

Optimal Solution of Discrete Resource Allocation Problems with SAS/OR ®Software

by LTC Doug McAllaster, US Army Logistics Management College, Fort Lee, VA

ABSTRACT

This paper is a tutorial on how to use SAS/OR PROC LP to solve integer programming problems. SAS/OR (operations research) software includes PROC LP which solves linear programming (LP) problems. This same procedure also includes the capability to solve integer programming (IP) problems, a very important class of problems which models discrete (binary: yes or no) decisions. This paper describes some classic integer programming models, all of which PROC LP can solve. Although the paper is primarily a verbal description of classic LP & IP models (and includes their algebraic formulations in the appendix), my presentation of this paper will focus on graphical representations (which are much more readily intelligible) like those in the appendix.

OVERVIEW

We begin with easier problems and work our way up to the more difficult ones. First, we describe three foundational LP models which are classic network flow problems: (1) assignment model, (2) transportation model, & (3) transshipment model. These problems have a special structure and are easier to solve using PROC NETFLOW rather than with the more general PROC LP. Then we examine three important and special applications of the transshipment model: (1) shortest path, (2) maximal flow, (3) maximal matching.

Second (having warmed up), we describe some pure integer programming models which are classic location models. These models are concerned with optimal placement of facilities to serve customers. We describe three classic models: (1) complete cover, (2) maximum cover, and (3) median placement.

CLASSIC NETFLOW MODELS

The assignment model gives an optimal assignment of people to jobs. Here we have N people and N jobs. There is some cost associated with a person performing a job. The model will assign a person to each job at minimal cost. We represent this model using a network where each node on the left represents one person and each node on the right represents one job. See the assignment model panel.

The transportation model has a very similar structure and network. This model is concerned with shipping goods from supply nodes (plants or warehouses) to demand nodes (customers) in order to minimize the transportation cost of meeting all the demand. Left hand nodes are plants, right hand nodes are customers. The numbers represent the supply of "widgets" at each plant and the demand for "widgets" at each customer. See the transport model panel.

Finally, the transshipment model is the most general network flow model. It permits a more general structure for the network. These networks can include intermediate nodes. Furthermore, the intermediate nodes may be supply, demand, or transshipment nodes. The latter is a node which simply sends out whatever comes in, i.e., it's neither a supply nor a demand, it's just a node

at which branching takes place, a "middleman," if you will. See the transship model panel.

The obvious application of the transshipment model is to meet customer demands from supplies at minimum cost. Thus, this is often called the minimum cost network flow problem (MCNFP). Less obvious, however, is that the MCNFP has very wide applicability. In fact, Glover, Klingman, & Phillips (see references) wrote an entire book describing the practical application of these models. For example, optimization analysts often use it to solve multiperiod production and inventory planning problems.

INPUT TABLES

PROC NETFLOW requires two data tables to solve this model, `nodedata` and `arcdata`. The `nodedata` table requires two columns: `node` and `supdem`. `node` is character, `supdem` is numeric, where positive integers are the supply at a node and negative integers are demands. The `arcdata` table requires three columns: `from`, `to`, and `cost`. Columns `from` and `to` are character and are the beginning and ending nodes of the arc, respectively. (Naturally, each `from` and `to` value in `arcdata` must match a node value in `nodedata`.) The `cost` column gives the cost to ship a unit along the arc.

From these two tables, PROC NETFLOW internally generates a single equation (or constraint) for each node. Each node's equation requires: (the sum of flows out of it) minus (the sum of flows into it) equal to (its `supdem`). The appendix has the algebraic formulation of the transshipment model. Both the assignment and transportation models are special cases of the transshipment model.

The output table is a listing of the flow on each arc. This flow minimizes total cost while meeting all of the nodal (supply and demand) constraints.

SEMINAL APPLICATIONS

We now consider two seminal applications of the MCNFP. The shortest path problem solves the problem of finding the shortest path from some source node to some demand node, through a set of intermediate transshipment nodes. We model this problem using the MCNFP with a supply of unity at the source node and a demand of unity at the sink node. Note that the solution to this problem is a single continuous path through the network from source to sink.

The maximal flow problem is similar, except that now our problem is to find out how many "widgets" we can get through our network from the source to the sink. This requires a simple (but not obvious) modification to the network. We add a new "return" arc from the sink back to the source and simply tell the model to maximize the flow along it. Note that all of the nodes are transshipment nodes, that is whatever flows in also flows out. One can think of the model as increasing the flow along the return arc until we achieve the maximum capacity of the network.

We conclude with a special and important application of the maximal flow problem to solve the maximal (bipartite) matching problem. In this situation, we have two disjoint sets, e.g., men and women. We indicate compatibility of man to woman with the existence of an arc between them. We introduce the return arc and simply maximize its flow to give us the matching which maximizes the number of couplings.

NETWORK ADVANTAGES

Network flow models have some great advantages over the more general linear programming models. These models are easy to comprehend, largely because they have a readily intelligible, graphical representation. Furthermore, these models are quite easy to code using SAS. The programmer simply provides a table of nodes and a table of arcs.

Next, these models enjoy a special internal structure which makes the solution times very fast (computationally easy). This structure also insures that the optimal flow values are integer. We get integer answers naturally; that is, without specifically constraining the optimizer to use much more complicated and potentially intractable integer programming solution (branch and bound) methods. That is, we get integer answers for free.

