SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

40.018 Heuristics and Systems Theory

Project 1 Report

CS02

Name	Student ID
Benedict Nai Rong Hui	1008173
Mika Ling	1007901
Muk Chen Eu Zachary	1007848

Contents

Executive Summary

This report summarises the processes involved in the project which concerns an optimization problem solving for multiple considerations to improve the flow of blood through external blood donation supply chains within a country. Blood donation is of high importance to the healthcare system, as fresh blood is needed daily for patients, its delivery and collection is of utmost importance as well.

The optimisation problem models the assignment of collection centres to blood banks across a country and the sequence of blood collection from each centre for each assignment within a single objective function. The problem is a mixed integer linear programme that is modelled after a combination of an assignment and TSP problems. Initial instances were solved using Gurobi optimiser, where numerous modifications were made to the model in various iterations to test it under different constraints. Initially modelled after Singapore, the problem can be scaled up to model larger countries by increasing the instances using random plot generators. However, due to added constraints such as the Miller-Tucker-Zemlin (MTZ) condition to the original problem to avoid subtours, complexity increased exponentially and rendered the problem unsolvable (e.g. Solver runs for more than 3 hours or produces a memory error) for instances within Singapore's context.

A metaheuristic using Ant Colony Optimisation (ACO) was hence developed to approximate solutions for the same optimisation problem. The algorithm was modified to model the same objective function, variables and constraints as the problem solved using Gurobi. ACO was chosen to handle the assignment and TSP problem which consists of nodes, which is ideal for this algorithm. The pheromone and ant exploration system produced promising results for the nature of the problem.

Both the Gurobi solver and the ACO algorithm were run several times across multiple iterations to produce a variety of results. The results for a small-scale instance on Gurobi were then utilised as a benchmark to determine the efficacy of the ACO algorithm performance. The lower bound optimal value generated by Gurobi is 153.00, which is benchmarked against 184.21 for the ACO solution, giving an error of about 20.4%. Comparing factors such as closeness of results, margin of errors, solution propriety and computational time, the ACO algorithm provided good approximations to the optimal solutions.

Considering the constraints set, the metaheuristic solutions from the ACO algorithm has produced results that are good approximations for the original problem using Singapore data. Hence, the value of 885.31 is also a good approximation of the optimal value using data from Singapore. The ACO algorithm is there proven to be useful for this problem, with potential for wider use cases within the healthcare system to attain more optimal solutions that can improve the processes of large-scale public health planning.

1. Problem Background

A perennial problem facing hospitals and medical institutions globally is the persistent shortage of donated blood. To maximize blood donations and improve the accessibility of willing donors, Singapore's 5 major blood banks regularly hold blood drives at respective Community Centres (CCs) to increase its network of donors. According to the Health Sciences Authority (HSA), "Every hour of the day, 15 units of blood are used in Singapore. We need to collect about 400 units (200 litres) of blood daily to meet the blood needs of patients in Singapore." Singapore faces the same problem as well.

This project aims to assign each blood bank to a group of CCs by minimizing the distance between CCs and the existing blood banks; while considering the demand for blood from each blood bank, the estimated volume of blood donated with each blood drive and the expected capacity of each CC. Thereafter, blood banks would need to decide the sequence or route they would take to organise blood drives, which is modelled as a Travelling Salesman Problem (TSP).

2. Data Collection

To begin the project, initial data collection procedures were performed. Key points of data that were required include:

- Geodata of various points of interest (POIs) pertaining to the blood supply chain in Singapore
    - Blood Banks (BBs), Community Centres (CCs), Public and Private Hospitals
- Daily blood requirements across the country
- Percentage of the population that donates blood

Most of this data was sourced from publicly available government sources, such as HSA's blood bank website, Singstat, Data.gov and others. This was required to replicate an accurate model of the blood supply chain in Singapore in modelling software such as Gurobi.

When all the necessary data was collected, plotting was conducted using the Matplotlib library in Python to obtain the graph as seen in Figure 1 below:

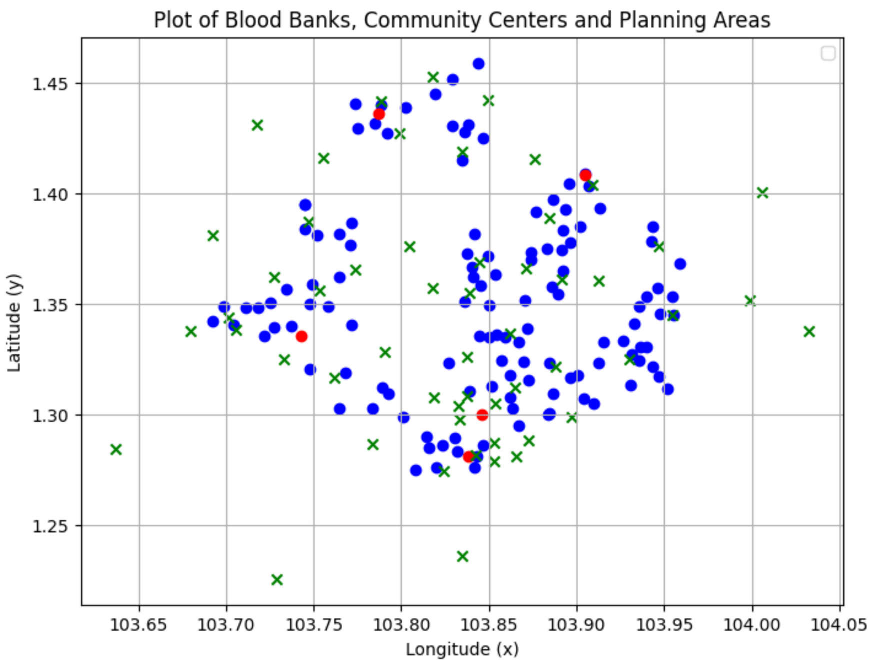


Figure 1 Matplotlib plot featuring the necessary Geodata

The next step for this data was to calculate distances between the various points. This is crucial for the optimisation problem, as TSP and assignment problems require computing distances between various POIs to minimise total distance. While initially using Euclidean distances, the physical distances between collection/community centres and the existing blood banks in the final model are calculated using their Geospatial coordinates (Longitude and Latitude) from real-world data. This takes advantage of the Haversine formula to compute the distances. The same formula was used to calculate the distances between the blood banks and hospitals.

