BUSN-6051 - Assignment 4 - Orion Foods

Shawn Berney

July 30, 2018

Orion Foods - Transport Decisions

Orion Foods distribution network was analysed in an effort to optimize operations and identify possible cost savings for the company. This analysis includes an evaluation of the current transport costs, carrying costs, as well as the location the companies distribution centres. Currently the company holds distribution centres in Burns, OR and Fresno, CA.

The first section deals with evaluating current operations. This report identifies opportunities to save money through providing specified routes to the transport company. The first section also evaluates the financial impact of changing the distribution channel for Salt Lake City. An alternative distribution centre location was also identified.

The second section in this report discusses the expansion of the Burns, Oregon warehouse. This section provides interpretation for the operational evaluation and identifies the optimal scenario given the current warehouses and transportation options.

Finally, this report looks at the costs associated with consolidating warehouses into a single location in Reno, Nevada. This consolidation was evaluated on the basis of load costs, carrying costs, and capital costs over the 5 year period of growth. A deeper analysis may be required to evaluate the financial return for consolidation. This analysis should include the value of investing the additional $1.7 million capital against the operational efficiency gained through consolidation. This report views the project strictly on the basis of total cost, including capital, spread over a five year period and does not include financial calculations such as net present value and required rate of return. There are certainly operational savings that can be realized through consolidation at Reno, Nevada. The recommendations of this report, however, is that consolidation would increase total scenario costs over a 5 year period. As a result of these increased costs, consolidation should not be undertaken.

Methodology

The cost of capital expansion was calculated as part of the total scenario cost. An important note here is that the total scenario costs are based on the linear growth of volume over the five year period. As such, the first year calculated will utilize the current load capacity (in cwt.) plus the one fifth of the difference between the current capacity and the projected capacity.

The calculations provided below were generated by modelling the requirements using Python. This model utilizes the X and Y coordinates of each city, combined with the driving distance of each route to find the shortest path - these shortest paths are then highlighted on the map and added to the data model. Once the shortest path has been identified for each route, a graph is generated. The graph also incorporates the center of gravity for all field warehouse routes submitted to the graph unless explicitly suppressed (`suppress_cog="Y"`). The data is then used for calculating costs by iterating over each year. Each iteration increases volume at each destination in a linear fashion. In other words, the linear growth at each warehouse is used to calculate shipping and carrying costs for each of the 5 years.

When calculating the costs, specific functionality has been added to; 1) limit the cost calculations to a specific year, and 2) pass in a parameter that will reduce the carrying cost percentage for consolidated cost savings at Reno, Nevada. Finally, costs are passed to another python method and summarized. Due to the extensive use of code for this assignment, in-line references and printout of results have not been provided for each step. A full copy of the data files, source code (classes) and a sample of class uses has been provided in the appendix.

0.1 Improving Current Distribution Operations

We began this assignment by determining the optimal route for Burns and Fresno warehouses. Initially, Excel was used to calculate the shortest route using Solver. Unfortunately, Solver was unable to handle the large number of 'edges' (or bidirectional routes) required for the entire dataset. Excel Solver is limited to 200 variables. As such, the dataset was loaded into Python and the NetworkX library was used to dynamically graph the locations, add labels, and calculate shortest routes.

Table 1 includes the calculations for current and future costs associated with all existing transportation routes if the shortest roads were consistently taken.

The cost is calculated by the following formulas: $Cost = \frac{L*D}{load} * r$

Where:

$$L = \text{Load of yearly goods in U.S. Short carat-weight (cwt.)}$$

$$D = \text{Distance in miles from distribution centre}$$

$$L * D = \text{Load-Distance}$$

$$c = \text{current year}$$

$$p = \text{5 year projected}$$

$$r = \$1.30 = \text{rate for hauling average load weight } (load) \text{ of goods 1 mile distance}$$

$$load = 300cwt. = \text{Average load weight of typical shipment}$$

The following graph with the optimal routes was generated:

Table 1: Optimal Travel Distances

	location	distance
1	Burns to Portland	293
2	Burns to Butte	676
3	Burns to Seattle	467
4	Fresno to Los Angeles	219
5	Fresno to Phoenix	588
6	Fresno to Salt Lake City	815
7	Fresno to San Francisco	183

In Table 2, the total load-distance was calculated using the optimal routes so that a comparison of costs can be undertaken. This allows us to determine if the shipping company is currently using the optimal route, as well as allows for scenario comparison between different distribution centre locations.

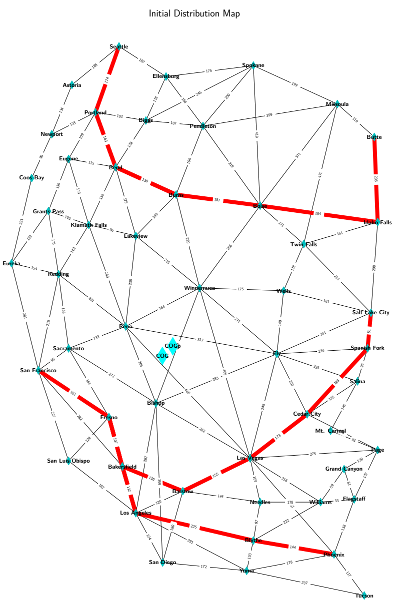


Figure 1: [Above] Optimal Routes for Orion distribution centres

By dividing Load-Distance by average shipment weight, we can arrive at the total travel distance required. This is then multiplied by the rate per mile to attain total load cost.

Cost Savings from routes and warehouse locations

The result of the shortest path calculations suggest that the transportation company is not utilizing the shortest routes. The total annual transport cost provided for current operations was $652,274. The amount is $22,216 higher than the amount that would have been accrued had the optimal routes identified above been used ($630,058).

In figure 1 you will notice that combined center of gravity calculated location for all warehouses does fall near the Reno location. We recalculate the ideal warehouse location using the center of gravity technique. This technique indicates that Burns and Fresno are not currently the ideal location given the current distribution channels.

