Süleyman Demirel Üniversitesi
Mühendislik Bilimleri ve Tasarım Dergisi
4(1), 15-20, 2016
e-ISSN: 1308-6693

Suleyman Demirel University
Journal of Engineering Sciences and Design
4(1), 15-20, 2016
e-ISSN: 1308-6693

Araştırma Makalesi Research Article

MATLOG: LOGISTICS ENGINEERING USING MATLAB

Michael G. Kay*

Fitts Department of Industrial and Systems Engineering North Carolina State University, Raleigh, NC, USA

Keywords
Matlog, Matlab, Facility Location, Freight Transport, Vehicle Routing, Networks.

Abstract
Matlog is a collection of computational routines and data that can be used within Matlab to implement a variety of different logistics-engineering-related tasks and provides a means to script the pre- and post-processing needed to interface with solvers like CPLEX. A detailed example of the use of Matlog to solve a facility location and allocation problem is presented and the use of Matlog in an educational environment is discussed.

1. Introduction

Matlog is a collection of computational routines and data that can be used within Matlab to implement a variety of different logistics-engineering-related tasks. It consists of Matlab functions that either directly implement solution techniques or provide a means to script the pre- and post-processing needed to interface with solvers like CPLEX. It provides a uniform, platform-independent environment and is available for download at Kay (2015). The routines in Matlog can be grouped into the following categories:

Facility location: Continuous minisum facility location, alternate location-allocation (ALA) procedure, discrete uncapacitated facility location, longitude and latitude to nearest city, Mercator projection plotting, and location aggregation based on distance

Freight transport: Transport charges for TL and LTL, minimum total logistics cost, and aggregate multiple shipments

Vehicle routing: VRP, VRP with time windows, traveling salesman problem (TSP), dial-a-ride, and long-haul truck routing problems

Networks: Shortest path, transportation, min cost network flow, minimum spanning tree, and multi-period multi-product production planning problems

General purpose: Linear programming using the revised simplex method procedure, mixed-integer linear programming, algebraic interface to formulate MILP for CPLEX, and steepest descent pairwise interchange (SDPI) heuristic for QAP

Data: U.S. cities, U.S. highway network (Oak Ridge National Highway Network); U.S. 3- and 5-digit ZIP codes; U.S. Census Block Group data

Matlog was developed initially for teaching purposes in a graduate-level Logistics Engineering course as a replacement for CAPS Logistics' commercial products Supply Chain Designer and RoutePro. The problem with using the CAPS products for teaching purposes was their lack of flexibility and the lack of transparency with respect to algorithmic details (see Campbell (2000) for another report of the CAPS products in an educational environment). Although the products had an easy-to-use GUI interface, when a new logistics network was to be analyzed, the required processing had to be specified using a specialized set of procedures, the CAPS Logistics Toolbox. In industrial applications, either trained CAPS engineers would use the Toolkit to customize the products to each customer's particular network as part of the purchase price of the product, or for a large customer, like Wal-Mart for example, CAPS could train the customer's own engineers in use of the Toolkit. As long as the changes to a customer's network were incremental modifications from the initial configuration, then CAPS would remain usable from just a GUI interface. In an educational environment, especially when teaching logistics engineering, it is import that students have a flexible computational tool and, when possible, that they can drill down if necessary and see all of the algorithmic details associated with the procedures that they are using. Instead of trying to teach students to use the Toolkit, Matlab was chosen as a suitable platform to try to develop a reasonable substitute for CAPS. As compared to the Toolkit, Matlab would provide students with knowledge of a general purpose computational environment that nicely complements Excel (what is easy to do in Matlab can be hard to do

* Corresponding author: kay@ncsu.edu

in Excel, and vice versa) and is easily available in most Colleges of Engineering,

Instead of trying to describe all of the features of Matlog at a high level, a single example of the use of Matlog will be discussed in detail and all of the code needed to duplicate the example will be described. This example is a very common task and will help illustrate the use of Matlog as a tool for solving a very common logistics engineering problem. If you are not familiar with Matlab, you can review the example to get the basic flavor of Matlog or, after reviewing a short tutorial on Matlab, Kay (2010), you will be able to understand most all of the code details. After the example, this paper concludes with a discussion of for using Matlog in an educational environment.