Haversine's equations were used for distance computation between all coordinate pairs:

$$\Delta lon_{ij} = x_{BB,i} - x_{CC,j}$$

$$\Delta \, \mathrm{lat}_{\mathrm{ij}} = \mathrm{y}_{\mathrm{BB,i}} - \mathrm{y}_{\mathrm{CC,j}}$$

$$a_{ij} = \sin^2\left(\frac{\Delta lat_{ij}}{2}\right) + \cos(y_{BB,i})\cos(y_{CC,j})\sin^2\left(\frac{\Delta lon_{ij}}{2}\right)$$

$$c_{ij} = 2 \cdot \mathrm{arctan2}\left(\sqrt{a_{ij}}, \sqrt{1 - a_{ij}}\right)$$

$$\mathrm{distance}_{ij} = R \cdot c_{ij}$$

3. Assumptions

Due to the high variability of the problem and the lack of concrete data, several assumptions were made to solve the problem.

1. The blood banks do not have a maximum capacity (infinite capacity)

2. Not all blood banks will have collection centres/community centres assigned to them

3. Singapore's 1.9% population donation rate is applied across all collection centres/community centres for all problem sizes

4. A conversion rate of 1 unit to 500 mL of blood is used for all blood amount calculations

5. Each individual only donates 350mL of blood

6. Singapore's daily blood demand of 328 units is used as a benchmark for all scales of the problem

7. Any randomised data used to account for lack of data is accurate for our use case

4. Mathematical Optimisation Model

Mathematical Optimisation Model

Sets

- I: Set of blood banks, indexed by i
- J: Set of collection centers, indexed by j
- H: Set of hospitals, indexed by h

Decision Variables

- $x_{ij} = \begin{cases} 1 & \text{if collection center } j \text{ is assigned to bank } i \\ 0 & \text{otherwise} \end{cases}$
- $y_i = \begin{cases} 1 & \text{if blood bank } i \text{ is open} \\ 0 & \text{otherwise} \end{cases}$
- $z_{ih} = \begin{cases} 1 & \text{if hospital } h \text{ is served by bank } i \\ 0 & \text{otherwise} \end{cases}$
- $u_{kij} = \begin{cases} 1 & \text{if blood bank } k \text{ travels from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$
- $mtz_{ki} \in [0, |J|]$: Continuous variable for MTZ subtour elimination.

Objective Function

$$\min \left(\sum_{i \in I} \sum_{j \in J} d_{ij}^{LC}\, x_{ij} + \sum_{k \in I} \sum_{i \in J} \sum_{j \in J,\, j \neq i} d_{ij}^{CC}\, u_{kij} \right)$$

Constraints

1. **Assignment:** Each collection center is assigned to exactly one bank.

$$\sum_{i \in I} x_{ij} = 1, \quad \forall j \in J$$

2. **Assignment only if open:** Collection centers can only be assigned to open banks.

$$x_{ij} \leq y_i, \quad \forall i \in I,\, j \in J$$

3. **Number of open banks:** Exactly a given number of banks must open.

$$\sum_{i \in I} y_i = \text{nbrblood}$$

4. **Minimum total collection:** Meet minimum blood collection requirement.

$$\sum_{i \in I} \sum_{j \in J} q_c[i]\, x_{ij} \geq 0.9 \times 328$$

5. **Bank capacity:** Do not exceed capacity of any open bank.

$$\sum_{j \in J} x_{ij} \leq q_b[i]\, y_i, \quad \forall i \in I$$

6. **Fuel budget:** Stay within the fuel cost budget.

$$\sum_{i\in I}\sum_{j\in J} f_{bc}\cdot d_{ij}^{LC}\, x_{ij} \le 10,000$$

7. **Hospital delivery:** Each hospital must be supplied by one bank.

$$\sum_{i\in I} z_{ih} = 1, \quad \forall h \in H$$

8. **Delivery only if open:** Blood can only be delivered from open banks.

$$z_{ih} \le y_i, \quad \forall i \in I,\, h \in H$$

9. **Hospital capacity:** Deliveries must not exceed hospital capacity.

$$\sum_{i\in I} z_{ih} \le q_c[h], \quad \forall h \in H$$

10. **Maximum delivery:** Blood supply from banks must not exceed their capacity.

$$\sum_{h\in H} z_{ih} \le q_c[i], \quad \forall i \in I$$

11. **TSP flow balance:** Routing flow must match assignment.

$$\sum_{j\in J,\, j\neq i} u_{kij} = x_{ki}, \quad \sum_{j\in J,\, j\neq i} u_{kji} = x_{ki}, \quad \forall k \in I,\, i \in J$$

12. **MTZ subtour elimination:** Prevent subtours in each route.

$$mtz_{ki} - mtz_{kj} + |J|\, u_{kij} \le |J| - 1, \quad \forall k \in I,\, i,j \in J \setminus \{0\},\, i \neq j$$

5. Initial Methods for Solving

Initially, the problem was formulated as a Mixed Integer Linear Program (MILP) that integrates a facility assignment problem and a TSP into a single objective function. The model assigns each community collection centre to an appropriate blood bank and then determines the most efficient routing sequence to collect and deliver blood donations while satisfying operational constraints.

