Robot Charging Station Construction on Antartica

Arin Srisajjalertwaja (S2670853), Puttiwat Wanna (S2745852),
Theeramet Sinthumongkolchai (S2652504)

April 8, 2025

Contents

1 Question 1: Deterministic Variant **1**

1.1 Question 1a: MINLP Formulation . . . 1

1.2 Question 1b: Construction Heuristic Formulation . . . 4

1.2.1 Results and Analysis . . . 5

1.3 Question 1c: Improvement Heuristics . . . 6

1.3.1 Improvement Heuristic Approaches . . . 6

1.3.2 Results and Analysis . . . 9

2 Question 2: Stochastic Version **11**

2.1 Model Formulation . . . 11

2.1.1 Additional Sets and Notation . . . 11

2.1.2 Adjusted Input Variables . . . 11

2.1.3 Adjusted Output Variables . . . 11

2.1.4 Adjusted Constraints . . . 12

2.1.5 Objective Function . . . 12

2.2 Modified Heuristics Approach . . . 12

2.3 Results . . . 13

3 Conclusion **15**

Introduction

This project tackles a large-scale logistical optimization problem involving the strategic placement and sizing of charging stations for 1,072 autonomous solar-powered robots operating in Antarctica. Each robot is equipped with a drone capable of reaching a charging station within a specific range. If a robot cannot reach a station due to insufficient remaining range, it must be manually transported at a higher cost.

The optimization aims to minimize total system cost, which includes fixed station construction costs, per-charger maintenance costs, energy usage for charging, and penalties for manual transport. The challenge is modeled in both deterministic and stochastic frameworks.

In the deterministic version, a Mixed-Integer Nonlinear Programming (MINLP) model is used to represent a simplified version of the problem due to computational complexity. To handle larger problem sizes, a construction heuristic—based on greedy allocation and weighted centroid calculations—is proposed. Additional refinement is performed using improvement heuristics including centroid adjustments, station reduction based on cost-efficiency thresholds, and local search via robot reassignment.

The stochastic version extends the deterministic model by incorporating uncertainty in robot range via multiple probabilistic scenarios. It adapts both the mathematical formulation and heuristic methods to handle these uncertainties while maintaining computational efficiency.

Overall, the study compares the performance of the exact MINLP model with the heuristic methods in terms of cost, runtime, and scalability, providing insights into practical approaches for deploying infrastructure in remote environments with dynamic energy constraints.

1 Question 1: Deterministic Variant

1.1 Question 1a: MINLP Formulation

The problem can be formulated as a Mixed-Integer Nonlinear Programming (MINLP), which is difficult to solve to find the solution for a large problem. Hence, this model is executed for only small subset of robots, approximately around 20 - 150 among 1072 robots. The model for executing the small sub-problem is formulated below.

1. **Sets and Notation**
   - S: Set of candidate charging station indices, $S = \{1, 2, \ldots, N_y\}$
   - V: Set of robot indices
2. **Input Variables**
   - X_v, Y_v: Longitude and latitude of robot v ; $v \in V$
   - D^{max} [km]: Maximum Distance among the robot sample, which is calculated by
   $$D^{max} = \sqrt{(\max\{X_v\} - \min\{X_v\})^2 + (\max\{Y_v\} - \min\{Y_v\})^2}.$$
   - R^{max} [km]: Maximum travel range per robot ; $R^{max} = 175$ km
   - R_v [km]: Current range of robot v ; $v \in V$
   - Q [robot/charger]: Maximum robots per charger ; $Q = 2$
   - M [charger/station]: Maximum chargers per station ; $M = 8$
   - L [station]: Minimum number of stations required:
   $$L = \left\lceil \frac{|V|}{M \cdot Q} \right\rceil,$$
   where $M \cdot Q$ is for the maximum amount of robot per 1 station, or 16.
   - C^b [£/station]: Built costs of each charging station ; $C^b = 5000$ £/station

- C^h [£/robot]: Penalty costs for robot being manually transported ; $C^h = 1000$ £/robot
- C^m [£/charger]: Charger Installation costs ; $C^m = 500$ £/charger
- C^c [£/km]: Charging Costs ; $C^c = 0.42$ £/km

