AN OPTIMIZATION MODEL FOR SELECTING TRAINING COURSE LOCATIONS, U. S. GEOLOGICAL SURVEY

By Timothy A. Cohn and William G. Baier

U.S. GEOLOGICAL SURVEY

Open-File Report 93-123

Reston, Virginia

1993

U. S. DEPARTMENT OF THE INTERIOR

BRUCE BABBITT, Secretary

U.S. GEOLOGICAL SURVEY

DALLAS L. PECK, Director

For additional information
write to:

Chief, Branch of Systems Analysis
U.S. Geological Survey
410 National Center
Reston, Virginia 22092

Copies of this report can be
purchased from:

U.S. Geological Survey
Books and Open-File Reports
Box 25425, Federal Center
Denver, Colorado 80225

CONTENTS

TABLES

AN OPTIMIZATION MODEL FOR SELECTING TRAINING COURSE LOCATIONS, U. S. GEOLOGICAL SURVEY

By Timothy A. Cohn and William G. Baier

ABSTRACT

This report describes a computer program, named KATALASE, originally designed to assist the Northeastern Region, Water Resources Division, U. S. Geological Survey, in minimizing training course costs. The program can be used by planners, at any level, to determine the least expensive location for meetings and training events. KATALASE employs a combination of enumeration and linear programming to minimize a cost function subject to constraints on number of attendees and site capacities. Up to twenty-five potential sites can be analyzed to determine which location, or set of locations, will support the desired training at the lowest cost. The program output provides total cost, site(s), and attendee routing schemes.

INTRODUCTION

In the Northeastern Region, Water Resources Division, U. S. Geological Survey, all new employees are required to take a series of training classes. Each of these classes can be taught at a number of different sites throughout the region. The desire to minimize overall costs raises two questions:

1) At which site(s) should the training classes be offered?
2) To which training site should each new employee be sent?

A computer program called KATALASE answers these questions. KATALASE can also be used to solve other similar linear network problems [Bradley and others, 1977].

This report provides instructions for using KATALASE, and describes its limitations and underlying assumptions. Discussions of input data file format, program execution, and interpretation of output files are provided. For those interested, a brief discussion of the program design is included.

THE PROBLEM

Assume we have a number of offices wishing to send new employees to training, and a number of sites that can provide training. The source sites are denoted $\{SS_1,...,SS_{NSNT}\}$, where NSNT is the total number of offices wishing to

send new employees to training. The sites that can provide training are denoted $\{TS_1,...,TS_{NPTS}\}$, where NPTS is the total number of sites that can provide training.

KATALASE finds optimal solutions for one training course at a time. The optimal solution may suggest teaching the course at multiple training sites. If several courses need to be scheduled, KATALASE must be run separately for each one.

The constraints and costs can be formulated as an optimization problem with linear constraints and linear objective function [see Bradley and others, 1977, p. 310]:

$$\text{minimize} \quad C = \Sigma_j N_j f_j + \Sigma_i \Sigma_j T_{ij} c_{ij} \qquad (1)$$

$$\text{subject to:} \quad \Sigma_j T_{ij} = d_i \qquad i=1, NSNT$$

$$\Sigma_i T_{ij} \leq N_j q_j \qquad j=1, NPTS$$

$$\Sigma_i T_{ij} \geq N_j ql_j \qquad j=1, NPTS$$

$$\Sigma_j N_j \leq NT \qquad i=1, NSNT$$

where:

- C = total cost,
- Σ_j = summation over $\{TS_j\}$,
- Σ_i = summation over $\{SS_i\}$,
- N_j = number of times training is held at TS_j,
- NT = maximum number of training classes permitted in the solution
- f_j = fixed costs of training at TS_j,
- T_{ij} = number of attendees from SS_i attending training at TS_j,
- c_{ij} = travel cost for one attendee from SS_i to attend training at TS_j,
- d_i = number of employees needing training from SS_i,
- q_j = maximum number of attendees at TS_j, and
- ql_j = minimum number of attendees at TS_j.

KATALASE ensures that demand and capacity constraints are fulfilled while cost is minimized. In short, KATALASE finds each attendee space in the class and ensures each class is not overbooked, while at the same time, minimizing total costs.

The KATALASE Algorithm

The KATALASE algorithm consists of four stages:

1) All possible combinations of training sites are listed;
2) For each combination of training sites, a linear program is used to find the least cost allocation of attendees to the training sites;
3) The least cost solution for each combination is recorded;
4) The combinations are ranked by cost.

