ORIE4100 Project 4

Zebang Wu

September 2025

1 Single-day demand optimization design

1.1 Assumptions for single-day version

First, we have assumptions for the procession of all our problem. They are proposed based on the actions we can take on this game, while not exactly the same of coverage as all the possible actions. We proposed these assumptions so that they could be understood better.

The assumptions and their reasons are:

1. If the driver's time for a trip has some digits, like the last day for the driver is half a day, don't schedule new tasks until a new day begins.

   In math, this will help us form more robust periodicity in days, making the behavior more predictable. For example, a period for a bus would be 4 days in this assumption, rather than 3.57 or some digits that could make the calculation more complex and involve some non-linear effect on certain dates. It shall be also more robust as that we have some space preserved at the same time of day for each shuttle. This allows us to rearrange easily.

   Also, in the consideration of bussiness competition, the periodicity would improve the experience of uses, that they could get what they want at the same time of day and having a steady delay time. So they can also schedule their distribution better with this stable expectation for products to come.

2. We send out the repositories instantly. No delay.

   This shall lesson the time that customer waits, since we don't wait to fully load our trunks and then start.

And by the mean of "single-day" strategy, we have this additional constraint:

- The demand of all the weekdays shall be the same as one of the days.

Table 1: Distance Matrix (Miles)

City	WHS	BNG	JAK	ORL	TPA	ATL	MAC	BTM	CHR	RAL	COL	MPH	NSH	ALX	RCH
WHS	0	150	351	426	452	10	82	654	240	377	214	382	246	618	532
BNG	150	10	463	544	539	150	200	771	391	479	364	255	194	735	634
JAK	351	463	10	140	194	351	273	751	390	456	412	694	563	715	611
ORL	426	544	140	10	82	426	344	892	530	596	552	776	672	856	751
TPA	452	539	194	82	10	452	370	945	584	650	593	792	698	909	805
ATL	10	150	351	426	452	10	82	654	240	377	214	382	246	618	532
MAC	82	200	273	344	370	82	10	736	322	450	223	455	328	700	605
BTM	654	771	751	892	945	654	736	10	418	298	517	891	696	37	143
CHR	240	391	390	530	584	240	322	418	10	136	99	622	486	397	292
RAL	377	479	456	596	650	377	450	298	136	10	227	661	533	261	155
COL	214	364	412	552	593	214	223	517	99	227	10	596	460	488	382
MPH	382	255	694	776	792	382	455	891	622	661	596	10	209	854	809
NSH	246	194	563	672	698	246	328	696	486	533	460	209	10	659	600
ALX	618	735	715	856	909	618	700	37	397	261	488	854	659	10	106
RCH	532	634	611	751	805	532	605	143	292	155	382	809	600	106	10

1.2 Loop formation

We have a table of cities, which is giving below, about the distances between each of them:

We're composing loops composing of these cities, as a route schedule for them.

For example, $\{ORL, BNG, JAK\}$ means $WHS \rightarrow ORL \rightarrow BNG \rightarrow JAK->$ WHS

There are many ways to compose these loops. And for our loop, we must include all of these 14 cities. For all the loops, the WHS(werehouse station) is always the start and the destination.

So this shall be the first task: for given loop number LN, we shall divide the set of 14 cities C into LN of subsets SC_i, each with unique elements, and summing up they get the uniform set ($\bigcup_i SC_i = C$, $SC_i \cap SC_j = \emptyset$).

Then for each of the subset, we shall determine a "sequence", that minimize the distance in total if we determine to traverse through it. For example, the $\{BNG, JAK, ORL\}$ shall choose $\{WHS \rightarrow BNG \rightarrow JAK \rightarrow ORL \rightarrow WHS\}$ or sth.

1.3 Index obtaining

Now we get an instance of schedule, denoted by r_i, that is the most efficient route loop we could have out of the SC_i. We can then calculate some indexes according to them:

Most basically, we can get the total distance for each loop: D_{r_i}. Is is implemented by traversing the loop we designed.

Then, we can get the time needed to travel through these loops, by applying a function $T_{r_i} = V \cdot D_{r_i}$.

