IOE510 - Linear Programming

Project Assignment

Jainabou B. Danfa, Sean T. Kelly
Due November 13th, 2019
University of Michigan

Contents

List of Figures

1 Introduction

The goal of the following work is to maximize the profit for a new company manufacturing and selling men's and women's jeans to a known set of customers. For this work, the assumed customers are department and clothing stores which are similar in size, and thus we expect customer demands to be similar in size and standard deviation. As such, customer demands are assumed to be based on a normal distribution with mean 500 units, or single pairs of jeans, and standard deviation of 100. Decisions to be made include deciding where to build factories from a given set of plausible locations, the capacity of the factories built, and the number of pairs of jeans, heretofore referred to as units, distributed to each customer from each factory for each hypothetical set of unknown future customer demands (i.e., a scenario). The primary method employed in this work is solving a two-stage stochastic optimization problem using a linear program with sample average approximation. This method is employed because factory locations and capacities need to be decided now, but future customer demands are currently unknown. Excel Solver is used to provide estimations for a reduced problem in which a smaller set of customers is considered to abide by the solver's limitations, and AMPL with the cplex solver is used to solve the full problem for 1, 3, 100, 1000, and 5000 scenarios used for sample average approximation. For each of these number of scenarios, 5 random iterations of the model is solved in AMPL, and using only a single scenario the reduced model is solved in Excel to fit within the solver's limitations.

2 Linear Program Model

A fully generalized linear program model is presented, followed by a description of a reduced model which is explicitly used for Excel due to the solver's limitations. Without loss of generality the same model can be applied on a reduced set of customers or factory locations, which is necessary to limit the number of variables such that the Excel Solver's limit is not exceeded (200 variables maximum).

2.1 Model Formulation

The following linear program utilizing sample average approximation is constructed, where the objective function value represents the expected profit and is based on pre-determined parameters and constraint equations. No iterations are performed over these known pre-determined parameters and they are assumed constant for all scenarios and for each model iteration. The variables and parameters used in the generalized model are defined as follows:

Variables:

- x_i: binary variable which is 1 if factory i is built and 0 if factory i is not built
- y_i: integer variable representing capacity of factory i
- z_{ij}^k: integer variable representing amount of units supplied to customer j from factory i in scenario k

Note that the location and capacity of a factory i, denoted by x_i and y_i respectively, are constant for all scenarios. Because factory locations and capacity need to decided before the demand from all customers is known, these variables represent stage-one decision. However, the decisions associated with the number of units sent from factory i to customer j for each scenario k assume the demand is known for each scenario k after the factory locations and capacities are determined. Thus, these represent stage-two decisions. The number of variables associated with z_{ij}^k equals the number of customers times the number of plausible factory locations times the number of scenarios. Consequently, the number of variables can get very large due to this term.

Parameters:

s: selling price of a single unit (in \$)

c_p: production cost associated with producing a single unit (in \$)

c_t: transportation cost of a single unit per mile when delivering to a customer (in \$ per mile)

c_f: cost to build a single factory (in \$)

c_u: cost per unit of capacity installed (in \$)

c_{total}: the total cost allowed for building factories and stocking factories, or maximum investment cost (in \$)

p_{min}: minimum capacity of a factory if built (integer)

p_{max}: maximum capacity of a factory if built (integer)

f_{min}: minimum number of factories that must be built (integer)

f_{max}: maximum number of factories that can be built (integer)

d_{ij}: distance from a factory i to customer j (in miles)

u_j^k: randomly generated demand of, or number of units order by, customer j for scenario k from a normal distribution with mean 500 and standard deviation of 100 (integer)

n_i: number of plausible factory locations (integer)

n_j: number of customers (integer)

n_k: number of scenarios used for sample average approximation (integer)

For the solutions provided in section 3, these parameters are assumed to be known with the exception of n_k which changes based on the number of scenarios considered.

