The Holy Rail

Kelly Tepper

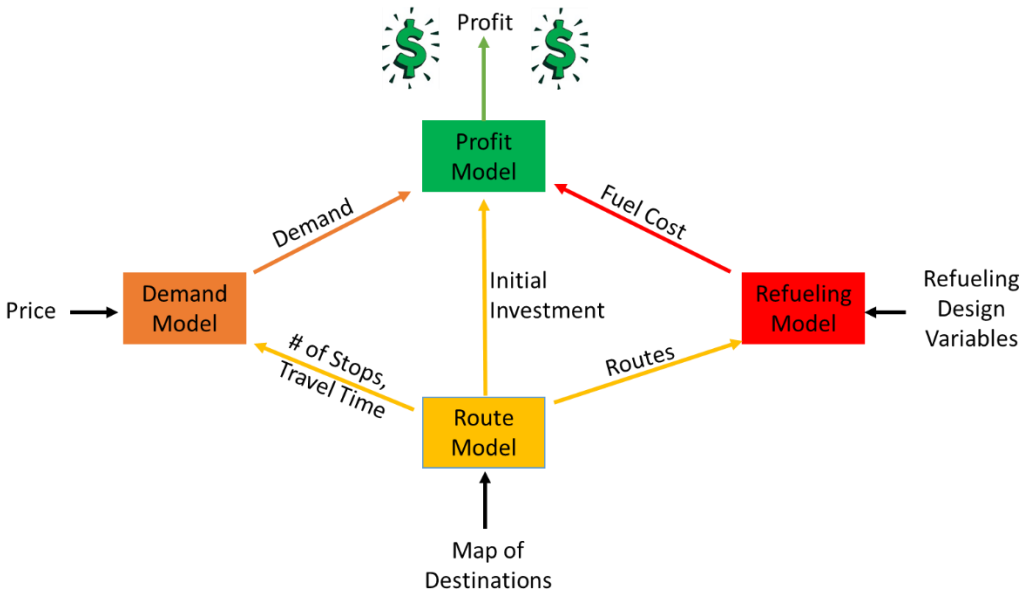
Suffian Hamzah

Pradhyumna Ramesh

Introduction

With the steady growth in population, the demand for safe and efficient public transport is ever-growing and provides a critical challenge for any of today's modern nations. The Railway is an important cog in this wheel and contributes up to 27% of public transport in some major countries. In this project we address the problem of routing and scheduling of trains in a model area. The objective of the project is to deliver a consistent and abstract algorithm for the route and schedule of trains given a set of points to connect. Ideally this system would be able to solve and find the optimal route for any given generic set of towns and stations, providing higher credibility in terms of abstraction.

We will optimize the profit of a train system that goes to a number of destinations. This will be done by considering the designated route(s) between each destination, the refueling cost based on schedule, and the demand for customers based on their satisfaction. The profit model for the system level optimization can be seen in the figure below.



Before the system level profit model was optimized the individual Route, Refueling, and Demand/Profit model were optimized to give us a base of where each subsystem results would be.

Subsystem 1

Route Optimization

The first basic requirement of any transportation system is an organised structure of routes on a map. In this subsystem, we hope to achieve the goal of delivering a comprehensive network of tracks connecting all the destinations (or railway stations) on a given map as effectively as possible. The main objective for this subsystem is, in addition to maintaining or ensuring good connectivity of the graph, we would like to minimize the infrastructure cost involved in building or laying the tracks. One of the assumptions we make of course is that the cost of laying a unit length of track at any point in the map is constant, so it is easy to calculate the total infrastructure cost of the railway track system by simply multiplying the total sum of edge lengths by the cost of unit length of track cost. The method that was used in performing the graph theory analysis was using a graph optimization toolbox on Matlab developed by Sergii Iglin. Before going into some of the algorithm details it is important to define a few graph theory terminology commonly used in our description of the toolbox.

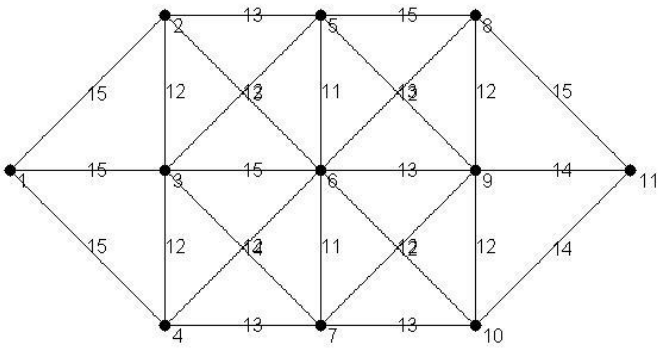
1. Directed graph - A **directed graph** is **graph**, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are **directed** from one vertex to another.
2. Undirected graph - An **undirected graph** is **graph**, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are bidirectional.
3. Complete graph -A **complete graph** is a **graph** in which each pair of **graph** vertices is connected by an edge. The **complete graph** with **graph** vertices is denoted and has (the triangular numbers) undirected edges, where is a binomial coefficient.
4. Subgraph - A **subgraph**, *H*, of a graph, *G*, is a graph whose vertices are a subset of the vertex set of *G*, and whose edges are a subset of the edge set of *G*.
5. Spanning tree - **spanning tree** *T* of an undirected graph *G* is a subgraph that includes all of the vertices of *G* that is a tree.
6. Minimal Spanning tree - Given a connected, undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together.

In our algorithm to solve the train route problem, we use the concept of the minimal spanning tree. In our code, we have as input a map of destination or vertices and we have to figure out the best way to connect the points using railway tracks and then find what is the best way for the train routes to be placed so that we minimize the operation costs. It is helpful to view this as a two step process.

First we use the corresponding function to calculate the minimal spanning tree of the given map. Our approach for this part, was initially to input a complete graph of 11 vertices with reasonable

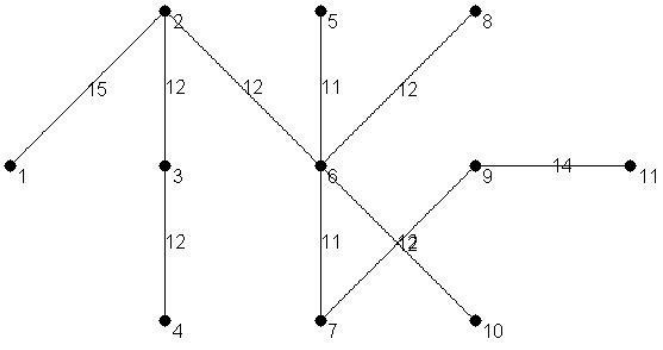
fictitious values of distances based on the map of Detroit. However it seemed to complex to connect every vertex to every other giving 66 edges, so to simplify we reduced the number of edges to a more reasonable number while still ensuring multiple redundant edges to connect the graph. The input graph with the edges is shown below:

The initial graph with weighed edges



As it can be seen from the above input map, we have 11 nodes or destination. Additionally, the edges are assigned weights. In our case, the edge weights can simply be considered as the distances in miles between the stations. The output from the minimal spanning algorithm is shown below. It uses a greedy algorithm by considering the smallest distance to connect all the vertices.

The minimal spanning tree



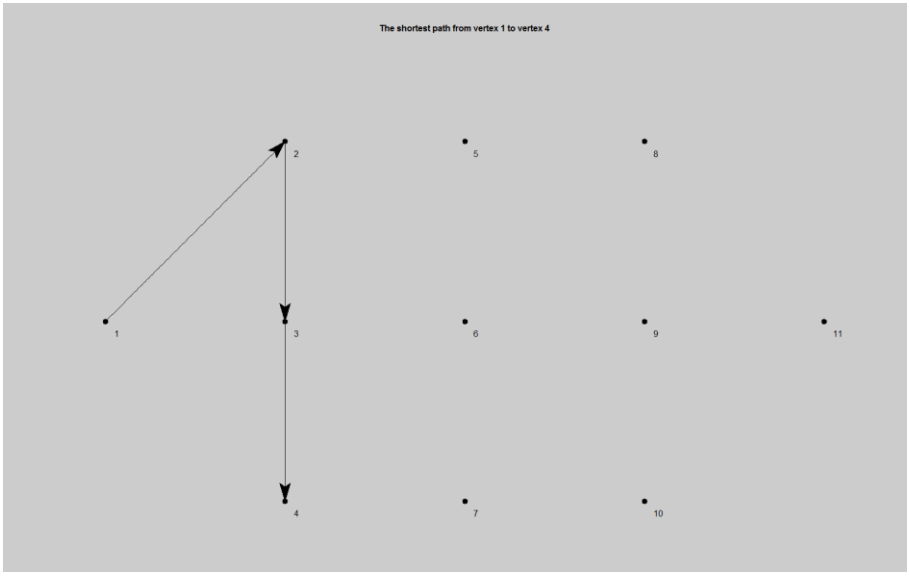
As evident the result of the algorithm ensures a connected graph, that is, all the vertices are connected to every other vertex. The redundant edges have been removed and the total sum of all the edges is minimized.