Also, we can get the information that by the end of each day, how many cities are visited, and how much of them are not. This shall be denoted by a sequence: $\{CW_i\}$. For example, for $\{ORL, BNG, JAK, ORL\}$, the first day the trunk visited the ORL (So there are 3 cities, BNG,JAK and ORL not visited and be waiting at this day), the sencond day it visited the BNG and JAK, and the last day it visited ORL and returned. Then the Customer Waiting sequence shall be $\{CW_i\} = \{3, 1, 0\}$.

We can further determine some quantities, that reflect how good overall (considering all the loops together) this route is. They are:

$$NT = \sum_i sup(T)_{r_i}$$

$$ML = \lambda \sum_i D_{r_i}$$

$$TDD = \lambda \sum_i \{glb(\frac{\lambda}{sup(T)_{r_i}}) \cdot T_{r_i} + [\lambda - glb(\frac{\lambda}{sup(T)_{r_i}}) \cdot sup(T)_{r_i}]\}$$

$$ACW = \sum_i \frac{1}{sup(T)_{r_i}} \sum_j (CW_j)r_i$$

$$MCW = \sum_i (CW_1)r_i$$

Where:

λ is the total days we need to consider here. In this game, $\lambda = 10$

glb is a function taking the round but for non-integer take the closest integer lower than it.

sup is a function taking the round but for non-integer take the closest integer higher than it.

NT is the Needed Trunk number. It is equal to the date that a loop is going to take since we're keeping sending trunks everyday before the last one comes back.

ML is the MiLeage took overall.

TDD is the total driver's days. For the actual case that $\lambda \to \infty$, the second term is negligible to the first term, so there's $\lim_{\lambda \to \infty} TDD = \frac{\lambda T_{r_i}}{sup(T)_{r_i}}$

ACW is the Average Customers Waiting. It shall be taken as: for loop i, the waiting guest number is 3,1,0, then the average waiting guest(city) shall be $\frac{3+1+0}{3} = \frac{4}{3}$. Then summing up the average waiting guest for all loops shall get us the overall ACW.

MCM is the maximum waiting customer number. It is obvious that at the first day of the delivery there will be the most customers. So we sum up the waiting cities at the first day as an estimation of the maximum waiting customer number.

Score calculation

According to the game setting, we have a final score to determine whether we're doing a great job:

$$S = w_1 NT + w_2 WL + w_3 TDD + w_4 ACW + w_5 MCM$$

In the .trk table, we obtain:

$$w_1 = 200,\ w_2 = 0.15,\ w_3 = 80,\ w_4 = 1900,\ w_5 = 400$$

So we use this as the metric to judge whether the loop design is good.

This basically gives us a metric $S = S(\{SC_i\}_\theta)$, and our purpose is to find the loop design that obtains the lowest score. That is:

$$\theta_0 = \arg\max_\theta \{S\{SC_i\}_\theta\}$$

And our method is to find all the possible loops design, compare their score, and then get the best result(s).

Before that, there shall be some more constraint to consider.

Constraints

The most clear constraint is that, the number of bus shall not exceed 10, since this is our capacity. That is:

$$NT \leq 10$$

The second constraint shall lead to the capacity of the trunks. The trunks carrying the shipments shall have enough capacity for the required loads. We have a table for the demand of each of the city:

Table 2: Orders by City and Day

Day	BNG	JAK	ORL	TPA	ATL	MAC	BTM	CHR	RAL	COL	MPH	NSH	ALX	RCH
0	2940	1313	1975	6589	5089	459	384	569	23877	803	132	1495	1851	1414
1	4304	1034	1121	12585	18097	355	750	7428	23913	3284	3772	1697	2243	1231
2	2191	1702	2805	17155	16740	1207	992	3780	2101	3762	1289	728	326	1696
3	2816	2126	2840	14059	13678	1783	878	7281	19296	63	2219	206	4706	5183
4	1798	243	2603	15391	14401	2757	782	2305	3204	973	1027	2351	4806	9005
5	2940	1313	1975	6589	5089	459	384	569	23877	803	132	1495	1851	1414
6	4304	1034	1121	12585	18097	355	750	7428	23913	3284	3772	1697	2243	1231
7	2191	1702	2805	17155	16740	1207	992	3780	2101	3762	1289	728	326	1696
8	2816	2126	2840	14059	13678	1783	878	7281	19296	63	2219	206	4706	5183
9	1798	243	2603	15391	14401	2757	782	2305	3204	973	1027	2351	4806	9005
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0

However, when calculating this program, we only consider the demand of a single day. i.e. a single row in this table. We will discuss later how to compose a more complex strategy that considers different requirements.

$$L = L_i(i \in r_i)$$

At last, we can estimate how much calculation would be implemented.