2. Example: Facility Location and Allocation

Four new facilities (NFs) are to be located anywhere in the continental U.S. to serve retail customers. The best locations for the facilities are those that minimize the average distance of a customer from its closest facility, where, for example, each facility could be distribution center and each customer a retail store. Since no other information or data is available, the centroid of each 3-digit ZIP code is used to represent each customer's location (EF, for existing facility) and the population of the ZIP code is used to represent relative demand. In the following solution, the shaded boxes contain Matlab code and the san-serif text and figures represent the resulting output associated with executing the code. In each code box, Matlog functions are italicized

Create Data

Load 3-digit ZIP codes with positive population in continental U.S., where *EF* is an 880 × 2 matrix of the longitude and latitude of each ZIP code centroid and *pop* is an 880-element vector of population. The Matlog function *uszip3* is used to read specific fields of ZIP code data as specified by the arguments '*XY*' and '*pop*', where ZIP codes outside of the continental U.S. (in Alaska, Hawaii, and Puerto Rico) are removed using Matlog's *mor* (multiple OR) command.

```
[EF,pop] = uszip3('XY','Pop',~mor({'AK','HI','PR'},uszip3('ST')) &
uszip3('Pop') > 0);
m = size(EF,1)              % number of codes

m = 880
```

Plot the resulting EF locations and then set the number of NFs.

```
makemap(EF)          % open new figure for subsequent plotting
pplot(EF,'r.');      % projective plot of EF locations
title([num2str(m) ' ZIP Code Locations'])
p = 4;               % number of NF
```

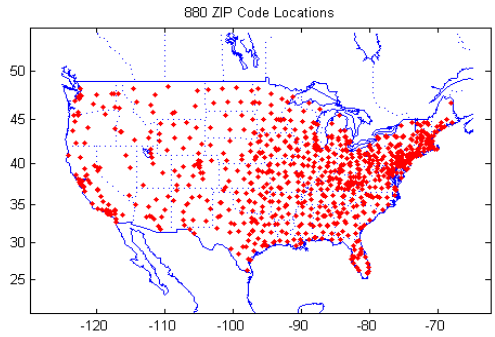

Figure 1. 880 ZIP Code Locations

Continuous Location: ALA Procedure

Given initial NF locations, the Alternate Location–Allocation (ALA) procedure (Cooper, 1963) finds optimal EF allocations and then finds optimal NF locations for these allocations, continuing to alternate until no further EF allocation changes are made. The following is a pseudocode description of the ALA algorithm:

ALA PROCEDURE: $(NF, W) \leftarrow ALA(NF_0, EF, w)$

1. Given initial NF locations NF_0
2. $TC \leftarrow \infty$
3. $W' \leftarrow allocate(NF, EF, w)$
4. $NF' \leftarrow locate(W', EF)$
5. If $TC(NF', W') > TC$, stop; otherwise $TC \leftarrow TC(NF', W')$, $NF \leftarrow NF'$, $W \leftarrow W'$, and go to step 3

There is a close correspondence possible between the ALA pseudocode and how it can be coded in Matlab, where the anonymous functions *TCh*, *allocate*, and *locate* provide a means of creating simple in-line functions without having to create separate files.

```
TCh = @(W,NF) sum(sum(W.*dists(NF,EF,'mi')));
allocate = @(NF)
full(sparse(argmin(dists(NF,EF,'mi')),1:m,pop',n,m));
opt =
optimset(fminsearch('defaults'),'Display','off','TolFun',1,'TolX',1);
locate = @(W,NF0) fminsearch(@(NF) TCh(W,NF),NF0,opt);
```

TCh determines the total people-miles given the allocations defined in the $p \times m$ matrix W, allocate determines the allocation based on the closest NF to each EF, and locate performs n 2-dimensional location searches using *fminsearch*, which is Matlab's default routine for nonlinear unconstrained optimization.

```
rng(1234)                % set random number seed so can duplicate
results
NF = randX(EF,p);        % generate random initial locations
TC = Inf;
done = false;
while ~done
  Wi = allocate(NF);
  NFi = locate(Wi,NF);
  TCi = TCh(Wi,NFi);
```

```
  fprintf('%e\n',TCi);
  if TCi < TC
    TC = TCi; NF = NFi; W = Wi;
  else
    done = true;
  end
end
lonlat2city(NF,uscity)       % determine closest city to each location
avgdist = TC/sum(pop)        % detemine average distance
makemap(EF)                  % plot locations and allocations
pplot(lev2list(W),[NF;EF],'g-');
pplot(EF,'r.');
pplot(NF,'kv')
title([num2str(p) ' NF Locations and Allocations to EFs'])
```

```
1.012094e+11
9.790048e+10
9.778027e+10
9.765950e+10
9.744964e+10
9.727697e+10
9.717181e+10
9.686568e+10
9.661784e+10
9.637384e+10
9.603856e+10
9.567937e+10
9.480952e+10
9.423325e+10
9.421143e+10
9.421143e+10
9.421143e+10
9.421143e+10

NF 1 is 11.33 mi SE of Ponderosa, CA
NF 2 is 4.08 mi W of Lawrenceburg, KY
NF 3 is 4.00 mi W of Richardson, TX
NF 4 is in Summit, NJ

avgdist =
  307.2082
```