The minimal spanning tree optimization can be formulated into a mathematical form as:

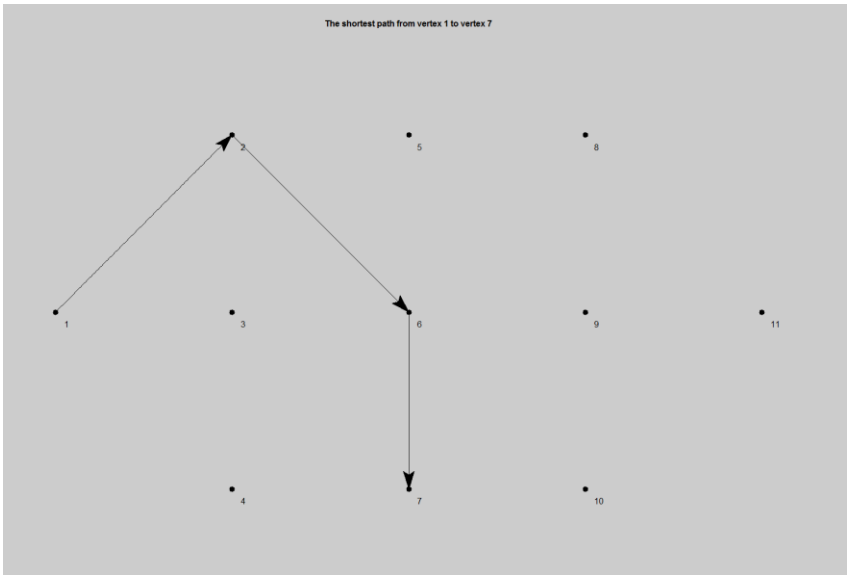
min $\sum e_{ij}$ where e_{ij} is the edge length between vertex i and j.

The constraints are, the edges are length zero when there exists no edge between the two vertices, otherwise the length of the edges is the distance between the points. Another constraint is to ensure connectivity between all the vertices.

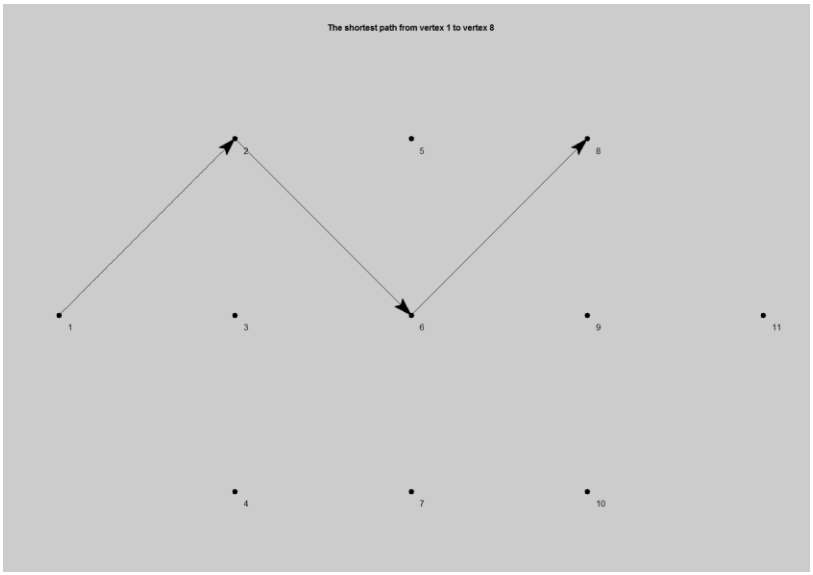
The next step is to use this spanning map as the input to the next step in the process, which is the shortest path algorithm. Here the program uses the map and is provided with a pair of source and destination nodes to find the shortest path to reach the destination from the source node. The code was also executed from the MATLAB toolbox mentioned above. Again it uses a greedy algorithm to solve this problem and finds the next step in a sequential process by picking the smallest distance to advance at every step. For our purpose, we used the minimal spanning tree as the input for the shortest path algorithm. The results with the input shown above and source and destination for combinations of points are shown below:



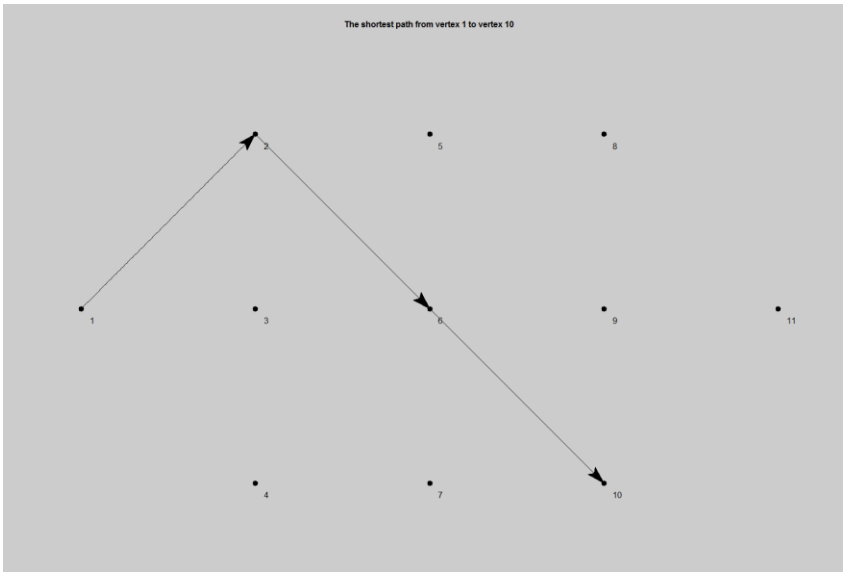
Shortest path from 1 to 4



Shortest path from 1 to 7



Shortest path from 1 to 8



Shortest path from 1 to 10

Analysis

The toolbox does a pretty good job of producing the results that we wanted. However, there are a couple of concerns with the nature of the greedy algorithm. The step wise advancing nature of the algorithm does not allow for it to move in the backward direction, that is from a higher numbered vertex to lower numbered one, so the numbering of the vertices affects the results of the code. For the minimal spanning tree, the input has to be a complete map connecting every vertex to every other vertex. It might be more efficient to get a system that could optimize the node connections with only a map of points as the input. Finally, one aspect that we did not consider for this portion was to try multiple different optimization methods to compare how well the methods work. The fact that this is a unique graph based non gradient system, it was difficult to find many algorithms that handle this type of variables. This also hinders the possibility of sensitivity analysis. Looking to the future, it may be a good idea to repeat the experiments with much more complex maps with higher dimensionality to check the robustness of the method.

Subsystem 2

Minimization of Refueling Cost

Refueling costs has been surging up and accounts for 11% of AMTRAK's operating cost in 2008[1]. This subsystem aims to reduce the costs, which would then maximize the main system's function of maximizing profit for train system.

The problem consists of a railroad network, a locomotive plan which describes locomotive assignments and a train schedule. Locomotives are engines that carry trains through the railroad network. Fuel costs can be divided into two parts. The first one is fuel cost that varies across yards. Yards are stations that are a train's origin, intermediate stop or destination. All yards can provide fuel to the trains. The second one is a fixed nominal cost for every stop a locomotive makes to fuel up. The third one is the cost of holding a fueling truck at a certain station. The objective of this problem is to reduce number of stops a train makes to refuel and the optimal location of placing fueling trucks and the amount across the railroad network. This subsystem is based on a research paper written by V. Prem Kumar and Michel Bierlaire [1] and has been simplified due to the complexity of its original model. The parameter values are based upon the data set provided by INFORMS for its Railway Application Section Problem Solving Competition 2010[2].

Assumptions of the problem

1. The amount of fuel per mile for a locomotive is known and does not change.
2. All locomotives are similar.
3. The train time table is fixed and there are no delays or deviations.
4. Capacity of locomotive fuel tank is known
5. Any fueling stop that is not origin or the end is an intermediate stop. The number of stops on a train route is bounded.
6. All locomotives operate during the whole schedule.

Problem inputs

1) A time schedule that provides a list of trains and time table
   a) Sequence of stations (identical for each train)
   b) The schedule is fixed over the period that has been set
2) Locomotive assignment plan for trains
   a) Locomotive assignments are feasible and cost effective
   b) Repeats over the week
   c) Assumption: The same locomotive is used for all trains, all trains are powered by one locomotive

Mathematical Model

The current model for this subsystem utilizes the data set[2] for its inputs such as the train schedule, railroad network, and the parameters that will be defined later in this section. First, a set of indexes will be defined.

Indexes

Let a locomotive visit a sequence of yards across all train routes and all days be represented by index s such that s= {1,2,3… S}. For example, a Locomotive 1 passes through a sequence of Yards = [Y1,Y2,Y3,Y4], where Y1 is the origin yard and Y4 is the destination yard. This sequence is called 1, and the next sequence is 2 and so on and so forth. Let j be the index used to identify a locomotive. Let i be the yard station.

Decision Variables

X_js : Flag to indicate refueling of locomotive j at a station in its route sequence s; Binary
Y_js: Amount of fuel in locomotive j at time entering the yard in appearing in sequence s; Linear
W_js: Amount of fuel filled in locomotive j at a station in its route sequence s; Linear

Parameters

Param_refuel_jis : 1 if locomotive j visits yard i on sequence s and 0 otherwise
Train_js : Flag for intermediate fueling stations; 1 if station sequence s for locomotive j is intermediate, 0 otherwise
D_js : Distance between yards in a sequence s for each locomotive
Rate : Amount of fuel consumed to run one mile
Min_fuel_js : Minimum fuel needed to reach next yard (d_js*rate)
C_i : Refueling fixed cost
TANK : Locomotive tank capacity
NFP : Max number of intermediate fueling yards in a train sequence

Objective function

The objective function has two components, as described in the subsystem description.

$$Min \sum_{j}\sum_{s} C_{fixed} * x\_js + \sum_{i}\sum_{j}\sum_{s} Param_{refuel_{jis}} * c_i * W_{js} \quad (2\text{-}1)$$

Constraints

1. A locomotive j on a yard sequence s may only be refueled if and only if there is a refueling place at that yard.

$$w_{js} \leq TANK * x_{js} \quad (2\text{-}2)$$

This ensures that the locomotive only refuels if the yard is a refueling place.