Dividing the cities into sets shall look like a combinatorial number problem. And if we consider all the possible number of loops, this shall require:

$$calculation = \sum_{k=1}^{10} C_{14}^{k} = C_{14}^{1} + C_{14}^{2} + \cdots + C_{14}^{10} = \sum_{k=1}^{10} C_{14}^{k}$$

Which shall be arount 14^{10}. This is too large.

More importantly, it is obvious that for the loop numbers that is large, it can hardly satisfy the two constraints above. For example, if there are 10 loops, each bus can work for at most 1 day. Which is rarely possible.

So we can give an empirical bound for the number of loops: 6.

$$Size(SC) \leq 6$$

We wouldn't consider the strategies that the number of loops exceeds 6.

Optimal route selection

After obtaining all the necessary parameters described above, we can use simply listing to get the optimal number.

The whole program shall be like this: (See Algorithm 1)

Algorithm 1 Maintain Top-k Loop Compositions with Minimum Score

```
Require: Desired leaderboard capacity $k = \texttt{record.size} \geq 1$
 1: record ← [ ] ▷ List of pairs (composition, score), kept ascending by score
 2: for $i \leftarrow 1$ to 6 do
 3:     use $i$ as the loop number
 4:     $\mathcal{C} \leftarrow$ GenerateCompositions($i$)
 5:     for all $c \in \mathcal{C}$ do                         ▷ each loop composition
 6:         idx ← ComputeIndexes($c$)
 7:         $s \leftarrow$ Score(idx)
 8:         if |record| < $k$ then
 9:             Append(record, ($c, s$))
10:             $p \leftarrow$ |record| − 1
11:         else if $s <$ record[|record| − 1].score then
12:             record[|record| − 1] ← ($c, s$)
13:             $p \leftarrow$ |record| − 1
14:         else
15:             continue                          ▷ worse than current tail; skip
16:         end if
17:         while $p > 0$ and record[$p$].score < record[$p - 1$].score do
18:             Swap(record[$p$], record[$p - 1$])         ▷ bubble upward to keep
    ascending order
19:             $p \leftarrow p - 1$
20:         end while
21:     end for
22:     print("[DEBUG] loop=", $i$, ", record=", record)
23: end for
24: return (record[0].composition, record)           ▷ optimal strategy and
    leaderboard
```

Modifications in coding

Best route creation algorithm

Originally, the optimal city visiting sequence was implemented by listing all permutations and picking the shortest tour. This is exact but incurs factorial time $\Theta(n!)$, which is prohibitive beyond a handful of cities (even after fixing the depot and removing rotational symmetry).

To keep exactness while cutting cost, we switch to the Held–Karp dynamic program, which runs in $\Theta(n^2 2^n)$ time and $\Theta(n 2^n)$ space. We memoize by city set (keyed by a frozenset) and use a compact depot+subset distance matrix so repeated subsets across partitions are solved once. In practice, loops with ≤ 8 cities are routed through Held–Karp (exact, fast enough).

For larger loops we add a deterministic approximate layer: build a tour via *Nearest Insertion* and then polish it with a single pass of *3-opt*. This yields near-optimal tours in roughly quadratic time for $n \leq 14$ (construction $O(n^2)$, light 3-opt), remains reproducible via fixed tie-breakers, and fits metric road distances. Our hybrid policy is therefore: Held–Karp for small loops; Nearest Insertion + 3-opt for larger ones.