The objective function, which represented the expected profit, is defined to be:

$$\text{Max} \quad \frac{1}{n_k}\sum_{k=1}^{n_k}\left(\sum_{i=1}^{n_i}\sum_{j=1}^{n_j}(s - c_p - c_t d_{ij})z_{ij}^k\right) - \sum_{i=1}^{n_i} c_f x_i - \sum_{i=1}^{n_i} c_u y_i \tag{1}$$

Within the first term, $(s - c_p - c_t d_{ij})z_{ij}^k$ represents cost associated of a single unit sent to customer j from factory i for scenario k. As can be seen, the shorter the distance to a customer j from a factory i, the smaller the total traveling cost and thus higher the profit. Additionally, for a smaller production cost c_p and larger selling price s, the profit increases as well as expected. The second term represents the total cost associated with building all factories, and the third term represents the total cost associated with stocking all factories. Additionally, note that although there are variables associated with stage-one and stage-two decisions, only a single objective function is used and only one linear program needs to be solved to calculate all variables.

The constraint equations are defined to be:

$$\sum_{i=1}^{n_i} x_i \geq f_{min} \tag{2}$$

$$\sum_{i=1}^{n_i} x_i \leq f_{max} \tag{3}$$

$$\sum_{i=1}^{n_i} c_f x_i + \sum_{i=1}^{n_i} c_u y_i \leq c_{total} \tag{4}$$

$$y_i \geq p_{min} x_i \quad \forall\ i \tag{5}$$

$$y_i \leq p_{max} x_i \quad \forall\ i \tag{6}$$

$$\sum_{i=1}^{n_i} z_{ij}^k = u_j^k \quad \forall\ j,\ \ \forall\ k \tag{7}$$

$$\sum_{i=1}^{n_j} z_{ij}^k \leq y_i \quad \forall\ i,\ \ \forall\ k \tag{8}$$

$$x_i = \{0, 1\} \quad \forall\ i \tag{9}$$

$$z_{ij}^k \geq 0 \quad \forall\ i,\ \ \forall\ j,\ \ \forall\ k \tag{10}$$

A description of each of the constraints is given below:

Eqn. (2): The number of factories built must be greater than or equal to the specified minimum number of factories that *must be* built, f_{min}. Thus, summing over x_i results in an integer value equal to the number of factories built

Eqn. (3): The number of factories built must be less than or equal to the specified maximum number of factories that *can be* built, f_{max}. Like in Eqn. (2), summing over x_i results in an integer value equal to the number of factories built

Eqn. (4): The total cost to build (first term on left hand side) and stock (second term on left hand side) the factories must be less than the total maximum investment allowed, c_{total}. If a factory is built, and hence $x_i = 1$, then the associated cost to build a factory c_f is included in the sum. If not built, and hence $x_i = 0$, then there is no effect on the sum since $c_f x_i = 0$ if this is the case. Regarding the capacity, note from Eqns. (5) and (6) that if $x_i = 0$, due to the bounds of y_i then $y_i = 0$ necessarily. Thus, if a factory i is not built, then y_i is 0 and does not add to the capacity term. However, if $x_i = 1$, the factory must be stocked and $y_i \neq 0$, and therefore the cost to stock factory i is included in the sum

Eqn. (5): This equation provides a lower bound on the capacity of a factory i. If factory i is built, then the lower bound is $y_i \geq p_{min}$, and if factory i is not built, then $y_i \geq 0$

Eqn. (6): This equation provides an upper bound on the capacity of a factory i. If factory i is built, then the upper bound is $y_i \leq p_{max}$, and if factory i is not built, then $y_i \leq 0$. Note that combined with Eqn. (5), if factory i is built then $p_{min} \leq y_i \leq p_{max}$. However, if factory i is not built, then $y_i = 0$ necessarily which follows because a factory that isn't built cannot hold a nonzero capacity

Eqn. (7): This constraint enforces that the sum of the number of units supplied across all factories to customer j in scenario k must equal the number of units demanded by customer j. That is, this constraint enforces that demand from customer j in scenario k is exactly met and not exceeded, and that this demand can come from any of the built factories. This constraint is applied for all customers in all scenarios

