Minimization of Trucker Idle Time and Transportation Cost in Container Delivery

Alexis Koch, Krystal Diaz, Matt Slouka / Dr. Christine Nguyen

Department of Industrial and Systems Engineering, College of Engineering and Engineering Technology, Northern Illinois University

Northern Illinois University

Introduction

A furniture company must deliver containers of furniture to their customers' locations. However, the furniture company does not have the necessary resources to carry out their own deliveries. Consequently, they outsource to a third-party logistics company that provides trucks, truckers, and containers. They will transport the containers using whatever policy works best for the furniture company. The current policy to deliver their products takes a lengthy amount of time. The driver will wait while the product is being unloaded and then returns to the depot with the now empty container to be loaded with product for another customer. The wait times make the transportation cost more expensive. The team investigated and produced a new policy to help utilize the truckers more efficiently.

Problem Description

Currently, the truckers wait extended periods of time for the customer to unload the product out of the containers. The issue is the wait time is too long and is variable, which adds to the cost of transportation. The delay also increases the need for more truckers to deliver containers to all customers.

Objective

The objective is to develop a new policy that will:

1. Minimize the overall transportation cost
2. Minimize the trucker idle time

Scope

The project only considers changes to how the truckers will transport the containers to and from the customer locations. The routing used, product packaging and the loading/unloading process are outside the scope of the project.

Current Policy

Under the current policy, the trucker begins at the depot/warehouse (blue box in figure 1) The warehouse is where the truckers retrieve the product already loaded into full containers. They transport it to the respected customer and wait for the customer to unload the container, which varies and can take hours. Once emptied, the trucker takes the empty container back to the warehouse to be reused, pick up another full container and continue with the deliveries. The cycle is repeated until they are done with the number of tasks in their work shift. This policy is modeled as an integer linear programming optimization problem.

Minimize

$$\sum_{v\in V}\sum_{i\in I}\sum_{t\in T}[c^R w_{vit} + c^F w_{vit} + c^E w_{vit}] + \sum_{v\in V}\sum_{t\in T} z_{vt}$$

Subject to

$$\sum_{V\in V} w_{vit} \geq f_{it}, \forall\, i \in L, t \in T$$

$$f_{it} = f_{it-1} + d_{it} - \sum_{V\in V} w_{vit-1}, \forall\, i \in L, t \in T$$

$$f_{i1} = d_{i1}, \quad \forall\, i \in L, t \in T$$

$$\sum_{i\in L} w_{vit}(2r_i + a) \leq K^M z_{vt}, \forall v \in V, t \in T$$

$$y_{vit}, f_{it} \geq 0, \forall v \in V, i \in L, t \in T$$

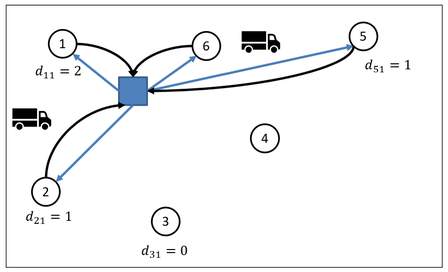

Figure 1. Current Policy Route Example

Proposed Policy

Instead of having the trucker wait at a customer location for the customer to unload it, the team created a policy where the trucker no longer had idle time. The trucker will leave the depot, arrive to the customer location, and leave the container at the customer location. The trucker will then leave to complete other tasks such as, picking up an empty container at another location or going to the depot to drop off a container to another customer location. The mathematical formulation is below:

Minimize

$$\sum_{t\in T}\sum_{v\in V}\sum_{i\in L}\sum_{j\in L}\left(c^R(w_{vijt} + x_{vijt}) + c^F s_{ij} w_{vijt} + c^E s_{ij} x_{vijt} + c^N s_{ij} y_{vijt}\right) + \sum_{t\in T}\sum_{v\in V} Z_{vt}$$