The pseudo code shall look like:

Algorithm 2 0→4 Multi-Day Transport Optimizer (ultra-compact)

```
    $D, Q, \mathcal{C}, \mathrm{Cap}, T, H, L_{\max}, K, \tau, \alpha, \beta, \mathbf{w}$    ▷ distances, demands, cities, capacity, trucks, horizon, limits
 1: function ROUTE($S$)
 2:     $(r,\ell) \leftarrow \begin{cases} \text{Held–Karp}(S), & |S| \le \tau \\ \text{NI+2opt}(S), & \text{else} \end{cases}$; $t \leftarrow \alpha\ell + \beta|S|$; return $(r, \ell, \lceil t \rceil)$
 3: end function
 4: function SCORE($M$)
 5:     return $w_1 M_{\max T} + w_2 M_{\mathrm{mi}} + w_3 M_{\mathrm{dd}} + w_4 M_{\mathrm{avgW}} + w_5 M_{\mathrm{maxW}}$
 6: end function
 7: function SIM($C, T, H$)
 8:     simulate weekday plan across $H$ days with $T$ trucks; if any day needs more trucks than available return $\perp$;
 9:     collect $(M_{\max T}, M_{\mathrm{mi}}, M_{\mathrm{dd}}, M_{\mathrm{avgW}}, M_{\mathrm{maxW}})$; return $M$
10: end function
11: 0 Setup  set daily truck-day cap $B$; for $d = 0..4$ initialize candidate list $\mathcal{T}_d$.
12: 1–3 Single-Day Candidates
13: for $d = 0$ to 4 do
14:     for $m = 1$ to $L_{\max}$ do
15:         for each partition $P = \{S_1, \dots, S_m\}$ of $\mathcal{C}$ do
16:             if $\sum_{c \in S_j} Q[d][c] \le \mathrm{Cap}\ \forall j$ and $\sum_j \mathrm{Route}(S_j).\mathrm{dur} \le B$, keep loops;
17:         end for
18:     end for
19:     $\mathcal{T}_d \leftarrow \mathrm{TOPK}(\text{feasible, key} = (\sum \mathrm{dur}, \sum \ell))$    ▷ 3 Keep
20: end for
21: 4 Calendar Composition
22: $(C^{,M^{,s}})^{\leftarrow(\varnothing,\varnothing,+\infty)}$
23: for each $C \in \mathcal{T}_0 \times \cdots \times \mathcal{T}_4$ do
24:     $M \leftarrow \mathrm{SIM}(C, T, H)$; if $M = \perp$ continue; $s \leftarrow \mathrm{SCORE}(M)$
25:     if $s < s$ then $(C^{,M^{,s}})^{\leftarrow(C,M,s)}$
26:     end if
27: end for
28: return $(C^{,M^{,s}})$
```

Driver time calculation

The time shall actually be calculated by:

$$Days \approx 0.00234 \times mileage + 0.053 \times (number\ of\ cities)$$

So the earlier assumption that $T_{r_i} = V \cdot D_{r_i}$ is actually not correct.