3. **Output Variables**
   - x_s, y_s: Coordinates of station s; $s \in S$
   - μ_s: Decision to build station s ; $s \in S$
   - η_s: Number of chargers at station s; $s \in S$
   - $\beta_{v,s}$: Decision whether robot v is assigned to station s; $v \in V$, $s \in S$
   - $\alpha_{v,s}$: Decision whether robot v is brought manually by human to station s; $v \in V$, $s \in S$
   - $d_{v,s}$: Distance robot v will travel to station s $v \in V$, $s \in S$

4. **Constraints**
   - General Constraints

$$\begin{aligned} x_s, y_s &\in R &&;\forall s \in S, \\ \mu_s &\in \{0,1\} &&;\forall s \in S, \\ \eta_s &\in \{0,1,2,...\} &&;\forall s \in S, \\ \alpha_{v,s}, \beta_{v,s} &\in \{0,1\} &&;\forall v \in V, \forall s \in S \\ d_{v,s} &\geq 0 &&;\forall v \in V, \forall s \in S \end{aligned}$$

   - Location of station s should be inside the robot area, so

$$\begin{aligned} x_s &\leq \max\{X_v\} &&;\forall s \in S \\ x_s &\geq \min\{X_v\} &&;\forall s \in S \\ y_s &\leq \max\{Y_v\} &&;\forall s \in S \\ y_s &\geq \min\{Y_v\} &&;\forall s \in S \end{aligned}$$

   - Robot v can only be assigned to built stations s, so

$$\beta_{v,s} \leq \mu_s \quad ;\forall v \in V,\ s \in S$$

   - Each robot v must be assigned to exactly one station, so

$$\sum_{s \in S} \beta_{v,s} = 1 \quad ;\forall v \in V$$

   - Total Number of stations must be more than the computed minimum number of station, so

$$\sum_{s \in S} \mu_s \geq L$$

   - Number of chargers at each stations is determined by

$$\eta_s \geq \frac{1}{Q} \sum_{v \in V} \beta_{v,s} \quad ;\forall s \in S$$

   - If a robot v is assigned to station s, it must either go by itself or be transported (by human manually)

$$\alpha_{v,s} \leq \beta_{v,s} \quad ;\forall v \in V,\ s \in S$$

- Moreover, distance that robot v will travel to station s is determined by constraints below,

$$\begin{aligned}
&d_{v,s} \geq \sqrt{(X_v - x_s)^2 + (Y_v - y_s)^2} - D^{max}(\alpha_{v,s} + (1 - \beta_{v,s})) &&;\forall v \in V,\ s \in S\\
&d_{v,s} \leq D^{max} \cdot \beta_{v,s} &&;\forall v \in V,\ s \in S\\
&d_{v,s} \leq R_v \cdot (1 - \alpha_{v,s}) &&;\forall v \in V,\ s \in S
\end{aligned}$$

  Combining with the previous constraint and $d_{v,s} \geq 0$, three cases can occur. The first case is where robot v is not allocated to charge at station s. This means $\beta v, s = 0$ and $d_{v,s} \leq D^{max} \cdot 0 = 0$, causing $d_{v,s} = 0$. The second case is where robot v is allocated and can travel to charging station v. This means $\beta_{v,s} = 1$ *and* $\alpha_{v,s} = 0$, resulting to $d_{v,s} \geq \sqrt{(X_v - x_s)^2 + (Y_v - y_s)^2}$, $d_{v,s} \leq D^{max}$, and $d_{v,s} \leq R_v$. Therefore, the travel distance, $d_{v,s}$, is bounded by,

$$\sqrt{(X_v - x_s)^2 + (Y_v - y_s)^2} \leq d_{v,s} \leq R_v,$$

  which in this case, $d_{v,s}$ should be at the lower bound, the euclidean distance, to minimize the charging cost (minimize the travel usage).

  The last case is where robot v is allocated but cannot travel itself to station s due to its insufficient travel range, R_v. This means $\beta_{v,s} = 1$ *and* $\alpha_{v,s} = 1$, resulting in $d_{v,s} \leq R_v \cdot (1 - \alpha_{v,s}) = 0$. Since $d_{v,s}$ is non-negative, it implies that, in this case, $d_{v,s} = 0$ (the robot doesn't travel by itself and is brought by human instead).