Subject to

$$\sum_{i\in L}\sum_{j\in L} r_{ij}\left(w_{vijt} + x_{vijt} + y_{vijt}\right) \leq K^M z_{vt}, \ \forall t \in T, v \in V$$

$$\sum_{i\in L}\sum_{j\in L} r_{ij} + x_{vi0t} + y_{vi0t} = \sum_{j\in L} w_{v0jt} + y_{v0jt}, \ \forall t \in T$$

$$\sum_{V\in V} w_{vit} \geq f_{it}, \forall\, i \in L, t \in T$$

$$\sum_{v\in V}\sum_{j\in L} y_{vijt} + w_{v0it} = \sum_{v\in V} x_{vi0t} + \sum_{j\in L} y_{vijt} + I_{it}, \ \forall i \in L, t \in T$$

$$f_{i1} = d_{i1}, \quad \forall\, i \in L, t \in T$$

$$f_{it} = f_{it-1} + d_{it} - \sum_{V\in V} w_{v,0,i,t-1}, \forall\, i \in L, t \in T$$

$$\sum_{v\in V} x_{vi0t} \geq I_{i,t-1}, \forall i \in L$$

$$I_{i1} = \sum_{v\in V} -x_{vi0t} + w_{v0i1}, \ \forall i \in L$$

$$I_{it} = I_{i,t-1} + \sum_{v\in V} -x_{vi0t} + w_{v0it}, \ \forall i \in L, t \in \{2 \dots |T|\}$$

Methodology

Python and IBM ILOG CPLEX optimization solver were used to solve the problem. In addition, the packages docplex, cplex, pandas, openpxyl, and numpy were used. Decision variables, constraints, and the objective function was defined in the python code. Then the code was executed to output the data into Excel files. Figures 2-4 show excerpts of the python code. In order to determine whether the proposed policy truly improved over the current state, 4 scenarios were created. The first 2 analyzed a network where customer locations were near and they had 5 and 10 locations respectively. (These are the "Near" scenarios in figures 5 and 6). The 3rd and 4th scenarios analyzed a network where customer locations were farther away, and they had 5 and 10 locations respectively. (These are the "Far" scenarios in figures 5 and 6).

```python
# Create constraints

for i in I:
    for t in range(1, TIME_HORIZON + 1):
        model.add_constraint(model.sum(y[v, i, t] for v in V) >= f[i, t])

for i in I:
    model.add_constraint(f[i, 1] == this_demand[i - 1][0])

for i in I:
    for t in range(2, TIME_HORIZON + 1):
        model.add_constraint(f[i, t] == f[i,t-1] + this_demand[i - 1][t - 1] - model.sum(y[v, i, t-1] for v in V_))
for i in I:
    model.add_constraint(
        model.sum(y[v, i, t] for v in V for t in T) == model.sum(this_demand[i - 1][t - 1] for t in T))

## Constraint: TOTAL_TIME is one way?
for v in V:
    for t in T:
        model.add_constraint(
            model.sum((y[v, i, t]) * (2 * this_travel_time[i - 1] + WAIT_TIME) for i in I) <= MAX_WORK_TIME * z[v, t])
```

Figure 2. Python: Current Policy Constraints

```python
# Create objective function
total_rental_cost = model.sum(COST_RENTAL * y[v, i, t] for v in V for i in I for t in T)
full_transportation_cost = model.sum(COST_TRANSPORT_FULL *
                                     this_travel_distance[i - 1] * y[v, i, t] for v in V for i in I for t in T)
empty_transportation_cost = model.sum(COST_TRANSPORT_EMPTY *
                                      this_travel_distance[i - 1] * y[v, i, t] for v in V for i in I for t in T)
total_wait_cost = model.sum(COST_WAIT / 60 * (WAIT_TIME * y[v, i, t]) for v in V for i in I for t in T)
total_truckers = model.sum(TRUCK_WEIGHT * z[v, t] for v in V for t in T)

model.add_kpi(total_rental_cost, "Total Rental Cost")
model.add_kpi(full_transportation_cost, "Full Transportation Cost")
model.add_kpi(empty_transportation_cost, "Empty Transportation Cost")
model.add_kpi(total_wait_cost, "Total Wait Cost")
model.add_kpi(total_truckers, "total truckers")

model.minimize(total_rental_cost + full_transportation_cost +
               empty_transportation_cost + total_wait_cost + total_truckers)
```

