A Dynamic-Programming-Based Approach for Optimizing Municipal Solid Waste Collection Routes

Yuxi Chen, Bole Pan, Haoran Yuan
Team 17

Columbia University,
New York City, NY,
USA, 10027

Summary

To address waste management inefficiencies in New York City, we consider garbage as heaps arranged in a given sequence awaiting collection by collection truck. We use dynamic programming to find optimal routes for garbage trucks given the constraint of having to depart from and return to one garage. Then, we employ a maximum-flow-minimum-cost algorithm to obtain the optimal routes when multiple garages are given. These calculations are performed on a 2x2 grid. We apply our algorithm on Scholes Street, Brooklyn to verify the feasibility of our finding. We generate the fitness function for Genetic Algorithms based on our formal models, and we try to compute the optimal sequence with Genetic Algorithm using real map data from Hell's Kitchen, Manhattan.

1 Introduction

The accumulation and mismanagement of municipal solid waste in New York City have persisted as a baleful issue. Deficiency in existing waste management arrangements and procedures continues to threaten public health. The tantamount quantities of waste produced by New York City each day burdens not only itself but neighboring areas socially, economically, and environmentally. Therefore, careful analysis is required to reconstruct, reform, and rethink the ways of how the city manages its waste.

According to the Department of Sanitation of New York City (DSNY), New Yorkers produce more than 12,500 tons of trash per day. These wastes are handled by more than 2000 DSNY and 4000 privately owned garbage trucks. However, there persists a stunningly inefficient allocation of garbage trucks in New York City. Currently, the average garbage truck can hold up to approximately 20 tons of waste. In contrast, one such truck operating in New York processes no more than 12,500/6,000 = 2.0833 tons on a daily average. Thus, one New York garbage truck employs no more than 2.0833/20 * 100% = 10.4% of its total capacity.

Therefore, in an effort to reduce expenditure and improve efficiency regarding garbage disposal in New York City, we seek to maximize the carrying capacity of individual garbage trucks, by which the overall truck fleet size will also be reduced. This will not only reduce annual costs of waste disposal in New York City, currently standing at $432 million, by lowering the number of trucks in service (including an initial purchase cost, ranging between $250,000 and $500,000, and subsequent maintenance costs) but also improve civil satisfaction by less occurrence of clamor, congestion, and stench associated with these trucks.

We first employ dynamic programming to generate the optimal paths for trucks when only one truck garage is concerned and when a sequence of visiting orders is given. Based on the first step, we employ maximum-flow-minimum-cost(MFMC) algorithm to calculate the optimal paths when multiple garages are considered. We demonstrate the first two steps in a 2x2 grid. In the third step, we attempt to use a Genetic Algorithm to compute the optimal sequence.

2 Problem Analysis

The urban pattern in New York largely conforms to a grid layout. Multiple garbage trucks start simultaneously to collect waste and conclude around approximately the same time (given relatively similar amounts of work). There exist multiple garages in the city, which leads to various possibilities for a garbage truck to depart from and return to.

Therefore, the model is dedicated to developing an efficient route management policy that involves multiple garbage trucks and garages in a grid layout.

3 Assumptions

1. Streets in New York can be represented as and conforms to a grid layout.

Justification: The urban layout in New York is extremely grid-like. For convenience in computing distance between locations, we decide to view streets in a grid layout.

2. All garbage trucks operate during similar time slots. The work has to be shared almost evenly between each truck.

Justification: The actual hours of operation for garbage trucks are typically constricted between before dawn and around midnight. The work should be relatively comparable, that is, the sanitation workers should finish and return at around the same time. No workers should be treated unjustly.

3. All garbage is located and will be collected at the midpoint of each street or avenue.

Justification: The status of each heap of garbage represents a location to be arrived at by a certain garbage collection vehicle. This is ideally achieved when there exists certain discrete locations spread evenly compared to a number of locations scattered continuously across the grid. The trucks therefore should return from the midpoint after collecting the last heap of garbage in lieu of completing the whole course of the last street.

4. The entirety of any heap of garbage at any location is to be collected by one garbage truck in one operation.

Justification: This is a realistic representation of how garbage is managed daily, that is, each heap at any location is either collected or neglected in one operation. Sanitation workers will not leave half heap at a location.

4 Modeling

4.1 Model Introduction

We identify the optimization of garbage truck routes as a NP-hard problem, which is too costly to be solved by a deterministic algorithm. Hence, we attempt to employ a Genetic Algorithm to solve it. To use a Genetic Algorithm, we need to generate a numerical sequence together with its fitness function. Therefore, we generate sequences by listing the trash heaps, which are located on the midpoints of the streets.

The fitness function of the sequence is characterized by the collective minimal time spent when trucks visit the trash heaps in the order of the sequence. Collective minimal time refers to the minimal time that the last truck completes its work. We then employ dynamic programming and maximal-flow-minimal-cost algorithm to calculate the collective minimal time of a given sequence——that is, in a given sequence of garbage heaps, we aim to let the last sanitation worker who completes his work finish his work at the earliest time possible.

Thus, we let the collective minimum time of the collection process be represented as T. It is determined by the time required for the last vehicle to complete collection duties at its assigned locations and return. T can be represented by the following equation:

$$T = \max(T_k, k = 1, 2, \ldots, m)$$

where T_k is the time needed for each vehicle (k = 1, 2, ..., m) to complete their assigned collection duties.

Table 1: Table of Variables

Variable Name	Meaning
T	The minimum time of the collection process
k	Generic Variable
n, q	The number of trucks in total
x	The number of trash heaps left
s	The number of trucks left
d_i, j	Time cost to travel from the i-th heap to j-th
$u(i, n)$	Time cost to travel from i-th heap to the $(i+n-1)$-th
p	The number of garages
c	The capacity of the garage

4.2 Dynamic Programming

We use dynamic programming to solve the case when there is only one garage.