This four-stage method guarantees integer solutions while avoiding more complicated integer techniques like "branch and bound" [Bradley and others, 1977, p. 387].

ASSUMPTIONS AND LIMITATIONS OF MODEL

Cost Functions

The first assumption is that fixed costs and travel costs are constant. This program does not recognize seasonal changes in airline, hotel, or other rates. If users wish to take seasonal rate changes into account, they may consider multiple entries for each site. For example, if fixed costs for Reston (U. S. Geological Survey National Center) are different in the summer than in the winter, then the user may wish to enter Reston(summer) and Reston(winter) as separate sites.

Discounting

The program does not permit non-linear cost functions. For example, a convention center might offer a discount if an organization commits to having more than one meeting per year at that facility. The program does not make use of this information.

Definition of Total Cost

The program is designed to minimize total costs for an event. For an event involving multiple organizations, the optimal solution may not minimize costs for each organization. If an organization hosting a multiple-organization event wishes to minimize its own cost, disregarding others' costs, only that organization's costs should appear in the input data. The fixed costs and travel costs for the other organizations should be set to zero.

Fixed costs and travel costs are, respectively, the sum of all costs associated with an event and the costs associated with transporting an attendee to the event. Determining the actual values for fixed cost and travel cost is the responsibility of the user. The optimal solution will be a function of these costs.

Problem Size

The number of combinations listed in stage 1 of the KATALASE algorithm (N_{com}) depends on NPTS and NT according to the following formula:

$$N_{com} = \binom{NPTS + NT}{NT} = \frac{(NPTS + NT)!}{NPTS!\ NT!} \qquad (2)$$

Because N_{com} can become very large for large values of NPTS and NT, KATALASE requires that:

1) NPTS not exceed 25; and
2) NT not exceed 10.

The user should choose a value for NT no larger than the maximum number of times that the course will be taught. Larger values of NT will substantially increase the amount of computer time required to solve the problem, and may result in solutions which suggest teaching the course more than the desired number of times. However, selecting values of NT which are too small may prevent KATALASE from considering lower-cost solutions which involve more classes. If NT is too small a feasible solution will not exist and the program output will include a string of asterisks for the minimum-cost solution.

KATALASE automatically adds one potential site with capacity, fixed cost, and travel costs equal to zero. This site allows the optimal solution to include fewer than NT sites. Combinations can contain the same training site multiple times. Permutations containing the same sites in different orders are considered only once.

PROGRAM OPERATION

Program Input Data File

The required input data format for the KATALASE program is shown in Table 1. Table columns are tab-separated and table rows are separated by a carrage return. Input files can be created using any spreadsheet or editor software which allows tab separation output (e.g. Excel[1], Lotus-123, Tactician). The input file must include the bold italicized words (For example, ***Minimum*** and ***ql***). These maintain data alignment, insuring correct interpretation of input data.

Blank spaces are not permitted between characters occupying the same cell. This is important when entering multiple word names. (New York entered as two words is not permitted. However, alternatives such as NewYork, New_York, and New.York are permitted).

An example of an input file containing information for 5 potential training sites (NPTS=5) and 7 sites needing training (NSNT=7) is shown in Table 2. Note that the potential training sites and the sites needing training can be different. 5 employees from Baltimore need training ($d_{Baltimore}$=5). For a course held in Boston, the minimum capacity is 15 (ql_{Boston}=15) and the maximum capacity is 32 (q_{Boston}=32). The fixed cost for an event held in Boston is $2487.00 ($f_{Boston}$=$2487.00). The travel cost for one employee traveling from Baltimore to train at Boston is $895.00 ($c_{Baltimore,Boston}$=$895.00).

---

1 The use of brand names in this report is for identification purposes only, and does not constitute endorsement by the U. S. Geological Survey.