The mathematical formulation defines:

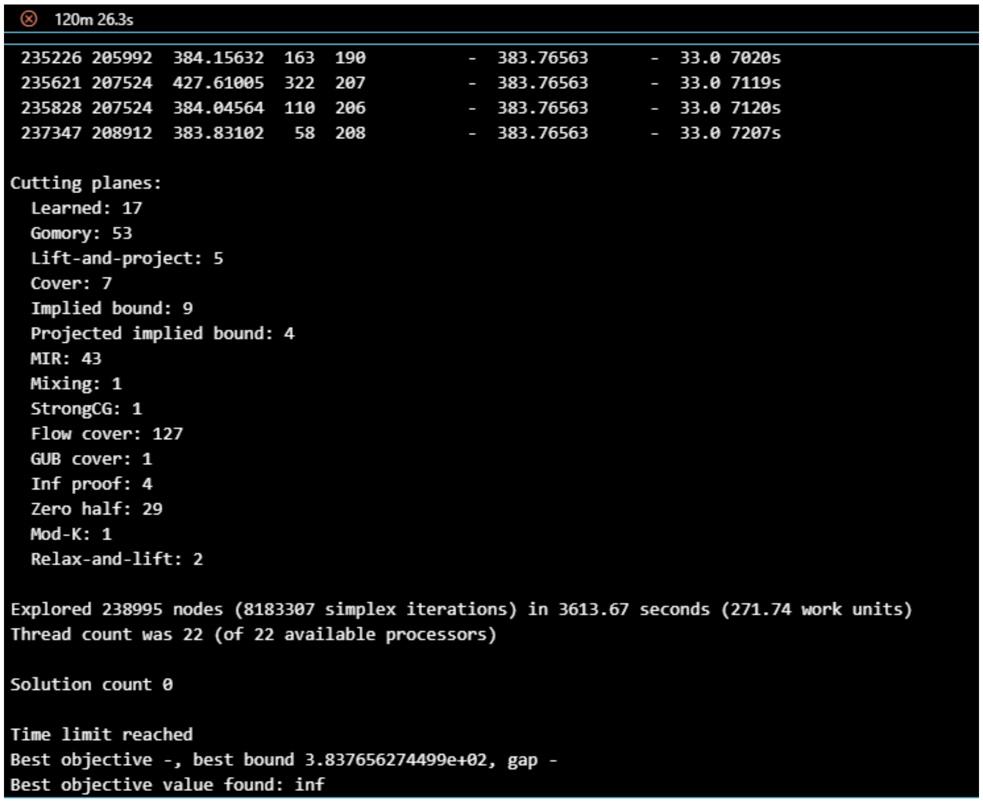
- Sets: blood banks i , collection centres j , hospitals h
- Parameters:
  - Distances between facilities
  - Capacities of blood banks and hospitals
  - Quantities of blood collected
  - Fuel cost per kilometre
- Decision Variables:
  - x_{ij} : assign collection centre j to blood bank i
  - y_i : open status of blood bank i
  - U_{kij} : routing link for vehicle routes
  - mtz_{ki}: subtour elimination for routing
- Objective: Minimise total transport distance and fuel cost for collection, delivery and routing
- Constraints: each collection centre is only assigned once; only open blood banks can be used; limit on the number of open blood banks; minimum blood collection is required; fuel budgets; each hospital is served once; feasible routes enforced without subtours.

The initial model was implemented using the Gurobi optimiser and tested on synthetic datasets representing small-scale supply chains, typically with 3 blood banks and 15 collection centres. For simple instances without subtour constraints, feasible assignments and routes were obtained quickly within 2 seconds.

To further improve the accuracy and complexity of our model, subsequent versions incorporated subtour elimination via MTZ constraints and realistic features such as minimum assignment conditions. These additions increased the model's size and complexity considerably, causing the time complexity to be $O(n^2)$. Enforcing minimally 1 assigned centre per blood bank also caused the problem to become infeasible or unbounded if demand and capacity constraints were not balanced correctly.

The final version of the Gurobi model successfully produced benchmark solutions for small to medium-sized instances within acceptable runtimes (less than 60 minutes). However, tests with larger instances revealed significant limitations, as the computational time increased exponentially, causing the optimiser to run for more than 6 hours without usable solutions. In addition, the memory usage often exceeded available resources, particularly when subtour constraints and time window constraints were active, throwing "Memory Error" for large instances due to extended run times and limited hardware computational capabilities. These results demonstrated that exact methods alone are not practical for large-scale or time-sensitive applications.