Eqn. (8): This constraint enforces that the sum of the number of units supplied to all customers from factory i in scenario k must be less than or equal to the capacity of factory i. That is, the number of units supplied to all customers from a factory i must not exceed the capacity of factory i, since we cannot supply more units from a factory than its given capacity. This constraints is applied for all factories in all scenarios

Eqn. (9): This constraint enforces that x_i be a binary variable which takes values 0 or 1

Eqn. (10): This constraint enforces all z_{ij}^k must be non-negative since we cannot send a negative number of units from factory i to customer j in any scenario k

For the full problem under consideration, the following parameter values are assumed constant and pre-determined for all model iterations:

Table I: Assumed parameter values for problem under consideration

s	c_p	c_t	c_f	c_u	c_{total}	p_{min}	p_{max}	f_{min}	f_{max}	n_i	n_j
\$125	\$5	\$1.50	\$10,000	\$5	\$250,000	5,000	10,000	4	8	10	25

where d_{ij} values are provided Table A1 in Appendix A for the range of values of n_i and n_j in Table I, u_j^k is determined randomly for each case for each model iteration, and n_k is varied depending on the number of scenarios used for sample average approximation. With this set of parameters, the number of variables to solve equals $2n_i + n_i n_j n_k$, where $2n_i$ contains all x_i and y_i and $n_i n_j n_k$ contains all z_{ij}^k.

2.2 Reduced Variable Model for Excel

Excel solver has a variable limit of 200, which is easily exceeded for this model primarily due to the number of variables contained in z_{ij}^k. The full problem consisting of $n_i = 10$ potential factory locations and $n_j = 25$ customers has 250 variables for each scenario from z_{ij}^k, 10 variables for the production levels (i.e., capacity y_i) at each location, and 10 for the binary variables, x_i, if the factory is opened or not for each location. Having a total of 270 variables was not feasible for Excel Solver's limitations. Thus, a subset of the full problem was constructed with 9 customers and 10 potential locations, which gives 200 variables in the model using two scenarios. In this reduced model, 9 random demands were produced for each scenario (i.e., u_j^k values) still normally distributed with mean 500 and a standard deviation of 100, however the equations and problem formulation are not adjusted and all other parameters match those in Table I. The distances d_{ij} for the 9 customers are for customers 1 through 9 given in Table A1 in the Appendix. Note that only two scenarios can be accommodated since z_{ij}^k contains 180 variables alone with this reduced formulation.

2.3 Model Implementation in Excel

The model was implemented in Excel using the Solver add-in. The MIT open solver tutorial[1] was used as a general guide for implementing the model using the Solver add-in. The Simplex LP engine was selected as the solving method and all unconstrained variables were made non-negative in the solver dialogue box. All other variables, constraints, and the objective value were input with the cell ranges of those specific values and/or formulas. The 9 random demands for the reduced problem (i.e., u_j^k values) were produced from a normal distribution with mean 500 and a standard deviation of 100, and were generated with the excel function [ROUND(NORM.INV(RAND(), 500, 100),0)]. Each run changed the various cells in the excel sheet so the analysis of the result was quite easy to understand and perform. To obtain 5 model iterations using different random demands, the solver was run 5 times and each run generated a different set of random demands.

2.4 Model Implementation in AMPL

The model implementation in AMPL consists of three separate files: a model .mod file, a data .dat file, and a run .run file. The model file, entitled Project_Final.mod is used to define the model discussed in sub-section 2.1. Initially, the parameters n_i and n_j are introduced representing the number of factories and customers, respectively, as shown below:

```
### Defining Parameters Used for Defining Sets
param n_i;      # Number of factory locations (integer)
param n_j;      # Number of customers (integer)
```

Figure 1: Defining parameters n_i and n_j within model file

Next, as was requested in the assignment[2], the number scenarios to generate for sample average approximation, n_k, was specified to be a user input. Thus, the following lines in the model file are included to define n_k. The user is prompted in the AMPL console window ”How many scenarios to be considered (integer)?”, at which point the user enters a value and presses ”Enter”. Because the number of scenarios must be an integer value, if the user does not input an integer, the value is rounded to the nearest integer and the rounded value instantiates n_k.

```
# Specifically getting user input for number of scenarios
param user_scenarios;
print "How many scenarios to be considered (integer)?";
read user_scenarios <- ;
param n_k := round(user_scenarios);
```