2. The fuel within a locomotive can never be above the tank capacity.

$$y_{js} + w_{js} \leq TANK \quad (2\text{-}3)$$

3. The amount of fuel in the locomotive before and after crossing a yard sequence s

$$y_{js} + w_{js} - \min\_fuel_{js} = y_{js+1} \quad (2\text{-}4)$$

This makes sure that the fuel amount in the locomotive during the period of entering a yard is the sum of fuel filled at the previous yard and the amount of fuel in the locomotive at that yard minus the fuel used to arrive at the current yard.

4. A locomotive can only be refueled at most NFP intermediate stations along a station route(excluding origin and the destination)

$$\sum_s Train_{js} x_{js} \leq NFP \quad (2\text{-}5)$$

5. Bounds on the decision variables

$$w_{js}, y_{js} \geq 0 \quad (2\text{-}6)$$
$$x_{js} \in \{0,1\} \quad (2\text{-}7)$$

Model Simulation

A Matlab file was created to simulate the mathematical model. The indexes values set are shown below:
s = 5 ; j = 2 ; r = 7 ; i =73 ;
This means that two locomotives with 5 different yard sequences, 7 train routes, 73 station yards were utilized as inputs. The train table schedule is made according to the indexes and from modifying the data set provided by INFORMS[2].

The values for parameters were also obtained from the INFORMS data set[2].

To test whether the model works, the decision variables were randomized but yet fulfilled the design constraints. The table below provides some of the outputs that were achieved using this model. Although it is not optimal, the results show that the current Matlab file successfully simulated the model.

Trial	Cost	Constraint 1	Constraint 2	Constraint 3	Constraint 4	Constraint 5	Constraint 6
#1	$47690	Met	Met	Met	Met	Met	Met
#2	$37455	Met	Met	Met	Met	Met	Met
#3	$98821	Met	Met	Met	Met	Met	Met

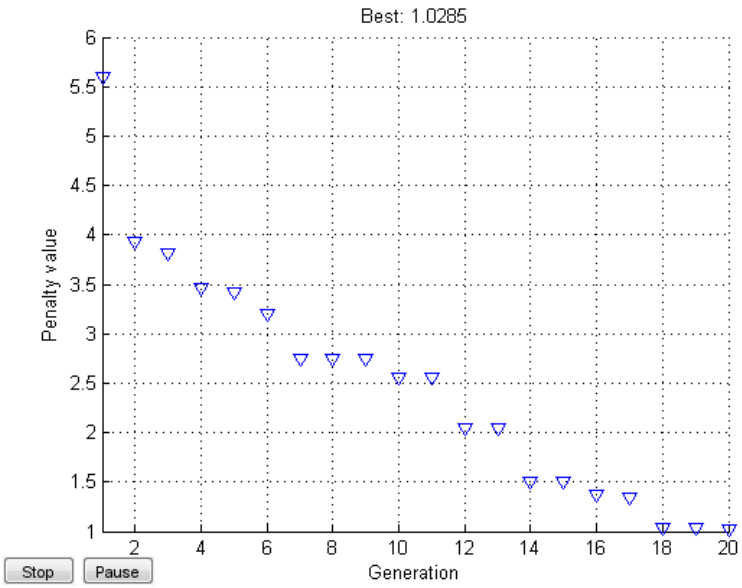
Model Optimization Results

For this subsystem, the Genetic Algorithm toolbox in Matlab was used to optimize the model. This algorithm was most suitable for the model since the model contains both binary and continuous variables. Initial conditions were set by the optimization tool itself with a random number generator

seed of 2 to reproduce results. The parameters set for these results were two trains(j =2) and each had a sequence of 5 stations(s).

The results are shown below for different generation values:

20 Generations



Based on the figure above, the algorithm begins to converge as the number of generation goes up. However, the algorithm did not manage to converge to an optimal value. Instead it exceeded the number of generations. Below is the design variables and the objective function evaluation that was obtained using the algorithm.

Design variables

X_js = [1,0,0,1,1; 1, 0, 1, 1, 1]

W_js = [1685.44, 137.39, 1920.66, 1270.32, 3038.93; 1197.35, 931.65, 2636.73, 738.85, 2443.58]

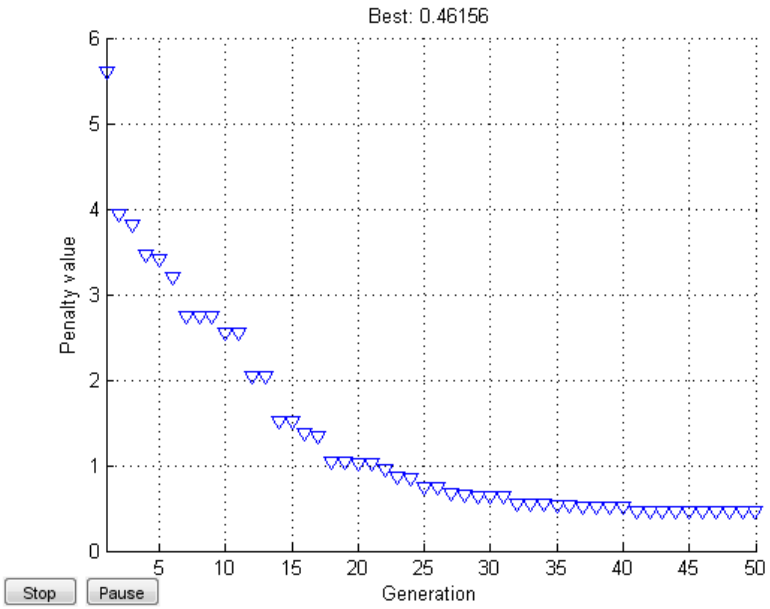
Y_js = [3660.83, 4255.82, 2734.42, 3245.28, 3973.10, 3453.35, 1958.84, 1720.76, 2221.65, 421.11]

Objective function value

F = 5.2792e+04

Using the decision variables, 10% of the constraints set were not met, thus making the values obsolete and variables are not the minimum point in the model.

50 Generations



Based on the figure above, the algorithm convergence goes lower as compared to the 20 generation run. However, the algorithm did not manage to converge to an optimal value. Instead it exceeded the number of generations. Below is the design variables and the objective function evaluation that was obtained using the algorithm.

Design variables

X_js = [1, 0, 0, 1, 1; 1, 0, 1, 0, 1]

W_js= [1776.73, 313.62, 760.35, 917.80, 1956.59; 979.83, 1004.74, 2726.77, 767.11, 1700.00]

Y_js = [3412.95, 3922.64, 2679.85, 2711.26, 1447.82; 3651.24, 2436.59, 1794.43, 1965.70, 754.90]

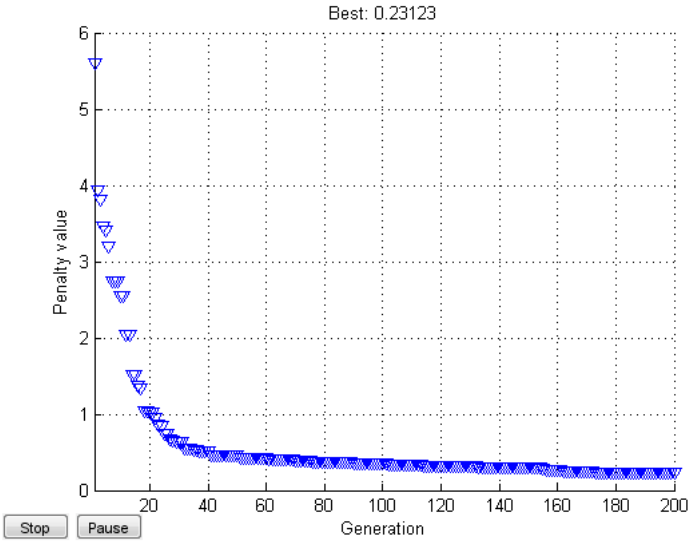
15 constraints violated

Objective function value

F = 4.2641e+04

Using the decision variables, 10% of the constraints set were not met, thus making the values obsolete and variables are not the minimum point in the model.

200 Generations



The figure above shows that with higher generations, the convergence rate of the algorithm improves. However, the algorithm still does not converge to an optimal value. Below is the design variables and the objective function evaluation that was obtained using the algorithm.