The figures below detail the outputs of the original problem statement using Singapore data (Figure 2) and small-scale randomised data (Figure 3).



```
120m 26.3s
 235226 205992  384.15632  163  190          -  383.76563      -  33.0 7020s
 235621 207524  427.61005  322  207          -  383.76563      -  33.0 7119s
 235828 207524  384.04564  110  206          -  383.76563      -  33.0 7120s
 237347 208912  383.83102   58  208          -  383.76563      -  33.0 7207s

Cutting planes:
  Learned: 17
  Gomory: 53
  Lift-and-project: 5
  Cover: 7
  Implied bound: 9
  Projected implied bound: 4
  MIR: 43
  Mixing: 1
  StrongCG: 1
  Flow cover: 127
  GUB cover: 1
  Inf proof: 4
  Zero half: 29
  Mod-K: 1
  Relax-and-lift: 2

Explored 238995 nodes (8183307 simplex iterations) in 3613.67 seconds (271.74 work units)
Thread count was 22 (of 22 available processors)

Solution count 0

Time limit reached
Best objective -, best bound 3.837656274499e+02, gap -
Best objective value found: inf
```

Figure 2 Gurobi Optimiser Output for real data

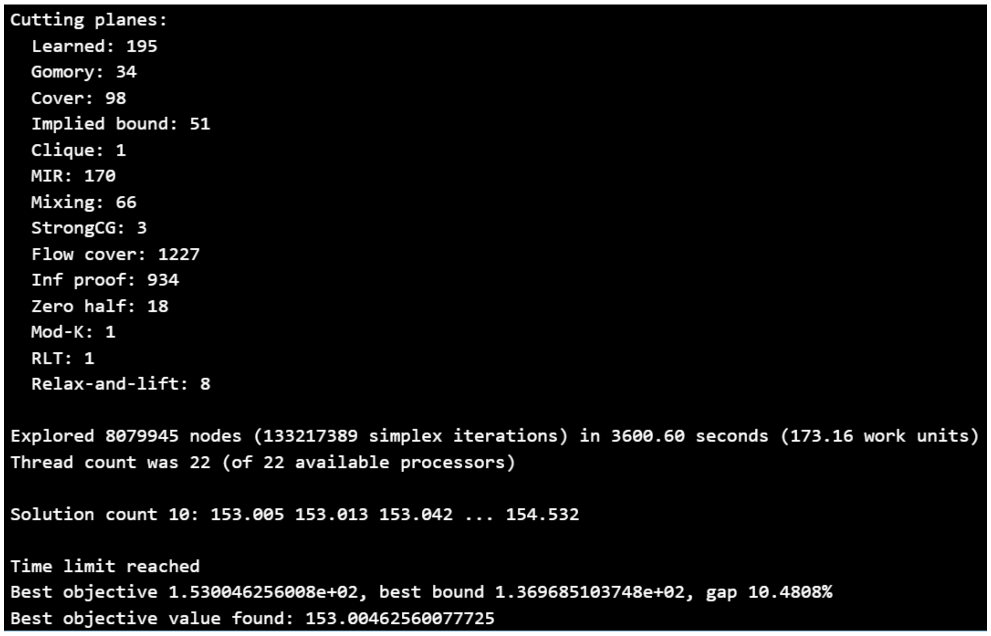

```
Cutting planes:
  Learned: 195
  Gomory: 34
  Cover: 98
  Implied bound: 51
  Clique: 1
  MIR: 170
  Mixing: 66
  StrongCG: 3
  Flow cover: 1227
  Inf proof: 934
  Zero half: 18
  Mod-K: 1
  RLT: 1
  Relax-and-lift: 8

Explored 8079945 nodes (133217389 simplex iterations) in 3600.60 seconds (173.16 work units)
Thread count was 22 (of 22 available processors)

Solution count 10: 153.005 153.013 153.042 ... 154.532

Time limit reached
Best objective 1.530046256008e+02, best bound 1.369685103748e+02, gap 10.4808%
Best objective value found: 153.00462560077725
```

Figure 3 Gurobi Optimiser Output for Small-scaled testing

The existing lower bound objective value of the small-scale model is 153.00, while running the model using data from Singapore proved to be too complex as it was unable to produce feasible solutions within the time parameter set. Hence, the small-scale test was used as a benchmark to test the accuracy of the metaheuristic model results.

6. Metaheuristic Method

Due to the limitations of the current approach, the Ant Colony Optimisation (ACO) was chosen for the metaheuristic model, for its suitability for problems with nodes and strong abilities for finding ideal paths to take across the various nodes. ACO utilises a population of 'ants' to explore the search space in detail. Each ant builds a solution and updates their pheromone trail, which stores memory of the path that they have taken. Good solutions are reinforced with stronger pheromone trails, while weaker solutions experience evaporation as the pheromone trails on those paths wear off. It is flexible in adapting to additional constraints such as capacity and demand constraints, which is suitable for the complexity of the problem.

To adapt the ACO into the specific problem, the basic pheromone update and path construction rules were modified to incorporate distance, demand and capacity data from the Gurobi formulation. Each 'ant' constructs a feasible route by probabilistically selecting the next node based on pheromone levels and the inverse of distance, while penalising routes that violate capacity or time window constraints. The pheromone trails are updated at the end of each iteration based on the quality of solutions found, encouraging the exploration of shorter, feasible routes.

The Ant Colony Optimisation algorithm used is largely based on the following formula for updating of the heuristic control and explorative controls of the 'ant' movements.

$$P_{ij}^k = \begin{cases} \dfrac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_{l \in N_i^k} (\tau_{il})^\alpha (\eta_{il})^\beta}, & \text{if } j \in N_i^k \\ 0, & \text{otherwise} \end{cases}$$