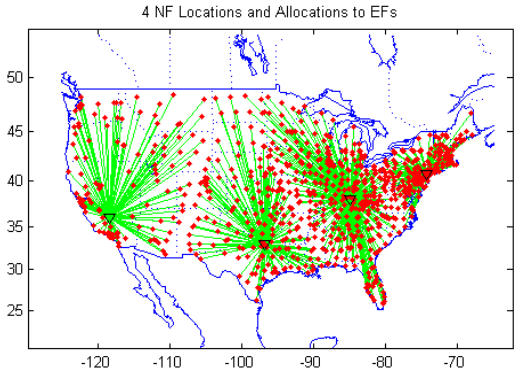

Figure 2. 4 NF Locations and Allocations to EFs

This average distance result of 307.2082 miles represents only a single run. Since the ALA procedure finds only a local optima, the procedure should be applied multiple times using different initial NF locations, keeping the best solution found as the final solution.

The Matlog function *ALA* implements the ALA procedure and produces the same results.

```
rng(1234)          % using same seed as before
[NF,TC] = ala(randX(EF,p),pop',EF,'mi');
lonlat2city(NF,uscity)
```

```
NF 1 is 11.33 mi SE of Ponderosa, CA
NF 2 is 4.08 mi W of Lawrenceburg, KY
NF 3 is 4.00 mi W of Richardson, TX
NF 4 is in Summit, NJ
```

Discrete + Continuous Location: p-Median Heuristic + ALA

An alternative to just using the ALA procedure is to first use the p-Median heuristic to find the best NF locations from among the set of EF locations and then use these NF locations as the initial locations for the ALA procedure.

```
C = repmat(pop',m,1) .* dists(EF,EF,'mi');  % m by m variable
cost matrix
[y,TC] = pmedian(p,C);
NF = EF(y,:);                          % select NFs from EFs
lonlat2city(NF,uscity)
avgdist = TC/sum(pop)
```

```
  Add: 95695328508.251694
 Xchg: 94496036279.333450
 Drop: 97544509467.758331
 Xchg: 93896189493.826721
Final: 93896189493.826721

NF 1 is in Doylestown, PA
NF 2 is in Nellieburg, MS
NF 3 is in Hometown, IL
NF 4 is 5.48 mi NE of Rosamond, CA

avgdist =
  306.1802
```

```
[NF,TC] = ala(NF,pop',EF,'mi');   % Use NF's from PMEDIAN as
intial locations
lonlat2city(NF,uscity)
avgdist = TC/sum(pop)
```

```
NF 1 is 4.37 mi SW of Stockton, NJ
NF 2 is 6.49 mi SE of Meridian, MS
NF 3 is in Merrionette Park, IL
NF 4 is 16.55 mi NE of Pearsonville, CA

avgdist =
  304.7786
```

Both the initial p-Median and the ALA improvement resulted in better average distance results than just the single ALA run. Although multiple runs of ALA would improve its results, they would still not likely be better than p-Median with ALA improvement. The reason that ALA is still valuable is the flexibility with which the allocation and location subprocedures can be defined. If the NFs are capacitated, then the anonymous function *allocate* can be a call to *trans*, the transportation problem procedure in Matlog; if some the NF's locations are fixed, then it is easy to modify anonymous function *locate*.