Finally, these models are very widely applicable to many business situations. Analysts often model a physical situation in which the main constraint is simply that "widgets" do not disappear. This "conservation of flow" is a fundamental fact in much of the physical universe.

CLASSIC LOCATION MODELS

We now describe some pure integer programming models which are classic location models. Location models enable us to determine the optimal placement of facilities to service some customers. Again, these models are quite easy to comprehend due to their physical and visual representation on the Euclidian plane. We consider three classics: complete cover, maximum cover, and median placement.

COMPLETE COVER

Consider the problem of locating fire stations to service cities. Here we obviously require a complete cover of all cities. We have a list of candidate sites for fire stations, a list of cities we must cover, distances between sites and cities, and some critical distance within which each city must have a fire station. We need to minimize the number of fire stations while providing complete coverage of the cities. This problem is a discrete one. That is, our solution must tell us whether or not to place a fire station at a candidate site. Also, note that we need to know both how many and where to put the fire stations.

We model this situation with a bipartite network. One side has the candidate fire station locations and the other has the cities we must cover. We derive the existence of an arc between a site and city from the distance table and the critical distance. See the complete cover panel in the appendix.

In this model we have several binary unknowns to consider simultaneously. That is, we first must consider the sites as being either "on" or "off." Then, we consider that turning a site node

"on" enables all the arcs which emanate from it, thus covering the respective cities.

An example of a covering problem on a map of the US is in the appendix. This example has 24 customer cities which must be covered by up to 8 candidate sites. Sites can cover cities which are within 999 miles. The solution shows that we can cover all 24 cities using only three sites. Also note that five cities are covered twice.

There are a few important differences between the former network flow models and this covering model. Here the fire stations do not exist yet! Furthermore, we are not flowing known quantities through the network. For example, one fire station can cover multiple cities. Likewise, a city may be covered by multiple fire stations. We cannot use network flow methods since we do not have fixed supplies and demands. Thus, we cannot use PROC NETFLOW.

The algebraic version of the model is in the appendix. It is fairly straightforward. Our binary analysis decision variable, X_i , is whether or not we place a facility at a candidate site. Our objective is to minimize the total number of facilities and our constraint is to cover each city at least once.

MAXIMUM COVER

Now we consider the less critical situation of locating libraries to provide service to cities. Here we still have candidate sites and cities to service. However, we do not have to completely cover each and every city (nobody burns without books). Rather, we consider the number of citizens in a city as the measure of importance for covering it. Our payoff is pleasing the most patrons. Another difference here is that we have some fixed number of libraries we will place. We still have the distance matrix between sites and cities as well as the some cutoff distance within which we consider a city covered. Thus, the only addition to our basic sites to cities graph is that we now have payoff values associated with the city nodes. See the maximum cover panel in the appendix.

The algebraic version of the model is also in the appendix. This model is more complicated in that we have an additional set of binary decision variables, Z_i , indicating whether a city is covered by some site. The objective is to maximize the grand total sum of covered patrons. The simple constraint is to place no more than the given (P) number of libraries. The other linking constraint insures that we only get credit for covering a city if it is within the cutoff distance of some site.

MEDIAN PLACEMENT

Our third classic location problem is concerned with minimizing the distance traveled from service sites to customer cities. Thus, the major difference is our method for handling distance. In the previous models, there was a critical or cutoff distance within which a city was covered, outside of which it was naked. Now, we consider distance in a linear (not binary) fashion, i.e., any service site can cover any city, but our objective is now to place our given (P) sites in the "weighted middle" of the cities. Thus, we are back to meeting the demand at every customer city, it's just that some folks have to make a long trip for service.

Of course, we know this is true in practice; for example, that medical facilities are located near population centers, small towns don't have retail super-centers. There are a plethora of applications of this model: for locating health clinics, post offices, grocery stores, et cetera.

Our basic network requires only slight modifications. We still have candidate sites on the left and cities (customers) on the right. Furthermore, we retain the demand weight (population) for each city as in the maximum cover problem. The major difference is that our network has more arcs since every site can service every city. (That is, our we cannot reduce our original dense distance matrix to a sparse binary one, as in the covering problems.) See the median placement panel in the appendix.

The algebraic model for this problem is also in the appendix. The algebraic model is significantly different from covering problems in that we now have a set of Y_{ij} binary decision variables for the assignment of sites to cities. We require this more extensive set of variables since we now must keep track of who goes where for service in order to accurately account for the total travel distance of customers. The objective minimizes the population weighted distance from cities to sites. The first constraints simply insures we build P sites. The second constraints insures that each city (customer) has a (service) site. The third linking constraint ensures that if a site doesn't exist, then no customer can go there for service.

NATIONAL GUARD

The Army National Guard (ARNG) annually faces a discrete location problem. I have simplified both the scope and details of the problem for this presentation. See Murty and Djang for a complete description. Each spring ARNG places a half-dozen mobile combat trainers at armories for the coming summer training cycle. The is the median placement problem; we we have a given number of trainers (P), candidate sites (large ARNG centers capable of housing incoming troops and supporting the training), and customer cities (consider each state's centroid, weighted by its ARNG population). See the appendix for a SAS/GRAPH map of this problem and its solution.

PROC LP

Although we were able to display the above pure integer programming problems on a network, these are not network flow models since we cannot fix the supply & demands a priori. As mentioned above, this is unfortunate since PROC NETFLOW is amenable to very easy coding.