Table 1.--Input data format for the KATALASE program

[NPTS, number of potential training sites; NSTN, number of sites needing training; TS, potential training site; q, maximum capacity; ql, minimum capacity; f, fixed cost; SS, source site; c, travel cost]

NPTS	NPTS					
NSNT	NSNT	TS_1	TS_2	TS_3	....	TS_{NPTS}
Capacity	*q*	q_1	q_2	q_3	....	q_{NPTS}
Minimum	*ql*	ql_1	ql_2	ql_3	....	ql_{NPTS}
Fixed_cost	*f*	f_1	f_2	f_3	....	f_{NPTS}
	Demand					
SS_1	d_1	c_{11}	c_{12}	c_{13}	....	c_{1NPTS}
SS_2	d_2	c_{21}	c_{12}	c_{13}	....	c_{2NPTS}
SS_3	d_3	c_{31}	c_{12}	c_{13}	....	c_{3NPTS}
....	....	....	....	....	....	....
SS_{NSNT}	d_{NSNT}	c_{NSNT1}	c_{NSNT2}	c_{NSNT3}	....	$c_{NSNT,NPTS}$

Table 2.--Sample input data table for the KATALASE program

NPTS	5					
NSNT	7	Hartford	Champaign	Louisville	Reston	Boston
Capacity	*q*	30	24	24	35	32
Minimum	*ql*	15	15	15	15	15
Fixed_cost	*f*	2499.00	3465.00	2727.00	0.00	2487.00
	Demand					
Hartford	1	0.00	937.00	987.00	833.00	499.00
Champaign	4	1041.00	0.00	779.00	1155.00	1025.00
Inianapolis	5	1037.00	685.00	775.00	863.00	779.00
Louisville	4	1067.00	755.00	0.00	909.00	841.00
Baltimore	5	853.00	897.00	741.00	514.10	895.00
Boston	30	447.10	869.00	709.00	829.00	0.00
Lansing	5	1109.00	757.00	845.00	1223.00	1093.00

Running the Program

To start KATALASE the user should:

1) Copy the program's executable file, katalase.out, into his/her directory;
2) Create the input data file, which should be checked for tab separation between entries, carrage returns between rows, and properly-placed Dummy variables;
3) Type **katalase.out**

The program will prompt the user for the input data file name. Entering an improper file name or problems with the input data file format may cause the program to stop execution. If this happens, a FORTRAN error statement will appear.

Next, the user will be asked to enter the maximum number of sites to be used in the solution (NT). If the planned event is a single occurrence event, the user's response should be 1. If the planned event is a series of meetings or multiple offerrings of the same training course, then the number enterred should reflect the maximum number of meetings or training course offerings.

The final prompt asks for the number of solutions to show in full (N_{show}). The complete optimal solution will be saved for the N_{show} least-cost solutions. Regardless of the response, the optimal cost and training sites for all combinations will be saved to the output file. No results are printed to the screen.

Interpretation of Program Output Files

The KATALASE program output is automatically saved in the file **results_*input filename***. A sample output file is presented in Appendix 2. Output files contain; input file name, output file name, NPTS, NT, NSNT, an echo of the input data file, and a listing of optimal solutions. Examining the echoed data is recommended.

The sample output file in Appendix 2 gives complete results for the first three optimal solutions. The remaining results only contain cost and sites. (The sample output was edited to show only the first and last ten solutions.) In this sample the maximum number of sites was set at three (NT=3).

The results for the optimal solution suggest that:

1. The minimum total cost for this training is $20634.00.
2. The training sites will be Louisville and Boston (only two sites are needed).
3. 31 employees will attend training in Boston. 30 will come from Boston, 1 will come from Hartford.

SUMMARY

In summary, KATALASE is an optimization program designed to assist planners in scheduling training courses. The user specifies costs associated with each potential training site. KATALASE then identifies the minimum-cost solution, costs of alternate solutions, and provides attendee routing schemes.

REFERENCES

Bradley, S. P., A. C. Hax and T. L. Magnanti, (1977), Applied Mathematical Programming, Addison-Wesley, Reading, Massachusetts, 716 pp.