Running result

```
============================================================
Transportation Route Optimization System
Optimizing for Day 0 demand
============================================================

[Processing] Analyzing 1 loop(s)...
  Found 0 valid configurations

[Processing] Analyzing 2 loop(s)...
  Found 82 valid configurations
  Current best score: 19811.47

[Processing] Analyzing 3 loop(s)...
  Found 55 valid configurations
  Current best score: 18761.67

[Processing] Analyzing 4 loop(s)...
    [Debug] Analyzed 10000000 loop settings for 4 loop(s)...
  Found 0 valid configurations
  Current best score: 18761.67

[Processing] Analyzing 5 loop(s)...
    [Debug] Analyzed 10000000 loop settings for 5 loop(s)...
    [Debug] Analyzed 20000000 loop settings for 5 loop(s)...
    [Debug] Analyzed 30000000 loop settings for 5 loop(s)...
    [Debug] Analyzed 40000000 loop settings for 5 loop(s)...
  Found 0 valid configurations
  Current best score: 18761.67

[Processing] Analyzing 6 loop(s)...
    [Debug] Analyzed 10000000 loop settings for 6 loop(s)...
    [Debug] Analyzed 20000000 loop settings for 6 loop(s)...
    [Debug] Analyzed 30000000 loop settings for 6 loop(s)...
    [Debug] Analyzed 40000000 loop settings for 6 loop(s)...
    [Debug] Analyzed 50000000 loop settings for 6 loop(s)...
    [Debug] Analyzed 60000000 loop settings for 6 loop(s)...
  Found 0 valid configurations
  Current best score: 18761.67

============================================================
OPTIMIZATION COMPLETE
Time taken: 1672.36 seconds
============================================================

TOP 3 SOLUTIONS:

============================================================
Rank #1 Solution - Score: 18761.67
============================================================

Loop Configuration (3 loops):
  Loop 1: WHS → COL → CHR → RAL → RCH → BTM → ALX → NSH → MPH → WHS
          Distance: 2034.0 miles
  Loop 2: WHS → ATL → BNG → WHS
          Distance: 310.0 miles
  Loop 3: WHS → JAK → ORL → TPA → MAC → WHS
          Distance: 1025.0 miles

Performance Metrics:
  • Needed Trucks (NT):           10
  • Total Mileage (ML):           3369.0 miles
  • Total Driver Days (TDD):     26.3 days
  • Avg Customers Waiting (ACW):  4.50
  • Max Customers Waiting (MCW):  14
  • Max Loop Demand:              30,525 units

Score Components:
  • NT component:  200 × 10 = 2000.00
  • ML component:  0.15 × 3369.0 = 505.35
  • TDD component: 80 × 26.3 = 2106.32
  • ACW component: 1900 × 4.50 = 8550.00
  • MCW component: 400 × 14 = 5600.00

============================================================
Rank #2 Solution - Score: 19201.43
============================================================

Loop Configuration (3 loops):
  Loop 1: WHS → COL → RAL → RCH → ALX → BTM → CHR → WHS
          Distance: 1397.0 miles
  Loop 2: WHS → MAC → ATL → WHS
          Distance: 174.0 miles
  Loop 3: WHS → NSH → MPH → BNG → TPA → ORL → JAK → WHS
```

```
          Distance: 1822.0 miles

Performance Metrics:
  • Needed Trucks (NT):           10
  • Total Mileage (ML):           3393.0 miles
  • Total Driver Days (TDD):     23.5 days
  • Avg Customers Waiting (ACW):  4.85
  • Max Customers Waiting (MCW):  14
  • Max Loop Demand:              28,898 units

Score Components:
  • NT component:  200 × 10 = 2000.00
  • ML component:  0.15 × 3393.0 = 508.95
  • TDD component: 80 × 23.5 = 1877.48
  • ACW component: 1900 × 4.85 = 9215.00
  • MCW component: 400 × 14 = 5600.00

============================================================
Rank #3 Solution - Score: 19238.22
============================================================

Loop Configuration (3 loops):
  Loop 1: WHS → COL → RAL → RCH → ALX → BTM → CHR → WHS
          Distance: 1397.0 miles
  Loop 2: WHS → ATL → WHS
          Distance: 20.0 miles
  Loop 3: WHS → NSH → MPH → BNG → TPA → ORL → JAK → MAC → WHS
          Distance: 1826.0 miles

Performance Metrics:
  • Needed Trucks (NT):           10
  • Total Mileage (ML):           3243.0 miles
  • Total Driver Days (TDD):     19.5 days
  • Avg Customers Waiting (ACW):  5.05
  • Max Customers Waiting (MCW):  14
  • Max Loop Demand:              28,898 units

Score Components:
  • NT component:  200 × 10 = 2000.00
  • ML component:  0.15 × 3243.0 = 486.45
  • TDD component: 80 × 19.5 = 1556.77
  • ACW component: 1900 × 5.05 = 9595.00
  • MCW component: 400 × 14 = 5600.00

============================================================
SUMMARY
```

```
============================================================
The optimal strategy uses 3 loop(s)
with a total score of 18761.67

This configuration requires 10 trucks
and covers 3369.0 miles in total.
```

2 Further improvement: multi-day demand design

The algorithm introduced above gives the simulation upon a single day demand. This is a simplification, since it allows the scores to be calculated easily, and we can avoid the combinatorial optimization problem. However, our ultimate goal is to find the best overall strategy design. So now we shall jump into this full version of problem.

Our strategy here shall be: we run the single day version of the optimization for each of the day in weekdays, and we keep the record for them respectively. In each of the record, it records the route design strategy that perform the best in each of the days. Then, we shall make them to further design the final delivery schedule.