You can see the individual calculation in figures 2 and 3. The Burns distribution center ideal location would be better suited in Ellensburg (x = 4.07596, y = 18.94904). Ellensburg would result in a current scenario load cost of $524,324.67, a substantial cost savings from Burns ($1,193,677.33)

The Fresno warehouse would be best located for this scenario in Barstow (x = 5.72976, y = 5.18408). A comparison of load costs revealed that altering the location to Barstow would result in total scenario load costs of $2,405,736.67, better than both Fresno ($2,755,995.67) and Bishop ($2,908,342.67) if shortest travel routes are used.

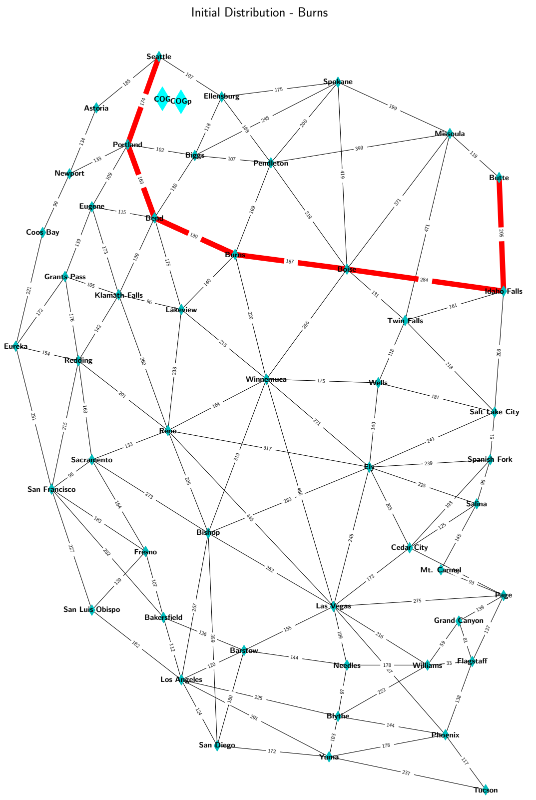


Figure 2: [Above] Burns Distributions Centre

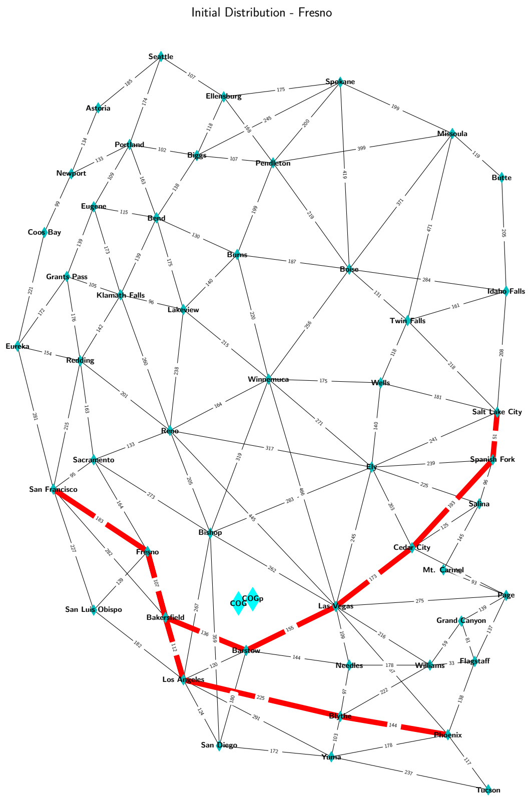


Figure 3: [Above] Fresno Distributions Centre

The total annual transport cost provided for current operations was $652,274. The amount is $22,216 higher than the amount calculated using the optimal routes identified above. With the revised routes, the newly calculated total load cost and total carry costs over five years can be seen in Table 3.

Table 2: Optimal Travel Costs based on Shortest Route

	location	distance	$L_c * D$	$L_p * D$	$Cost_c$	$Cost_p$
1	Burns to Portland	293	12,599,000	16,701,000	54595.67	72371.00
2	Burns to Butte	676	3,380,000	10,140,000	14646.67	43940.00
3	Burns to Seattle	467	26,152,000	36,893,000	113325.33	159869.67
4	Fresno to Los Angeles	219	24,090,000	28,908,000	104390.00	125268.00
5	Fresno to Phoenix	588	35,280,000	49,392,000	152880.00	214032.00
6	Fresno to Salt Lake City	815	28,525,000	45,640,000	123608.33	197773.33
7	Fresno to San Francisco	183	15,372,000	19,215,000	66612.00	83265.00
		3,241	145,398,000	206,889,000	630,058.00	896,519.00

Table 3: Total Cost Calculations for Base Scenario (Optimized Travel Route)

Route	Total Load Costs ($)	Total Carry Costs ($)
Burns to Portland	326,304.33	674,625.00
Burns to Butte	161,113.33	144,375.00
Burns to Seattle	706,259.67	916,125.00
Fresno to Los Angeles	584,584.00	1,617,000.00
Fresno to Phoenix	947,856.00	976,500.00
Fresno to Salt Lake City	840,536.67	624,750.00
Fresno to San Francisco	383,019.00	1,267,875.00

Total Scenario Load Cost = 3,949,673.00

Total Scenario Carrying Cost = 6,221,250.00

Capital Costs Required = $300,000 (upgrade to Burns warehouse to meet projected capacity)

Total Scenario Cost = 10,470,923.00

Cost savings from altering distribution channels

Load-cost savings can be realized if Salt Lake City was served from the Burns warehouse instead of Fresno. The following calculation identifies the total cost savings for at the projected volume in year 5.

$$Cost = \frac{L * D}{load} * r$$

$$Cost = \frac{56000 * 279}{300} * 1.3$$

$$Cost = \$67,704$$

Fresno is 279 miles further away from Salt Lake City than the Burns distribution centre. Given the projected volume of 56,000 cwt. in 5 years, this works out to substantial added costs on a per year basis.