To understand the p-Median heuristic, see Daskin (1995); to understand how the heuristic has been implemented in Matlog, the *type* command in Matlab can be used to list all of the code for *pmedian*. The heuristic first adds 1 to *p* NFs using a greedy add procedure *ufladd*, followed by the *uflxchg* pairwise exchange improvement procedure; this result is then compared to the result of the greedy drop procedure *ufldrop* and *uflxchg* and the best result is reported.

```
type pmedian
```

```
function [y,TC,X] = pmedian(p,C,dodisp)
%PMEDIAN Hybrid algorithm for p-median location.
% [y,TC,X] = pmedian(p,C)
%          = pmedian(p,C,dodisp),  display intermediate results
%     p = scalar number of NFs to locate
%     C = n x m variable cost matrix,
%         where C(i,j) is the cost of serving EF j from NF i
%dodisp = true, default
%     y = p-element NF site index vector
%    TC = total cost
%       = sum(sum(C(X)))
%     X = n x m logical matrix, where X(i,j) = 1 if EF j allocated
to NF i
%
% Algorithm: Report the minimum of UFLADD(0,C,[],p) followed
by
% UFLXCHG(O,C,yADD) and UFLDROP(0,C,[],p) followed by
UFLXCHG(O,C,yDROP),
% where yADD and yDROP are the p-element NF site index
vector returned by
% UFLADD and UFLDROP for a fixed number of NFs
%
% Example (Example 8.8 in Francis, Fac Layout and Loc, 2nd
ed.):
% p = 2
% C = [0    3    7   10    6    4
%      3    0    4    7    6    7
%      7    4    0    3    6    8
%     10    7    3    0    7    8
%      6    6    6    7    0    2
%      4    7    8    8    2    0]
% [y,TC,X] = pmedian(p,C);       y,TC,full(X)

% Copyright (c) 1994-2014 by Michael G. Kay
% Matlog Version 16 03-Jan-2014
(http://www.ise.ncsu.edu/kay/matlog)

% Input Error Checking
**************************************************
*
error(nargchk(2,3,nargin));

[n,m] = size(C);
if nargin < 3 || isempty(dodisp), dodisp = true; end

if ~isscalar(p) || p > n || p < 1
   error('"p" must be between 1 and "n".')
elseif ~isscalar(dodisp) || ~islogical(dodisp)
   error('"dodisp" must be a logical scalar.')
end
% End (Input Error Checking)
**********************************************

[y,TC] = ufladd(0,C,[],p); if dodisp, fprintf('  Add: %f\n',TC),
end
[y,TC,X] = uflxchg(0,C,y); if dodisp, fprintf(' Xchg: %f\n',TC),
end

[y1,TC1] = ufldrop(0,C,[],p); if dodisp, fprintf(' Drop:
%f\n',TC1), end
[y1,TC1,X] = uflxchg(0,C,y1); if dodisp, fprintf(' Xchg:
%f\n',TC1), end

if TC1 < TC, TC = TC1; y = y1; end
if dodisp, fprintf('Final: %f\n',TC), end

y = sort(y);
X = logical(sparse(y(argmin(C(y,:),1)),1:m,1,n,m));
```

Aggregate Demand Data to Reduce Size

In order to determine if the p-Median result is the optimal result, the problem can be formulated as a mixed-integer linear program (MILP) and solved to optimality using a MILP solver like CPLEX. Based on experience, the 880 × 880 variable cost matrix C will cause CPLEX to run out of memory. To avoid this, locations can be aggregated to reduce the number of EF locations. The procedure combining the pair of locations that are closest together and replacing both locations by their population-weighted average location and then adds together both population values, continuing until a third of the locations (294) are left.

```
D = triu(dists(EF,EF,'mi'));
D(D==0) = Inf;
maxsize = ceil(size(D,1)*(1/3));    % aggregate to 1/3 original
size
while size(D,1) > maxsize
   [i,j] = argmin(D);
   popij = pop([i j]);
   EF(i,:) = (popij(:)'*EF([i j],:))/sum(popij);  % pop-weighted
centroid location
   EF(j,:) = [];
   pop(i) = pop(i) + pop(j);
   pop(j) = [];
   D = triu(dists(EF,EF,'mi'));
   D(D==0) = Inf;
end
makemap(EF)
pplot(EF,'r.');
m = size(EF,1)
title([num2str(m) ' ZIP Codes Locations after Aggregation'])

m =
   294
```