Figure 2: Defining parameter n_k via user input within model file

In order to define the range of values to iterate over for the summations shown in the objective function and constraint equations (2), (3), (4), (7), and (8), the following sets are defined, representing factory location, customer, and scenario indices respectively:

```
### Defining Sets ###
set L := {1..n_i} ordered;      # Set of all factory locations
set C := {1..n_j} ordered;      # Set of all customers
set T := {1..n_k} ordered;      # Set of all cases
```

Figure 3: Defining sets for summations within model file

With sets defined, all of the remaining parameters are defined within the model file.

```
### Parameters ###
param s;            # Selling price of a single unit (in $)
param c_p;          # Production cost associate with producing a single unit (in $)
param c_t;          # Transportation cost of single unit per mile (in $ per mile)
param c_f;          # Cost to build a single factory (in $)
param c_u;          # Cost per unit of capacity installed (in $)
param c_total;      # Max total cost allowed for building and stocking factories; maximum investment cost (in $)
param p_min;        # Minimum capacity of factory if built (integer)
param p_max;        # Maximum capacity of factory if built (integer)
param f_min;        # Minimum number of factories that must be built (integer)
param f_max;        # Maximum number of factories that must be built (integer)
param d {i in L, j in C};                              # Distance from factor i to customer j
param u {k in T, j in C} := round(Normal (500, 100));  # Number of units ordered by customer j (col) in set k (row)
```

Figure 4: Defining remaining parameters within model file

With all parameters and necessary sets for summations now defined, the variables x_i, y_i, and z_{ij}^k are now defined within the model file.

```
### Variables ###
var x {i in L} binary;                  # Variables determining whether factory is built (1) or not (0)
var y {i in L} integer;                 # Variables determine capacity of factory i
var z {i in L, j in C, k in T} integer; # Variables determining how many units go to customer j from factory i in scenario k
```

Figure 5: Defining variables x_i, y_i, and z_{ij}^k within model file

With variables now defined, the objective function is now defined within the model file.

```
### Objective Function ###
maximize Total_Profit: (1/n_k)*(sum {k in T, i in L, j in C} ((s - c_p - c_t*d[i, j])*z[i, j, k])) - (sum {i in L} c_f*x[i]) - (sum {i in L} c_u*y[i]);
```

Figure 6: Defining objective function within model file

The only remaining portion of the model file remaining is to define constraint equations (2)−(10), which are defined in the model file as shown below.

```
### Constraints ###
subject to min_Fact: (sum {i in L} x[i]) >= f_min;      # Min number of factories must be >= f_min
subject to max_Fact: (sum {i in L} x[i]) <= f_max;      # Max number of factories must be <= f_max
subject to Total_Cost: c_f*(sum {i in L} x[i]) + (sum {i in L} c_u*y[i]) <= c_total; # Limit total cost for building and stocking factories
subject to Capacity_Limit_L {i in L}: p_min*x[i] <= y[i];                        #If built, must have greater than or equal to p_min units in factory i
subject to Capacity_Limit_H {i in L}: y[i] <= p_max*x[i];                        #If built, must have less than or equal to p_max units in factory i
subject to Meet_Demand {k in T, j in C}: (sum {i in L} z[i, j, k]) = u[k, j];    #Must find how much to give each customer from factory j
subject to Enforce_Dem {k in T, i in L}: (sum {j in C} z[i, j, k]) <= y[i];      #Ensuring demand from a customer j in scenario k is less than capacity
subject to Enforce_NonNeg {k in T, i in L, j in C}: z[i, j, k] >= 0;             #Non-negativity constraint for all z
```