APPENDIX 1, SOURCE CODE LISTING FILE

```
      program katalase
c=====================================================================
c     program designed to optimize training for northeast region
c
c
c     employs IMSL linear programming method to allocate student
c     to training centers, evaluate all possible sites, and
c     report costs as a function of where the training is held.
c
c     author.....tim cohn & greg baier
c     date.......march 9, 1993
c
c
c     variable size     definition
c     -------- ----     ----------
c     c        i,j         cost for one student to travel from
c                        site i to site j for training
c     d        i        demand for training course at site i
c     f        j        fixed cost of conducting course at site j
c     q        j        capacity limit at site j
c     ql       j        minimum class size at site j
c     npts              number of potential training sites
c     nsnt              number of sites needing training
c
c=====================================================================
      implicit double precision (a-h,o-z)
      parameter (is_i=100,is_j=26,is_k=100000,lda=126,n_sol=10)
      parameter (i_dv=2500)

      common /wgb001/
     1  c(is_i,is_j),d(is_i),f(is_j),q(is_j),ql(is_j)

      common /wgb002/
     2  a(lda,is_i*n_sol),irtype(lda),bl(lda),bu(lda),cost(i_dv)

      common /wgb003/
     1  xsol(i_dv),dsol(i_dv),xlb(i_dv),xub(i_dv),xcost(is_k),
     2  jsol(n_sol,is_k),xsort(is_k),iperm(is_k),xsolsum(n_sol)

      common /wgb004/
     1  nvars,nconstr,nn,nt,tdemand

      dimension
     1  jj(n_sol)

      equivalence
     1  (j1,jj(1)),(j2,jj(2)),(j3,jj(3)),(j4,jj(4)),(j5,jj(5)),
     2  (j6,jj(6)),(j7,jj(7)),(j8,jj(8)),(j9,jj(9)),(j10,jj(10))

      character*100 fname,fname2
      character*16 name_pts(is_i),name_snt(is_i)
      character*16 cdum,cdum1,cdum2,cdum3

      call dset(i_dv,0.d0,xlb,1)
      call dset(i_dv,-1.0d30,xub,1)
```

```
      call dset(lda,9999.d0,bu,1)
      call dset(lda,-9999.d0,bl,1)
c
c     read in the input data for travel costs, fixed costs, demand, capacity
c
      write(*,*) ' enter the data file name'
      read(*,'(a100)') fname
      open(unit=11,file=fname,status='old')
      write(fname2,'(a8,a92)') 'results_',fname(1:92)
      open(unit=12,file=fname2,status='new')
c
c     Get maximum number of training sites to consider simultaneously
c
      write(*,*) ' enter max. no. training locations allowed in solution'
      read(*,*) NT
         if(NT .gt. n_sol) then
            write(*,*) ' maximum number of training locations is:',n_sol
            stop
         endif
      write(*,*) ' enter the number of solutions to present in full'
      read(*,*) ncheck
c
c     write out banner for page 1
c
      write(12,110)
110   format(/,t20,'LP OPTIMIZATION FOR TRAINING LOCATIONS')
      write(12,111)
111   format(t20,'  Timothy A. Cohn and William G. Baier')
      write(12,112)
112   format(t20,'      version 1.0, March 9, 1993',//)
      write(12,114) fname
114   format(' Input File Name:  ',a100)
      write(12,115) fname2
115   format(' Output File Name: ',a100)
c
c     Read number of training sites;  add dummy null site at end
c
      read(11,*) cdum1,npts
         npts  =  npts+1
         if(npts .gt. is_i) then
            write(*,*) '  number of sites exceeds program capability'
            stop
         endif
      write(12,'(a36,i6)') ' No. Potential Training Sites (NPTS)',npts-1
      write(12,'(a36,i6)') ' No. Sites in Solution (NT)          ',NT
c
c     Read training site names, nsnt
c
      read(11,*) cdum1,nsnt,(name_pts(k),k=1,npts-1)
      name_pts(npts) =  '----'
      write(12,'(a36,i6)') ' No. Sites Needing Training (NSNT)   ',nsnt
c
c     read in capacity at each training site, fixed costs
c
         write(12,'(//,t33,''Possible Training Sites'')')
         write(12,122) (name_pts(k)(1:7),k=1,npts-1)
122      format(/,9(t24,7(1x,a7)/))
         read(11,*) cdum,cdum2,(q(j),j=1,npts-1)
```

```
         write(12,121) cdum,(q(j),j=1,npts-1)
121      format(/,7x,a16,9(t24,7f8.2/))
         read(11,*) cdum,cdum2,(ql(j),j=1,npts-1)
         write(12,121) cdum,(ql(j),j=1,npts-1)
         read(11,*) cdum,cdum2,(f(j),j=1,npts-1)
         write(12,121) cdum,(f(j),j=1,npts-1)
         q(npts)  =  0.d0
         f(npts)  =  0.d0
c
c     Read in cost data
c
         read(11,'()')
         write(12,
     1   '(//,t4,''Office'',t17,''Demand'',t35,''Total Travel Costs'')')
