A Method for Scheduling Railroad Fueling Operations

Ed Ramsden

Dec 14, 2010

www.edscave.com

edr@sensorlytics.com

This paper is an expanded version of the 'report' section of my entry to the 2010 INFORMS RAS Problem Solving Competition held in Austin, Texas at INFORMS' 2010 Annual Meeting. This technique and associated presentation won 2nd place in a field of over 30 entries from operations research experts from all over the world. The competition was organized by the Railway Applications Section of INFORMS, and was co-sponsored by BNSF, CSX, Union Pacific, and Norfolk Southern.

Problem Overview

The goal of this problem is to minimize the total fueling costs for a railroad system. Total fueling cost comprises the following three components: (1) The sale price of the fuel consumed, (2) A fixed overhead cost for taking on fuel at a given yard (potential fueling point), and (3) A cost for establishing fueling capacity at a given yard. The sale price of fuel varies from yard to yard, and for this problem is in the general vicinity of $3/gallon. The amount of fuel required by a given locomotive is proportional to the length of its route, and assumed to be 3.5 gallons/mile regardless of the type of locomotive, number of cars being hauled or the train loading. The fixed cost for stopping at a yard to take on fuel is $250. The cost for establishing fueling capacity at a given yard is a non-linear function of the maximum amount of fuel that must be dispensed. In this problem, up to 25,000 gallons can be dispensed daily from a single fueling truck, contracted at a cost of $4000/week. If the amount of fuel required is greater than 25,000 gallons on a given day, additional trucks must be contracted. If contracted at all, a fuel truck must be contracted for the entire problem horizon (2 weeks), so the contracting cost for fuel trucks increases in increments of $8000 ($4000/week x 2 weeks).

Additional constraints and conditions are also placed on the problem. First, it is assumed that the details of the individual train schedules (the order in which yards are visited, and the routes between yards) are fixed, so the yards at which a given locomotive may take on fuel are limited to a small subset of the total set of yards. Second, there is a hard limit (2) on the number of times a train may fuel between 'origin' yards (where the locomotive picks up and discharges cars). It is not required that a train fuel at origin yards, but it may not stop more than twice in the course of a trip between an origin yard and its next destination yard to fuel. Finally, the maximum tank capacity of a locomotive is 4500 gallons. At 3.5 gallon/mile, the tank capacity limits a locomotive to a maximum range of ~1285 miles between fueling stops.

The primary source of problem complexity is its scale: 214 locomotive may fuel at any of 5264 potential stops over 73 different yards. The need to select fueling stops also adds a major non-linear combinatorial component to the problem.

Solution Overview

To solve this problem, I used a combination of two techniques, organized hierarchically, as illustrated in Figure 1. At the level of individual locomotives, (Level-1) a deterministic greedy algorithm is used to make decisions determining at which yards each locomotive should take on fuel, and how much fuel should be taken on, yielding cost information. At the top level (Level-2), a stochastic search algorithm is used to determine which yards should be used as fuel depots by the Level-1 algorithm. At each iteration of the Level-2 algorithm, the Level-1 algorithm is re-executed for each locomotive and a new set of costs calculated.
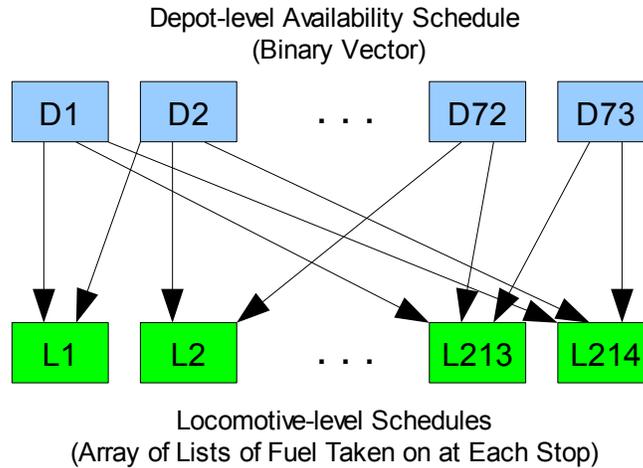
Figure 1 – Hierarchical Decomposition of Problem

Level-1 Optimization Algorithm

Locomotives pass from yard-to-yard in the course of running their scheduled routes. At any given yard they have the option of stopping and taking on fuel. The maximum amount of fuel that may be taken on at a given yard ($F(y)$, Figure 2) is limited by the fuel at arrival at that yard ($FA(y)$) and the the tank capacity. The locomotive departs a yard with a departure fuel level ($FD(y)$) that is the sum of the arrival fuel and the fuel taken on. As the locomotive moves to the next yard, fuel is consumed proportional to distance, and it arrives at the next yard with a partially depleted tank. At no point can the locomotive fuel level exceed the tank capacity. If the fuel level drops to zero except when stopped at a yard, this indicates an infeasible solution (the train runs out of gas between stops).
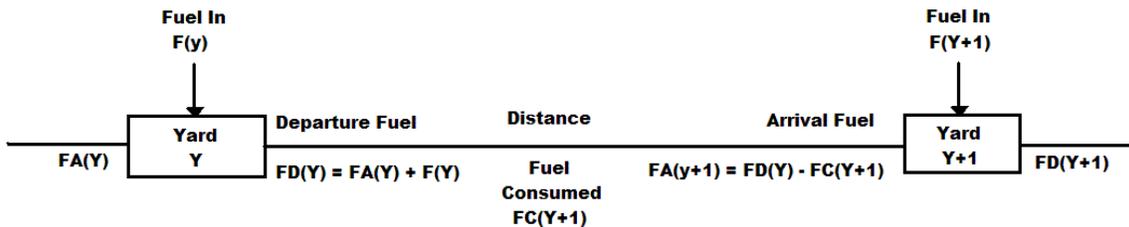
Figure 2 – Locomotive Fueling Relations