Let there be n trucks in total. Given a sequence of the garbage heaps, to share work between n trucks is to compartmentalize the sequence into n pieces. We decide a car as well as the garbage heaps it (visits and) collects; therefore, the problem can be regarded as a n-stage decision problem. The status at each stage is defined as (x, s), where x is the number of trash heaps left and s is the number of trucks left. The decision set is defined as $\{1, 2, ..., k\}$, which means that the truck is able to collect all the garbage from the next k heaps and we can decide to let it collect from the next k_1 heaps, where $k_i \in \{1, 2, ..., k\}$.

Based on our assumption of the collective minimum time T, we generate a function of x and s to represent the collective minimal time:

$$f(x, s) = \min\{\max\{u(i, n), f(x-n, s-1)\}\}$$

where i suggests that we are at the i-th garbage heap in the sequence, that is, we are done with dispatching the former $i-1$ to the trucks. n indicates the number of heaps that will be visited by the next truck. $u(i, n)$ indicates the time cost to travel from the i-th garbage heap to the $(i+n-1)$-th. $f(x-n, s-1)$ stands for the situation when the next truck will visit n heaps – so there will be $x-n$ heaps and $s-1$ trucks left. This formula of $f(x, s)$ thus stands because it says: the last truck which completes its duties will either be the next one, who will collect garbage from the i-th, $i+1$-th, ..., $i+n-1$-th garbage heap, or will be in the following trucks awaiting to be dispatched, in $f(x-n, s-1)$.

Let $d_{i,j}$ represent the time cost to travel from the i-th garbage heap to the (j)-th. $u(i,n)$ can be defined as follows:

$$u(i,n) = d_{0,i} + \Sigma_i^{i+n-2} d_{t,t+1} + d_{i+n-1,0}$$

because the truck departs from the garage, collects the garbage, and returns to the garage.

Note that when the s trucks left are not able to carry all the garbage in the x heaps left, $f(x,s) = \infty$

4.2.1 Demonstration of Dynamic Programming

Here we present our dynamic programming model on a 2x2 grid, where there are 12 garbage locations and 3 trucks. See appendix for the our code.

Let the garbage heaps be labeled 1-2-...-11-12 as shown in **Figure 1**. We set the sequence to be 1-2-3-...-10-11-12 as well. We visit heaps in this sequence. Note that the sequence can be different, for instance 12-...-1, and 1-...-12 is not the optimal sequence.

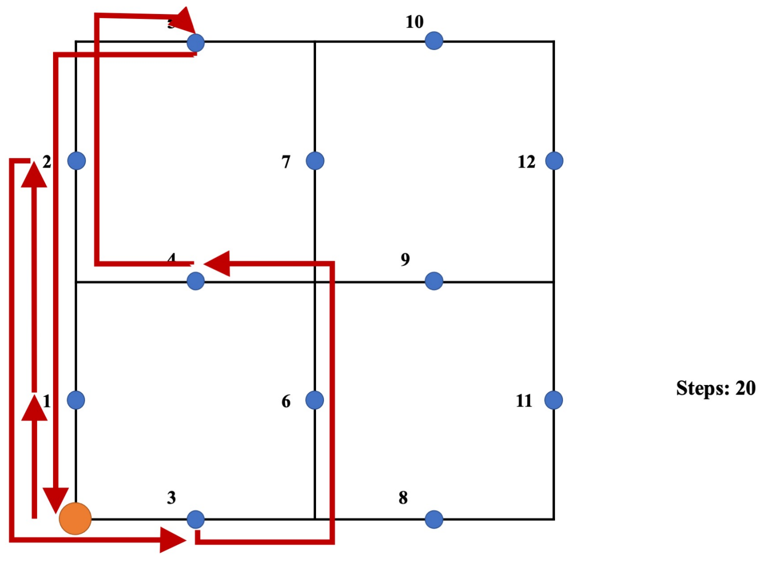

Figure 1: Collection Route of Truck 1.

The speed of the truck is set to be 1 unit of distance/1 unit of time. The length of a grid is set to be 2, that is, it takes the truck 2 unit of time to go over an edge of the grid. The capacity of the truck to be 5 units, and the quantity of garbage at each location is set to be 1 unit.

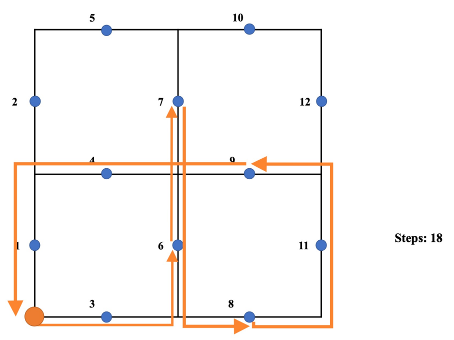

Figure 2: Truck 2 Route.

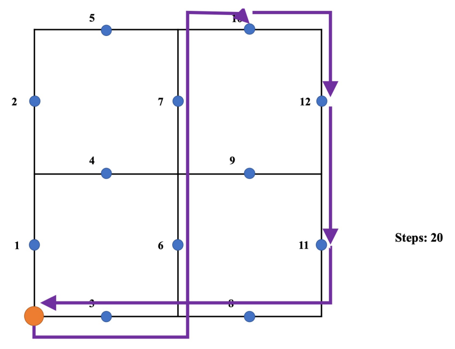

Figure 3: Truck 3 Route.

f(12,3) is the minimum amount of time needed for these three trucks to collect all garbage from these 12 locations.

The decision set of the first stage is $\{1,2,3,4,5\}$. In other words, the first truck can choose to visit k_i heaps starting from the 1-st heap (1-st heap included), where $k_i \in \{1,2,3,4,5\}$.