Figure 7: Defining variables x_i, y_i, and z_{ij}^k within model file

Though the parameters have been defined in the model file, with the exception of n_k and u_j^k which are defined with values in the model file, the parameters need to be given numerical values. These values are contained in the data file, entitled Project_Final.dat, and for the full model match the values given in Table I. Additionally, this data file contains the values for d_{ij} which are provided in Appendix A. The instantiation of the parameters with numerical values in shown below except for d_{ij} which is not shown for brevity.

```
### Defining Parameters for Defining Sets ###
param n_i := 10;    # Number of factory locations (integer)
param n_j := 25;    # Number of customers (integer)

### Defining Constant Parameters ###
param s := 125;             # Selling price of a single unit (in $)
param c_p := 5;             # Production cost associate with producing a single unit (in $)
param c_t := 1.50;          # Transportation cost of single unit per mile (in $ per mile)
param c_f := 10000;         # Cost to build a single factory (in $)
param c_u := 5;             # Cost per unit of capacity installed (in $)
param c_total := 250000;    # Max total cost allowed for building and stocking factories; maximum investment cost (in $)
param p_min := 5000;        # Minimum capacity of factory if built (integer)
param p_max := 10000;       # Maximum capacity of factory if built (integer)
param f_min := 4;           # Minimum number of factories that must be built (integer)
param f_max := 8;           # Maximum number of factories that must be built (integer)
```

Figure 8: Parameter instantiation within data file

Lastly, with the model defined and the parameters instantiated with numerical values, the final step is to create a .run file, which for this work is entitled Project_Final.run. The code within this .run file is shown below:

```
## Project Optimization Problem - IOE510 ##

reset;              # Reseting workspace
option randseed 0;  # Setting current seed to random value

### Defining Model and Data and Solving with cplex ###
model Project_Final.mod;            # Defining model
data Project_Final.dat;             # Defining data
option solver cplex;                # Setting solver to cplex
option cplex_options 'mipgap=1e-6'; # Setting optimality gap
solve;                              # Solving model

### Displaying Objective Value and Factory Locations and Capacity ###
display Total_Profit;   # Objective value (expected profit)
display x;              # Factory locations
display y;              # Capacity of each factory
```

Figure 9: Entire .run file as submitted

Initially, the workspace is reset to make sure any model and data is cleared, and the seed for generating random values for the customer demands is set to a random value. Note that this seed is used to create a matrix containing all customer demands. Next, the model and data files are defined, and the solver set to cplex. For the cplex solver, the optimality gap for the cplex algorithm, denoted by 'mipgap' in the code above, is set to the value of 1e-6 (the default value). Lastly, the model is solved. Once the model is solved, the objective function value stored in `Total_Profit` is displayed along with the factory locations and corresponding capacities which are contained in `x` and `y` respectively.

To run the model in AMPL, a user simply needs to type "include Project_Final.run" and press "Enter" within the AMPL console window. They will be prompted for the number of scenarios to consider, and the model and data files are loaded automatically. For the reduced model, separate model and data files are included entitled Project_Final_Reduced.dat and Project_Final_Reduced.run. The only differences between the reduced model .dat file and the full order .dat file is that the values of d_{ij} are limited to only the first 9 customers and all 10 factory locations and n_j shown in Figure 8 is instantiated to be 9 instead of 25. The reduced model .run file then loads the same model file (which can be loaded because it is fully generalized) and the reduced .dat file, with all other lines staying the same. References [3]−[8] were used as primary references for generating all codes.

3 Results and Discussion

3.1 Excel Results - Reduced Model

Using the Excel Solver, the reduced model of 9 potential customers and 10 potential factory locations was solved 5 separate times all using the Simplex LB solver with randomly generated customer demands. All scenarios opened 4 locations with minimum production levels (i.e., minimum capacities) of 5000 for each selected location. The specific locations which were opened and their capacities which were constant for all 5 runs are shown in Table II.