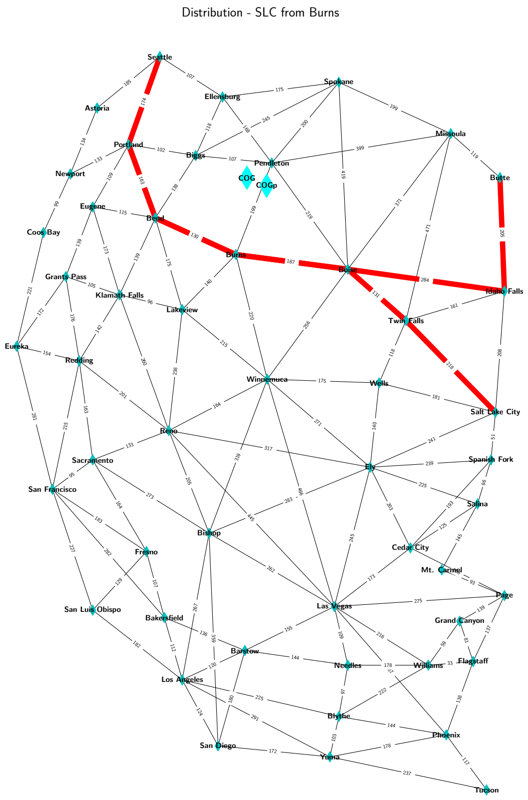


Figure 4: [Above] Burns distribution channels with Salt Lake

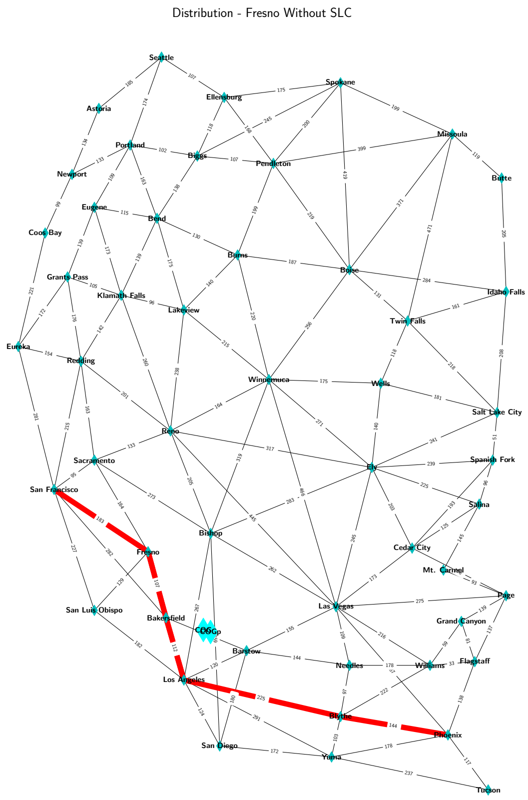


Figure 5: [Above] Fresno Distributions Centre after Salt City changed to Burns warehouse

Table 4: Travel Costs for distribution to Salt Lake City from Burns warehouse

	location	distance	$L_c * D_m$	$L_p * D_m$	$Cost_c$	$Cost_p$
1	Burns to Portland	293	12599000	16701000	54,595.67	72,371.00
2	Burns to Butte	676	3380000	10140000	14,646.67	43,940.00
3	Burns to Seattle	467	26152000	36893000	113,325.33	159,869.67
4	Burns to Salt Lake City	536	18760000	30016000	81,293.33	130,069.33
5	Fresno to Los Angeles	219	24090000	28908000	104,390.00	125,268.00
6	Fresno to Phoenix	588	35280000	49392000	152,880.00	214,032.00
7	Fresno to San Francisco	183	15372000	19215000	66,612.00	83,265.00
		2,962	135,633,000	191,265,000	587,743.00	828,815.00

After altering the distribution center for Salt Lake City, both warehouse locations are better suited to serve their regional centers. Although the Burns warehouse may still be better served from Pendleton, and the Fresno warehouse from Bakersfield or Barstow, their is a definite improvement and cost savings. In order to accomplish this, the existing infrastructure as Burns would need to be upgraded. As the upgrade is costly working out to \$300,000 for an additional 80,000 cwt. yearly (\$300,000 for 10,000 cwt. capacity with a turnover of 8 / year).

A single increment of 10,000 cwt. upgrade would provide just short of the required capacity at year 5. If the growth is spread evenly over the 5 year period, this would result in a yearly growth at burns (including the Salt Lake City distribution) of 13,600 cwt. The shortfall of 7,000 cwt in year 5 could be shipped from the extra capacity currently available at the Fresno warehouse.

Baseline Scenario

The lowest operational cost that can be found using the Burns and Fresno warehouse will be our baseline scenario moving forward.

Total Scenario Load Cost = 3,661,931.00 + $8,463

Total Scenario Carrying Cost = 6,221,250.00

Capital Costs = $300,000

Total Scenario Costs = 10,191,644.00

Fresno to Salt Lake City (S.L.C.) - Year 5, 7000 cwt.

$$Cost = \frac{L * D}{load} * r \tag{1}$$

$$Cost = \frac{7000 * 279}{300} * 1.3 \tag{2}$$

$$Cost = \$8,463 \tag{3}$$

0.2 Benefits to expanding Burns, Oregon warehouse

The expansion of warehouse capacity will be necessary in order to accommodate the projected growth. The current combined capacity of the warehouses can serve 520,000 cwt of goods in one year. This is accomplished through having a turnover rate of 8, resulting in 400,000 cwt annual capacity at Fresno and 120,000 cwt annual capacity at Burns.

Currently, Burns does not have sufficient capacity to meet projected annual capacity over the next 5 years. The expansion of the Burns location with 10,000 cwt upgrade for $300,000 would result in the ability to meet future demand for existing distribution channels.

Expanding the Burns warehouse will allow the company to alter the distribution channels used and serve Salt Lake City from Burns, Oregon. The initial scenario - our scenario with only optimized routes - costed $10,470,923.00 over a 5 year period if we assume that the growth in load volume is linear. With the upgrade to Burns, Oregon warehouse and using this capacity to serve Salt Lake City, the total scenario costs are reduced to $10,191,644. This includes upgrading the Burns facility with only the minimum upgrade and serving the remaining shortfall from Fresno when capacity is reached in year 5.