Rather, we have to solve these IP models with PROC LP. Now, PROC LP can solve any general linear and integer programming problem. However, as a result of its general problem solving capability, the analyst must specify the objective and constraint equations explicitly. This is not a trivial task. But, of course, DATA STEP or PROC SQL programming provides adequate capability to generate the input table.

INPUT TABLE

PROC LP can accept the input table in either dense or sparse format. The appendix has both formats for our complete cover model. The dense format has a row for each equation in the

model. Each variable (column) is an analysis decision variable. This format directly reflects the algebraic equations. However, since an LP may have hundreds of decision variables, the more common format is the sparse (or coefficient) format. Here we have an observation for each non-zero coefficient and identify the coefficient's location in the model equations with a row and col variable. The type variable identifies the type of equation.

CONCLUSION

SAS/OR software can solve integer programming problems. This paper describes some classic integer programming applications, their formulations, graphical representations, and the input data required for PROC LP to solve a problem.

REFERENCES

The Wiley-Interscience Series in Discrete Mathematics and Optimization includes two excellent books which are readily intelligible to the interested reader.

Glover F., Klingman D., and Phillips N., *Network Models in Optimization and their Applications in Practice*, 1992, John Wiley and Sons.

Daskin, Mark. *Network and Discrete Location: Models, Algorithms, and Applications*, 1995, John Wiley and Sons.

SAS Institute Inc. *SAS/OR User's Guide Version 6 First Edition*, Cary, NC: SAS Institute Inc. 1989. 479pp.

Murty, K. and Djang, P., 1999. The US Army National Guard's Mobile Training Simulators Location and Routing Problem. *Operations Research* 47(2) March- April, 175-182.

TRADEMARK

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

AUTHOR CONTACT

You may contact me at: McAllasterD@Lee.Army.Mil

TRANSSHIPMENT MODEL determine flows thru network to meet demands but not exceed supplies

analysis decision variables and input data

- X_{ij} – advar – # units to flow from node i to node j thru arc(i,j)
- B_i – input – net supply or 'balance' at node i
- C_{ij} – input – cost per unit flow thru arc (i,j)

objective minimize grand sum of all flows*costs

$$\sum_{ij} C_{ij}X_{ij}$$

subject to at every node \boxed{i} :

$$\begin{aligned} \text{SumFlowOut} - \text{SumFlowIn} &= \text{Balance} && \text{– for every node } i \\ \sum_j X_{ij} - \sum_k X_{ki} &= B_{\boxed{i}} && \text{– for every node } i \end{aligned}$$

COMPLETE COVER locate minimum # of sites to cover every customer node

analysis decision variables and input data

- X_j – advar – whether (binary) we place facility at candidate site j
- A_{ij} – input – whether (binary matrix) site j can cover node i

derive binary matrix A_{ij} from a table of site to city distances and a critical distance D_c which defines possible coverage

objective minimize number of facilities at candidate sites

$$\text{minimize } \sum_j X_j$$

subject to cover every customer node i at least once

$$\sum_j A_{ij}X_j \geq 1 \quad \text{– for every customer node } i$$

MAXIMUM COVER locate **P** facilities to maximize covered demand (without necessarily covering every demand node) vice covering all nodes with minimum # of sites

analysis decision variables and input data

X_j – advar	– whether (binary) we place facility at site j
Z_i – advar	– whether (binary) node i is covered by some site j
A_{ij} – input	– whether (binary matrix) site j covers node i
H_i – input	– demand at node i
P – input	– number of facilities to place

objective maximize covered demand

$$\text{maximize } \sum_i H_i Z_i$$

subject to

$$\sum_j A_{ij} X_j \geq Z_i \quad \text{– for every customer node } i$$
$$\sum_j X_j \leq P$$

MEDIAN PLACEMENT locate P facilities to minimize weighted distance between customers & facilities

analysis decision variables and input data

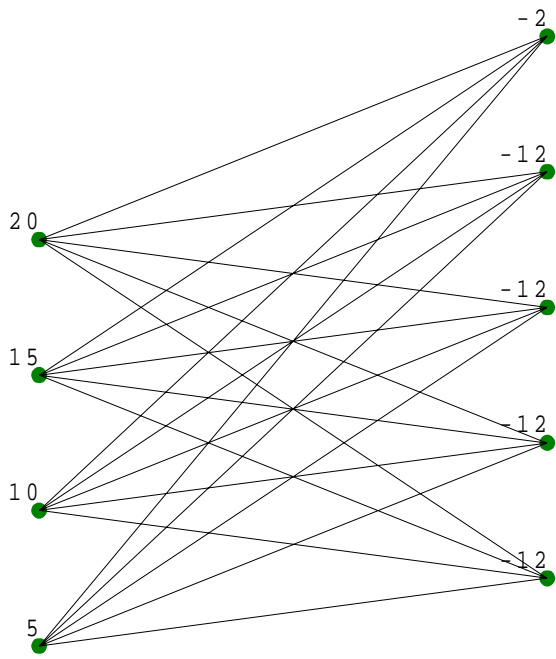
H_i – input	– demand load at customer node i
D_{ij} – input	– distance from customer node i to candidate site j
P – input	– number of facilities to place
X_j – advar	– whether (binary) we locate a facility at candidate site j
Y_{ij} – advar	– whether (binary) facility at site j serves entire demand at node i