In the Level-1 optimization, each locomotive schedule is treated as an independent problem, subject to the constraints of fuel cost at each depot and the requirement to minimize the total number of fueling stops. An additional requirement not to refuel more than twice between 'origin' stops also exists. To solve this problem, I used a greedy approach: After a fueling operation, fuel is to be acquired at the most distant yard with the lowest price within range. To discourage excessive fueling stops, a per-gallon penalty cost based on the fuel remaining at arrival is added to the nominal fuel price. The effective cost for fuel at all stops within range is calculated as:

$$\text{EffectiveCost}(y) = \text{NominalCost}(y) + \text{Stopcost} / (\text{TankCapacity} - FA(y))$$

In this problem, StopCost=\$250 and TankCapacity=4500 Gallons. Arriving at a yard with an empty tank results in an additional \$0.06 per gallon cost, while arriving with a half-empty tank ups this to \$0.11. Arriving with the tank $9/10^{th}$ full results in an adder of \$0.56. This cost adder proved to be such an effective deterrent to over-frequent refueling, that the issue of the two-stops constraint seldom had to be separately enforced in a solution. At each fueling stop, the tank is topped off, until the trip is complete. Figure 3 shows how the raw fuel cost and fuel-remaining penalty combine to discourage premature refueling.

Figure 3 – Combination of Fuel Cost and Fuel-remaining Penalty

While sequentially applying the above rule results in a feasible (and usually a fairly good) solution for an individual locomotive schedule, room for further improvement exists. The most obvious case to consider is if the locomotive arrives at a fueling stop with fuel in the tank, and fuel at this stop is less expensive than fuel acquired at the previous fueling stop. In this case it is possible to transfer the excess fuel dispensed at the previous stop to the the present stop, resulting in an immediate cost reduction. This process is applied sequentially to each stop in the locomotive schedule.

Including all of the above considerations, the Level-1 algorithm can be summarized:

1) Calculate total fuel required for route (14-day horizon). Fuel dispensed will be deducted from this balance.

2) Find yard with least expensive fuel – use as starting point yard and fill tank

3) Calculate arrival gallons for potential fueling points (yards) within range

4) Calculate effective fuel cost for each yard, find least expensive yard at furthest range.

5) Top-off tank at this yard, repeat from 3 until no fuel remains

6) Beginning with starting point yard, determine if fueling can be shifted from more expensive fueling points to less expensive, successor fueling points.

Level-2 Optimization Algorithm

The goal of the Level-2 optimization algorithm is to minimize the number of fuel trucks required. The overall approach I employed was to randomly perturb the availability of fueling yards, and retain the best solutions discovered along this random walk – essentially a randomly-guided hill-climbing search. For each candidate set of available yards, all individual locomotive fueling schedules (Level-1) are recalculated, and the overall fueling requirements are gathered to determine the number of required fuel trucks at each yard and total problem costs (fuel+stops+trucks). At each test cycle, a certain percentage of yards are randomly selected and changed in state from 'available' to 'unavailable' or vice-versa. Perturbing 10% of the yards seemed to work well. If the total cost (fuel+stops+trucks) is *less than or equal* to that of the current solution, the current yard-available configuration is replaced by the new yard-available configuration. Permitting the algorithm to move to solutions that are merely equal to older solutions allows it greater freedom to explore the potential solution space than a strict hill-climbing requirement.

Because making a yard totally unavailable for fueling would often result in an inability to obtain *any* feasible solution for individual locomotive schedules, the state of unavailability is presented to the Level-1 algorithm as a per-gallon cost penalty incurred for refueling at the unavailable yard. This effectively discourages refueling at particular yards, but does not prevent it. Interestingly, a surprisingly modest penalty ($0.10/gallon) was found to result in obtaining the best overall solutions.

The Level-2 optimization process can be summarized:

1) Initialize the YardAvailable vector to all TRUE

2) Calculate Nominal Cost for fuel at each yard based on YardAvailable and penalty.

3) Run Level-1 Algorithm for All Locomotives, Calculate TotalCost.

4) If TotalCost <= BestCost, Update YardAvailable Vector, else undo perturbation

5) Perturb P % of yard available

6) Repeat from step 2 until converged.