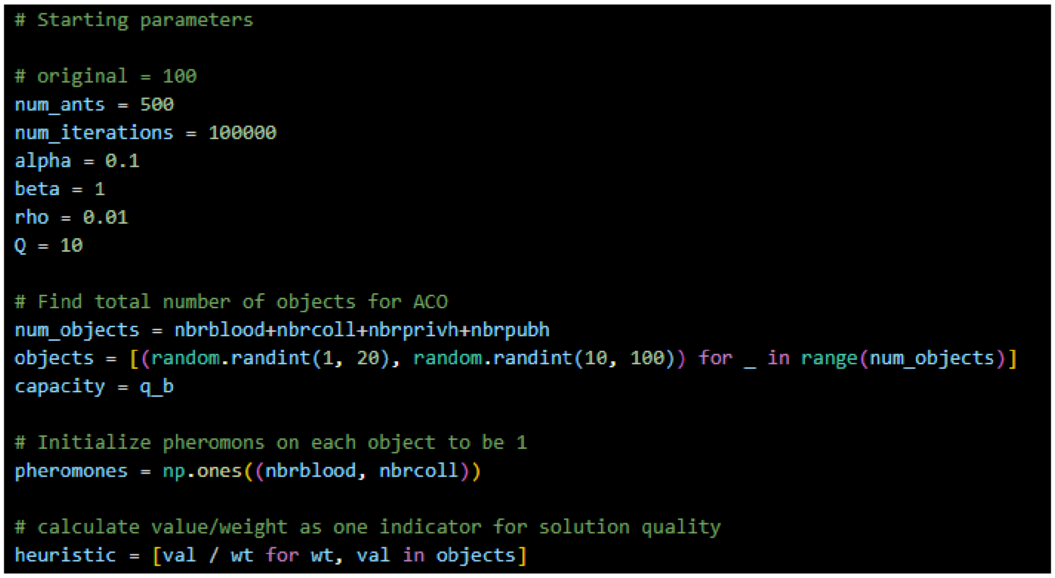
$$\tau_{ij} \coloneqq (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k$$

$$\Delta\tau_{ij}^k = \begin{cases} \dfrac{Q}{L_k}, & \text{if ant } k \text{ used edge } (i,j) \\ 0, & \text{otherwise} \end{cases}$$

$$\eta_{ij} = \frac{1}{d_{ij}}$$

$$\tau_{ij}(0) = \tau_0$$

On initialisation of the ACO algorithm, the parameters set are as seen in Figure 4 below.



```
# Starting parameters

# original = 100
num_ants = 500
num_iterations = 100000
alpha = 0.1
beta = 1
rho = 0.01
Q = 10

# Find total number of objects for ACO
num_objects = nbrblood+nbrcoll+nbrprivh+nbrpubh
objects = [(random.randint(1, 20), random.randint(10, 100)) for _ in range(num_objects)]
capacity = q_b

# Initialize pheromons on each object to be 1
pheromones = np.ones((nbrblood, nbrcoll))

# calculate value/weight as one indicator for solution quality
heuristic = [val / wt for wt, val in objects]
```

Figure 4 ACO Initialisation Parameters

To ensure that the solutions obtained are robust and prevent local optima traps, the pheromone levels are updated if stagnation occurs when the objective value does not improve after 50 iterations. This helps to increase exploration and improve the solution.

Although no explicit local search or swap operators were used, constructive assignments are used in the algorithm above, as each ant builds a solution by sequentially assigning each CC to a BB. Additionally, random exploration moves are included, which makes the assignments probabilistically random by constantly updating the value "exploration_prob".

The addition of subtour checking and penalties for solutions with subtours helped to integrate the MTZ condition into the metaheuristic, to produce better solutions that were closer to the optimised values in the small-scale test.

7. Performance Benchmark

To evaluate the metaheuristic's performance, a benchmark test was conducted using a small-scale instance for which Gurobi could still generate optimal or near-optimal results. The smaller scale was required for Gurobi to be able to solve in a reasonable amount of time. Larger-scale models were too intense for Gurobi to handle, leading to errors deriving tangible solutions or unrealistically long runtimes for the solver to complete, thus resulting in no useful solutions.

For this project, the small-scale instance used was 3 blood banks and 15 collection centres. This instance proved reasonably hard for Gurobi to optimise within a single objective function, but still solvable within a reasonable duration. Hence, considering the original scale of Singapore's context (5 blood banks and 125 community/collection centres), the small-scale instance is a good benchmark for comparison of the metaheuristic algorithm accuracy against the actual optimised value.