      do 10 i=1,nsnt
         read(11,*) name_snt(i),d(i),(c(i,j),j=1,npts-1)
         do 15 j=1,npts-1
            c(i,j)   =  max(0.00001d0,c(i,j))
15       continue
         c(i,npts)   =  0.d0
         write(12,120) name_snt(i),d(i),(c(i,j),j=1,npts-1)
120      format(/,2x,a16,f5.0,9(t24,7f8.2/))
10    continue
c
         tdemand     =  0.d0
      do 20 j=1,nsnt
         tdemand  =  tdemand + d(j)
20    continue
c
c
      NN =  nsnt
c
c     set up constraint matrices
c
         nconstr  =  NT+NN
         nvars =  NT*NN
         call dset(lda*n_sol*is_i,0.d0,a,1)
      do 40 i=1,NN
            bl(i)     =  d(i)
            bu(i)     =  d(i)
            irtype(i)    =  0
         do 40 j=1,NT
            a(i,i+NN*(j-1))    =  1.d0
40    continue
c
      do 50 j=1,NT
            irtype(NN+j)    =  3
         do 50 i=1,NN
            a(j+NN,i+NN*(j-1))    =  1.d0
50    continue
c
         ict    =  0
c
c     Begin major do-loops
c
      do 30 j1=1,npts
      do 30 j2=max(j1,sign(npts,1-nt)),npts
      do 30 j3=max(j2,sign(npts,2-nt)),npts
```

```
      do 30 j4=max(j3,sign(npts,3-nt)),npts
      do 30 j5=max(j4,sign(npts,4-nt)),npts
      do 30 j6=max(j5,sign(npts,5-nt)),npts
      do 30 j7=max(j6,sign(npts,6-nt)),npts
      do 30 j8=max(j7,sign(npts,7-nt)),npts
      do 30 j9=max(j8,sign(npts,8-nt)),npts
      do 30 j10=max(j9,sign(npts,9-nt)),npts
c
         ict   =  ict+1
            if(ict .gt. is_k) then
               write(*,*) ' number of cases exceeds program capability'
               stop
            endif
c
         call runit(jj,xcost(ict))
c
         do 30 k=1,nt
            jsol(k,ict) =  jj(k)
30    continue
c
c     Sort results with IMSL dsvrgp; prepare output
c
         do 65 k=1,ict
            iperm(k) =  k
65       continue
         call dsvrgp(ict,xcost,xsort,iperm)
c
c     Output Results
c
      write(12,113)
113   format(//////,t31,'OPTIMAL SOLUTIONS',//)
      write(12,110)
      write(12,111)
      write(12,112)
      write(12,116)
116   format(' Solutions listed in order of increasing cost')
      write(12,117)
117   format(' Infeasible solution indicated by:"Cost: $**********"',//)
      do 70 i=1,ict
         index =  iperm(i)
         write(12,100) i,xcost(index),(name_pts(jsol(k,index)),k=1,NT)
100      format(i4,'  Cost: $',f11.2,6x,10(a16))
         if(i .le. ncheck) then
               call dset(NT,0.d0,xsolsum,1)
               call runit(jsol(1,index),tcost)
            do 60 jt=1,nsnt
               write(12,101) name_snt(jt),(int(xsol(jt+(k-1)*NN)),k=1,NT)
101            format(6x,a16,3x,10(i9,7x))
               do 60 k=1,NT
                  ind            =  jt+(k-1)*NN
                  xsolsum(k)     =  xsolsum(k)+xsol(ind)
60          continue
               write(12,101) ' Total Attendees',(int(xsolsum(k)),k=1,NT)
               write(12,'(//)')
         endif
70    continue
      stop
      end
```

```
c
      subroutine runit(jj,tcost)
c=======================================================================
c
c     subroutine to implement lp
c
c=======================================================================
      implicit double precision (a-h,o-z)
      parameter (is_i=100,is_j=26,is_k=100000,lda=126,n_sol=10)
      parameter (i_dv=2500)

      common /wgb001/
      1  c(is_i,is_j),d(is_i),f(is_j),q(is_j),ql(is_j)

      common /wgb002/
      2  a(lda,is_i*n_sol),irtype(lda),bl(lda),bu(lda),cost(i_dv)

      common /wgb003/
      1  xsol(i_dv),dsol(i_dv),xlb(i_dv),xub(i_dv),xcost(is_k),
      2  jsol(n_sol,is_k),xsort(is_k),iperm(is_k),xsolsum(n_sol)

      common /wgb004/
      1  nvars,nconstr,nn,nt,tdemand

      dimension jj(n_sol)
c
c     Set up matrices
c
      do 10 i=1,NN
         do 10 j=1,NT
            cost(i+(j-1)*NN)  =  c(i,jj(j))
10    continue
c
c     Set up right-hand side constraints
c
      do 30 j=1,NT
         bu(NN+j) =  q(jj(j))
         bl(NN+j) =  ql(jj(j))
30    continue
c
c     Check to see if feasible solution exists and compute fixed costs
c
            tcapacity   =  0.d0
            fcost    =  0.d0
         do 91 js=1,nt
            tcapacity   =  tcapacity + q(jj(js))
            fcost    =  fcost + f(jj(js))
91       continue
      if(tdemand .gt. tcapacity) then
         tcost =  1.d30
      else
         call ddlprs
      1     (nconstr,nvars,a,lda,bl,bu,cost,irtype,xlb,xub,obj,xsol,dsol)
         tcost =  obj + fcost
      endif
      return
      end
```