It shall be like this: in 10 days we consider, for the first day, we pick one of the strategy from the best route designs in the monday record:

```
Loop Configuration (3 loops):
  Loop 1: WHS → COL → CHR → RAL → RCH → BTM → ALX → NSH → MPH → WHS
          Distance: 2034.0 miles
  Loop 2: WHS → ATL → BNG → WHS
          Distance: 310.0 miles
  Loop 3: WHS → JAK → ORL → TPA → MAC → WHS
          Distance: 1025.0 miles
```

We have 10 trunks in total. Then we will send the buses of number 1, 2, 3 to carry out the request. They shall be went out to deliver the goods, and during that they shall be unavailable. For this example, the bus one will deliver the work for loop 1 for 6 days, the bus two will deliver the work for loop 2 for 1 day, and the bus three will be deliver the work for loop 3 for 1 day.

Then on the Tuesday, we can use the one of the best designs from the result of Tuesday. Now the 1st and 3rd buses are ocupied, so we can use 2, 4, 5 and sth to carry out the mission.

This shall repeat until all 10 days are implemented. We will try each of the compositions composed of the best strategies obtained from the demand of every single day. So there will be

$$(record\ size)^5$$

of the compositions in total.

We will calculate the scores for each of these compositions, and get the best route design composition.

In this multi-day simulation, the calculation of indexes, and the setting of the constraints shall be different.

Originally:

$$NT = \sum_i sup(T)_{r_i}$$

$$ML = \lambda \sum_i D_{r_i}$$

$$TDD = \lambda \sum_i \{glb(\frac{\lambda}{sup(T)_{r_i}}) \cdot T_{r_i} + [\lambda - glb(\frac{\lambda}{sup(T)_{r_i}}) \cdot sup(T)_{r_i}]\}$$

$$ACW = \sum_i \frac{1}{sup(T)_{r_i}} \sum_j (CW_j) r_i$$

$$MCW = \sum_i (CW_1) r_i$$

And for this new version:

NT: can't be calculated. Record the bus used in simulation, and try not to use new buses unless necessary.

$$TDD = \sum_{date} DD - \sum_{end} DD$$

(That is, the TDD shall equal to the sum of driver's day for all dates' strategy, minus the undone days at the end of the calculated data (the 10th for this case))

And it can be proven that:

$$\lim_{Date \to \infty} TDD = \frac{Date}{5} \sum_{i=1}^{5} \text{DD on the i-th weekday}$$

$$ACW = Average(\sum_{\text{date i}} \frac{1}{sup(T)_{r_i}} \sum_j (CW_j) r_i)$$

MCW needs to be counted everyday and take the maximum

For simplicity, we neglect the MCW index. That is, we take MCW = 14 for all cases.

Also, the constraint for the overall case shall be:

There shall be always bus available to carry all the works, otherwise this combination fails.

If a combination satisfies this, we calculate its score, and compare it with the combinatorial route strategy recorder, and get the best combinatorial strategies.