The algorithm converges quickly on a locally optimum solution, with most of the improvement typically occurring within the first few hundred iterations. Figure 4 shows the solution quality (lower is better) as a function of iterations for one example run.

Algorithm Convergence Example
(Team EdRamsden)
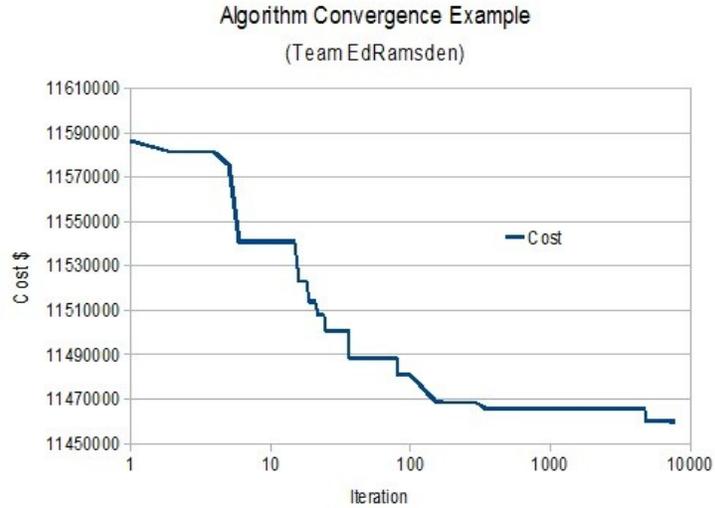
Figure 4 – Convergence Example

Because the solution landscape for this problem is highly non-linear, there are numerous local minima, making it difficult to determine whether one has really found a global minimum cost. This algorithm will tend to settle into a local minima and become 'stuck' there. Running the algorithm for 200 trials with 10,000 iterations per trial resulted in the distribution of solutions shown in Figure 5. Note that it most frequently converged to one of a few solutions, and that all of these 'attractor' solutions were of very similar (+/- 0.1%) quality.

Frequency of Common Solution Attractors
(200 Runs @ 10,000 Iterations/run)
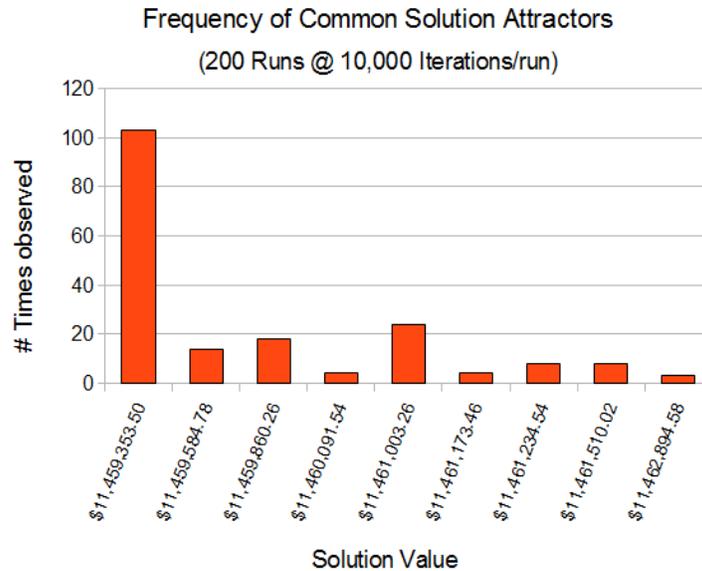
Figure 5 – Frequency of Discovering Particular Local Minima

Implementation & Results

The optimization program was implemented entirely in Microsoft Visual Basic 2008, with a total length of approximately 600 lines (including comments). It reads in three separate text files containing the yard-to-yard mileage for each pair of yards, the nominal ($/gallon) price of fuel at each yard, and the schedule for each locomotive. It outputs a text file containing the fueling schedule for each locomotive, the initial fuel required by each locomotive at the start of the problem horizon, the number of tucks that should be contracted for each yard, and a cost summary.

When run for 20,000 iterations on a single core of an dual-core Intel E5300 processor with 4GB of main memory, the program required approximately 15 seconds to execute. To a first approximation, one could expect execution time to be roughly proportional to both the number of iterations performed by the top-level algorithm multiplied by the total number of stops in the problem. Although memory utilization was not explicitly measured, this algorithm requires little storage beyond that required to hold the problem description data and a single description of the solution state (available yards, fuel at arrival, fuel dispensed for each possible stop, etc..). For these reasons, this technique should be readily scalable to larger problems.

Because the optimization algorithm is stochastic in nature, and the problem has numerous local minima, different results may be obtained from different runs. Table 1 summarizes the best observed results, and the ones submitted as part of this competition entry:

Table 1 – Summary of Best Observed Solution Costs

Parameter	Value
Overall Solution Cost	$11,459,353.50
Fixed Refueling 'Stop' Cost	$281,250.00
Truck Contracting Cost	$296,000.00
Total Fuel Cost	$10,882,103.50

For more information on this problem and the competition, go to the INFORMS RAS Competition website http://www.informs.org/Community/RAS/Problem-Solving-Competition