APPENDIX 2, SAMPLE OUPUT FILE

LP OPTIMIZATION FOR TRAINING LOCATIONS
Timothy A. Cohn and William G. Baier
version 1.0, March 9, 1993

Input File Name: sample.input
Output File Name: results_sample.input
No. Potential Training Sites (NPTS) 5
No. Sites in Solution (NT) 3
No. Sites Needing Training (NSNT) 7

	Possible Training Sites				
	Hartfor	Champai	Louisvi	Reston	Boston
Capacity	30.00	24.00	24.00	35.00	32.00
Minimum	15.00	15.00	15.00	15.00	15.00
Fixed_cost	2499.00	3465.00	2727.00	0.00	2487.00

Office	demand	Total Travel Costs				
Hartford	1.	0.00	937.00	987.00	833.00	499.00
Champaign	4.	1041.00	0.00	779.00	1155.00	1025.00
Inianapolis	5.	1037.00	685.00	775.00	863.00	779.00
Louisville	4.	1067.00	755.00	0.00	909.00	841.00
Baltimore	5.	853.00	897.00	741.00	514.10	895.00
Boston	30.	447.10	869.00	709.00	829.00	0.00
Lansing	5.	1109.00	757.00	845.00	1223.00	1093.00

OPTIMAL SOLUTIONS

LP OPTIMIZATION FOR TRAINING LOCATIONS
Timothy A. Cohn and William G. Baier
version 1.0, March 9, 1993

Solutions listed in order of increasing cost
Infeasible solution indicated by:"Cost: $**********"

1	Cost: $ 20634.00	Louisville	Boston	----
	Hartford	0	1	0
	Champaign	4	0	0
	Inianapolis	5	0	0
	Louisville	4	0	0
	Baltimore	5	0	0
	Boston	0	30	0
	Lansing	5	0	0
	Total Attendees	23	31	0

2	Cost: $ 21164.00	Champaign	Boston	----
	Hartford	0	1	0
	Champaign	4	0	0
	Inianapolis	5	0	0
	Louisville	4	0	0
	Baltimore	4	1	0
	Boston	0	30	0
	Lansing	5	0	0
	Total Attendees	22	32	0

3	Cost: $ 23121.00	Louisville	Boston	Boston
	Hartford	0	0	1
	Champaign	4	0	0
	Inianapolis	5	0	0
	Louisville	4	0	0
	Baltimore	5	0	0
	Boston	0	16	14
	Lansing	5	0	0
	Total Attendees	23	16	15

4	Cost: $ 23643.00	Champaign	Boston	Boston
5	Cost: $ 24112.50	Reston	Boston	----
6	Cost: $ 24785.00	Champaign	Louisville	Boston
7	Cost: $ 25007.50	Louisville	Reston	Boston
8	Cost: $ 25021.50	Champaign	Reston	Boston
9	Cost: $ 25689.50	Reston	Boston	Boston
10	Cost: $ 26564.60	Hartford	Champaign	Boston
. . . .				
47	Cost: $ 52117.00	Champaign	Champaign	Champaign
48	Cost: $**********	Champaign	----	----
49	Cost: $**********	Champaign	Louisville	----
50	Cost: $**********	Louisville	Louisville	----
51	Cost: $**********	Champaign	Champaign	----
52	Cost: $**********	Reston	----	----
53	Cost: $**********	Louisville	----	----
54	Cost: $**********	Hartford	----	----
55	Cost: $**********	Boston	----	----
56	Cost: $**********	----	----	----