objective minimize demand weighted distance from customers to facilities

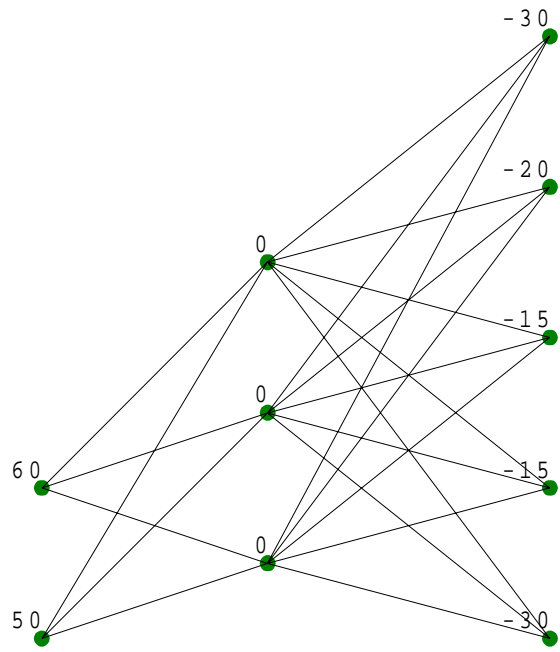
$$\text{minimize } \sum_{ij} H_i D_{ij} Y_{ij}$$

subject to

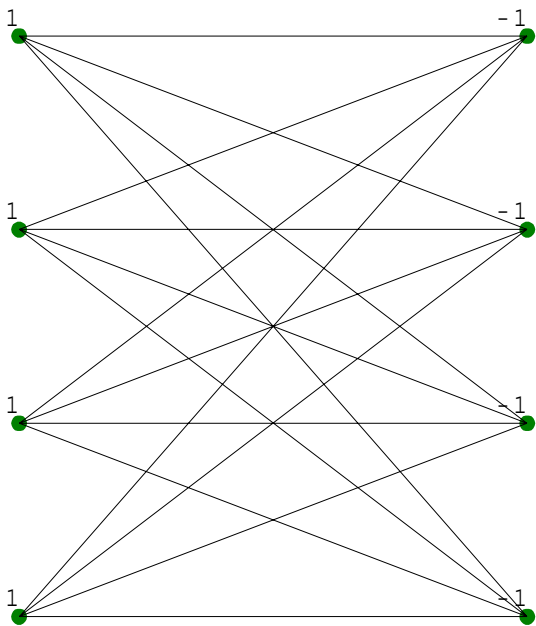
$$\sum_j X_j = P$$
$$\sum_j Y_{ij} = 1 \quad \text{– for every customer node } i$$
$$Y_{ij} \leq X_j \quad \text{– for every } ij \text{ pair}$$