8. Insights Into Metaheuristic Performance

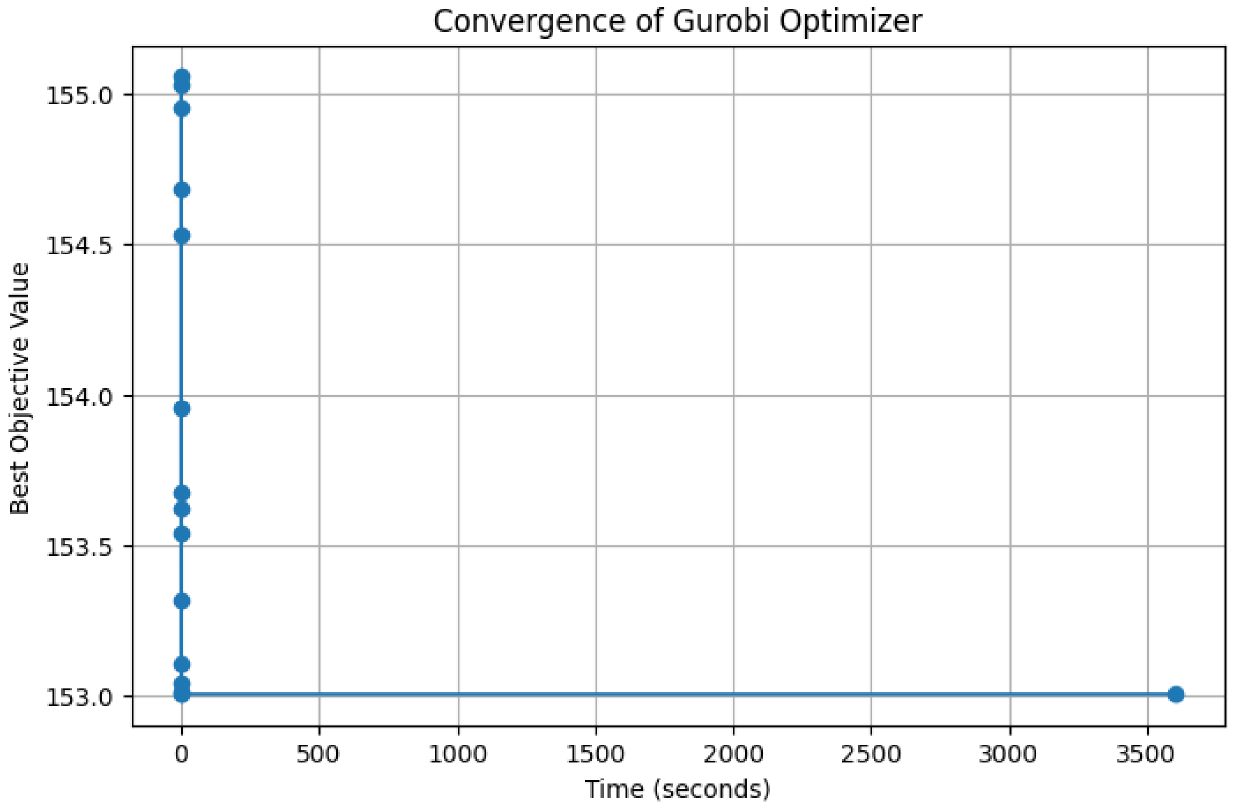


Figure 5 Convergence plot of Gurobi optimiser

The results indicate that the ACO method converges rapidly to good quality solutions, typically within 150 iterations. The convergence pattern in Figure 5 depicts the stabilisation of pheromone trails around efficient routes that satisfy the capacity and time constraints performed on the Gurobi optimiser. The convergence plot enables the observation of certain trends, such as when the optimiser is stuck on various local optima, or how rapidly it obtains a stable solution. A stable solution would be defined as a local optimum that the solution does not deviate heavily any further for the duration of the solve time. As seen in Figure 5, a stable solution is found very rapidly and remains there for the remainder of the program running. This can then be compared to the performance of the metaheuristic.

Multiple versions of the metaheuristic algorithm were tested, with the final ones being shown below. It was tested on two different sets of data, the first being the one derived directly from the real-world Singapore data. The second was tested on a scaled-down version of the data for more efficient solve times.