Design Variables

X_js = [1, 0, 0, 1, 1; 1, 0 , 1, 0, 1]

W_js= [1530.65, 9.09, 178.33, 2064.21, 1019.31; 1147.92, 144.73, 3044.87, 362.62, 1257.41]

Y_js = [3342.48, 4097.27, 2401.08, 333.80, 505.26; 3395.21, 2473.13, 1455.00, 2135.01, 436.64]

Objective function value

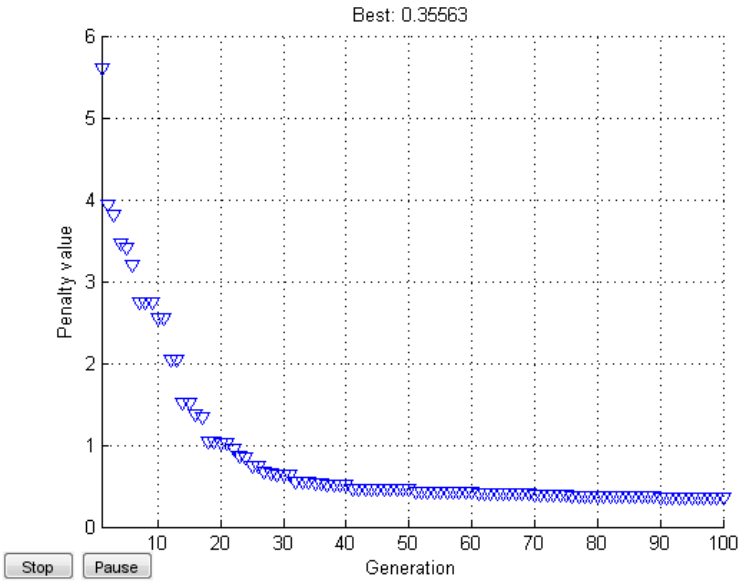
F = 3.5990e+04

Using the decision variables, 10% of the constraints set were not met, thus making the values obsolete and variables are not the minimum point in the model.

To summarize, the number of generations improved the convergence rate of the algorithm, however none of the values were optimal since the algorithm ended due to exceeding the number of generations set. As for the design variables obtained, none of the set managed to fulfill all constraints. One reason that this happens is that the feasible domain is small that it requires a strictly feasible initial value in order to get a feasible optimal solution. Another one would be that some of the constraints are strict.

Instead of varying generations, the initial values for the algorithm were now varied to check if the optimal value can be achieved with different starting points. Different initial values were set by setting different seeds for the random generator and the number of generations were fixed to 100. A seed of 3, 5 and 9 were used to obtain the results. The results are shown below.

Random initial variables with random seed =3



The figure above shows that the algorithm converges fairly close with a low penalty value. However, the algorithm did not manage to solve the model and exited since the number of generations was exceeded.

Design Variables

X_js = [1, 0, 0, 1, 1, 1, 0, 1, 0, 1]

W_js =[1762.02, 68.92, 530.86,993.11, 2500.94; 1097.94, 650.85, 3005.79, 447.88, 1231.85]

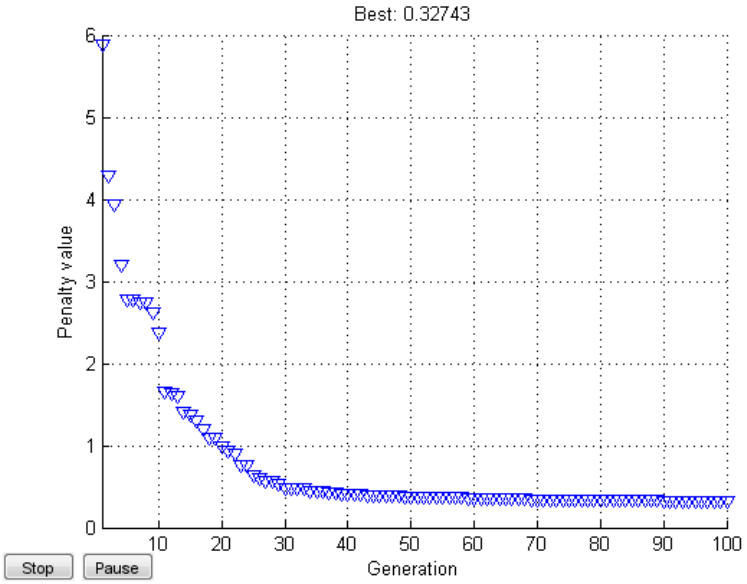
Y_js =[3245.75, 3956.72, 3250.06, 2046.37, 977.975; 3381.85, 2511.09, 1494.31, 2139.65, 523.09]

Objective function value

F = 4.0477e+04

The results were not optimal as 10% (4 out 38) constraints were violated.

Random initial variables with random seed =5



The figure above shows that the algorithm converges fairly close with a low penalty value. However, the algorithm did not manage to solve the model and exited since the number of generations was exceeded.

Design Variables

X_js = [1, 0, 0, 1, 1, 1, 0, 0, 1, 1]

W_js = [887.03, 325.47, 8.87, 1376.05, 2905.21; 1017.22, 14.73, 14.05, 2805.42, 2295.50]

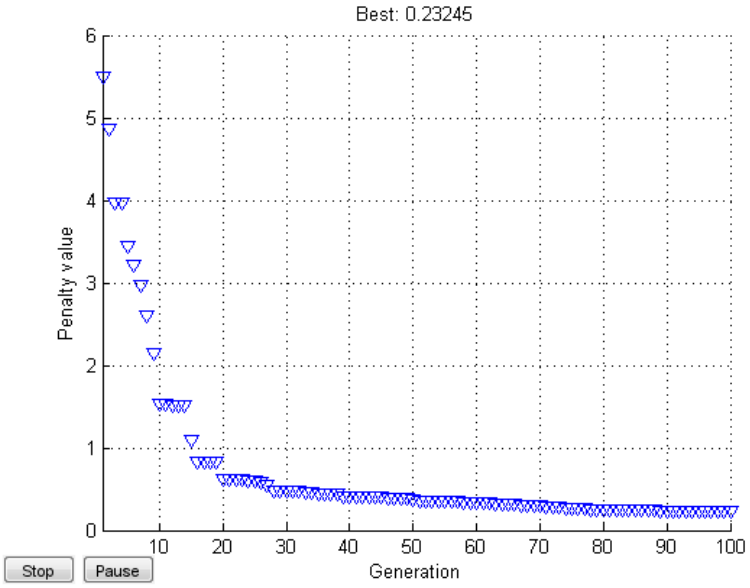
Y_js = [3625.26, 2550.17, 2018.83, 1005.49, 176.84; 3989.27, 3601.58, 2711.97, 337.50, 1112.56]

Objective function value

F = 3.6700e+04

The results were not optimal as 10% (4 out 38) constraints were violated.

Random initial variables with random seed =11



The figure above shows that the algorithm converges fairly close with a low penalty value. However, the algorithm did not manage to solve the model and exited since the number of generations was exceeded.

Design Variables

X_js =[1,0,1,0,1,1,0,1,0,1]

W_js = [3863.26, 73.86, 3137.07, 605.83, 3278.25; 2488.09, 57.19, 1930.55, 3.19, 1537.69]

Y_js = [691.65, 2535.27, 1107.97, 1894.61, 314.89; 2299.58, 3362.31, 2568.31, 2098.42, 62.35]

Objective function value

F = 5.4700e+04

The results were not optimal as 10% (4 out 38) constraints were violated.

Results Analysis

Since this problem contains integer constraints and non-linear constraints, it is fairly difficult for the genetic algorithm to pinpoint a feasible solution. Although the penalty values were close to zero, the solutions obtained from the different number of generations and different initial values. Another reason for the unsolved model is that the initial values set by the algorithm toolbox are unfeasible, thus leading to unfeasible solutions.

Using feasible initial variables

Thus, the algorithm was set with a feasible initial set of variables that fulfill all the constraints. Then, the algorithm was ran with a generation of 100 and 500 respectively. The solution obtained were 3.7102e+04 and 3.4899e+04 respectively, which are lowest among all the solution values obtained However, both were not optimal did not fulfill all constraints as each was violating four out of the 38 constraints set.

Setting objective function to zero to find feasible initial points

Another method was used to find a feasible point was to set the objective function to zero using the feasible initial variables to find other feasible variables. By setting a generation of 500, the algorithm was ran. However, the results was not satisfactory as the design variables obtained violated four out of 38 constraints.

Relaxing and tightening constraints

One of the reasons for the model to have non-optimal solutions when solved using the genetic algorithm is that the constraints might be too strict. To check if this was the case, the constraints were relaxed (which in this case was done by adding a positive value 'α' to the constraints. For example, when the constraint function is called, it returns the value of the constraints with an additional value. If the points are feasible (fulfills the constraints), then α is reduced which symbolizes the tightening of the constraints until the points become infeasible. The results are shown below with varying α using the same feasible initial variables

Trial	A	Generations	F evaluation	Violated constraints
#1	5	51	4.2077E+04	None
#2	4	51	4.2077E+04	None
#3	2	51	4.2077E+04	None
#4	1	200	4.2097e+04	Four

For trial 1-3, the solution converged to the value of the initial variables, which means that the constraints were too relaxed. However, as they were tightened, the solution becomes infeasible again when α is set to 1 and the four constraints were violated Thus, this tells us that the constraints are too strict and that the feasible region is small.

Analysis of constraint violations

Based on the results, the four constraints violated consist of constraints 2-2 and 2-3. One of the reasons why they cannot be met is because of the strict constraint set by 2-4, which limits one of the decision variables, x_js. This affects the other constraints that contain x_js (2-2 and 2-3) thus causing them to be violated. To verify this claim, constraint 2-5 was removed from the constraint function and by using the feasible initial variables, the results obtained were optimal. Another test was done by letting the genetic algorithm function to set the initial variables. The results obtained were not satisfactory as the solution was not optimal as compared with using feasible initial variables yet all constraints were met. In terms of the decision variables representation of the system, removing 2-4 would allow trains to refuel at every stop, thus causing them to incur more costs, which explains the higher objective function value as compared to solving the model using an initial set of design variables.

Conclusion of Optimization

The model contains very strict constraints. The non-linearity of some of the constraints caused difficulty for the genetic algorithm to solve the model. Also, the inclusion of binary and continuous variables affected the performance of the solver used for the model. Although a feasible initial set of variables were used, the results were still non-optimal.