Transport



Transship



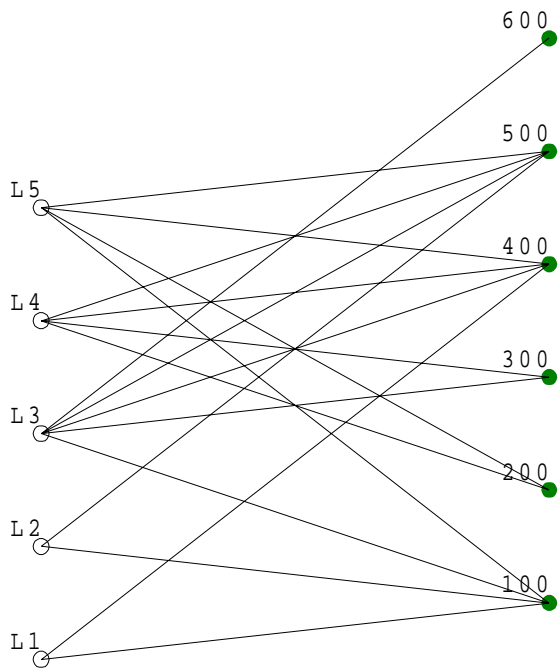
Assign

Network Flow Models

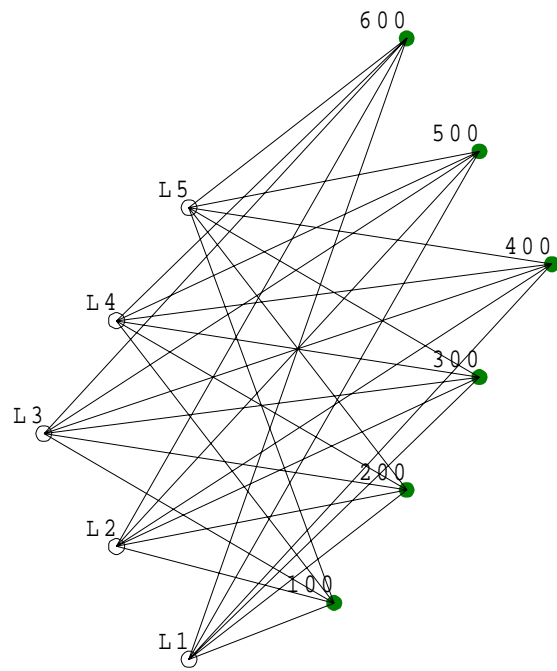
Assignment Model

Transportation Model

Transshipment Model



Max_Cover



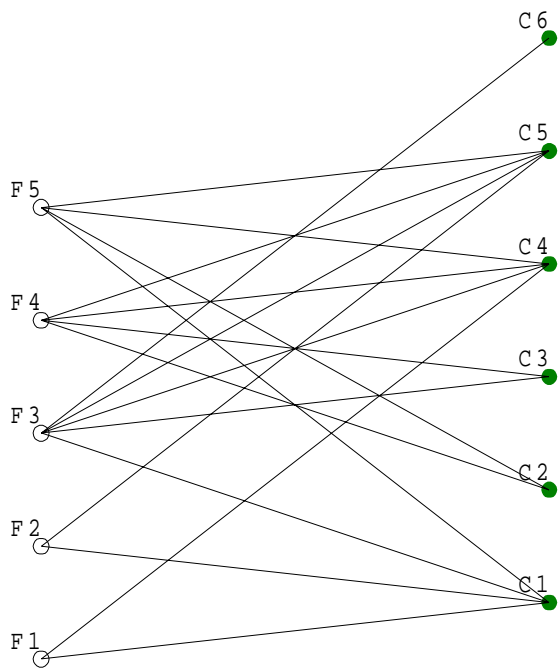