2.1 Result

```
============================================================
OPTIMAL MULTI-DAY SOLUTION
============================================================

Overall Score: 45439.00

Weekday Strategy Selection:

Monday:
  Loop 1: WHS → CHR → RAL → RCH → ALX → BTM → COL → BNG → MPH → NSH → WHS
          Duration: 6 days, Distance: 2265.0 miles
  Loop 2: WHS → ATL → JAK → ORL → TPA → MAC → WHS
          Duration: 3 days, Distance: 1035.0 miles

Tuesday:
  Loop 1: WHS → MPH → NSH → ALX → BTM → RCH → RAL → WHS
          Duration: 5 days, Distance: 1962.0 miles
  Loop 2: WHS → BNG → ATL → WHS
          Duration: 1 days, Distance: 310.0 miles
  Loop 3: WHS → COL → CHR → JAK → ORL → TPA → MAC → WHS
          Duration: 4 days, Distance: 1377.0 miles

Wednesday:
  Loop 1: WHS → MAC → TPA → ORL → JAK → CHR → RAL → RCH → ALX → BTM → COL → WHS
          Duration: 6 days, Distance: 2229.0 miles
  Loop 2: WHS → ATL → BNG → MPH → NSH → WHS
          Duration: 3 days, Distance: 870.0 miles

Thursday:
  Loop 1: WHS → MPH → NSH → ALX → BTM → RCH → RAL → WHS
          Duration: 5 days, Distance: 1962.0 miles
  Loop 2: WHS → BNG → ATL → WHS
          Duration: 1 days, Distance: 310.0 miles
  Loop 3: WHS → COL → CHR → JAK → ORL → TPA → MAC → WHS
          Duration: 4 days, Distance: 1377.0 miles

Friday:
  Loop 1: WHS → ATL → CHR → BTM → ALX → RCH → RAL → COL → WHS
          Duration: 4 days, Distance: 1407.0 miles
  Loop 2: WHS → MAC → JAK → ORL → TPA → BNG → MPH → NSH → WHS
          Duration: 5 days, Distance: 1826.0 miles

Performance Metrics:
  • Maximum Trucks Used:        10
  • Total Mileage:             33860.0 miles
  • Total Driver Days:         77
  • Average Customers Waiting: 14.00
  • Maximum Customers Waiting: 14

Daily Truck Usage:
  Day  1 (Mon): 2 trucks active
  Day  2 (Tue): 5 trucks active
  Day  3 (Wed): 6 trucks active
  Day  4 (Thu): 8 trucks active
  Day  5 (Fri): 9 trucks active
  Day  6 (Mon): 9 trucks active
  Day  7 (Tue): 10 trucks active
  Day  8 (Wed): 10 trucks active
  Day  9 (Thu): 9 trucks active
  Day 10 (Fri): 9 trucks active

Score Breakdown:
  • Trucks:    200 × 10 = 2000.00
  • Mileage:   0.15 × 33860.0 = 5079.00
  • Driver Days: 80 × 77 = 6160.00
  • Avg Wait:  1900 × 14.00 = 26600.00
  • Max Wait:  400 × 14 = 5600.00

============================================================
Total optimization time: 5910.60 seconds
============================================================
```

Cost optimization

If we want to directly optimize the cost of the game, we can simply adjust the scoring weights to the cost calculated to hold the bus, pay the driver and other mileage related cost. The weight we adapted according to our calculation should be:

$$w_1 = 4431,\ w_2 = 0.345,\ w_3 = 690,\ w_4 = 0,\ w_5 = 0$$

The running result shall look like:

```
============================================================
MULTI-DAY TRANSPORTATION OPTIMIZATION SYSTEM
============================================================

Initializing optimizer...
Starting optimization process...
============================================================
```