Subsystem 3

In this subsystem, the profit will also be maximized in order to determine the best combination of price, # of stops along the route, and travel time of the train. First one route was optimized that ran 30 miles. Later on the code was updated to run with two routes of 15 and 30 miles to determine how the price would be altered.

Objective Function

Subsystem 3 will be maximizing profit for The Holy Rail Train system. The profit function to be maximized is seen below for one route.

$$Profit = 2 * Price * Demand - Cost \quad (3\text{-}1)$$

The income price and demand is multiplied by two because it is assuming price is for one way and the demand will be the same for commuters traveling to and from.

However when working with more than one route the profit function to be maximized has to include the sum of the different priced routes.

$$Profit = 2 * Sum(Price * Demand) - Cost \quad (3\text{-}2)$$

When working with two routes, price and demand would be a 1X2 array and the individual components are multiplied by the reciprocal in the profit/demand array.

Design Variables

By maximizing the profit function we are changing three design variables: price ($), # of stops, and travel time (min). The first variable can be seen very easily in the profit function but all three variables are in the Demand function, where U_A is the utility of The Holy Rail Train system, U_B is the utility of a commuter bus, and U_C is the utility of a personal car.

$$Demand = \frac{e^{U_A}}{e^{U_A} + e^{U_B} + e^{U_C}} * Population \quad (3\text{-}3)$$

The utility function is a function of price, # of stops, and travel time. This utility function was found by using Conjoint Analysis software by Sawtooth Software [3]. By creating a survey with the Sawtooth Software, the part-worth or beta of the variables was determined. The survey that was created contained the following levels.

	Best			Worst
Price	$1	$3	$5	$8
# of Stops	0	1	3	5
Travel Time	30 min	60 min	120 min	180 min

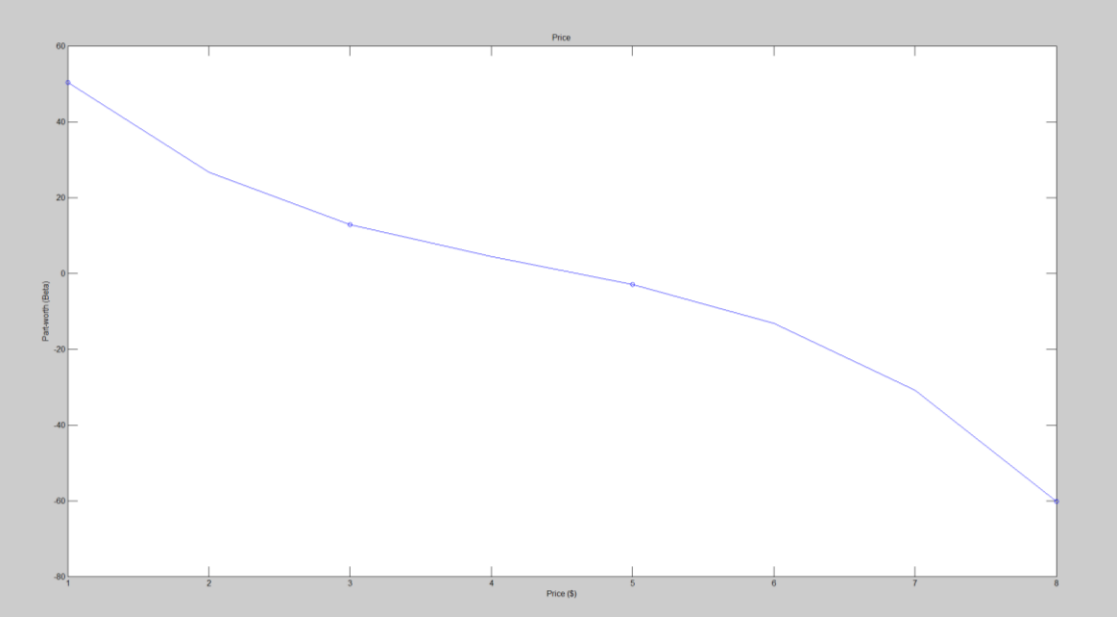
With 18 surveys taken of 11 questions the following part-worth for each level and variable was determined.

	Best			Worst
Price	**$1**	**$3**	**$5**	**$8**
Part-Worth	50.3	12.8	-2.9	-60.2
# of Stops	**0**	**1**	**3**	**5**
Part-Worth	30.7	21.4	-7.3	-44.8
Travel Time	**30 min**	**60 min**	**120 min**	**180 min**
Part-Worth	49.8	23.9	-9.5	-64.2

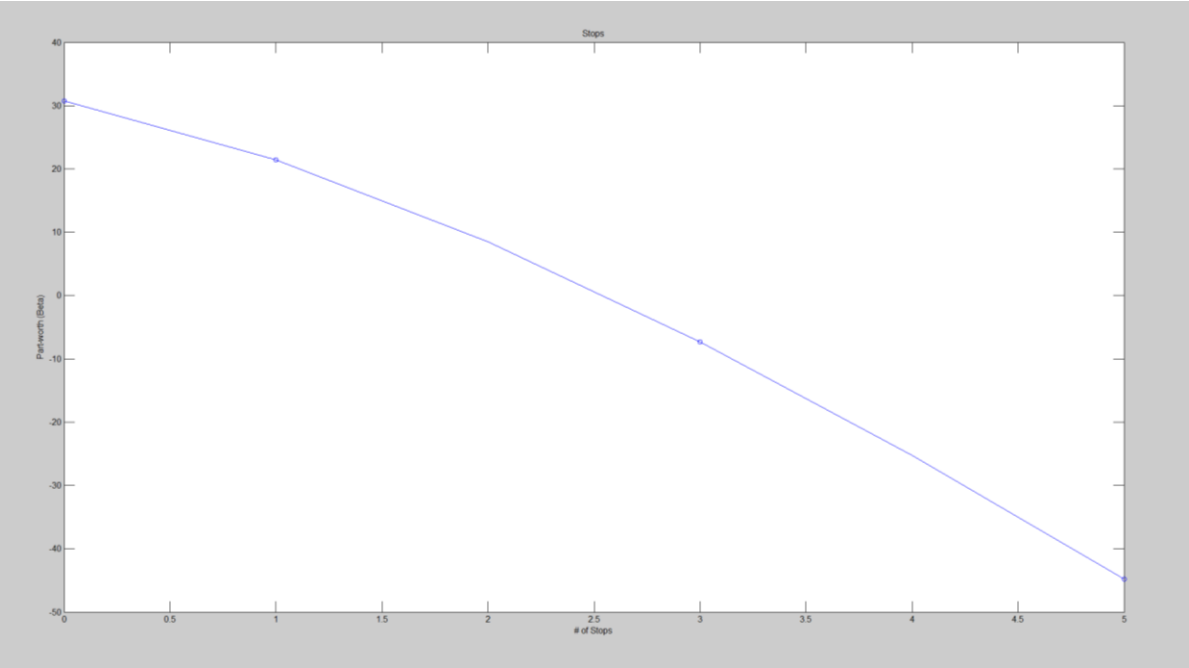
With these part-worths a continuous function using spline was used to help find the utility functions. Below are the splines of each design variable. The total utility function adds up the result of each spline for that designated "product."

$$U = Beta_{price} + Beta_{stops} + Beta_{wait} \quad (3\text{-}4)$$

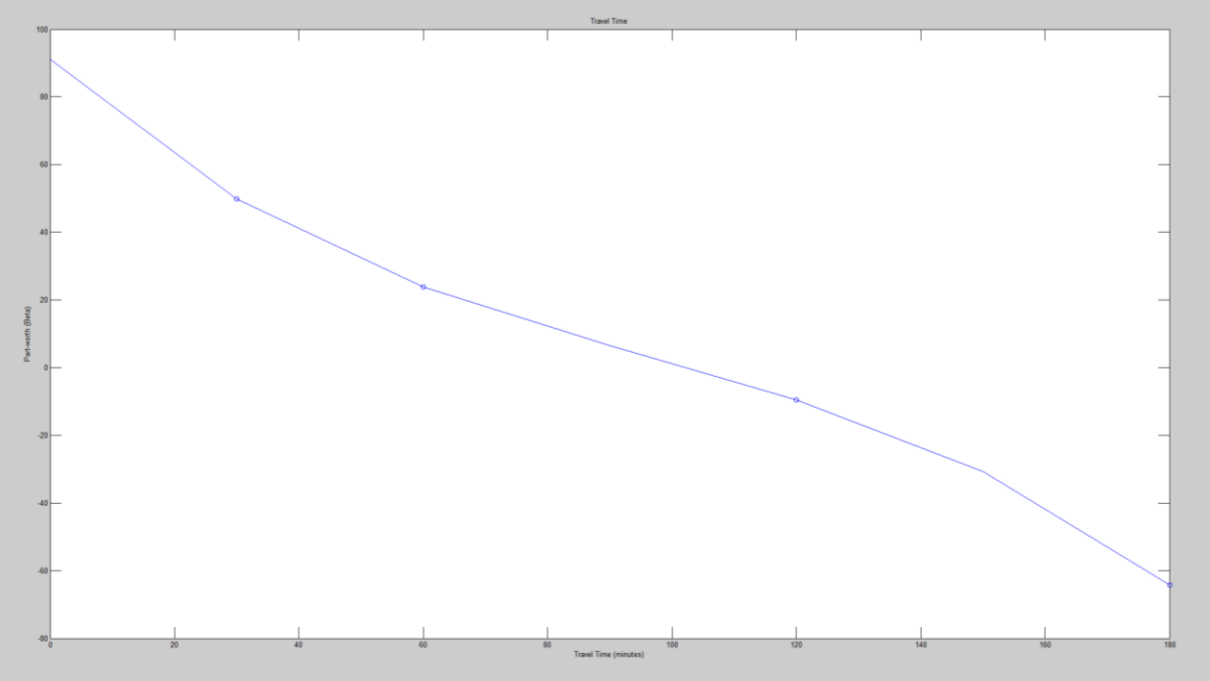
Price



Stops



Travel Time



When calculating one route of 30 miles, for the utility of the Bus these assumptions were made: price = $1.25, stops = 5 stops, travel time = 60 min. Plugging in these values into the spline equations, the utility

of the Bus = 22.35. For the utility of the Car these assumptions were made: price = $1.88 (calculated for a 20 mile journey, with average US gas price and new car average mpg), stops = 5 stops, travel time = 45 min. Plugging in these values into the spline equations, the utility of the Car = 19.5.