So we have the following equation:

$$\begin{aligned} f(12,3) = &\min\{\max\{2, f(11,2)\}, \max\{6, f(10,2)\},\\ &\max\{8, f(9,2)\}, \max\{14, f(8,2)\}, \max\{24, f(7,2)\}\end{aligned}$$

Note that we can neglect the term f(11,2) because 2 trucks cannot load the all 11 units of garbage left. This term will eventually give an ∞. We neglect this to simplify the calculation process.

Next, we have:

$f(10,2) = \min\{\max\{2, f(9,1)\}, \max\{8, f(8,1)\}, \max\{14, f(7,1)\}, \max\{16, f(6,1)\}, \max\{20, f(5,1)\}\}$

$f(9,2) = \min\{\max\{6, f(8,1)\}, \max\{12, f(7,1)\}, \max\{14, f(6,1)\}, \max\{18, f(5,1)\}, \max\{20, f(4,1)\}\}$

$f(8,2) = \min\{\max\{10, f(7,1)\}, \max\{12, f(6,1)\}, \max\{16, f(5,1)\}, \max\{18, f(4,1)\}, \max\{24, f(3,1)\}\}$

$f(7,2) = \min\{\max\{6, f(6,1)\}, \max\{10, f(5,1)\}, \max\{12, f(4,1)\}, \max\{18, f(3,1)\}, \max\{24, f(2,1)\}\}$

We eventually have $f(10,2) = 24$, $f(9,2) = 22$, $f(8,2) = 22$, $f(7,2) = 20$, which gives that $f(12,3) = 20$. See appendix for a full example.

Given $f(12,3)$, the paths of trucks are illustrated. In **Figure 1** we demonstrate the path of the first truck, the truck which got assigned the first piece of sequence. Truck 1 visits the garbage heaps in the order of: 0-1-2-3-4-5-0. **Figure 2** and **Figure 3** show the paths of Truck 2 and 3.

4.3 MFMC Algorithm

4.3.1 MFMC for Multiple Garages

Now we are considering the condition of multiple garages. In real life situations, the time spent to launch from and return to the garage is minor relative to the actual time spent collecting garbage. Thus, it is enough to just share the collection duties equally to achieve a fair distribution of work. We adjusted $u(i,n)$, the time spent to visit n garage heaps starting from the i-th, so that the time travelling from the starting garage and the time returning to the ending garage are left out of our consideration.

Therefore we have

$$u(i,n) = \Sigma_{i}^{i+n-2} d_{t,t+1}$$

The new definition of $u(i,n)$ would give us new $f(x,s)$ and new planning strategies, which orchestrate the trucks without considering the starting and ending garages.

With the new programming strategies, we would be able to develop the following "GSEG" graph:

Assume there are p garages and q trucks. The graph consists of a starting point, four layers, and an ending point. The starting garage layer represents the garages upon departure. The S-layer represents the starting locations for each garbage collection vehicle algorithmically decided by our model. The E-layer represents the restive ending locations of these vehicles. The distance between each corresponding "S" point and "E" point is calculated by the model with $u(i,n) = \Sigma d_{t,t+1}$. The ending garage layer represents garages upon return.

Flow-cost sets are labelled on the arcs. The first component in each set designates the maximum flow allowed on the arc, whereas the latter denotes the distance between the two locations connected by the arc. c_1, c_2, ... ,c_p on arcs between the starting point and the S-layer denote the capacity of each garage. d_{ij} retains its meaning

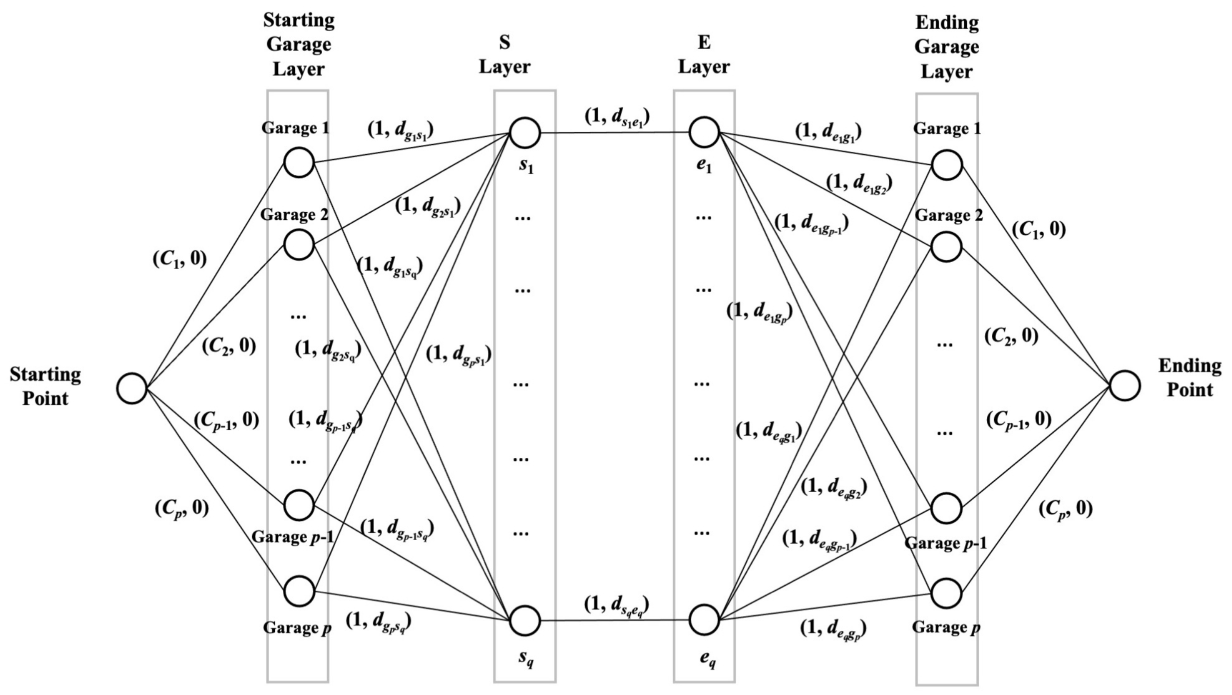

Figure 4: GSEG Graph

as the time spent covering the distance between the i-th and j-th node.