Med_Place

Covering Models

Complete Cover

Maximum Cover

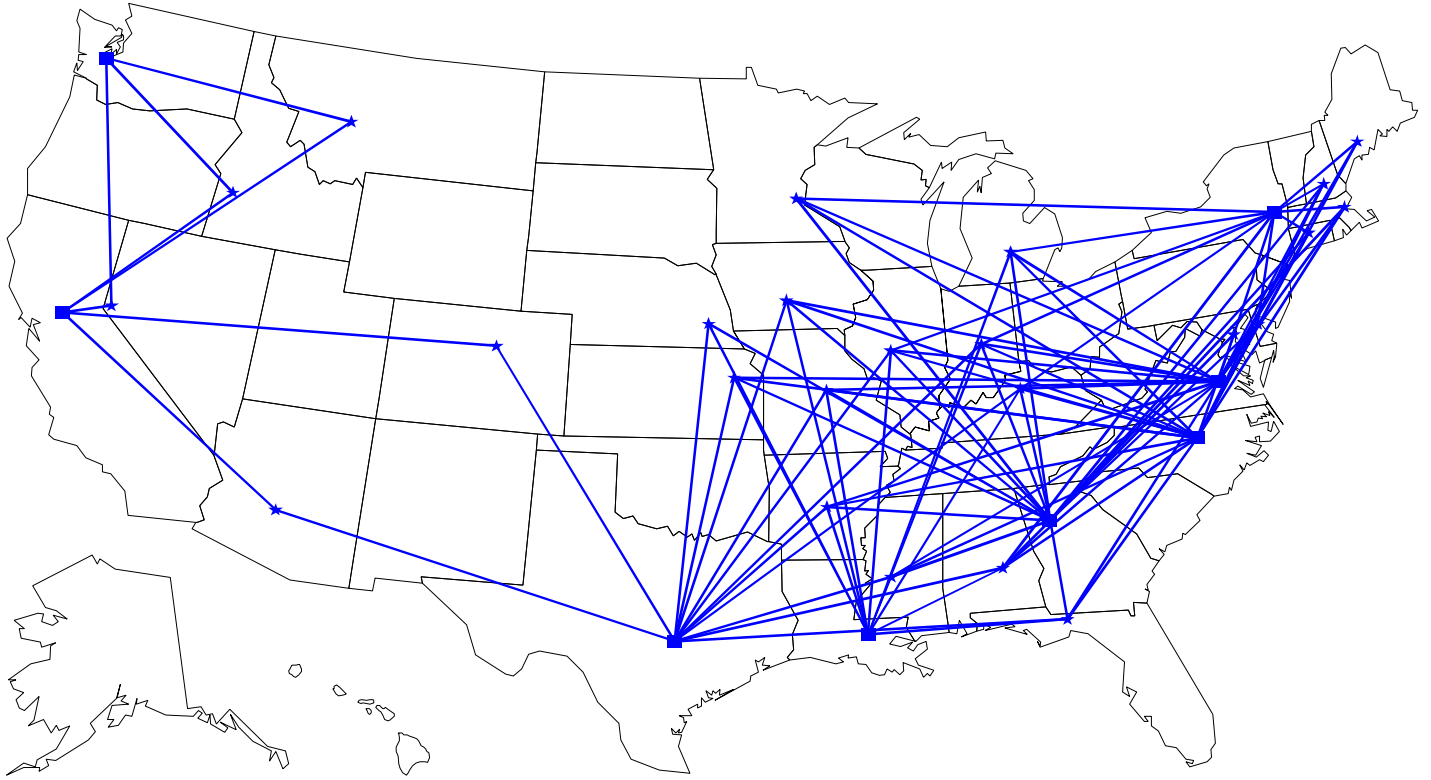
Median Placement



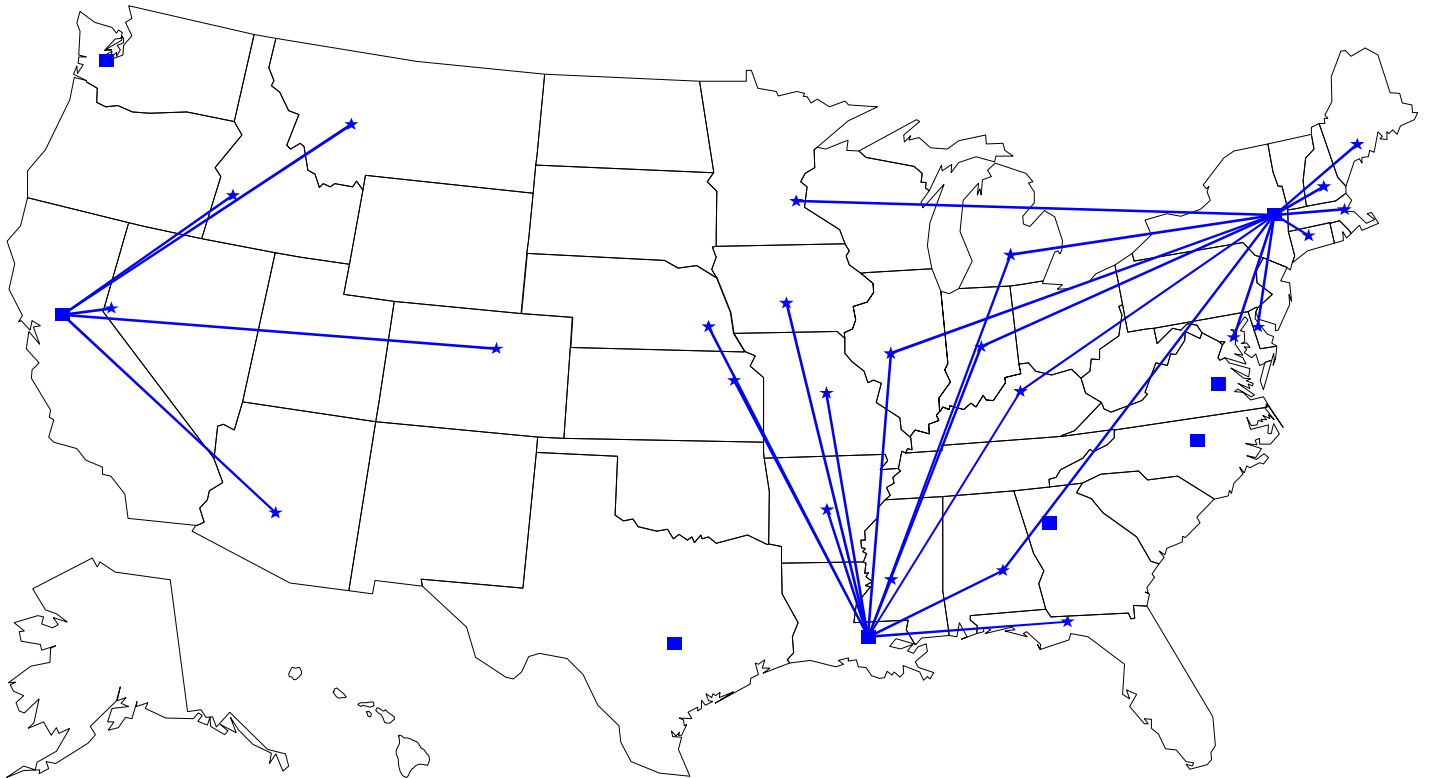
Com_Cover

--

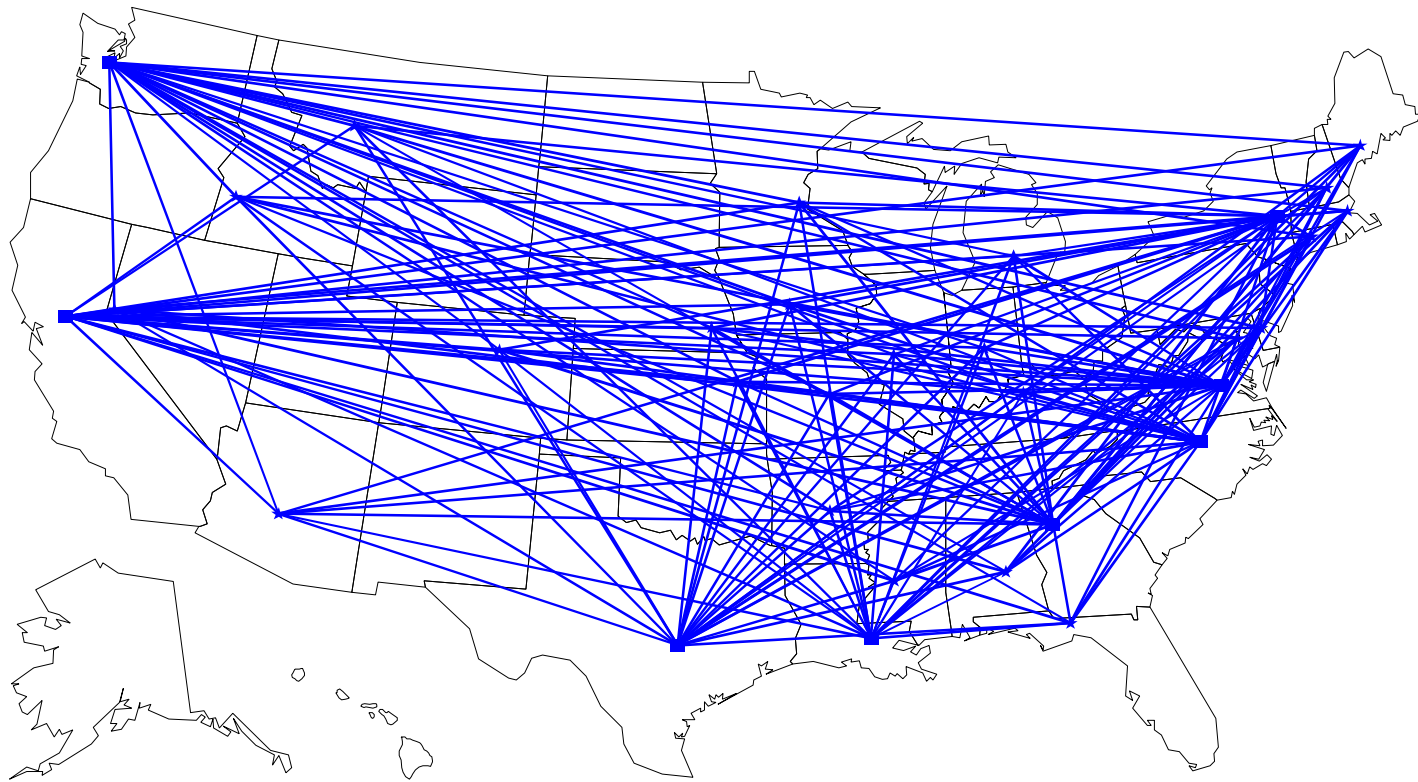
complete cover problem (crit dist = 999 mi)