Table II: Factory locations and capacities from Excel Solver

Factory Location	x_i	y_i
1	1	5000
2	1	5000
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	1	5000
9	1	5000
10	0	0

From Table II, it can clearly be seen that only the minimum number of factories needed to be built, and that for those four factories, the minimum capacity was allocated. This stands to reason that likely the cost of opening another factory, which includes both the cost to build and stock the factory, costs more than the transportation cost from a factory that is opened but may be farther away (on average) from all customers considered. Locations 1, 2, 8, and 9 were always selected for each scenario for all iterations. This solutions yields \$110,000 of slack because the maximum investment cost, c_{total}, of \$250,000 was not reached (i.e., the constraint was not binding). Based on of these results alone, this could either be due to reducing the number of customers to 9 so no more factory locations than the minimum f_{min} needed to be opened, or due to the nature of the problem. However, using the results from AMPL later discussed, it will be shown that the number of factories open and their capacity is more a facet of this problem rather than due to the reduced number of customers.

The object function values which are the maximum expected profit for this model averaged \$238,324.75 with a standard deviation of \$21,668.10 for 5 iterations. The objective function values from each run are shown in Table III.

Table III: Expected profit and selected factory locations from Excel Solver

	Expected Profit	Factories Locations
Run 1	$251,512.75	1, 2, 8, 9
Run 2	$249,395.50	1, 2, 8, 9
Run 3	$234,226.75	1, 2, 8, 9
Run 4	$202,153.00	1, 2, 8, 9
Run 5	$254,335.75	1, 2, 8, 9
Mean	$238,324.75	N/A
std.	$21,668.10	N/A

3.2 AMPL Results - Reduced Model

To compare the results from Excel Solver and AMPL, the reduced problem using 10 factory locations and 9 customers was also implemented in AMPL along with the full model. Because the Excel solver could only use a two scenarios to fit within the solver's limitations, the 5 separate runs of the AMPL simulation using only a single case and three cases can be used for direct comparison, though the cases using 1, 3, 100, 1000, and 5000 scenarios are also performed for context. For all AMPL results presented, an optimality gap of 1e-6 is used (i.e., `mipgap` = 1e-6).

From AMPL, as with Excel Solver and for the full model in AMPL (results presented in sub-section 3.3), the locations of the factories and their respective capacities were consistent across all runs and number of scenarios considered. From AMPL, the factory locations and variable outputs from the model are given in Table IV.

Table IV: Factory locations and capacities from AMPL - reduced model

Factory Location	x_i	y_i
1	0	0
2	1	5000
3	1	5000
4	0	0
5	0	0
6	1	5000
7	0	0
8	0	0
9	1	5000
10	0	0

Interestingly, though the parameters values including d_{ij} were consistent between the Excel and AMPL solutions, the number of built factories and their respective capacities match, except they differ in which locations to open. The Excel solver indicated that for all runs to open factories at locations 1, 2, 8, and 9, however from all runs AMPL indicates to open factories at location 2, 3, 6, and 9.

This discrepancy could be due to differences in the solution method used by Excel and AMPL (Excel solver vs. cplex), and/or possibly differences in the optimality gaps. However, though they differ in this aspect, as shown in Table V the expected profit values are very similar and within the same range.

Table V: Expected profit and factory locations from AMPL - small model

	Expected Profit with k Scenarios					Factory Locations
	$k = 1$	$k = 3$	$k = 100$	$k = 1000$	$k = 5000$	
Run 1	$263,525.50	$251,173.00	$250,684.26	$249,476.18	$250,536.20	1, 2, 3, 10
Run 2	$246,742.00	$240,846.50	$253,884.01	$250,051.28	$250,573.22	1, 2, 3, 10
Run 3	$224,162.50	$259,321.00	$250,126.23	$250,660.93	$251,113.80	1, 2, 3, 10
Run 4	$208,423.00	$248,989.50	$249,230.85	$250,983.82	$249,775.14	1, 2, 3, 10
Run 5	$265,541.50	$232,068.00	$249,560.68	$251,508.78	$250,273.40	1, 2, 3, 10
Mean	$241,678.90	$246,479.60	$250,697.21	$250,536.20	$250,454.35	N/A
std.	$24,925.45	$10,400.18	$1,865.73	$793.82	$487.28	N/A