We aim to minimize the time spent for the system. The problem is thus converted into calculating the path with minimum cost while the flow in the graph is maximized, that is, a Maximum Flow Minimum Cost problem. There is a well-developed solution to such problems and we directly apply the algorithm to it.

4.3.2 Demonstration of Two Garages in 2x2 Grid

We again demonstrate our model in a 2x2 grid. We assume the trash heaps are laid out in the same pattern as the previous 2x2 grid. However, there are two garages, Garage1 at (0,0) and Garage2 at (2,2). Garage1 can contain 2 trucks and Garage2 can hold 1. Trucks can start and return to either of the garage – they can start and return to a different garage. The location of the garage largely reflects the lo-

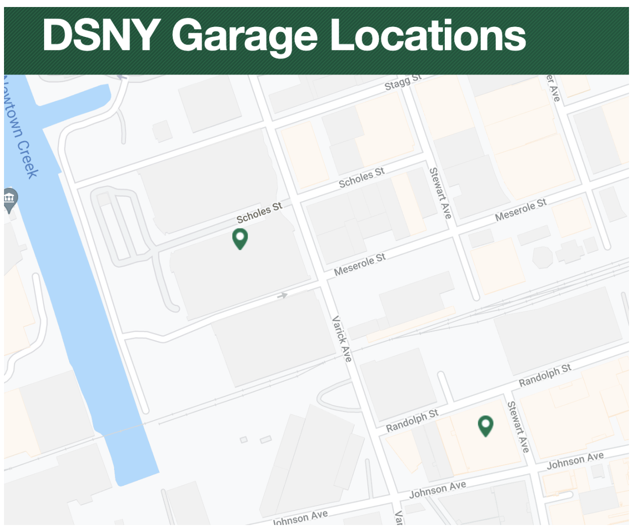

Figure 5: Locations of Two DSNY Garages in Brooklyn

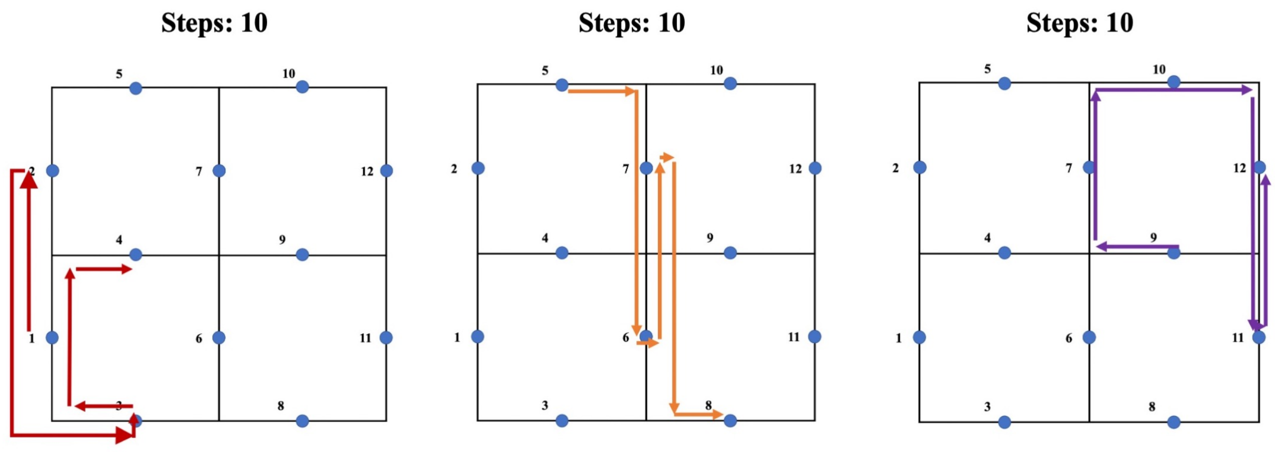

Figure 6: Route without Origin Constraints for Truck 1, 2, and 3

cation of two DSNY (The City of New York Department of Sanitation) garages situated on near Meserole St. and Randolph St. respectively that are separated by only one avenue in Brooklyn.

First, we calculate $f(12, 3)$ in this case, which gives $f(12, 3) = 10$, indicating that the collective minimal time is 10 unit of time for 3 trucks to traverse all the garbage heaps without concerning the time spent of leaving and returning to the garage. In **Figure 6** we show the paths of the three trucks.

The locations of garages, starting points and ending points are graphed below in **Figure 7**. Thus we are able to decide the costs of arcs between G-S and E-G layers as all "S" points and "E" points are represented. Hence, we are able to generate the GSEG graph, as shown in **Figure 8**.

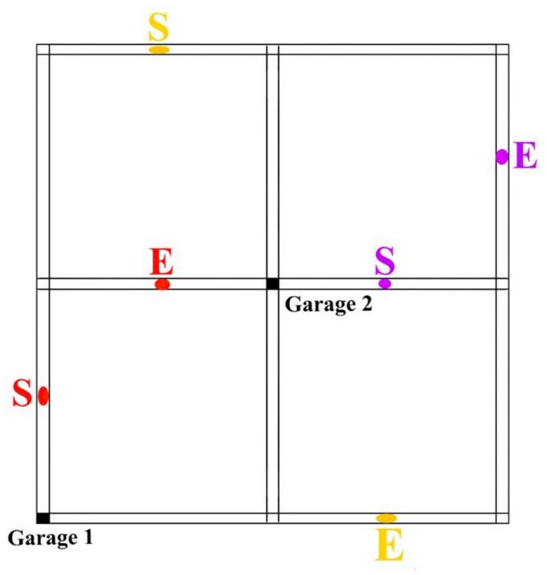

Figure 7: GSEG Points Demo

To illustrate this, the set (1, 1) displayed on the arc between Garage 2 and the purple “S” shows that only one truck is permitted to travel to start at Garage 2. (Apparently, ONE truck can at maximum allows ONE truck to start from a certain garage.) The distance of travel from Garage 2 to purple "S" is 1.