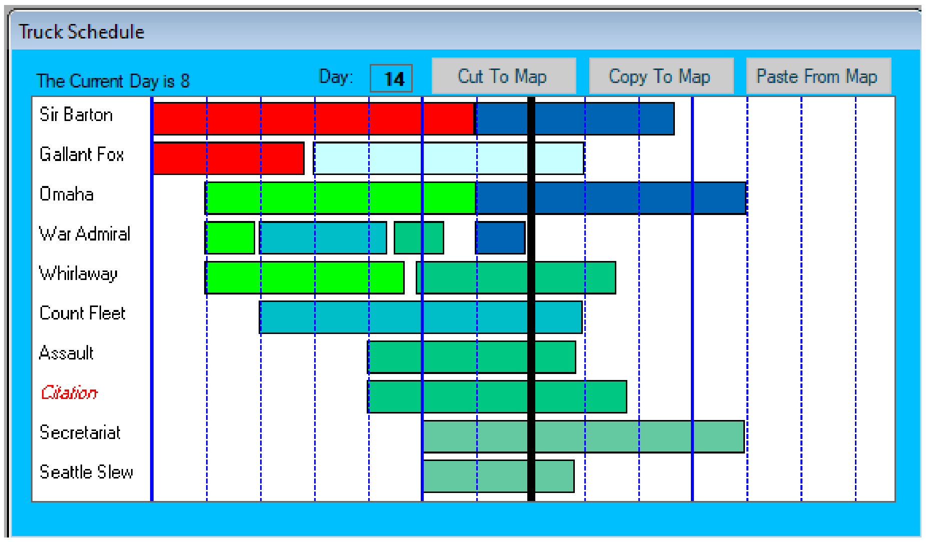

Figure 1: Demo for the final optimization

```
Phase 1: Single-Day Optimization
============================================================

Optimizing for Monday (Day 0)...
  Found 5 top strategies
  Best: 2 loops, total distance: 3233.0 miles

Optimizing for Tuesday (Day 1)...
  Found 1 top strategies
  Best: 3 loops, total distance: 3649.0 miles

Optimizing for Wednesday (Day 2)...
  Found 5 top strategies
  Best: 2 loops, total distance: 3099.0 miles

Optimizing for Thursday (Day 3)...
  Found 1 top strategies
  Best: 3 loops, total distance: 3649.0 miles

Optimizing for Friday (Day 4)...
  Found 5 top strategies
  Best: 2 loops, total distance: 3233.0 miles

============================================================
Phase 2: Multi-Day Composition Search
```

```
============================================================

Total compositions to evaluate: 3,125
  Evaluated: 100/3125 (3.2%), Feasible: 80

Evaluation complete!
  Total evaluated: 125
  Feasible solutions: 100
  Best score: 109121.70

============================================================
OPTIMAL MULTI-DAY SOLUTION
============================================================

Overall Score: 109121.70

Weekday Strategy Selection:

Monday:
  Loop 1: WHS → CHR → RAL → RCH → ALX → BTM → COL → BNG → MPH → NSH → WHS
          Duration: 6 days, Distance: 2265.0 miles
  Loop 2: WHS → JAK → ORL → TPA → MAC → ATL → WHS
          Duration: 3 days, Distance: 1035.0 miles

Tuesday:
  Loop 1: WHS → MPH → NSH → BTM → ALX → RCH → RAL → WHS
          Duration: 5 days, Distance: 1962.0 miles
  Loop 2: WHS → BNG → ATL → WHS
          Duration: 1 days, Distance: 310.0 miles
  Loop 3: WHS → MAC → TPA → ORL → JAK → CHR → COL → WHS
          Duration: 4 days, Distance: 1377.0 miles

Wednesday:
  Loop 1: WHS → MAC → TPA → ORL → JAK → CHR → RAL → RCH → ALX → BTM → COL → WHS
          Duration: 6 days, Distance: 2229.0 miles
  Loop 2: WHS → ATL → BNG → MPH → NSH → WHS
          Duration: 3 days, Distance: 870.0 miles

Thursday:
  Loop 1: WHS → MPH → NSH → BTM → ALX → RCH → RAL → WHS
          Duration: 5 days, Distance: 1962.0 miles
  Loop 2: WHS → BNG → ATL → WHS
          Duration: 1 days, Distance: 310.0 miles
  Loop 3: WHS → MAC → TPA → ORL → JAK → CHR → COL → WHS
          Duration: 4 days, Distance: 1377.0 miles
```

```
Friday:
  Loop 1: WHS → ATL → CHR → BTM → ALX → RCH → RAL → COL → WHS
          Duration: 4 days, Distance: 1407.0 miles
  Loop 2: WHS → NSH → MPH → BNG → TPA → ORL → JAK → MAC → WHS
          Duration: 5 days, Distance: 1826.0 miles

Performance Metrics:
  • Maximum Trucks Used:        10
  • Total Mileage:             33860.0 miles
  • Total Driver Days:         77

Daily Truck Usage:
  Day  1 (Mon): 2 trucks active
  Day  2 (Tue): 5 trucks active
  Day  3 (Wed): 6 trucks active
  Day  4 (Thu): 8 trucks active
  Day  5 (Fri): 9 trucks active
  Day  6 (Mon): 9 trucks active
  Day  7 (Tue): 10 trucks active
  Day  8 (Wed): 10 trucks active
  Day  9 (Thu): 9 trucks active
  Day 10 (Fri): 9 trucks active

Score Breakdown:
  • Trucks:    4431 × 10 = 44310.00
  • Mileage:   0.345 × 33860.0 = 11681.70
  • Driver Days: 690 × 77 = 53130.00

============================================================
Total optimization time: 5985.01 seconds
============================================================
```