1]');
         CALL SYMPUT('chk_asg_'||TRIM(LEFT(agt))||'_'||TRIM(LEFT(dst)),
           'IF assign_'||PUT(agt,Z2.)||'_'||PUT(dst,Z2.)||' = 1 THEN DO; agent='||LEFT(agt)||';
district='||LEFT(dst)||';OUTPUT;END');
      END;
   END;
     /* This set of loops produces the constraints that each district must be assigned to one agent */
   cnt = 0;
   DO dst = 1 TO &M.;
      cnt = cnt + 1;
      constr = 'LINCON';
      DO agt = 1 TO &N.;
         constr = TRIM(constr) || ' assign_' || PUT(agt,Z2.) || '_' || PUT(dst,Z2.);
         IF agt < &N. THEN constr = TRIM(constr) || ' +';
      END;
      constr = TRIM(constr) || ' = 1';
      CALL SYMPUT('dist_assign_'||TRIM(LEFT(cnt)),TRIM(constr));
   END;
     /* This set of loops produces the constraints that each agent must be assigned at least one district */
   cnt = 0;
```

```
   DO agt = 1 TO &N.;
      cnt = cnt + 1;
      constr = 'LINCON';
      DO dst = 1 TO &M.;
         constr = TRIM(constr) || ' assign_' || PUT(agt,Z2.) || '_' || PUT(dst,Z2.);
         IF dst < &M. THEN constr = TRIM(constr) || ' +';
      END;
      constr = TRIM(constr) || ' >= 1';
      CALL SYMPUT('agent_assign_'||TRIM(LEFT(cnt)),TRIM(constr));
   END;
RUN;


  /* Merge the array data together into a single one-row dataset */
DATA arrays;
   MERGE ag_arrays ds_arrays;
   ARRAY ag_nums{*} ag_num1 - ag_num&N.;
   ARRAY tgt_lds{*} tgt_ld1 - tgt_ld&N.;
   ARRAY ds_nums{*} ds_num1 - ds_num&M.;
   ARRAY dst_lds{*} dst_ld1 - dst_ld&M.;
   ARRAY dst_tzs{*} dst_tz1 - dst_tz&M.;
RUN;

DATA _NULL_;
   SET arrays;
   ARRAY ag_nums{*} ag_num1 - ag_num&N.;
   ARRAY tgt_lds{*} tgt_ld1 - tgt_ld&N.;
   ARRAY ds_nums{*} ds_num1 - ds_num&M.;
   ARRAY dst_lds{*} dst_ld1 - dst_ld&M.;
   ARRAY dst_tzs{*} dst_tz1 - dst_tz&M.;
   LENGTH constr $ 10000;
   cnt = 0;
    /* First do the lower and upper bounds for each agent's total allocation of leads */
   DO i = 1 TO &N.;
      cnt = cnt + 1;
      agt = ag_nums{i};
      tgt_leads = tgt_lds{i};
      lb = MAX(FLOOR((1-&p_threshold.)*tgt_leads),0);
      ub = CEIL((1+&p_threshold.)*tgt_leads);
      constr = 'LINCON';
      DO j = 1 TO &M.;
         dst = ds_nums{j};
         dst_leads = dst_lds{j};
         constr = TRIM(constr) || ' ' || LEFT(dst_leads) || '*assign_' || PUT(agt,Z2.) || '_' || PUT(dst,Z2.);
         IF j < &M. THEN constr = TRIM(constr) || ' +';
      END;
      CALL SYMPUT('agent_lb_'||TRIM(LEFT(cnt)),TRIM(constr)||' >= '||PUT(lb,5.));
      CALL SYMPUT('agent_ub_'||TRIM(LEFT(cnt)),TRIM(constr)||' <= '||PUT(ub,5.));
   END;
```

```
    /* Now handle the notion that each agent needs at least one district in each time zone */
   cnt = 0;
   DO i = 1 TO &N.;
      cnt = cnt + 1;
      agt = ag_nums{i};
      DO tz = 1 TO &Z.;
         constr = 'LINCON';
         did_first_one = 0;
         DO j = 1 TO &M.;
            dst = ds_nums{j};
            dst_tz = dst_tzs{j};
            IF dst_tz = tz THEN DO;
               IF did_first_one THEN
                 constr = TRIM(constr) || ' + assign_' || PUT(agt,Z2.) || '_' || PUT(dst,Z2.);
               ELSE
                 constr = TRIM(constr) || ' assign_' || PUT(agt,Z2.) || '_' || PUT(dst,Z2.);
               did_first_one = 1;
            END;
         END;
         CALL SYMPUT('agent_tz_'||TRIM(LEFT(cnt))||'_'||TRIM(LEFT(tz)),TRIM(constr)||' >= 1');
      END;
   END;
RUN;




%MACRO run_clp;
PROC CLP OUT=out;

%DO agt = 1 %TO &N.;
   %DO dst = 1 %TO &M.;
      &&var_dcl_&agt._&dst.;
   %END;
%END;

%DO dst = 1 %TO &M.;
   &&dist_assign_&dst.;
%END;

%DO agt = 1 %TO &N.;
   &&agent_assign_&agt.;
%END;

%DO i = 1 %TO &N.;
   &&agent_lb_&i.;
   &&agent_ub_&i.;
%END;
```

```
%DO i = 1 %TO &N.;
   %DO tz = 1 %TO &Z.;
      &&agent_tz_&i._&tz.;
   %END;
%END;

RUN;



  /* Now parse out the actual assignments of agents to districts */
DATA assignments;
   SET out;
%DO agt = 1 %TO &N.;
   %DO dst = 1 %TO &M.;
      &&chk_asg_&agt._&dst.;
   %END;
%END;
   KEEP agent district;
RUN;

%MEND run_clp;




%run_clp;



  /* Lastly, match up the assigned agents to their target lead counts, the actual leads from the district(s), and the
time zones as well */

PROC SORT DATA=assignments;
   BY agent;
RUN;

DATA assignments;
   MERGE assignments (IN=inasg) agents (IN=inagt);
   BY agent;
   IF inasg AND inagt;
   KEEP agent district tgt_leads;
RUN;

PROC SORT DATA=assignments;
   BY district;
RUN;
```

```
DATA assignments;
   MERGE assignments (IN=inasg) districts (IN=indst);
   BY district;
   IF inasg AND indst;
   KEEP agent district tgt_leads dst_leads tz;
RUN;

PROC SORT DATA=assignments;
   BY agent district;
RUN;
```

ACKNOWLEDGMENTS & DISCLAIMERS

Lindsey Puryear, SAS ® Institute, who provided invaluable help and assistance.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Email: stephen.b.sloan@accenture.com
kevin.k.gillette@accenturefederal.com