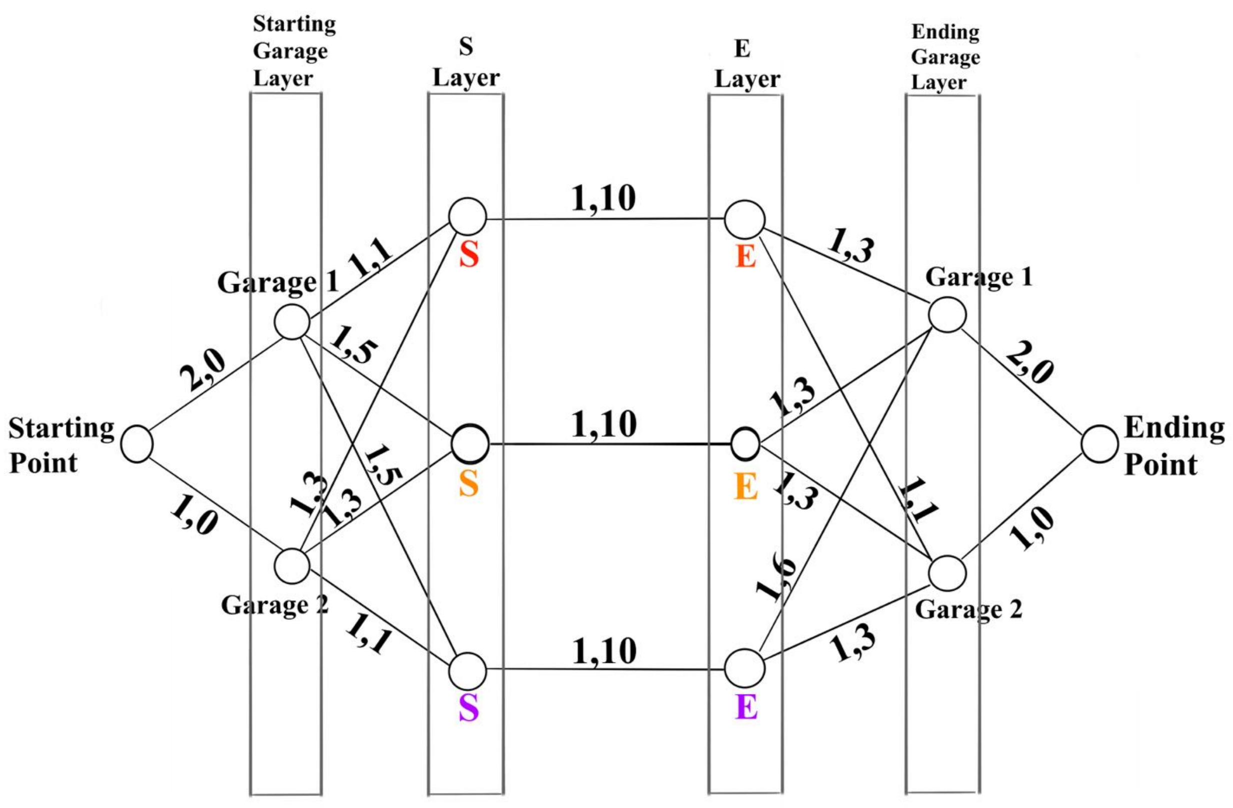

Figure 8: GSEG Graph For 2x2 Grid

We employ MFMC algorithm and compute the allocation of garages. Results are shown below in **Figure 10**.

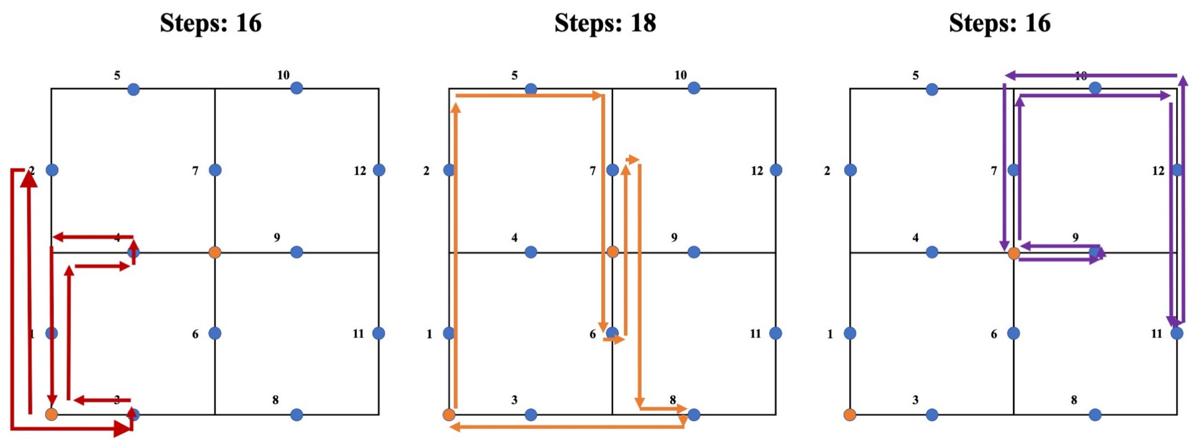

Figure 9: MFMC Results

4.4 Genetic Algorithm

4.4.1 Algorithm Explanation

Genetic Algorithm is invented by John Holland in an attempt to mimic the process of genetic evolution. It is often employed to tackle with NP-hard problems, which are too costly for deterministic algorithms.