When working with two routes, the design variables of the Bus was calculated to be: price = $0.625*length of journey, stops = 5 stops, travel time = 3*length of journey. For the car the design variables were calculated to be: price = $0.094*length of journey, stops = 5 stops, and travel time = $2.25*length of journey.

The Utility of The Holy Rail train will be found the same way as the different variables change to find the maximum profit.

Constraints

If this problem had no constraints the price would go to the optimum that would keep a high demand, and the stops and wait time would go to zero (since the problem is constrained with a lower bound of 0).

To keep this from happening the price, stops, and wait time need to be constrained.

Price

Price will be constrained by the cost of operating the train which includes the initial investment cost and the O&M cost per day. The price constraint is actually incorporated into the objective function. This constraint is making sure the income is greater than the break-even point.

$$Income > break\ even \tag{3-5}$$

$$break\ even = Investment\ cost * \left(\frac{CFR}{365}\right) + O\&M\ Cost \tag{3-6}$$

Where CFR is the capital recovery factor based on a 20 year, 5% interest load.

Stops

Stops will be constrained by an equation making sure that all "stations" along the route have been stopped at by the amount of trains running that day. The amount of trains running per day is an assumption that will be made for subsystem 3.

$$Stops > \frac{\#\ of\ destinations}{\#\ of\ trains} \tag{3-7}$$

Travel Time

Travel time will be constrained by an equation based on the maximum speed a train can go caring passengers. I assumed Class 4 passenger train can go a speed of 80 mph.

$$Train\ time = distance\ time + stop\ time * \#\ of\ stops \tag{3-8}$$

Where distance time is the time is takes a train to go the whole length of the route without stopping and stop time is the time it takes up when a train slows down to stop plus waiting for passengers to embark/disembark.

The wait time constraint is below.

$$Travel\ time > Train\ time \quad (3\text{-}9)$$

Analysis of Constraints

When looking at the utility functions for price, stops, and travel time, it can be seen that for the demand model that if each design variable increases the demand would decrease. The monotonicity analysis of the objective function can be seen in the table below.

	Price	# of Stops	Travel Time
Demand Model	(-)	(-)	(-)
Profit Model	(+)	(+)	(+)
G1			(-)
G2		(-)	

Since the profit model is maximizing the profit, the objective function is negative when minimizing. This results in g_1 and g_2 to be active constraints.

Solution

Due to the fact that the second design variable, number of stops, needs to be an integer, the best method for optimizing this subsystem was to use the genetic algorithm.

Solution One Route

When the model was solved working with one route, the following parameters were assumed:

Investment cost = $1,000,000

Loan Rate = 0.080243 (20 year loan at 5% interest)

O&M cost/day = $1,000

Population = 4,296,250 people (population of Metro Detroit in 2010 census)

# of Trains = 4

# of Destinations = 10

Length Train Travels = 30 miles

Stop Time = 1 min

To determine the optimal result for one route length the GA code was run for 5 iterations which can be seen in the table below.

Iteration	Price	# Stops	Travel Time	Demand	Profit
1	$6.47	3	25.5 min	4,257,800	$55,119,000
2	$6.47	3	25.5 min	4,257,700	$55,119,000
3	$6.47	3	25.5 min	4,257,800	$55,119,000
4	$6.47	3	25.5 min	4,257,800	$55,119,000
5	$6.47	3	25.5 min	4,257,800	$55,119,000

From the above table it was obvious to see that he optimized profit turned out to be $55,119,000 with a demand of 4,257,800 people. The design variables to obtain this result was:

Price = $6.47

Stops = 3

Travel Time = 25.5 min

Due to the assumed parameters that the number of trains = 4 and number of destinations = 10, makes g2 $Stops > 2.5$. However, due to the fact that stops needs to be an integer the "active" constraint for number of stops has to be 3 stops.

Solution Two Routes

When the model as solved working with two routes, the following parameters were assumed:

Investment cost = $1,000,000

Loan Rate = 0.080243 (20 year loan at 5% interest)

O&M cost/day = $1,000

Population = 2,148,900 people per route (half of the population of Metro Detroit in 2010 census)

# of Trains = 4

# of Destinations = 10

Length Train Travels = [15 miles, 30 miles]

Stop Time = 1 min

To determine the optimal result for the two routes the GA code was ran for 10 iterations. This was not as consistent as the one route code. However, the results are very close to each other where the price for route 1 has a range of $0.37 and route 2 has a range of $0.12. The travel time has a bit more of a range but stays relatively close. The travel time for route 1 has a range of 3.06 minutes and route 2 has a range of 2.07 minutes.

Iteration	Price 1	Price 2	# Stops 1	# Stops 2	Travel Time 1	Travel Time 2	Demand 1	Demand 2	Profit
1	$5.92	$7.78	3	3	17.32	26.04	2137700	2139000	$58,580,000
2	$6.17	$7.77	3	3	15.05	26.05	2131400	2139900	$59,598,000
3	$6.20	$7.79	3	3	14.95	25.52	2124700	2142100	$59,723,000
4	$5.96	$7.79	3	3	14.54	25.50	2148400	2142100	$58,986,000
5	$5.98	$7.69	3	3	15.57	27.38	2148300	2146400	$58,687,000
6	$6.04	$7.74	3	3	14.48	27.44	2147600	2136200	$59,022,000
7	$6.19	$7.79	3	3	14.26	25.51	2141500	2140400	$59,865,000
8	$6.18	$7.72	3	3	14.81	27.02	2135200	2144700	$59,495,000
9	$6.29	$7.78	3	3	14.60	25.53	2094700	2143800	$59,697,000
10	$6.24	$7.68	3	3	14.34	27.58	2132000	2147000	$59,547,000

From the ten iterations above the optimized profit turned out to be $59,865,000 with a demand of 2,141,500 and 2,140,400 people respectively. The design variables to obtain this result were:

15 mile:

Price = $6.19

Stops = 3

Travel Time = 14.26 min

30 mile:

Price = $7.79

Stops = 3

Travel Time = 25.51 min

As can be seen the result with dealing with two routes for the same 30 mile route has changed the price by $1.31 but the travel time and # of stops has stayed constant. I believe this is due to the different size demand that the route now sees. This shows that having more than one route can affect the prices of the different routes due to figuring out the demand and the relationship between the two or more routes.

System Level Profit Optimization

For the system level profit optimization the objective function is below.

$$Profit = 2 * Sum(Price * Demand) - Cost \quad (4\text{-}1)$$

Because we are going to optimize multiple routes, the price for each route multiplied by demand needs to be summed together to create the full income for all the different routes.

At the system level, the cost is not a parameter like it was in Subsystem 3, it is instead a response from the optimizations of the first and second subsystem, where the infrastructure cost comes from Subsystem 1 and the Refueling or O&M cost comes from Subsystem 2.

$$Cost = Investment\ cost * \left(\frac{CFR}{365}\right) + O\&M\ Cost \quad (4\text{-}2)$$

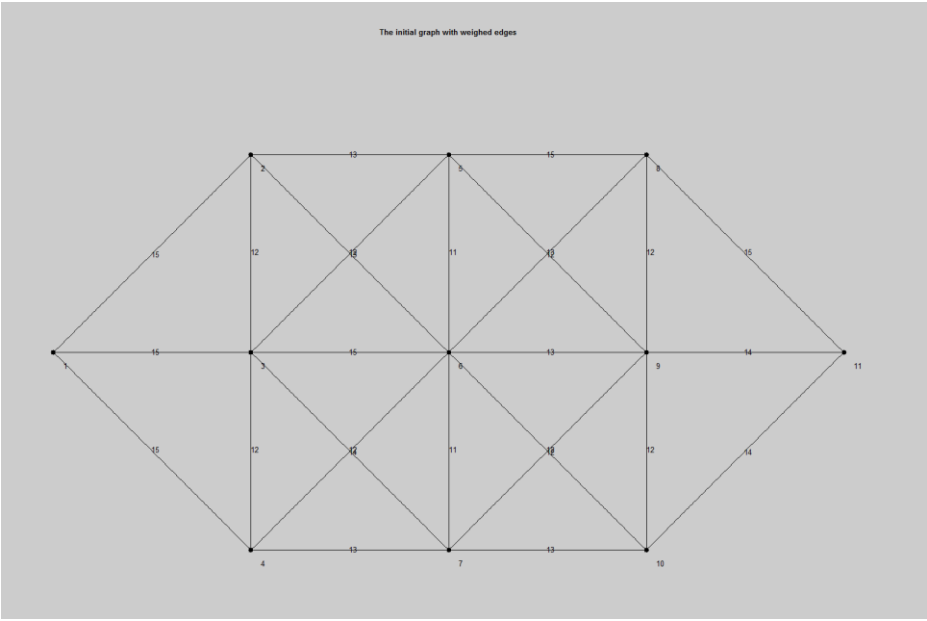
The design variables for the system level optimization are cost per ticket, route which includes the distance between stations and the order, and the refueling design variables.