For two scenarios (i.e., $k = 2$) the Excel results have a mean value of $238,324.75 while the AMPL results for $k = 1$ have a mean value of $ 241,678.90, which are very close especially given that so few scenarios are considered and each run uses an independently generated set of normally distributed random values. Similarly, the standard deviation from the Excel results is $21,668.10 and the standard deviation from AMPL is $24,925.45, both of which are of the same order of magnitude. Additionally, though not comparable to the Excel model directly, as the number of scenarios considered in the sample average approximation increases for this reduced model, the objection function values from each iterations are closer to the mean as indicated by the decreasing standard deviation as k is increased. This is expected given that as the number of scenarios is increased (assuming equal weighting of all scenarios), the effects of outliers within the randomly generated data is minimized and the averaged results converge to a mean value.

Even with the discrepancy in the factory locations, because the capacity of the opened factories in identical between the solvers, and the expected profit values from 5 separate runs are of the same order of magnitude and show similar mean and standard deviation, the model appears to be implemented correctly in both methods and shows that using either method, one can obtain sample average results of similar nature and order of magnitude with either program.

3.3 AMPL Results - Full Model

The full model using the parameters given in Table I is now considered using 10 factory locations and 25 customers and is implemented in AMPL. The results presented consists of 5 separate iterations of the AMPL model using 1, 3, 100, 1000, and 5000 scenarios. For all AMPL results presented, an optimality gap of 1e-6 is used (i.e., `mipgap` = 1e-6).

As with the Excel and reduced AMPL results, the locations of the factories and their respective capacities were consistent across all runs and number of scenarios considered. From AMPL, the factory locations and variable outputs from the model are given in Table VI.

Table VI: Factory locations and capacities from AMPL - full model

Factory Location	x_i	y_i
1	1	5000
2	1	5000
3	1	5000
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	1	5000

The expect profits (i.e., objective function values) from running the model using 1, 3, 100, 1000, and 5000 scenarios 5 independent times are presented in Table VII. The mean and standard deviations associated with each number of scenarios are provided as well in Table VII.

Table VII: Expected profit and factory locations from AMPL - full model

	Expected Profit with k Scenarios					**Factory Locations**
	$k = 1$	$k = 3$	$k = 100$	$k = 1000$	$k = 5000$	
Run 1	\$937,013.50	\$938,028.00	\$934,927.92	\$934,037.89	\$935,327.48	1, 2, 3, 10
Run 2	\$948,358.00	\$915,106.50	\$924,898.35	\$936,516.79	\$934,851.16	1, 2, 3, 10
Run 3	\$928,712.50	\$949,542.00	\$937,781.59	\$936,238.29	\$935,763.67	1, 2, 3, 10
Run 4	\$916,342.00	\$922,640.50	\$928,014.03	\$934,259.12	\$935,438.47	1, 2, 3, 10
Run 5	\$910,004.50	\$944,902.00	\$934,803.31	\$935,585.32	\$935,289.32	1, 2, 3, 10
Mean	\$928,086.10	\$934,043.80	\$930,746.69	\$935,327.48	\$935,334.02	N/A
std.	\$15,458.89	\$14,685.24	\$9,631.28	\$1,130.81	\$328.14	N/A

Using these results, it is clear that that as more scenarios are included in the sample average approximation, the closer the expected profits are among runs, and hence the distribution of the expected profits are closer to the mean. This is demonstrated by the standard deviation which decreases as the number of scenarios increases. As with the results presented for the reduced model, this is expected given that as the number of scenarios is increased (assuming equal weighting of all scenarios), the effects of outliers within the randomly generated data is minimized and the averaged results converge to a mean value. However, it should be noted that as the number of scenarios increases, even though the standard deviation decreases, the time to reach a solution within the default cplex optimality gap within AMPL (1e-6) drastically increases. This can be attributed to needing to calculate the variables associated with z_{ij}^k, which for the largest model using 5000 scenarios accounts for 1,250,000 variables alone.