The algorithm entails a sequence of numerical symbols. By generating a fitness function to the sequence, we are able to pick out the fitter individuals of the population. Then, we renew the population by marrying pairs of individuals in the population and mutating individuals of them. Eventually, we will have a population of the fittest individuals.

4.4.2 Model On Real New York

We ran our model on Hell's Kitchen, Manhattan. The region is on 4x15 grid of blocks in between 8 Avenue and 12 Avenue and between 57th street and 42th street. The region can be approximated as a square with sides of length 0.7 mile. The region can be subdivided into 60 (4x15) blocks, each block with length of 0.175 mile and width of 0.047 mile. We approximate the time of truck to traverse the length of the block to be 30 seconds. The time of truck to traverse the width of the block to be 10 seconds (Calculations are based on the assumption that speed of the truck is around 19 mph).

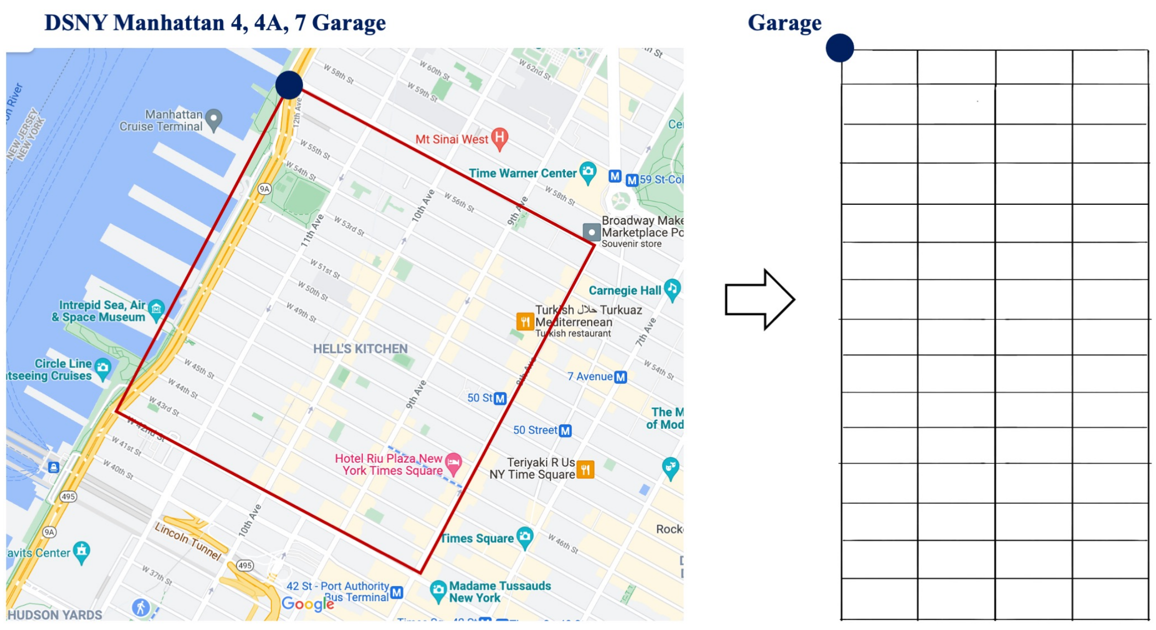
DSNY Manhattan 4, 4A, 7 Garage
Garage
HELL'S KITCHEN
Manhattan Cruise Terminal
Intrepid Sea, Air & Space Museum
Circle Line Sightseeing Cruises
Mt Sinai West
Time Warner Center
Carnegie Hall
Times Square
Madame Tussauds New York
Hotel Riu Plaza New York Times Square
42 St - Port Authority Bus Terminal
Lincoln Tunnel
HUDSON YARDS
11th Ave
10th Ave
9th Ave
7 Avenue
50 St
50 Street

Figure 10: Model On Hell's Kitchen, Manhattan [Google Maps]

Unfortunately, due to the limitation of computing power and time, we are only able to calculate the fitness function of the algorithm, which can be seen in Appendix.

Reference

“Annual Report: New York City Curbside and Containerized Municipal Refuse and Recycling Statistics by Borough and District: FISCAL YEAR 2021.” The City of New York Department of Sanitation, 2021, https://dsny.cityofnewyork.us/wp-content/uploads/2021/09/ about_dsny- collections-annual-2021-Fiscal-Year-2021-DSNY-Curbside-Collections.pdf. Accessed 22 Jan 2022.

“What You Should Know About the Four Major Types of Garbage Trucks.” Route Ready, Oct. 10, 2019, https://routereadytrucks.com/ blogs/know-4-major-types-garbage-trucks/#: :text=Most%20of%20the %20standard%20trucks,28%20cubic%20yards%20of%20garbage.

“Discover Our Fleet.” The City of New York Department of Sanitation, 2022, https://www1.nyc.gov /assets/dsny/site/about/fleet. Accessed 21 Jan 2022.

“More Funds for Growing Waste Export Budget And Organics Processing Capacity.” New York City Independent Budget Office, Mar. 2020, https://ibo.nyc.ny.us/iboreports/more-funds-for- growing-waste-expor-budget-and-organic-processing-capacity-fopb-march-2020.pdf. Accessed 22 Jan 2022.

“Solid Waste.” PlaNYC, 2011, http://s-media.nyc.gov/agencies/ planyc2030 /pdf/planyc$_2011_solid_waste.pdf.Accessed22Jan2022.$

“Genetic Algorithm.” ScienceDirect, https://www.sciencedirect.com/ topics /engineering/genetic-algorithm. Accessed 22 Jan 2022.