If we refer back to Figure 4., we can see that with the inclusion of Salt Lake City as part of the Burns distribution channel, the calculated centre of gravity for both Fresno and Burns is more closely aligned with the warehouse location. This results in lower load-costs and substantial savings over time. In summary, there are financial benefits to expanding the Burns warehouse.

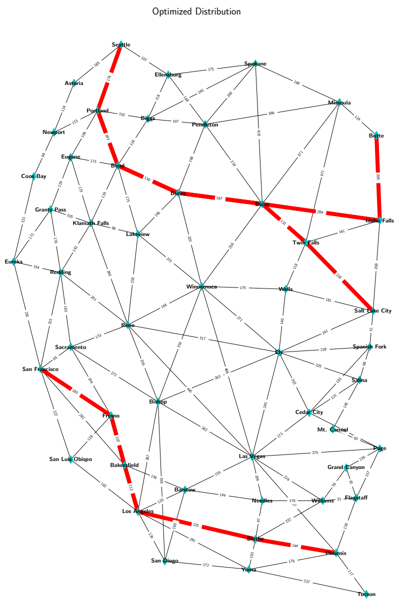
Table 5: Optimized distribution channels (baseline)

Distribution Channels	Load Cost	Additional Costs	Carrying Cost	Capital Cost	TOTAL
Original	$3,949,673	$0	$6,221,250	$300,000	10,470,923
Optimized (Burns & Fresno)	$3,661,931	$8,463	$6,221,250	$300,000	10,191,644

0.3 Consolidating regional warehouse operations at Reno

Utilizing the model generated above, we can simply enter in the paths for Reno to each of the various locations. The model built in python will automatically calculate out the shortest route, calculate costs for load and carrying over the five year period. To account for the carrying cost reduction, the model was updated with a multiplier of .6 (a reduction of .4 or 40% from the original value) that is applied to the carrying cost percent. This value is set by passing the parameter "consolidation='Reno'" within the calling method.

The result of this calculation is that the total scenario costs for consolidating warehouse operations at Reno, NV are greater than the costs associated with the optimized routes and distribution channels.



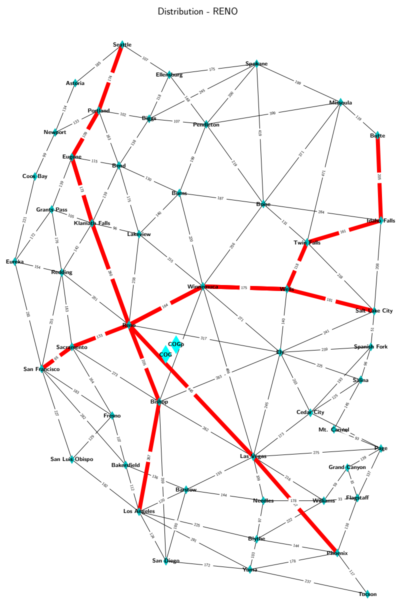


Table 6: Analysis of Reno Consolidation

	Load Cost	Carrying Cost	Capital Cost	TOTAL
Optimized - Burns & Fresno	$3,670,394	$6,221,250	$300,000	10,191,644
Consolidation at Reno	$5,335,993	$3,732,750	$2,000,000	11,068,742

0.4 Recommendation

Although there are many limitations and assumptions within the analysis provided here, the scenario comparison provided in table 5 demonstrates that optimization and expansion of existing distribution channels will result in lower total cost over a 5 year period than undertaking consolidation at Reno.

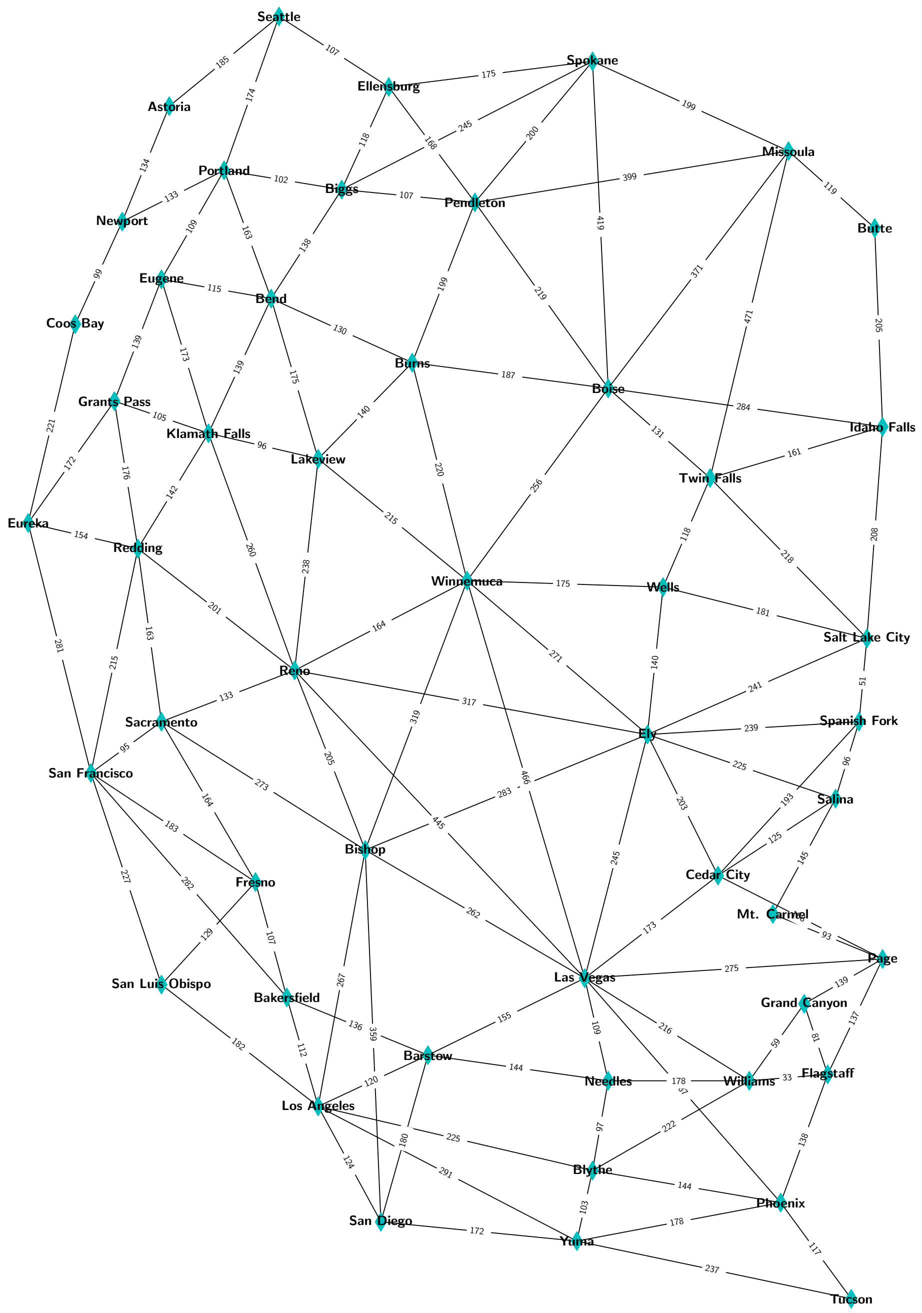
Although operational costs are lower overall from the Reno location in both year-one ($130,654) and year-five ($192,840) as a result of substantially reduced carrying costs, the increased load-costs and large amount of capital required to undertake this project results in the return on investment associated with this project far into the future. Further financial analysis as to the required rate of return and net present value of the project should be undertaken to evaluate costs and benefits more closely.