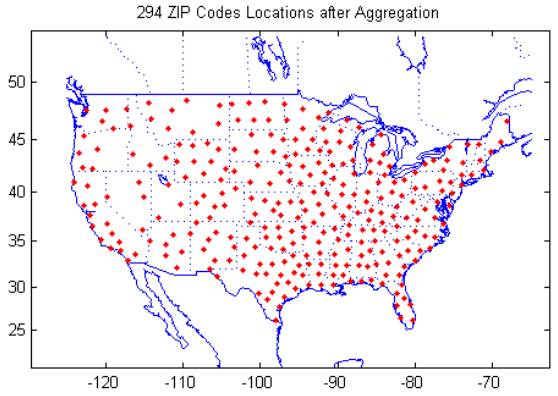

Figure 3. 4 249 ZIP Codes Locations after Aggregation

Re-run p-Median on Aggregated Data

In order to get a baseline for the MILP result, the p-Median and ALA improvement are re-run using the new variable cost data.

```
C = repmat(pop',m,1) .* dists(EF,EF,'mi');  % m by m variable
cost matrix
[y,TC] = pmedian(p,C);
NF = EF(y,:);             % select NFs from EFs
lonlat2city(NF,uscity)
avgdist = TC/sum(pop)
```

```
  Add: 95815142943.656967
 Xchg: 94051364921.479156
 Drop: 96038290150.573212
 Xchg: 93622738138.367325
Final: 93622738138.367325
```

```
NF 1 is 4.07 mi E of Hoboken, NJ
NF 2 is 7.42 mi NW of Orrville, AL
NF 3 is in La Grange Park, IL
NF 4 is 5.48 mi NE of Rosamond, CA

avgdist =
  305.2885
```

```
[NF,TC] = ala(NF,pop',EF,'mi');
lonlat2city(NF,uscity)
avgdist = TC/sum(pop)
```

```
NF 1 is in Martinsville, NJ
NF 2 is 8.74 mi W of Demopolis, AL
NF 3 is in La Grange Park, IL
NF 4 is 14.73 mi NE of Pearsonville, CA

avgdist =
  303.6785
```

Discrete + Continuous Location: Cplex + ALA

Given m EFs and n sites at which NFs can be established, the p-Median problem can be formulated as the following MILP:

$$\text{Min } TC = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \quad ,m$$

$$y_i \geq x_{ij} \qquad i = 1, \quad ,n;\ j = 1, \quad ,m$$

$$\sum_{i=1}^{n} y_i = p$$

$$0 \leq x_{ij} \leq 1 \qquad i = 1, \quad ,n;\ j = 1, \quad ,m$$

$$y_i \in \{0,1\} \qquad i = 1, \quad ,n$$

where

c_{ij} variable cost to serve all of EF j's demand from site i
y_i 1, if NF established at site i; 0, otherwise
x_{ij} fraction of EF j's demand served from NF at site i.

This MILP for p-Median is termed a "strong formulation" due to the second set of nm constraints, which results in an LP relaxation that gives a tight lower bound that quite often is optimal. When these constraints are replaced with the n constraints

$$my_i \geq \sum_{j=1}^{m} x_{ij}$$

,

$i = 1, \quad ,n.$

The formulation is termed "weak" since the lower bound from the LP relaxation is not very tight, resulting in a large branch-and-bound tree.

The MILP for this example can be created using Matlog's *Milp* class, which provides a means of inputting data similar to an algebraic modeling language like OPL.

```
mp = Milp('pMedian');        % constructor for Milp object
c = repmat(0,1,m);           % no fixed cost
mp.addobj('min',c,C)         % Add cost arrays to objective
function
for j = 1:m
   mp.addcstr(0,{':',j},'=',1)     % add m constraints
end
for i = 1:m
   for j = 1:m
      mp.addcstr({i},'>=',{i,j});  % add m^2 constraints (strong
formulation)
   end
end
mp.addcstr(1,0,'=',p)        % add constraint to force p NFs
mp.addub(Inf,1)              % upper bounds
mp.addctype('B','C')         % c is binary, C is continuous
```

If CPLEX is available on your computer (using version 12.5 in this example), then the *Milp* model can be copied to a *Cplex* object and then, using the basic CPLEX API methods, the model can be solved.

```
cp = Cplex;                  % constructor for Cplex object
cp.Model = mp.Model;         % copy Milp model to Cplex model
cp.solve();                  % solve Cplex model
```