The constraints for the system level optimization are that income needs to be greater than cost, the stops are at fixed locations, and the many refueling constraints mentioned in Subsystem 2 section.

Due to the discrete variables to calculate the refueling cost, genetic algorithm optimization method was used. Also, because the Subsystem 1 is a graphical optimization approach, Subsystem 1 was optimized first then its output was put in the model to optimize Subsystem 2 and 3 together.

Parameters

The train system will be optimized with the input map of stations as seen below.



To predict the utility function for the transportation competitors of a car and a bus the following equations were used.

Bus

$$Price = 0.625 * Length\ of\ route \quad (4\text{-}3)$$

$$Travel\ Time = 3 * Length\ of\ route \quad (4\text{-}4)$$

Car

$$Price = 0.575 * Length\ of\ route \quad (4\text{-}5)$$

$$Travel\ Time = 2.25 * Length\ of\ route \quad (4\text{-}6)$$

Where the price per mile of the car is due to the fact that IRS will reimburse 57.5 cents per mile in year 2015(4).

The number of stops for both bus and car were kept consistent at 5 stops for every length route.

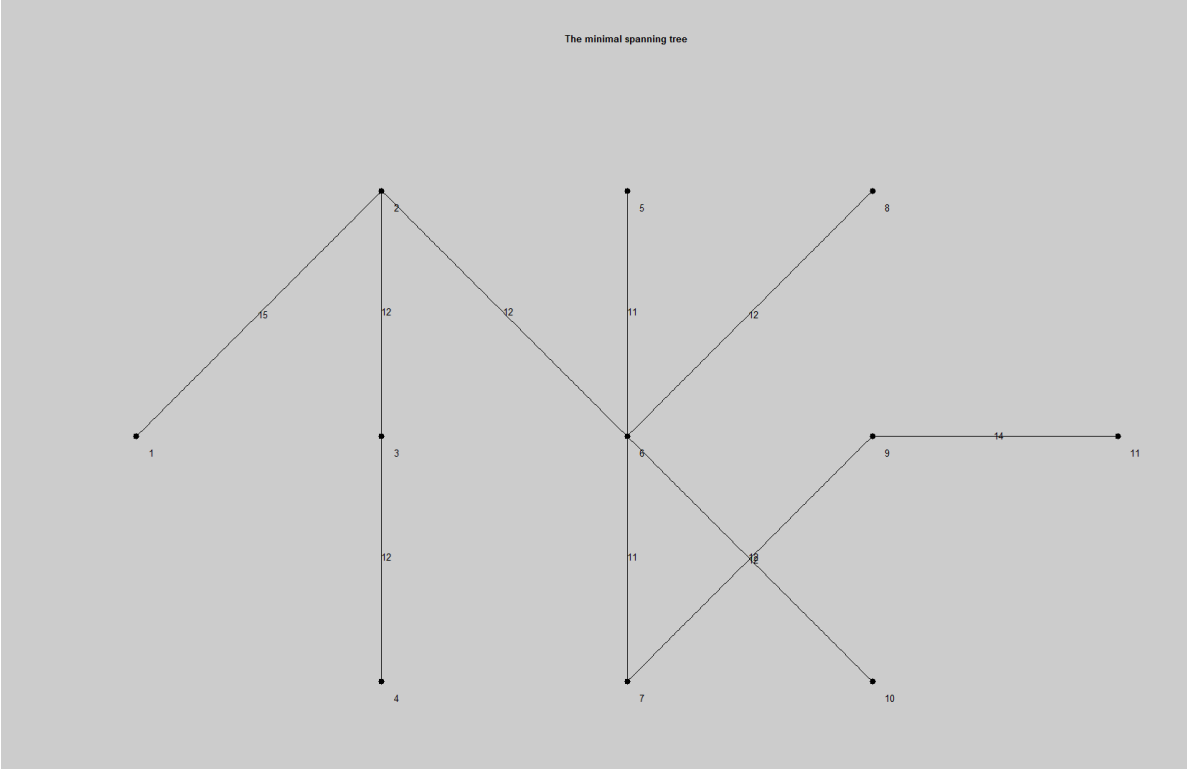
The assumed loan the company would receive for infrastructure cost would be a 20 year loan at 5% interest.

The parameters for subsystem 2 was maintained for system level optimization.

Solution

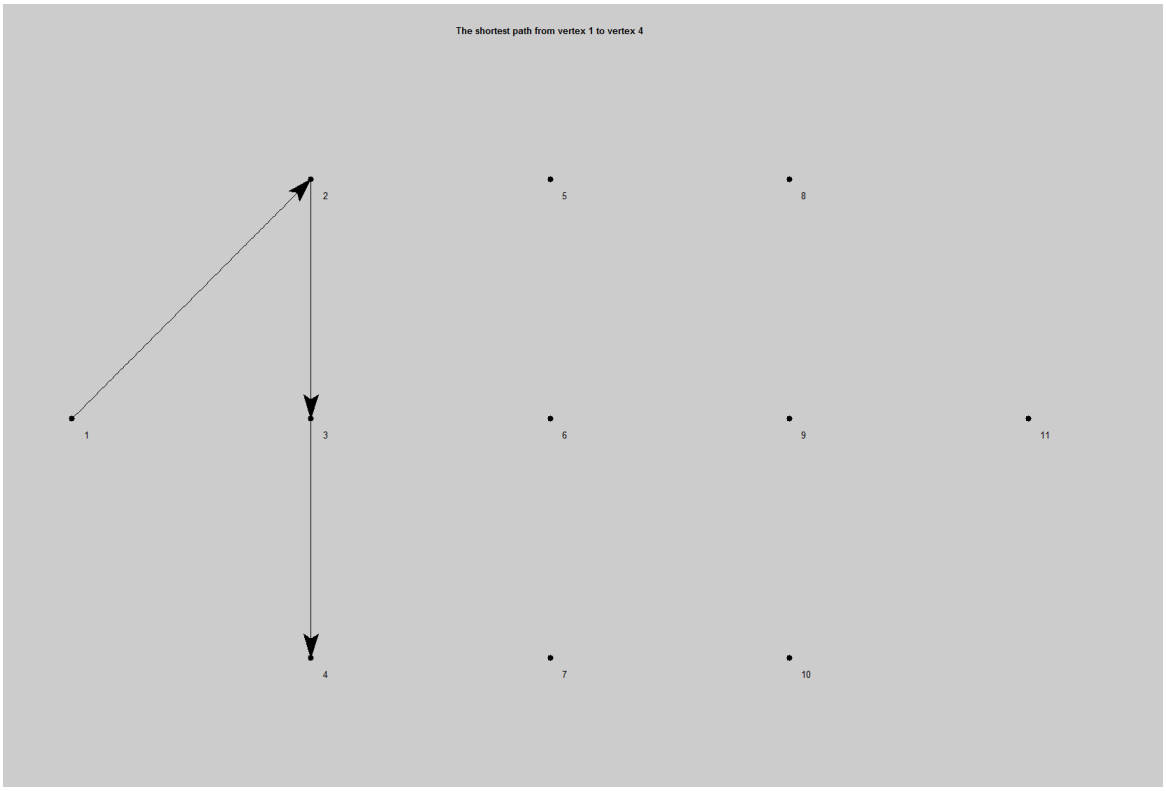
The first step in this optimization was to find the optimal spanning tree of the original network of stations.

This can be seen in the figure below.