Figure 3. Python: Current Policy Objective Function

```python
# Create objective function

total_rental_cost = model.sum(
    COST_RENTAL * (w[v, i, j, t] + x[v, i, j, t]) for v in V for i in I for j in J for t in T)
full_transportation_cost = model.sum(
    COST_TRANSPORT_FULL * this_travel_distance[i - 1][j - 1] * w[v, i, j, t] for v in V for i in I for j in J for t
    in T)
empty_transportation_cost = model.sum(
    COST_TRANSPORT_EMPTY * this_travel_distance[i - 1][j - 1] * x[v, i, j, t] for v in V for i in I for j in J for t
    in
    T)
no_container_cost = model.sum(
    COST_NO_CONTAINER * this_travel_distance[i - 1][j - 1] * y[v, i, j, t] for v in V for i in I for j in J for t in
    T)
total_truckers = model.sum(TRUCK_WEIGHT * z[v, t] for v in V for t in T)

model.add_kpi(total_rental_cost, "Total Rental Cost")
model.add_kpi(full_transportation_cost, "Full Transportation Cost")
model.add_kpi(empty_transportation_cost, "Empty Transportation Cost")
model.add_kpi(no_container_cost, "Total No Container Cost")
model.add_kpi(total_truckers, "Total Truckers Cost")
```

Figure 4. Python: Proposed Policy Constraints

Results

The optimization program was set up to perform 20 random instances of each scenario, which was analyzed with the current and proposed policies. The rental cost, travel cost, and total truckers were compared between the current state and proposed state. The first cost associated with the system is the rental cost, which was higher in every state of the proposed policy. An increase in capacity leads to an increase in container rentals. The total transport cost of the proposed state is 1.5% to 4% lower in all states of the proposed policy. The number of truckers used in the system was reduced by 28% to 60% in the proposed policy.

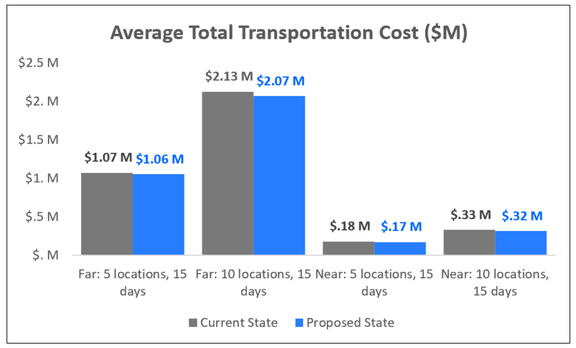

Figure 5. Average Total Transportation Cost Proposed vs. Current Policy

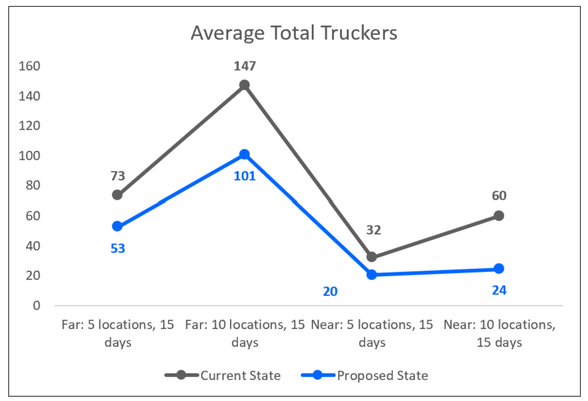

Figure 6. Average Total Truckers Proposed vs. Current Policy

Conclusions

The team has concluded that the proposed policy is beneficial for both near and far locations. Regardless the transportation cost did increase, the idle time has decreased. In addition, the proposed policy is more efficient for the furniture company.

Acknowledgments

The team would like to thank Northern Illinois University Department of Engineering and Engineering Technology. Thank you to the Department of Industrial and Systems faculty. The team would like to give a special thanks to Dr. Christine Nguyen for the constant help, knowledge, and support.