4 Conclusion

From this project, two solvers were used to solve a two-stage stochastic optimization problem with sample average approximation. Using Excel Solver, limitations in the solver variable capacity meant only a reduced model of the problem could be solved, which lead to a high variation in the profit estimation. When using the AMPL solver, such limitations did not exist, which allowed for several iterations of the model to be considered from 1 up to 5000 customer demand scenarios, which minimized the variation in our profit estimation significantly compared to our Excel Solver profit variation. However, running the model for the higher simulations drastically increased the time needed for the solver to compute a optimal solution. This highlights the trade-off between model run time and estimation accuracy when determining the ”right” number of scenarios to run of a sample average approximation model.

References

[1] "Optimization Methods in Management Science/Operations Research - Excel Techniques." MIT. https://ocw.mit.edu/courses/sloan-school-of-management/15-053-optimization-methods-in-management-science-spring-2013/tutorials/MIT15_053S13_tut03.pdf

[2] University of Michigan Department of Industrial and Operations Engineering. *IOE510 - Project Assignment Handout*. Fall 2019.

[3] Fourer, Robert, and Brain W Kernighan. *"Production Models: Maximizing Profits."* AMPL: A Modeling Language for Mathematical Programming, by David M Gay, 2nd ed., AMPL, 2003. ISBN 0-534-38809-4

[4] Fourer, Robert, and Brain W Kernighan. *"Simple Sets and Indexing"* AMPL: A Modeling Language for Mathematical Programming, by David M Gay, 2nd ed., AMPL, 2003. ISBN 0-534-38809-4

[5] Fourer, Robert, and Brain W Kernighan. *"Parameters and Expressions"* AMPL: A Modeling Language for Mathematical Programming, by David M Gay, 2nd ed., AMPL, 2003. ISBN 0-534-38809-4

[6] Fourer, Robert, and Brain W Kernighan. *"Linear Programs: Variables, Objectives and Constraints"* AMPL: A Modeling Language for Mathematical Programming, by David M Gay, 2nd ed., AMPL, 2003. ISBN 0-534-38809-4

[7] Fourer, Robert, and Brain W Kernighan. *"Display Commands"* AMPL: A Modeling Language for Mathematical Programming, by David M Gay, 2nd ed., AMPL, 2003. ISBN 0-534-38809-4

[8] Fourer, Robert, and Brain W Kernighan. *"AMPL Reference Manual"* AMPL: A Modeling Language for Mathematical Programming, by David M Gay, 2nd ed., AMPL, 2003. ISBN 0-534-38809-4

Appendix - Table of Distances from Factory Locations to Customers

Table A1: Customer distances d_{ij} for full model (in miles)

Customer j	Factory Location i									
	1	2	3	4	5	6	7	8	9	10
1	87	20	96	85	84	76	43	69	78	33
2	92	70	19	37	33	28	40	40	24	94
3	97	94	54	37	100	63	57	70	27	100
4	93	94	16	17	52	32	26	89	30	60
5	35	41	49	89	66	17	47	86	58	16
6	67	26	73	62	28	70	49	76	69	99
7	33	84	36	99	37	89	32	29	33	52
8	59	94	31	55	87	89	29	56	24	57
9	76	24	23	22	76	17	83	54	20	41
10	24	14	19	13	17	74	100	48	81	31
11	58	48	68	87	72	44	91	91	94	50
12	11	99	29	27	28	88	14	95	36	59
13	43	70	45	95	100	60	52	42	61	46
14	92	80	86	97	33	66	92	99	75	31
15	20	97	65	49	98	41	82	49	80	70
16	23	62	80	60	40	36	79	80	70	23
17	57	20	66	69	37	66	86	18	48	94
18	82	21	35	47	77	39	90	80	28	26
19	11	41	84	51	42	14	38	91	16	40
20	59	66	23	84	23	20	15	34	30	16
21	19	11	77	34	71	66	31	30	73	13
22	47	23	95	69	83	85	10	83	66	29
23	68	82	13	86	74	83	91	98	99	91
24	43	44	75	69	41	40	69	69	68	10
25	13	16	52	37	60	37	32	63	26	39