Appendix

A Letter to the Mayor

Dear Mr. Adams:

Since the past few years we have all become witnesses of tremendous upheaval. The world has been plagued by not only the pestilence of COVID-19, beset by political turmoil and societal fracture, but also the ever-looming shadow climate change. New York, towering as the world's fulcrum, serving as exemplar after which so many follow, must present itself as resolutely progressive, dauntlessly innovative, and conscientiously in the lead.

Our research focuses on New York's current trash disposal policies. Existing data suggests that garbage trucks in New York operate inefficiently, leading to excessively high and unnecessary costs in purchase and maintenance. We believe our model can help reduce these expenditures and divert greater budgets towards other environmental initiatives in New York City.

Our model is based on dynamic programming and genetic algorithms, two computer science methods employed in optimization. We first illustrate our model with a simple grid with one garbage truck and one garage, then, we proceed to complicate our model by lifting movement restrictions and increasing the number of trucks and garages. Finally, we extrapolate our previous findings to Hell's Kitchen, Manhattan and Scholes Street, Brooklyn. We find that, with three trucks, we will be able to trasverse the area in a short amount of time.

Therefore, we sincerely hope you heed our advice, for not only will New York benefit from these arrangements, but a standard shall be set for metropolises to achieve.

Sincerely Yours,

Yuxi Chen, Bole Pan, Haoran Yuan

Appendix A: Demonstration of Dynamic Programming: Full Calculations

Following from Section 4.2.1, we have for the next stage:

$f(9,1)=\infty; f(8,1)=\infty; f(7,1)=\infty; f(6,1)=\infty$

$f(5,1)=\min\{\max\{6,f(4,0)\},\max\{12,f(3,0)\},\max\{18,f(2,0)\},\max\{20,f(1,0)\},$
$\max\{24,f(0,0)\}\}$

$f(4,1)=\min\{\max\{10,f(3,0)\},\max\{16,f(2,0)\},\max\{18,f(1,0)\},\max\{22,f(0,0)\}\}$

$f(3,1)=\min\{\max\{14,f(2,0)\},\max\{16,f(1,0)\},\max\{20,f(0,0)\}\}$

$f(2,1)=\min\{\max\{10,f(1,0)\},\max\{14,f(0,0)\}\}$

For the next stage, we have:

$f(4,0)=\infty; f(3,0)=\infty; f(2,0)=\infty; f(1,0)=\infty; f(0,0)=0$

Thus, we can calculate terms from the previous stage:

$f(5,1)=\min\{\max\{6,\infty\},\max\{12,\infty\},\max\{18,\infty\},\max\{20,\infty\},$
$\max\{24,0\}=24$

$f(4,1)=\min\{\max\{10,\infty\},\max\{16,\infty\},\max\{18,\infty\},\max\{22,0\}\}=22$

$f(3,1)=\min\{\max\{14,\infty\},\max\{16,\infty\},\max\{20,0\}\}=20$

$f(2,1)=\min\{\max\{10,\infty\},\max\{14,0\}\}=14$

We then calculate terms from one more stage before:

$f(10,2)=\min\{\max\{2,\infty\},\max\{8,\infty\},\max\{14,\infty\},\max\{16,\infty\},$
$\max\{20,24\}\}=24$

$f(9,2)=\min\{\max\{6,\infty\},\max\{12,\infty\},\max\{14,\infty\},\max\{18,24\},$
$\max\{20,22\}\}=22$

$f(8,2)=\min\{\max\{10,\infty\},\max\{12,\infty\},\max\{16,24\},\max\{18,22\},$
$\max\{24,20\}\}=22$

$f(7,2)=\min\{\max\{6,\infty\},\max\{10,24\},\max\{12,22\},\max\{18,20\},$
$\max\{24,14\}\}=20$

Appendix B: Codes

Dynamic Programming

```python
from math import sin

class Station:
    def __init__(self, id, amount, position):
        self.id = id
        self.amount = amount
        a,b = position
        self.pos = (a,b)


stations=[Station(0,0,(0,0))]
for i in range(12):
    stations.append(Station(i+1, 1, divmod(2*i+1,5)))
truckLoad = 5

# customers current truck is able to visit in this round. Input: current
                                     heap
def cusCovered(currentCustomer):
    global stations
    global truckLoad

    i = 0
    burden = stations[currentCustomer].amount
    while truckLoad>=burden:
        i+=1
        if currentCustomer+i>=len(stations):
            return i
        burden += stations[currentCustomer+i].amount
    return i

# Manhatton distance sum. Visit b heaps from the a-th, a-th included.
def u(a,b):
    global stations
    mahattanDistance = 0
    for i in range(b-1):
        x1,y1 = stations[a+i].pos
        x2,y2 = stations[a+i+1].pos
        mahattanDistance += abs(x1-x2)+abs(y1-y2)
        if x1==x2 and x1%2==1 and y1!= y2:
            mahattanDistance +=2
        if y1==y2 and y1%2==1 and x1!= x2:
            mahattanDistance +=2
    x1,y1 = stations[a].pos
    x2,y2 = stations[a+b-1].pos
    backhome = x2+y2
    goto = x1+y1
    mahattanDistance += backhome +goto
    return mahattanDistance

# dynamic programming. Input: number of heaps left, number of trucks left
def minTime(cusN, truckN):
    global stations
    # terminating condition
    if cusN <= 0:
        return 0
    elif truckN == 0:
        return 999999999

    currentCus = len(stations)-cusN
    onerun = cusCovered(currentCus)

    bestPath = []
    for i in range(onerun):
        bestPath.append(max(u(currentCus, i+1), minTime(cusN-(i+1), truckN
                                                        -1)))

    return(min(bestPath))

print(minTime(12,3))
```

