Medical Resource Allocation During Pandemics

Hamid Foroughi

Department of Civil and Systems Engineering

Introduction

In December, 2019, Wuhan, Hubei province, China, became the centre of an outbreak, which raised intense attention not only within China but internationally. In a short time, COVID-19 became a new pandemic with nearly three million confirmed cases all around the world [1]. In the current COVID-19 outbreak, infection control and clinical management efforts are necessarily being implemented on a larger scale than in any previous outbreak, and it is therefore appropriate to reassess the allocation methods for medical resources. Deep-learning algorithms has been used by many researchers to provide more insight into different aspects of previous pandemics from infection to disease management [2][3]. In the current study, a geographic dataset of Maryland counties [4] has been used to develop a model that utilizes the minimum vertex cover set as a solution to optimal resource allocation. Based on the minimum cover set, a traveling salesman problem (TSP) model is used to determine the optimal path of delivery which incurs minimum total cost.

Model Setting

1. Let $G = (V, E)$ be a graph object consisting of a distance function $w : E \to \mathbf{R}^+$.
2. The node set V represents the set of agents we are studying for and can be in state, county, or even individual level.
3. Edge set E contains all the edges between each pair of nodes (i.e. it is a complete graph).
4. The distance function $w(e)$ maps the traveling distance between two agents connected by edge e.

Model Assumptions

1. If physical distance is assumed to be a diving parameter in disease propagation, it can be said that once the physical distance among agents exceeds a certain level, the possibility of getting infected becomes quite low. Mathematically speaking, Let the distance between two agents connected by edge e be denoted as $w(e)$. Then we make an assumption that when $w(e) > t$ for some threshold t, there is no virus spreading between the pair of agents connected by e.
2. Each agent requires the same amount of resources.
3. The model determines whether to deliver medical resources to each agent.
4. There are enough resources for all the agents chosen by the model.

Model Algorithm

Step 1: Set a threshold t. Build a new graph $G_1 = (V, E_1)$, where $E_1 = \{e \in E : w(e) \leq t\}$.
Step 2: Solve the minimum vertex cover problem on G_1 with a solver (e.g. Gurobi) to get a minimum vertex cover set $S \subseteq V$.
Step 3: Build another graph $G_2 = (S, E_2)$ where $E_2 = \{e \in E : e$ connects a pair of nodes in S$\}$.
Step 4: Solve a traveling salesman problem on the combination of G_2 and the distance function $w(e)$ by applying Gurobi and deep learning methods separately to get two economical routes for delivery for comparison.

How to solve The Minimum Vertex Cover (MVC) Problem

We solve MVC by implementing the following pure Integer Programming (IP) problem in Gurobi.

$$\begin{aligned} \min \quad & \sum_{u \in V} x_u \\ \text{s.t} \quad & x_u + x_v \geq 1 \text{ for all } e = uv \in E \\ & x_u \in \{0,\ 1\} \text{ for all } u \in V \end{aligned}$$

How to solve TSP

Gurobi

We solve TSP using the following IP formulation

$$\begin{aligned} \min \quad & \sum_{e \in E_2} w(e) \cdot x_e \\ \text{s.t} \quad & x_{uv} + x_{vz} = 2 \text{ for all } u, v, z \in S \\ & \delta(S') \geq 2 \text{ for all } S' \subseteq S \quad \text{(Lazy Constraints)} \\ & x_e \in \{0, 1\}, \end{aligned}$$

where $\delta(S')$ means the number of selected edges across between $S \backslash S'$ and S', and lazy constraints are only generated when needed. Typically lazy constraints are unlikely to be violated.

HNN [6, 7]

As an alternative to MIP, chaotic neural network (CNN) is also being used as a heuristic solution to combinatorial optimization of TSP problem. We name the neural network Hopefield neural network (HNN) as it was first applied by Hopefiled and Tank in 1985 to study TSP problem [6]. To minimize the cost function that reflects total traveling distance of the feasible routes of delivery, HNN relies on chaotic simulated annealing (CSA) as an efficient searching method.

Modeling assumptions
1: Distance matrix is symmetric: $d_{ij} = d_{ji} V_{i,j}$
2: $N \times N$ permutation matrix Y is used to represent routing selection as the output of HNN. Each entry Y_{ij} of the permutation matrix is a binary variable, with $Y_{ij=1}$ represents visiting city i in order j: **(a)** $\sum_{i(a_{ij})} = 1$, **(b)** $\sum_{i(a_{ij})} = 1$
3: A computational energy function is included to minimize the total tour length while simultaneously satisfying all constraints:

$$E = \frac{W_1}{2} \left\{ \sum_{i=1}^{n} \left(\sum_{j=1}^{n} x_{ij} - 1 \right)^2 + \sum_{j=1}^{n} \left(\sum_{i=1}^{n} x_{ij} - 1 \right)^2 \right\} \textit{ 1 st term represents two constraints}$$
$$+ \frac{W_2}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \left(x_{kj+1} + x_{kj-1} \right) x_{ij} d_{ik} \textit{ 2 nd term represents cost function associate with distance}$$

where W_1 and W_2 are fixed coupling parameters corresponding to the constraints and the cost function of the tour length.
4: Simulation stopping criteria:
(a) The CSA will stop either when program reaches maximum number of iterations, or
(b) when energy function converges to global minimum, in other words when, the permutation matrix Y becomes stable.