Table 1. Parameters

Parameters	Value
Tried aggregator	1 time.
Presolve time	0.19 sec. (91.45 ticks)
Found incumbent of value	3.4020985e+011 after 0.33 sec. (169.65 ticks)
Probing time	0.02 sec. (8.37 ticks)
Tried aggregator	1 time.
Presolve time	0.11 sec. (92.20 ticks)
Probing time	0.03 sec. (8.37 ticks)
MIP emphasis: balance optimality and feasibility. MIP search method: dynamic search. Parallel mode: deterministic, using up to 8 threads.	
Root relaxation solution time	21.84 sec. (7303.04 ticks)

```
        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                       3.40210e+011        0.0000    25637  100.00%
*     0+    0                       3.38632e+011        0.0000    25637  100.00%
*     0     0      integral     0  9.36227e+010  9.36227e+010    25637    0.00%
Elapsed time = 22.50 sec. (7711.35 ticks, tree = 0.00 MB, solutions = 3)

Root node processing (before b&c):
  Real time             =   22.01 sec. (7433.78 ticks)
Parallel b&c, 8 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =   22.01 sec. (7433.78 ticks)
```

Figure 4. Result's screen

The solution can then be displayed, where the *Milp* method *namesolution* is used to extract the y variables of the solution corresponding to the zero-valued coefficients c in the objective function.

```
disp(cp.Solution.statusstring)
TC = cp.Solution.objval
x = mp.namesolution(cp.Solution.x);
y = find(x.c);
NF = EF(y,:);
lonlat2city(NF,uscity)
avgdist = TC/sum(pop)
```

```
integer optimal solution
TC =
   9.3623e+10

NF 1 is 4.07 mi E of Hoboken, NJ
NF 2 is 7.42 mi NW of Orrville, AL
NF 3 is in La Grange Park, IL
```

```
NF 4 is 5.48 mi NE of Rosamond, CA

avgdist =
  305.2885
```

This result matches the reduced p-Median result and confirms that it is the optimal result.

If CPLEX is not available, then the Matlab's MILP solver *intlinprog* can be used. Its input format is similar to Matlab's LP solver *linprog*, with the addition of a second input vector of indices corresponding to the integer variables. The *Milp* method *milp2lp* is used to convert the instance to LP arguments. In the p-Median formulation, only the *y* variables are integer, corresponding to indices 1 to *n*, which are inserted as the second input argument to *intlinprog*.

```
lp = mp.milp2lp;
n = length(c)
intcon = 1:n;       % only firts n are integer
milp = {lp{1},intcon,lp{2:end}}  % add incon to make lp into
milp
[x,TC2,exitflag,output] = intlinprog(milp{:});
```

```
n =
   294
milp =
  Columns 1 through 4
    [86730x1 double]    [1x294 double]    [86436x86730 double]
[86436x1 double]
  Columns 5 through 8
    [295x86730 double]    [295x1 double]    [86730x1 double]
[86730x1 double]
LP:              Optimal objective value is 9.362274e+10.

Optimal solution found.

Intlinprog stopped at the root node because the
objective value is within a gap tolerance of the optimal value,
options.TolGapAbs = 0
(the default value). The intcon variables are integer within
tolerance,
options.TolInteger = 1e-05 (the default value).
```

This result matches the CPLEX result.

3. Conclusions

One of the major unexpected benefits of using Matlog in educational environment for the past fifteen years is the stability that it has provided. Over this time period, CAPS Logistics was first purchased by Baan and now is part of Infor's Strategic Network Design software product (Funaki, 2009). Trying to keep up with these changes while trying to provide a stable software environment would have been a challenge. Matlab has been a very stable coding environment. Many of the Matlog functions developed over fifteen years ago still run unchanged. The major difficulty is using Matlog has been a steep learning curve for those students without programming experience, even though most of the functions in Matlog eliminate detailed programming. It remains a challenge to try to find the right level of detail for each procedure, balancing ease of use with flexibility.

Conflict of Interest

No conflict of interest was declared by the authors.

References

Campbell, A., Goentzel, J., Savelsbergh, M. (2000), "Experiences with the use of supply chain management software in education," *Production and Operations Management*, Vol. 9, No. 1, pp. 66–80.

L. Cooper, L. (1963), "Location-allocation problems," *Operations Research,* Vol. 11, pp. 331–343.

Daskin, M. (1995), *Network and Discrete Location: Models, Algorithms, and Applications*, Wiley, New York.

Kay, M. (2015), "Matlog: Logistics Engineering Matlab Toolbox," http://www.ise.ncsu.edu/kay/matlog (last accessed on July 9, 2015).

Funaki, K. (2009), "State of the Art Survey of Commercial Software for Supply Chain Design," Supply Chain and Logistics Institute, Georgia Tech, https://www.scl.gatech.edu/downloads/

GTSCL_SCDesign_Software_Survey.pdf (last accessed on July 9, 2015).