Table 7: Year 5 - Operational Cost Comparison

	Load Cost	Additional Costs	Carrying Cost	TOTAL
Optimized - Burns & Fresno	$828,815	$8,463	$1,386,000	$2,223,278
Consolidation at Reno	$1,198,838	$0	$831,600	$2,030,438

Appendix

Map - No Routes



City Coordinates

City	x	y
Astoria	2.60	18.70
Bakersfield	4.10	4.80
Barstow	5.90	3.90
Bend	3.90	15.70
Biggs	4.80	17.40
Bishop	5.10	7.10
Blythe	8.00	2.10
Boise	8.20	14.30
Burns	5.70	14.70
Butte	11.60	16.80
Cedar City	9.60	6.70
Coos Bay	1.40	15.30
Ellensburg	5.40	19.00
Ely	8.70	8.90
Eugene	2.50	16.00
Eureka	0.80	12.20
Flagstaff	11.00	3.60
Fresno	3.70	6.60
Grand Canyon	10.70	4.70
Grants Pass	1.90	14.10
Idaho Falls	11.70	13.70
Klamath Falls	3.10	13.60
Lakeview	4.50	13.20
Las Vegas	7.90	5.10
Los Angeles	4.50	3.10
Missoula	10.50	18.00
Mt. Carmel	10.30	6.10
Needles	8.20	3.50
Newport	2.00	16.90
Page	11.70	5.40
Pendleton	6.50	17.20
Phoenix	10.40	1.60
Portland	3.30	17.70
Redding	2.20	11.80
Reno	4.20	9.90
Sacramento	2.50	9.10
Salina	11.10	7.90
Salt Lake City	11.50	10.40
San Diego	5.30	1.30
San Francisco	1.60	8.30
San Luis Obispo	2.50	5.00
Seattle	4.00	20.10
Spanish Fork	11.40	9.10
Spokane	8.00	19.40
Tucson	11.30	0.10
Twin Falls	9.50	12.90
Wells	8.90	11.20
Williams	10.00	3.50
Winnemuca	6.40	11.30
Yuma	7.80	1.00

City Distances

From	To	Distance
Astoria	Seattle	185
Astoria	Newport	134
Bakersfield	Barstow	136
Bakersfield	Los Angeles	112
Bakersfield	San Francisco	282
Barstow	Las Vegas	155
Barstow	Needles	144
Barstow	San Diego	180
Barstow	Los Angeles	120
Barstow	Bakersfield	136
Bend	Biggs	138
Bend	Portland	163
Bend	Eugene	115
Bend	Klamath Falls	139
Bend	Lakeview	175
Bend	Burns	130
Biggs	Ellensburg	118
Biggs	Portland	102
Biggs	Bend	138
Biggs	Pendleton	107
Biggs	Spokane	245
Bishop	Las Vegas	262
Bishop	San Diego	359
Bishop	Los Angeles	267
Bishop	Sacramento	273
Bishop	Reno	205
Bishop	Winnemuca	319
Bishop	Ely	283
Blythe	Yuma	103
Blythe	Los Angeles	225
Blythe	Phoenix	144
Blythe	Williams	222
Blythe	Needles	97
Boise	Burns	187
Boise	Twin Falls	131
Boise	Winnemuca	256
Boise	Idaho Falls	284
Boise	Missoula	371
Boise	Spokane	419
Boise	Pendleton	219
Burns	Bend	130
Burns	Boise	187
Burns	Winnemuca	220
Burns	Lakeview	140
Burns	Pendleton	199
Butte	Missoula	119
Butte	Idaho Falls	205
Cedar City	Spanish Fork	193
Cedar City	Salina	125
Cedar City	Page	158
Cedar City	Las Vegas	173
Cedar City	Ely	203
Coos Bay	Newport	99
Coos Bay	Eureka	221
Ellensburg	Seattle	107