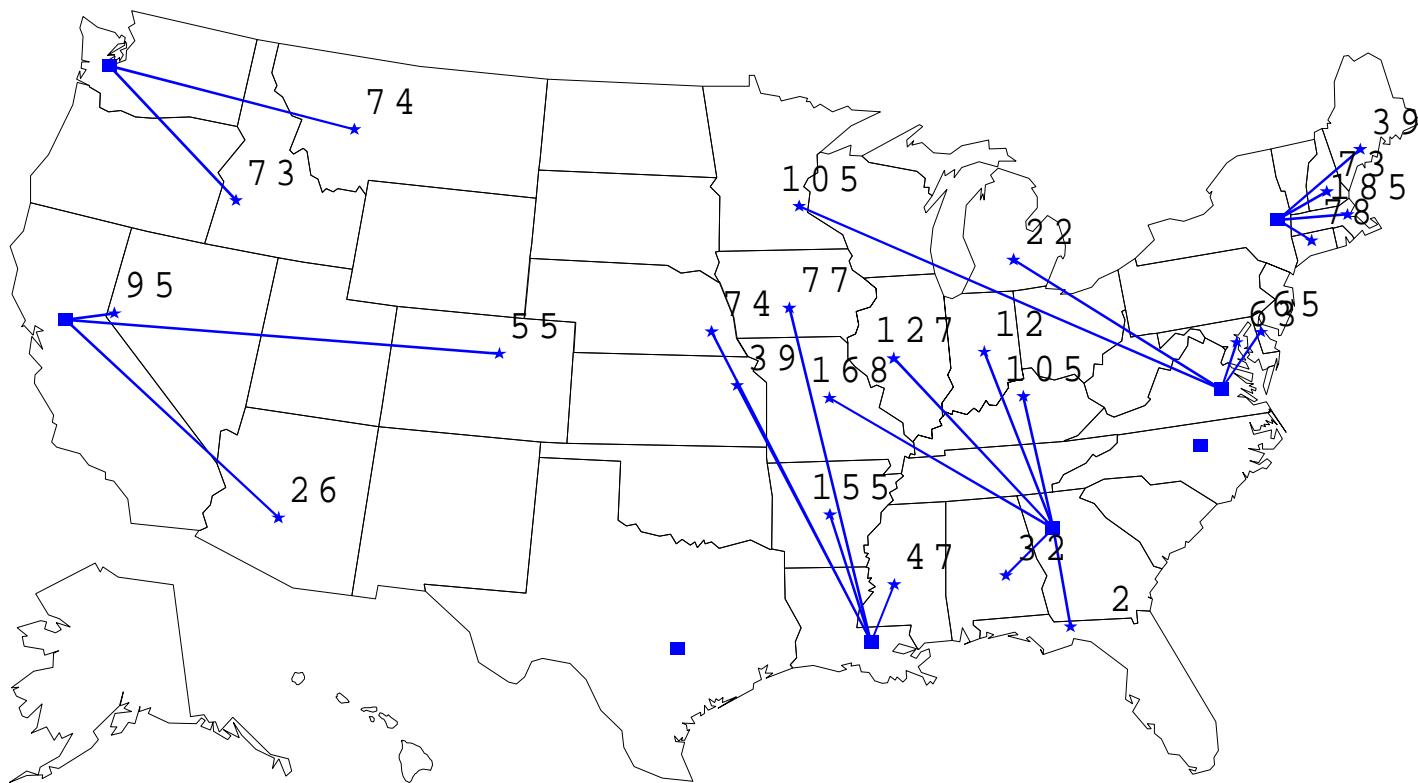
complete cover solution – five customers covered twice