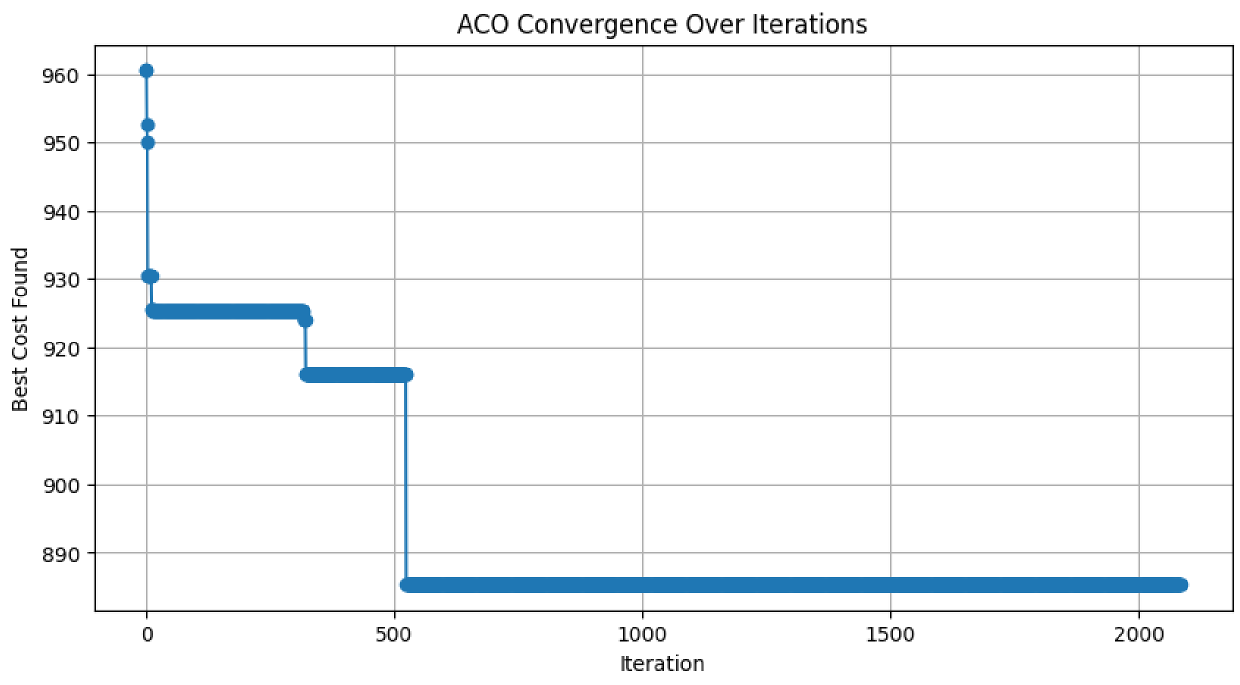


Figure 6 Convergence plot of ACO over iterations using real data

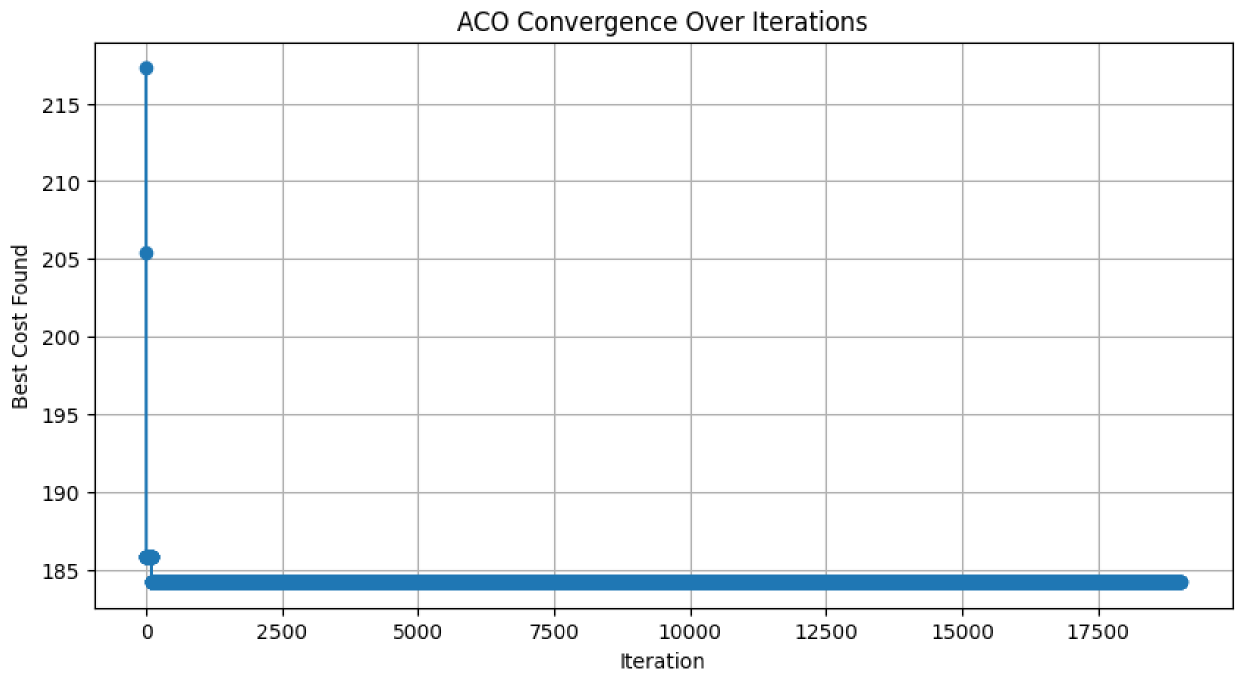


Figure 7 Convergence plot of ACO over iterations using small-scale simulated values

In Figure 6, the ACO algorithm uses Singapore data. This greatly increased the complexity of the algorithm, requiring a significantly higher solve time. It produced higher variation in iterations as well as requiring longer computational time to derive the same number of iterations as the scaled-down version. The bulk of solutions are found within the first 200 iterations, with drastic changes to the value of the solutions and shorter time spent in local optima in this period. After around the 200th iteration, the iterations converge to the same local optimum for the remaining duration of the algorithm's runtime.

In Figure 7, the ACO algorithm is run on a smaller-scale model. It converges very rapidly to a local optimum within the first few hundred iterations. It maintains this local optimum for most of its runtime, staying there for nearly 20,000 iterations. It can generate a significantly higher number of iterations due to lower complexity enabling the more rapid generation of new iterations.

It can be observed that using real data allows for more exploration of the search space, yet the higher complexity yields fewer solutions. A simplified model yields a better response but fails to explore the search space as thoroughly, likely due to fewer potential solutions being available.

In comparing the output of the small-scale test results, the optimiser produced a lower bound objective value of 153.00 and the metaheuristic objective value after 100 minutes of run time was 184.21. This gives an error margin of 20.4%, which is a fair result, as the convergence on Gurobi for even small test cases would take upwards of 2 hours (120 minutes).

Comparing the visual plots of the solutions generated, we realised that the optimised solution from Gurobi assigned all collection centres to a single blood bank (Figure 8). An interpretation of this result could be that for 15 collection centres, only 1 blood bank is necessary, rendering an excess of 2 blood banks built. However, the solution given by the metaheuristic algorithm was able to assign at least 1 collection centre to each blood bank, which is a more appropriate solution than that of the optimiser (Figure 9). In our attempt to override the solution generated from Gurobi, we added a constraint to ensure that at least one collection centre should be assigned to each blood bank, which significantly raised the computational complexity of the problem. Hence, this constraint was dropped for the purpose of this project.

Considering the quality of the metaheuristic solution with the time complexity and context of the problem, the metaheuristic solution provides a good approximation to an optimal solution.