Ellensburg	Biggs	118
Ellensburg	Pendleton	168
Ellensburg	Spokane	175
Ely	Wells	140
Ely	Salt Lake City	241
Ely	Spanish Fork	239
Ely	Salina	225
Ely	Cedar City	203
Ely	Las Vegas	245
Ely	Bishop	283
Ely	Reno	317
Ely	Winnemuca	271
Eugene	Portland	109
Eugene	Bend	115
Eugene	Klamath Falls	173
Eugene	Grants Pass	139
Eureka	Grants Pass	172
Eureka	Coos Bay	221
Eureka	San Francisco	281
Eureka	Redding	154
Flagstaff	Page	137
Flagstaff	Grand Canyon	81
Flagstaff	Williams	33
Flagstaff	Phoenix	138
Fresno	Sacramento	164
Fresno	San Francisco	183
Fresno	Bakersfield	107
Fresno	San Luis Obispo	129
Grand Canyon	Page	139
Grand Canyon	Flagstaff	81
Grand Canyon	Williams	59
Grants Pass	Klamath Falls	105
Grants Pass	Redding	176
Grants Pass	Eureka	172
Grants Pass	Eugene	139
Idaho Falls	Butte	205
Idaho Falls	Boise	284
Idaho Falls	Twin Falls	161
Idaho Falls	Salt Lake City	208
Klamath Falls	Grants Pass	105
Klamath Falls	Redding	142
Klamath Falls	Reno	260
Klamath Falls	Lakeview	96
Klamath Falls	Bend	139
Klamath Falls	Eugene	173
Lakeview	Klamath Falls	96
Lakeview	Bend	175
Lakeview	Burns	140
Lakeview	Winnemuca	215
Lakeview	Reno	238
Las Vegas	Ely	245
Las Vegas	Cedar City	173
Las Vegas	Page	275
Las Vegas	Williams	216
Las Vegas	Phoenix	287
Las Vegas	Needles	109
Las Vegas	Barstow	155
Las Vegas	Bishop	262

Las Vegas	Reno	445
Las Vegas	Winnemuca	466
Los Angeles	Barstow	120
Los Angeles	Bishop	267
Los Angeles	Bakersfield	112
Los Angeles	San Luis Obispo	182
Los Angeles	San Diego	124
Los Angeles	Blythe	225
Los Angeles	Yuma	291
Missoula	Spokane	199
Missoula	Pendleton	399
Missoula	Boise	371
Missoula	Twin Falls	471
Missoula	Butte	119
Mt. Carmel	Page	93
Mt. Carmel	Salina	145
Needles	Las Vegas	109
Needles	Williams	178
Needles	Blythe	97
Needles	Barstow	144
Newport	Coos Bay	99
Newport	Astoria	134
Newport	Portland	133
Page	Mt. Carmel	93
Page	Cedar City	158
Page	Las Vegas	275
Page	Grand Canyon	139
Page	Flagstaff	137
Pendleton	Spokane	200
Pendleton	Missoula	399
Pendleton	Boise	219
Pendleton	Burns	199
Pendleton	Biggs	107
Pendleton	Ellensburg	168
Phoenix	Blythe	144
Phoenix	Yuma	178
Phoenix	Tucson	117
Phoenix	Flagstaff	138
Phoenix	Las Vegas	287
Portland	Biggs	102
Portland	Seattle	174
Portland	Newport	133
Portland	Eugene	109
Portland	Bend	163
Redding	Klamath Falls	142
Redding	Grants Pass	176
Redding	Eureka	154
Redding	San Francisco	215
Redding	Sacramento	163
Redding	Reno	201
Reno	Lakeview	238
Reno	Klamath Falls	260
Reno	Redding	201
Reno	Sacramento	133
Reno	Bishop	205
Reno	Las Vegas	445
Reno	Ely	317
Reno	Winnemuca	164

Sacramento	Reno	133
Sacramento	Redding	163
Sacramento	San Francisco	95
Sacramento	Bishop	273
Salina	Spanish Fork	96
Salina	Ely	225
Salina	Cedar City	125
Salina	Mt. Carmel	145
Salt Lake City	Ely	241
Salt Lake City	Wells	181
Salt Lake City	Twin Falls	218
Salt Lake City	Spanish Fork	51
Salt Lake City	Idaho Falls	208
San Diego	Los Angeles	124
San Diego	Barstow	180
San Diego	Bishop	359
San Diego	Yuma	172
San Francisco	Sacramento	95
San Francisco	Redding	215
San Francisco	Eureka	281
San Francisco	San Luis Obispo	227
San Francisco	Bakersfield	282
San Luis Obispo	Los Angeles	182
San Luis Obispo	San Francisco	227
Seattle	Astoria	185
Seattle	Portland	174
Seattle	Ellensburg	107
Spanish Fork	Salt Lake City	51
Spanish Fork	Ely	239
Spanish Fork	Cedar City	193
Spanish Fork	Salina	96
Spokane	Ellensburg	175
Spokane	Pendleton	200
Spokane	Boise	419
Spokane	Missoula	199
Tucson	Phoenix	117
Tucson	Yuma	237
Twin Falls	Boise	131
Twin Falls	Idaho Falls	161
Twin Falls	Wells	118
Twin Falls	Missoula	471
Twin Falls	Salt Lake City	218
Wells	Twin Falls	118
Wells	Salt Lake City	181
Wells	Ely	140
Wells	Winnemuca	175
Williams	Grand Canyon	59
Williams	Flagstaff	33
Williams	Blythe	222
Williams	Needles	178
Williams	Las Vegas	216
Winnemuca	Lakeview	215
Winnemuca	Burns	220
Winnemuca	Boise	256
Winnemuca	Wells	175
Winnemuca	Ely	271
Winnemuca	Las Vegas	466
Winnemuca	Bishop	319

Winnemuca	Reno	164
Yuma	San Diego	172
Yuma	Blythe	103
Yuma	Phoenix	178
Yuma	Tucson	237

Call Method Examples - Scenario 1: Alternative Location

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jul 29 14:31:02 2018

@author: Shawn

Scenario 1 - Comparison of Warehouse Locations (FRESNO)
"""

import network_classes

pths = [
    ['Barstow', 'Salt Lake City', '35000', '56000'],
    ['Barstow', 'Los Angeles', '110000', '132000'],
    ['Barstow', 'Phoenix', '60000', '84000'],
    ['Barstow', 'San Francisco', '84000', '105000']
]

sbg = sbGraph()
length = sbg.getNetworkGraph(pths, 'Optimized Distribution Map - Barstow Alternative',
    suppress_cog='Y')

costs = sbg.getCosts(length)
getSumCosts = sbg.getSumCosts(costs, pths)

pths = [
    ['Bishop', 'Salt Lake City', '35000', '56000'],
    ['Bishop', 'Los Angeles', '110000', '132000'],
    ['Bishop', 'Phoenix', '60000', '84000'],
    ['Bishop', 'San Francisco', '84000', '105000']
]

sbg = sbGraph()
length = sbg.getNetworkGraph(pths, 'Optimized Distribution Map - Bishiop Alternative',
    suppress_cog='Y')

costs = sbg.getCosts(length)
getSumCosts = sbg.getSumCosts(costs, pths)