CSA algorithm

1. Random pick initial state $Y = y_o$, y_o is a $n \times n$ matrix with each entry random pick from range [0,1]
2. Updates Y according to the specified dynamics on the right hand side derived by the Euler method.
3. Keep updating Y until E converges or the maximum number of iterations being reached.
4. Output: final state Y

$$x_{ij}(t) = \frac{1}{1+e^{-y_{ij}(t)/\varepsilon)}}$$
$$y_{ij}(t+1) = k y_{ij}(t) - z(t)(x_{ij}(t) - I_o)) + \alpha \{ - W_1 \left(\sum_{l \neq j}^{n} x_{il}(t) + \sum_{k \neq i}^{n} x_{kj}(t) \right)$$
$$- W_2 \left(\sum_{k \neq i}^{n} d_{ik} x_{kj+1}(t) + \sum_{k \neq i}^{n} d_{ik} x_{kj-1}(t) \right) + W_1 \}$$
$$(i, j = 1, ..., n)$$
$$z(t+1) = (1 - \beta) z(t)$$

Model Demonstration

Assume we have nine counties as the agents that form the nodes of a graph. The distance function is defined by real data. Then, if we set a threshold t, we can build the graph G_1 as mentioned above. (See Figure 1)

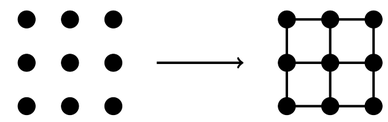

Figure 1

Solve the minimum vertex cover problem based on graph G_1. Then build graph G2 as shown in Figure 2 based on the red nodes form the solution.

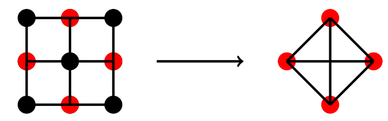

Figure 2

Solve the traveling salesman problem on G_2 to get the final solution.

Computational Experiments

Dataset

The dataset considered in this study consists of the pairwise distance data of all the towns in the state of Maryland. This set of data is used as initial input for the MVC problem to finding a set of towns deemed best for initial allocation of resources.

Results of Solving MVC with Gurobi

In order to control the size of the solution of the MVC problem, the we set the threshold t equal to 99% quantile of all the edge weights in the pairwise matrix of distances. This allocation serves another purpose of making the HNN algorithm computationally efficient. The result is a set of 20 towns as the initial allocation ports.

Results of Solving TSP with Gurobi and HNN

We use the results of the MVC problem to construct a clique of all the selected towns. We modified the algorithm proposed by [9]. By solving the TSP on this graph and using GUrobi, we get a tour with length of 710.365 miles. Once more, modifying the approach from [8], we get tours of length around 720 miles, which is sufficiently close to the true optimal solutions resolved by Gurobi for small networks such as this. It is worth to note that the true potential of HNN will probably be shown in larger networks where solving for the optimal tours with Gurobi becomes nearly impossible.

Comparison between Gurobi and HNN

We compare the performance of Gurobi and HNN as the following table. Note that the result of HNN is based on 30 tries and we take the shortest tour length, while the time is the total running time of 30 tries.

	Tour Length	Time Consumption
Gurobi	710.365 miles	0.3 seconds
HNN	731.410 miles	around 3 seconds

Conclusions

- HNN requires much heavier computations than solving IP with Gurobi to derive an acceptable results for smaller networks. However, HNN becomes more and more efficient as the size of the problem grows in comparison to Gurobi.
- Due to the nature of the HNN algorithm, the solutions derived from this method are not as good as the IP solutions given by Gurobi in smaller networks where solving the TSP IP problem is not computationally intensive.
- MVC can be solved efficiently by Gurobi.
- By selecting a relatively high threshold t, we can shrink the size of MVC solution efficiently.

References

[1] Dong, E., Du, H. and Gardner, L., 2020. An interactive web-based dashboard to track COVID-19 in real time. The Lancet infectious diseases.

[2] Fujita, H. and Herrera-Viedma, E., 2018, September. Deep Learning employed in the recognition of the vector that spreads Dengue, Chikungunya and Zika viruses. In New Trends in Intelligent Software Methodologies, Tools and Techniques: Proceedings of the 17th International Conference SoMeT18 (Vol. 303, p. 108). IOS Press.

[3] Akhtar, M., Kraemer, M.U. and Gardner, L.M., 2019. A dynamic neural network model for predicting risk of Zika in real time. BMC medicine, 17(1), p.171.

[4] Mileage-Charts.com. (n.d.). Retrieved from https://www.mileage-charts.com/chart.php?p=index&a=NA&b=US&c=MD

[5] Chen, L. and Aihara, K., 1995. Chaotic simulated annealing by a neural network model with transient chaos. Neural networks, 8(6), pp.915-930.

[6] Hopfield, J.J. and Tank, D.W., 1985. "Neural" computation of decisions in optimization problems. Biological cybernetics, 52(3), pp.141-152.

[7] Chen, L. and Aihara, K., 1995. Chaotic simulated annealing by a neural network model with transient chaos. Neural networks, 8(6), pp.915-930.

[8] Tunnell, James. GitHub, 25 Oct. 2017, https://github.com/jamestunnell/chaotic_tsp

[9] Cook, William, AMS 467/667: Hello MIP World (HW1), http://www.ams.jhu.edu/w̃cook12/dl/hw1/index.html