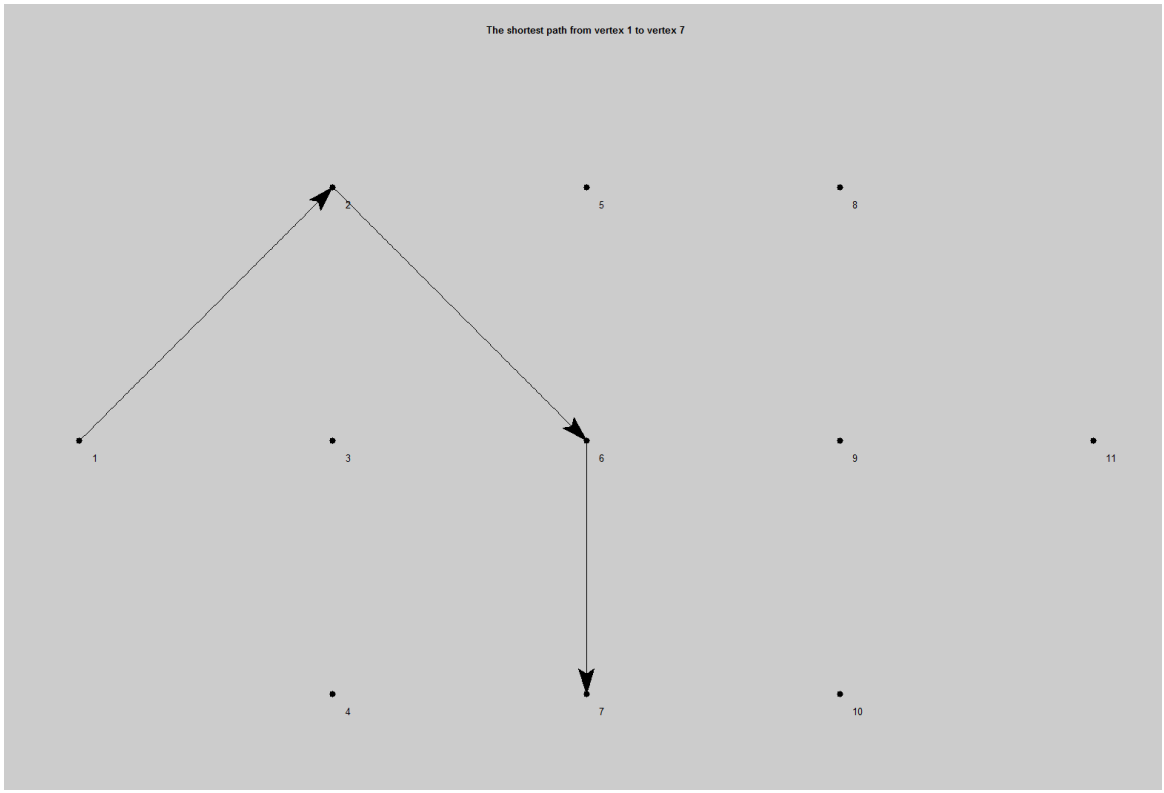


The figure above was used to determine the minimal length of train track needed for the Holy Rail Train system. It was determined that it costs about $1 million dollars per mile of track to install (5). Therefore the final infrastructure cost amounted to $123 million.

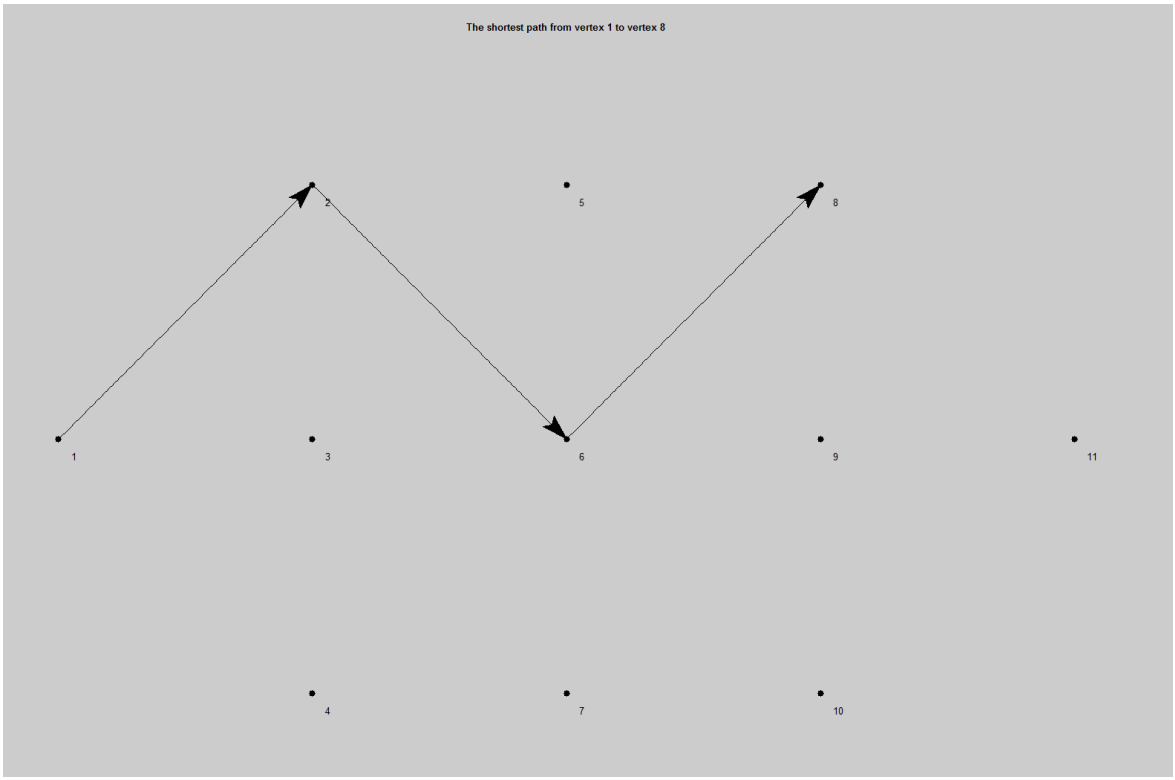
After the minimal spanning tree was developed four routes were optimized that each started from station 1. These routes can be seen below.



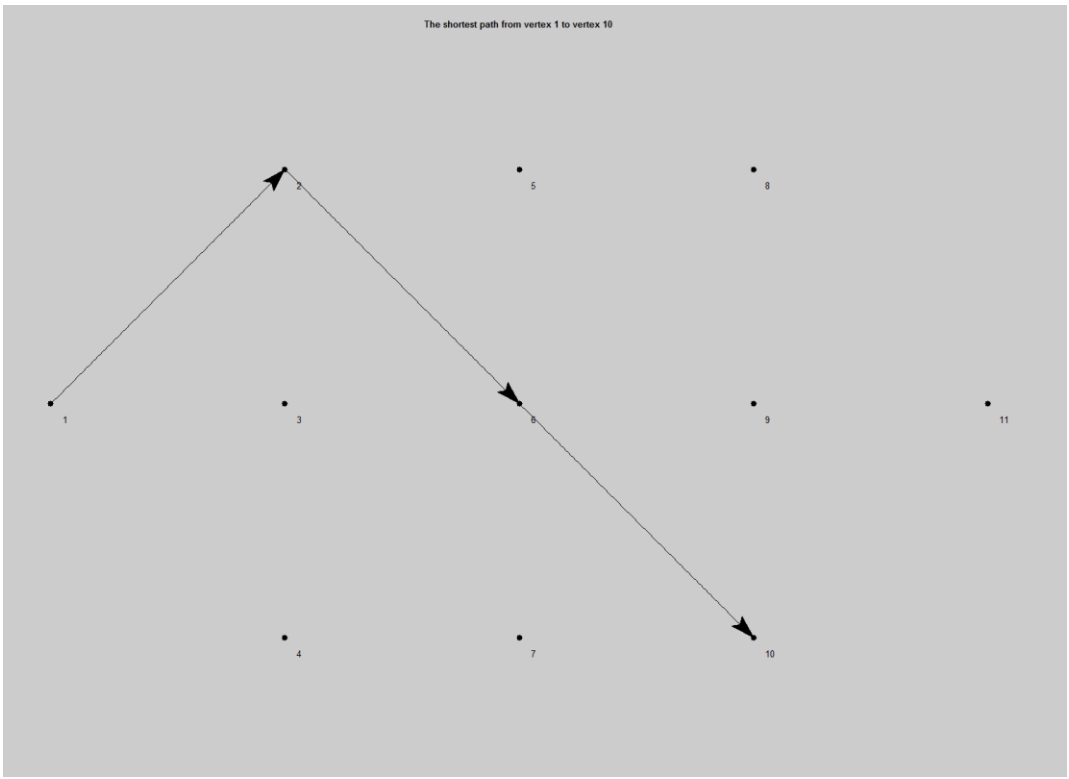
Route 1 – [1,2,3,4] – Totals 39 miles



Route 2- [1,2,6,7] – Totals 38 miles



Route 3- [1,2,6,8] – Totals 39 miles



Route 4- [1,2,6,10] – Totals 39 miles

Each of these routes were inputted into the combination of Subsystem 2 and 3 to determine the optimal Profit for the Holy Rail based on the price for each route.

Analysis

Due to 48 initial population size of variables that need to be hand calculated to be inputted as the initial variables for the genetic algorithm to accurately find a solution within the feasible space, the current method of solving for the optimal solution is not ideal.

Based on the analysis done for subsystem 2, it seems that the constraints used were too strict, causing the algorithm to provide non-optimal solutions that was not even close to the feasible region. This makes the model unsolvable without violating any of the constraints that were set.

Future Work

In order to have a better outcome for the model, one of the ways is too modify the constraints for subsystem 2, which may have been too complex for the model. Also, the constraints could be relaxed in order to be satisfied and increase the feasible solution region of the model. However, this would provide a not so optimal solution as the constraints are relaxed.

Optimization Code
Attached with this document is the code for our system optimization. Run the 'All_Objective.m' to see code run in action.

References

1. Kumar, V., & Bierlaire, M. (2013). Optimizing Fueling Decisions for Locomotives in Railroad Networks. *Transportation Science,* 131105054348001-131105054348001. Retrieved February 27, 2015, from http://transp-or.epfl.ch/documents/technicalReports/KumarBier11.pdf
2. Railway Applications Section. (n.d.). Retrieved February 27, 2015, from https://www.informs.org/Community/RAS/Problem-Solving-Competition/2010-RAS-Competition
3. Sawtooth Software. (n.d.). Retrieved February 27, 2015, from https://discover.sawtoothsoftware.com
4. "New Standard Mileage Rates Now Available; Business Rate to Rise in 2015." New Standard Mileage Rates Now Available; Business Rate to Rise in 2015. IRS, 10 Dec. 2014. Web. 20 Apr. 2015. <http://www.irs.gov/uac/Newsroom/New-Standard-Mileage-Rates-Now-Available;-Business-Rate-to-Rise-in-2015>.
5. Hollowell, Carl. "Costs of Rail Siding." Costs of Rail Siding. N.p., n.d. Web. 20 Apr. 2015. <www.acwr.com/economic-development/railroads-101/rail-siding-costs>.