# Determine path & draw on graph for Fresno  Distribution
# During iteration, calculate distances, Load-Distances, and Costs.
pths = [
        ['Fresno', 'Los Angeles', '110000', '132000'],
        ['Fresno', 'Phoenix', '60000', '84000'],
        ['Fresno', 'Salt Lake City', '35000', '56000'],
        ['Fresno', 'San Francisco', '84000', '105000']
]

sbg = sbGraph()
length = sbg.getNetworkGraph(pths, 'Initial Distribution - Fresno')
costs = sbg.getCosts(length)
getSumCosts = sbg.getSumCosts(costs, pths)
```

Call Method Examples - Reno: Year 5 Comparison

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jul 25 10:40:44 2018

@author: Shawn

Original Dataset:
```

```
pths = [
    ['Burns', 'Portland', '43000', '57000'],
    ['Burns', 'Butte', '5000', '15000'],
    ['Burns', 'Seattle', '56000', '79000'],
    ['Fresno', 'Los Angeles', '110000', '132000'],
    ['Fresno', 'Phoenix', '60000', '84000'],
    ['Fresno', 'Salt Lake City', '35000', '56000'],
    ['Fresno', 'San Francisco', '84000', '105000']
]

Reno Site Evaluation - Year 5 Comparison to optimized scenario
"""

import network_classes

pths = [
    ['Reno', 'Portland', '43000', '57000'],
    ['Reno', 'Butte', '5000', '15000'],
    ['Reno', 'Seattle', '56000', '79000'],
    ['Reno', 'Los Angeles', '110000', '132000'],
    ['Reno', 'Phoenix', '60000', '84000'],
    ['Reno', 'Salt Lake City', '35000', '56000'],
    ['Reno', 'San Francisco', '84000', '105000']
]
sbg = sbGraph()
length = sbg.getNetworkGraph(pths, 'Distribution - RENO')
costs = sbg.getCosts(length, consolidation='Reno', years=[5])
print(costs)
sbg.getSumCosts(costs, pths)

pths = [
    ['Burns', 'Portland', '43000', '57000'],
    ['Burns', 'Butte', '5000', '15000'],
    ['Burns', 'Seattle', '56000', '79000'],
    ['Burns', 'Salt Lake City', '35000', '56000'],
    ['Fresno', 'Los Angeles', '110000', '132000'],
    ['Fresno', 'Phoenix', '60000', '84000'],
    ['Fresno', 'San Francisco', '84000', '105000']
]
sbg = sbGraph()
length = sbg.getNetworkGraph(pths, 'Optimized Distribution', suppress_cog="Y")
costs = sbg.getCosts(length, years=[5])
print(costs)
sbg.getSumCosts(costs, pths)
```

Class Methods in Python

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 26 21:26:54 2018

@author: Shawn
"""

import itertools
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt

class sbGraph:
    __instance = None
    def __init__(self):
        if not sbGraph.__instance:
            print("Initiated")
        else:
            print("Instance already created:", self.getInstance())

    @classmethod
    def getInstance(cls):
        if not cls.__instance:
            cls.__instance = sbGraph()
            self.COG_x = 0
            self.COG_y = 0
```

```
            self.COGp_x = 0
            self.COGp_y = 0
            self.COG = []

        return cls.__instance

    @classmethod
    def getNetworkGraph(self, pths, graph_name, **keyword_param):

        edgelist = pd.read_csv(
                'C:/Users/Shawn/SkyDrive/Documents/Learning/BUSN_6051/Assign_4/City_Distances.csv')
        nodelist = pd.read_csv(
                'C:/Users/Shawn/SkyDrive/Documents/Learning/BUSN_6051/Assign_4/City_Coords.csv')

        g=nx.Graph()
        group_attr = []
        group_attr_dict = dict(set(sorted(group_attr)))
        nx.set_node_attributes(g, "group", group_attr_dict)

        if ('suppress_cog' in keyword_param):
            if(keyword_param['suppress_cog'] != 'Y'):
                self.setCOG(pths, nodelist)
                for COG_loc in self.COG:
                    g.add_node('COG', x=self.COG_x, y=self.COG_y)
                    g.add_node('COGp', x=self.COGp_x, y=self.COGp_y)
        else:
            self.setCOG(pths, nodelist)
            for COG_loc in self.COG:
                g.add_node('COG', x=self.COG_x, y=self.COG_y)
                g.add_node('COGp', x=self.COGp_x, y=self.COGp_y)

        #g.add_path(nodelist['City'])
        for i, nlrow in nodelist.iterrows():
            g.add_node(nlrow[0], x=nlrow[1], y=nlrow[2])

        for i, elrow in edgelist.iterrows():
            g.add_edge(elrow[0], elrow[1], weight=elrow[2])

        #Add Node Labels
        node_positions = {node[0]: (node[1]['x'], node[1]['y']) for node in g.nodes(data=True)}

        node_color = []
        node_size = []
        for node in g.nodes(data=True):
            if 'COG' in node[0]:
                    node_color.append('cyan')
                    node_size.append(1000)
            else:
                node_color.append('c')
                node_size.append(200)

        pos=nx.circular_layout(g)
        pos=nx.spring_layout(g,dim=2,pos=pos)

        plt.figure(figsize=(14,21))
        nx.draw(g, pos=node_positions, with_labels=True, font_size=14, font_weight='bold', node_color=node_color, node_shape='d', node_size=node_size)

        #Add Edge Labels:
        labels = nx.get_edge_attributes(g,'weight')
        nx.draw_networkx_edge_labels(g, pos=node_positions, with_lables=True,edge_labels=labels)

        length = []
        for path in pths:
            # Determine shortest paths
            edges = nx.shortest_path(g,source=path[0],target=path[1],weight='weight')
            #Create list of path tuples for highlighting
            zipped = list(zip(edges, edges[1:]))
            #higlight optimal route on graph for given paths
```