ARNG median placement problem



ARNG median placement solution



PROC LP, INPUT TABLE, DENSE FORMAT

row	CA	GA	LA	NC	NY	TX	VA	WA	type	rhs
OBJ	1	1	1	1	1	1	1	1	MIN	.
BIN	1	1	1	1	1	1	1	1	BIN	.
AL	0	1	1	1	1	1	1	0	GE	1
AR	0	1	1	1	0	1	1	0	GE	1
AZ	1	0	0	0	0	1	0	0	GE	1
CO	1	0	0	0	0	1	0	0	GE	1
CT	0	1	0	1	1	0	1	0	GE	1
DE	0	1	0	1	1	0	1	0	GE	1
FL	0	1	1	1	0	1	1	0	GE	1
IA	0	1	1	1	0	1	1	0	GE	1
ID	1	0	0	0	0	0	0	1	GE	1
IL	0	1	1	1	1	1	1	0	GE	1
IN	0	1	1	1	1	1	1	0	GE	1
KS	0	1	1	1	0	1	1	0	GE	1
KY	0	1	1	1	1	1	1	0	GE	1
MA	0	1	0	1	1	0	1	0	GE	1
MD	0	1	0	1	1	0	1	0	GE	1
ME	0	0	0	1	1	0	1	0	GE	1
MI	0	1	1	1	1	0	1	0	GE	1
MN	0	1	0	1	1	0	1	0	GE	1
MO	0	1	1	1	0	1	1	0	GE	1
MS	0	1	1	1	0	1	1	0	GE	1
MT	1	0	0	0	0	0	0	1	GE	1
NE	0	1	1	0	0	1	0	0	GE	1
NH	0	1	0	1	1	0	1	0	GE	1
NV	1	0	0	0	0	0	0	1	GE	1

PROC LP, INPUT TABLE, SPARSE FORMAT

ROW	COL	TYPE	COEFF	ROW	COL	TYPE	COEFF	ROW	COL	TYPE	COEFF	ROW	COL	TYPE	COEFF
AL	GA	GE	1	DE	VA	GE	1	KS	VA	GE	1	MO	LA	GE	1
AL	LA	GE	1	DE	RHS_	GE	1	KS	RHS_	GE	1	MO	NC	GE	1
AL	NC	GE	1	FL	GA	GE	1	KY	GA	GE	1	MO	TX	GE	1
AL	NY	GE	1	FL	LA	GE	1	KY	LA	GE	1	MO	VA	GE	1
AL	TX	GE	1	FL	NC	GE	1	KY	NC	GE	1	MO	RHS_	GE	1
AL	VA	GE	1	FL	TX	GE	1	KY	NY	GE	1	MS	GA	GE	1
AL	RHS_	GE	1	FL	VA	GE	1	KY	TX	GE	1	MS	LA	GE	1
AR	GA	GE	1	FL	RHS_	GE	1	KY	VA	GE	1	MS	NC	GE	1
AR	LA	GE	1	IA	GA	GE	1	KY	RHS_	GE	1	MS	TX	GE	1
AR	NC	GE	1	IA	LA	GE	1	MA	GA	GE	1	MS	VA	GE	1
AR	TX	GE	1	IA	NC	GE	1	MA	NC	GE	1	MS	RHS_	GE	1
AR	VA	GE	1	IA	TX	GE	1	MA	NY	GE	1	MT	CA	GE	1
AR	RHS_	GE	1	IA	VA	GE	1	MA	VA	GE	1	MT	WA	GE	1
AZ	CA	GE	1	IA	RHS_	GE	1	MA	RHS_	GE	1	MT	RHS_	GE	1
AZ	TX	GE	1	ID	CA	GE	1	MD	GA	GE	1	NE	GA	GE	1
AZ	RHS_	GE	1	ID	WA	GE	1	MD	NC	GE	1	NE	LA	GE	1
BIN	CA	BINA	1	ID	RHS_	GE	1	MD	NY	GE	1	NE	TX	GE	1
BIN	GA	BINA	1	IL	GA	GE	1	MD	VA	GE	1	NE	RHS_	GE	1
BIN	LA	BINA	1	IL	LA	GE	1	MD	RHS_	GE	1	NH	GA	GE	1
BIN	NC	BINA	1	IL	NC	GE	1	ME	NC	GE	1	NH	NC	GE	1
BIN	NY	BINA	1	IL	NY	GE	1	ME	NY	GE	1	NH	NY	GE	1
BIN	TX	BINA	1	IL	TX	GE	1	ME	VA	GE	1	NH	VA	GE	1
BIN	VA	BINA	1	IL	VA	GE	1	ME	RHS_	GE	1	NH	RHS_	GE	1
BIN	WA	BINA	1	IL	RHS_	GE	1	MI	GA	GE	1	NV	CA	GE	1
CO	CA	GE	1	IN	GA	GE	1	MI	LA	GE	1	NV	WA	GE	1
CO	TX	GE	1	IN	LA	GE	1	MI	NC	GE	1	NV	RHS_	GE	1
CO	RHS_	GE	1	IN	NC	GE	1	MI	NY	GE	1	OBJ	CA	MIN	1
CT	GA	GE	1	IN	NY	GE	1	MI	VA	GE	1	OBJ	GA	MIN	1
CT	NC	GE	1	IN	TX	GE	1	MI	RHS_	GE	1	OBJ	LA	MIN	1
CT	NY	GE	1	IN	VA	GE	1	MN	GA	GE	1	OBJ	NC	MIN	1
CT	VA	GE	1	IN	RHS_	GE	1	MN	NC	GE	1	OBJ	NY	MIN	1
CT	RHS_	GE	1	KS	GA	GE	1	MN	NY	GE	1	OBJ	TX	MIN	1
DE	GA	GE	1	KS	LA	GE	1	MN	VA	GE	1	OBJ	VA	MIN	1
DE	NC	GE	1	KS	NC	GE	1	MN	RHS_	GE	1	OBJ	WA	MIN	1
DE	NY	GE	1	KS	TX	GE	1	MO	GA	GE	1				