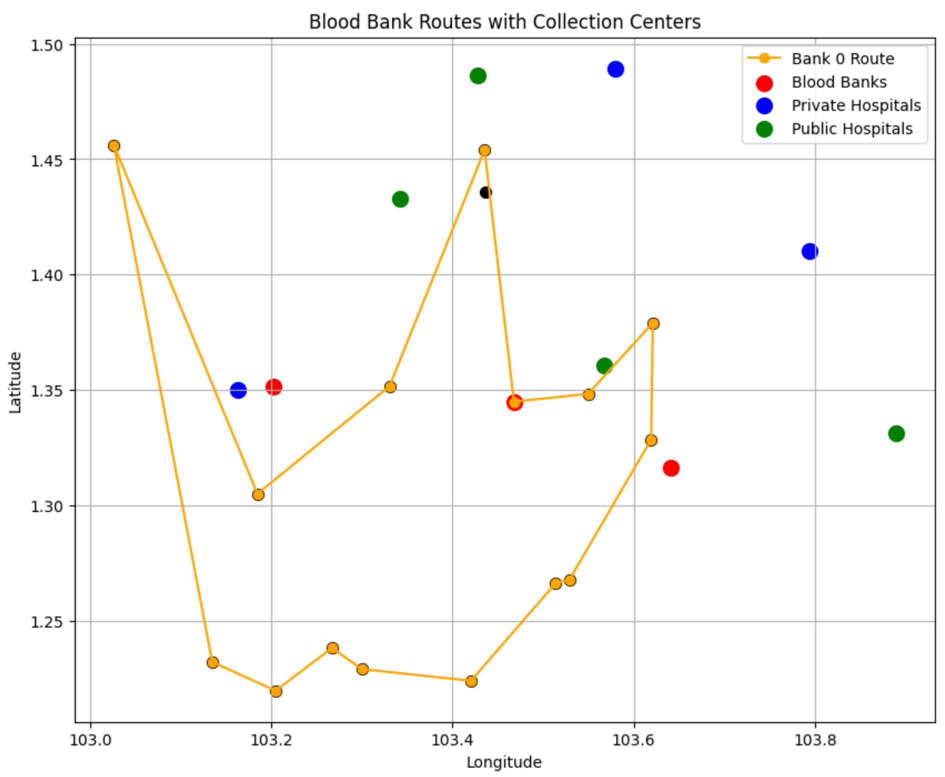


Figure 8 Visual plot of solution from Gurobi of small-scale instance

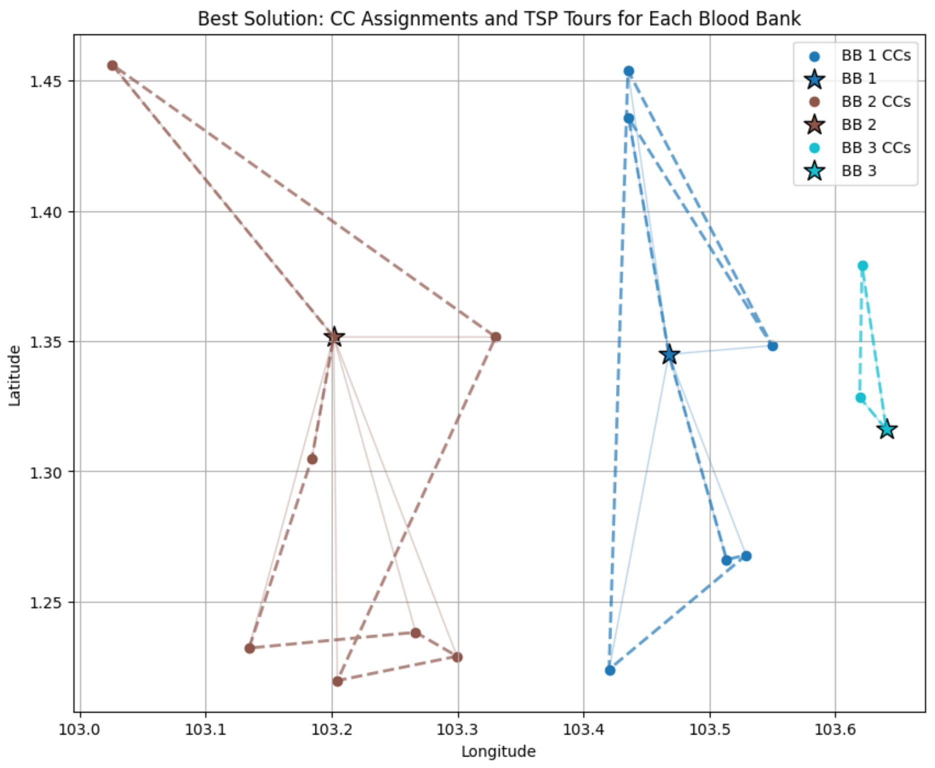


Figure 9 Visual plot of ACO solution for small-scale instance

9. Conclusion

The ACO algorithm provides a close modelling of the original problem and a good suboptimal solution, perhaps even better. Considering the closeness of small-scale results, computational time and margin of error, the solutions given by the ACO algorithm are good approximations to the actual optimal solutions. Hence, the solution for the original instance (Singapore's data), can be extrapolated using the metaheuristic solution.

A further realistic improvement to the objective function would be to split the assignment and TSP components of the problem into separate optimisation problems, such that the optimiser would be able to individually solve TSP problems for each optimal assignment solution. This would also greatly reduce the time complexity of the problem and give the solver more space to accommodate tighter constraints such as ensuring at least 1 collection centre is assigned to each blood bank.

The exploratory nature of the algorithm makes it highly suitable to explore problems such as blood bank-collection centre assignments and TSP that involve various paths that can be taken and produces results very efficiently as well. Applying the ACO algorithm to solve complex problems possess immense potential to improve existing systems in sectors such as healthcare, which is the focus of this problem. It can be suggested as a recommendation for analysts in these fields to explore such metaheuristic methods in detail to derive positive predictive statistical outcomes.

References

(n.d.). Retrieved from https://vrl.lta.gov.sg/lta/vrl/action/pubfunc?ID=FuelCostCalculator

Community Clubs — PA. (n.d.). Retrieved from https://data.gov.sg/

data.gov.sg. (n.d.). *Blood donors and blood donations, annual*. Retrieved from SINGSTAT: https://data.gov.sg/datasets/d_440e6268902c6ef0abab0d1b647c29f0/view

Give Blood Save Lives. (n.d.). Retrieved from Give Blood Save Lives: https://giveblood.sg/

HSA. (2018, December 31). *HSA*. Retrieved from Blood facts and figures: https://www.hsa.gov.sg/blood-donation/facts-and-figures