MFMC Flow Cost

```
from math import sin

class Station:
    def __init__(self, id, amount, position):
        self.id = id
        self.amount = amount
        a,b = position
        self.pos = (a,b)


stations=[Station(0,0,(0,0))]
for i in range(12):
    stations.append(Station(i+1, 1, divmod(2*i+1,5)))
truckLoad = 5

# How many heaps current truck is able to visit in this round
def cusCovered(currentCustomer):
    global stations
    global truckLoad

    i = 0
    burden = stations[currentCustomer].amount
    while truckLoad>=burden:
        i+=1
        if currentCustomer+i>=len(stations):
            return i
        burden += stations[currentCustomer+i].amount
    return i

# Manhatton distance sum. Visit b heaps from the a-th, a-th included.
def u(a,b):
    global stations
    mahattanDistance = 0
    for i in range(b-1):
        x1,y1 = stations[a+i].pos
        if a+i == len(stations)-1:
            x2,y2= stations[len(stations)-1].pos
        else:
            x2,y2 = stations[a+i+1].pos
        mahattanDistance += abs(x1-x2)+abs(y1-y2)
        if x1==x2 and x1%2==1 and y1!= y2:
            mahattanDistance +=2
        if y1==y2 and y1%2==1 and x1!= x2:
            mahattanDistance +=2

    return mahattanDistance

# dynamic programming.
def minTime(cusN, truckN):
    global stations

    # terminating condition
    if cusN <= 0:
        return 0
    elif truckN == 0:
        return 999999999

    currentCus = len(stations)-cusN
    onerun = cusCovered(currentCus)

    bestPath = []
    for i in range(onerun):
        bestPath.append(max(u(currentCus, i+1), minTime(cusN-(i+1), truckN
                                                        -1)))

    return(min(bestPath))

print(minTime(12,3))
```

MFMC Algorithm

```
from ortools.graph import pywrapgraph

start_nodes = [ 0, 0, 1, 1, 1, 2, 2, 2, 3, 4, 5, 6, 6, 7, 7, 8, 8, 9,10]
end_nodes   = [ 1, 2, 3, 4, 5, 3, 4, 5, 6, 7, 8, 9,10, 9,10, 9,10,11,11]
capacities  = [ 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1]
unit_costs  = [ 0, 0, 1, 5, 5, 3, 3, 1,10,10,10, 3, 1, 3, 3, 6, 3, 0, 0]

# Define an array of supplies at each node.
supplies =    [ 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -3]

# Instantiate a SimpleMinCostFlow solver.
min_cost_flow = pywrapgraph.SimpleMinCostFlow()

# Add each arc.
for i in range(0, len(start_nodes)):
    min_cost_flow.AddArcWithCapacityAndUnitCost(start_nodes[i], end_nodes[
                                                i],capacities[i], unit_costs[i])
```

```
# Add node supplies.
for i in range(0, len(supplies)):
    min_cost_flow.SetNodeSupply(i, supplies[i])

 # Find the minimum cost flow between node 0 and node 4.

if min_cost_flow.Solve() == min_cost_flow.OPTIMAL:
    print('Minimum cost:', min_cost_flow.OptimalCost())
    print('')
    print('  Arc    Flow / Capacity  Cost')
    for i in range(min_cost_flow.NumArcs()):
      cost = min_cost_flow.Flow(i) * min_cost_flow.UnitCost(i)
      print('%1s -> %1s   %3s  / %3s       %3s' % (
          min_cost_flow.Tail(i),
          min_cost_flow.Head(i),
          min_cost_flow.Flow(i),
          min_cost_flow.Capacity(i),
          cost))
else:
    print('There was an issue with the min cost flow input.')
```

Fitness Function Prep

```
from math import sin

class Station:
    def __init__(self, id, amount, position):
        self.id = id
        self.amount = amount
        a,b = position
        self.pos = (a,b)


stations=[Station(0,0,(0,0))]
for i in range(139):
    a,b = divmod(10*i-5,155)
    position = 15*a, b
    stations.append(Station(i+1, 40+40*sin(i), position))
truckLoad = 3000
# truck load 15ton
# 200(1+sin(i))kg per street, takes 1s to load 5kg

# number of heaps current truck is able to visit in this round
def cusCovered(currentCustomer):
    global stations
    global truckLoad

    i = 0
    burden = stations[currentCustomer].amount
    while truckLoad>=burden:
        i+=1
        if currentCustomer+i>=len(stations):
            return i
        burden += stations[currentCustomer+i].amount
    return i
```

```
# Manhatton distance sum. Visit b heaps from the a-th, a-th included.
def u(a,b):
    global stations
    mahattanDistance = 0
    for i in range(b-1):
        x1,y1 = stations[a+i].pos
        x2,y2 = stations[a+i+1].pos
        mahattanDistance += abs(x1-x2)+abs(y1-y2)
        if x1==x2 and x1%2==1 and y1!= y2:
            mahattanDistance +=2
        if y1==y2 and y1%2==1 and x1!= x2:
            mahattanDistance +=2
    x1,y1 = stations[a].pos
    x2,y2 = stations[a+b-1].pos
    backhome = x2+y2
    goto = x1+y1
    mahattanDistance += backhome +goto
    return mahattanDistance

# renew 'stations' array
def stationsequence(*sequence):
    new_station=[Station(0,0,(0,0))]
    for i in sequence:
        new_station.append(stations[i])

    return new_station

# dynamic programming
def minTime(cusN, truckN):
    global stations

    # terminating condition
    if cusN <= 0:
        return 0
    elif truckN == 0:
        return 999999999

    currentCus = len(stations)-cusN
    onerun = cusCovered(currentCus)

    bestPath = []
    for i in range(onerun):
        bestPath.append(max(u(currentCus, i+1), minTime(cusN-(i+1), truckN
                                                    -1)))

    return(min(bestPath))
```

Fitness Function

```
import fitnessprep as ft

def fitness(*sequence):
    new_stations = ft.stationsequence(*sequence)
    for i in len(ft.stations):
        ft.stations[i] = new_stations[i]

    return ft.mintime(139,5)

nlist = range(1,139)
print(fitness(nlist))
```