```
            nx.draw_networkx_edges(g, pos=node_positions, edgelist=zipped, edge_color='r'
                , width=10)

            # Output path lengths and Calculate Costs
            dist = nx.shortest_path_length(g,source=path[0],
                                target=path[1], weight='weight')
            length.append([
                    str(path[0]), str(path[1]),
                    str(path[0]) + ' to ' + str(path[1]),
                    int(dist),
                    int(dist) * int(path[2]),
                    int(dist) * int(path[3]),
                    (int(dist) * int(path[2]))/(300) * 1.3,
                    (int(dist) * int(path[3]))/(300) * 1.3,
                    path[2],
                    path[3]
            ])

        plt.subplots_adjust(top=.8)
        plt.title(graph_name, size=60)
        plt.savefig('../Assign_4/' + graph_name + '.pgf', bbox_inches='tight')
        plt.show()

        return length

    @classmethod
    def setCOG(self, pths, nodelist):
        x_value = []
        x_weight = []
        y_value = []
        y_weight = []

        xp_weight = []
        yp_weight = []

        for node in pths:
            print(node[1])
            nl = nodelist.loc[nodelist['City'] == node[1]]
            x_value.append(float(nl['x']))
            x_weight.append(float(node[2]))
            xp_weight.append(float(node[3]))

            y_value.append(float(nl['y']))
            y_weight.append(float(node[2]))
            yp_weight.append(float(node[3]))
            #y_value.append([int(nl['y']), int(pths[2]])

        print(x_value)
        self.COG_x = sum(p*q for p,q in zip(x_value, x_weight))/sum(x_weight)
        self.COG_y = sum(p*q for p,q in zip(y_value, y_weight))/sum(y_weight)

        self.COGp_x = sum(p*q for p,q in zip(x_value, xp_weight))/sum(xp_weight)
        self.COGp_y = sum(p*q for p,q in zip(y_value, yp_weight))/sum(yp_weight)

        self.COG = ['COG', self.COG_x, self.COG_y, self.COGp_x, self.COGp_y]

    @classmethod
    def getCosts(self, length, **keyword_param):

        total_load_cost = 0
        total_carry_cost = 0
        costs = []

        # Enable ability to query a specific year
        if ('years' not in keyword_param):
             years = [1,2,3,4,5]
        else: years = keyword_param['years']

        # Calculate Carrying Costs
        produce_cost_cwt = 60
        carrying_cost_percent = .35
        consolidation_savings = 1 # No savings by default

```

```python
        # Enable Operational Savings for Reno
        if ('consolidation' in keyword_param):
            if(keyword_param['consolidation'] == 'Reno'):
                consolidation_savings = .6 # alter multiplier to accoung for consolidation
                print(consolidation_savings)

        avg_load_cwt = 300
        mileage_rate = 1.3
        turnover = 8

        route_tc = []
        route_costs = []
        rc = pd.DataFrame(columns = ['route', 'load', 'year', 'load_cost', 'yearly_carry_cost', 'total_load_costs', 'total_carry_costs'])
        for route in length:
            route_length = route[3]
            load_current = int(route[8])
            load_projected = int(route[9])
            incremental = (load_projected - load_current)/5
            for year in years:
                load = load_current + (year * incremental)
                load_distance = load * route_length
                load_cost = (load_distance / avg_load_cwt) * mileage_rate
                yearly_carry_cost = (load * produce_cost_cwt *
                                     (carrying_cost_percent*consolidation_savings)) / turnover
                total_load_cost = total_load_cost + load_cost
                total_carry_cost = total_carry_cost + yearly_carry_cost
                print(total_load_cost)
                route_costs.append([route[2], f'{load:.0f}', year, f'{load_cost:.2f}', f'{yearly_carry_cost:.2f}'])

            route_tc.append([route[2], f'{total_load_cost:.2f}', f'{total_carry_cost:.2f}'])

            rcosts = pd.DataFrame(route_costs, columns=['route', 'load', 'year', 'load_cost', 'yearly_carry_cost'])
            rtcosts = pd.DataFrame(route_tc, columns=['route', 'total_load_costs', 'total_carry_costs'])
            rc = rc.append(rcosts.merge(rtcosts, left_on='route', right_on='route', how='inner'))
            print(route_costs)
            route_costs = []

            # Clear Summary variables for next iteration
            costs = []
            total_load_cost = 0
            total_carry_cost = 0

        return rc

    @classmethod
    def getSumCosts(self, costs, pths):
        tc = pd.DataFrame(columns=['route', 'total_load_costs', 'total_carry_costs'])
        wh_load = []
        for paths in pths:
            cc = costs.loc[costs['route'] == paths[0] + ' to ' + paths[1],
                'route':'yearly_carry_cost'].to_latex()
            # print(cc)
            tc = tc.append(pd.DataFrame(costs.loc[costs['route'] == paths[0] +
                ' to ' + paths[1]], columns=['route', 'total_load_costs',
                'total_carry_costs']))
            wh_load.append([paths[0], int(paths[3])])

        tc1 = tc.drop_duplicates().reindex()

        # Determine the total yearly capacity required for distribution scenario.
        # Divid by 8 to attain capacity warehouse after turnover
        print("Distribution center yearly cwt:")
        wh_load_df = pd.DataFrame(wh_load, columns = ['Warehouse', 'Projected Load'])
        print(wh_load_df.groupby('Warehouse')['Projected Load'].sum())
```

```
        # print(tc1.to_latex())
        lcost = 0
        ccost = 0
        for c in tc1['total_load_costs']:
            lcost = lcost + float(c)

        print('Total Scenario Load Cost = ' + f'{lcost:.2f}')

        cost = 0
        for c in tc1['total_carry_costs']:
            ccost = ccost + float(c)
        print('Total Scenario Carrying Cost = ' + f'{ccost:.2f}')
        tcost = ccost + lcost
        print('Total Scenario Costs = ' + f'{tcost:.2f}')

        return
```