5. **Objective Function**

$$\min(\sum_{s \in S} C^b \mu_s + \sum_{s \in S} C^m \eta_s + \sum_{v \in V} \sum_{s \in S} \left[C^h \alpha_{v,s} + \ C^c \left[R^{max} \beta_{v,s} - (R_v \beta_{v,s} - d_{v,s})\right]\right])$$

   The first and second terms denote the cost of installing stations and chargers, respectively. The last term determines the operation cost of each robot. It includes the penalty cost for robots being transported manually to the charging station and the charging cost. When robot v is allocated to station s, and can commute to that station, the operation cost depends on how far it travels and its fuel remaining, $C^c \left[R^{max} - (R_v - d_{v,s})\right]$. If it is allocated but cannot travel to that station, the operation cost should include the penalty, which is $C^h + C^c \left[R^{max} - R_v\right]$.

Due to the complexity of the MINLP model, the experiment was conducted on a small subset of robots and the model was executed using Xpress solver, with time limit of 5 minutes. The resulting outputs are shown in the table below.

N_Sample	Running_Time (s)	Status	%Gap	Obj	Lower Bound
25	85.41	Auto Terminate	2.05	17681.08	17318.33
50	27.39	Auto Terminate	0.68	34755.97	34517.93
75	336.65	Timeout	1.33	47428.93	46793.88
100	339.98	Auto Terminate	0.83	64541.94	64000.09
125	391.17	Timeout	1.37	77353.22	76287.23
150	525.62	Auto Terminate	0.70	94364.06	93700.93

Table 1: Model performance for different sampled robot size

From observation, the distance constraints in this model is Non-convex function. So, it is uncommon for the solver to provide an optimal solution. However, the solver can automatically terminate itself if it cannot find improving direction (case N = 25), or if the optimality %Gap is sufficiently small (case N = 50, 100, 150), assuming that the solution at that iteration is approximately optimal. Therefore, the cases where the solver provides near-optimal solutions are selected as benchmarks for comparison with the output from heuristics algorithms. The geoplot and result comparison will also be presented in section 1.2.1.

1.2 Question 1b: Construction Heuristic Formulation

Noted that the definitions of all notations used will be the same as in the MINLP formulation. The construction heuristic uses greedy algorithm to greedily construct stations by avoiding penalties (i.e. avoid the use of human-operated vehicle). It first begins by iterating through all robots by finding the next robot as the one closest to the current one. This was done by pre-processing the data to get the distance matrix. Then for each robot, try putting it in all existing opened stations. Then in each station that the robot was put in, calculate the new centroid (i.e. the new location for this station) and total cost for all robots assigned (i.e. not including the unassigned robots). The cost can be calculated follows.

$$C_s^{total} = C_b + \lceil \frac{|v \in (V|s)|}{Q} \rceil \cdot C_m + \sum_{v \in (V|s)} C_v^{charge}$$

where C_v^{charge} is the charging cost of robot v once it reaches the station taking the penalty into account, which can be calculated as follows.

$$C_v^{charge} = \begin{cases} C_c \cdot (R^{max} - R_v + d_{v,s}), & \text{if } R_v \geq d_{v,s}, \\ C_c \cdot (R^{max} - R_v) + C_h, & \text{otherwise} \end{cases}$$

To pick which station the robot should be in, follow the conditions below.

- The stations has less than maximum robots per station ($M \times Q = 8 \times 2 = 16$).
- The new centroid for the station should not make any robots assigned (including the new robot) become penalized (i.e. every robots can travel to the station by themselves).
- The total cost should be lowest as possible (i.e. pick the station with the lowest cost that satisfies all conditions).

If the above conditions cannot be satisfied for any existing stations, a new station is opened for that robot.

The pseudo code below shows the main steps in this approach.

Algorithm 1: Greedy Construction

```
1  Initialize $stations$ = [ ]
2  for $each\ v$ do
3      Initialize $best\_cost = -1$
4      Initialize $station\_to\_insert$ = -1
5      for $s \in stations$ do
6          $centroid$ = find weighted centroid
7          if $exist\ penalized\ robot$ then
8              skip this and continue to the next station
9          $cost$ = calculate cost of assigning the robot $v$ to this stations $s$
10         if $cost < best\_cost$ then
11             Set $best\_cost = cost$
12             Set $station\_to\_insert = s$
13     if $station\_to\_insert \neq -1$ then
14         Add $v$ to $stations[station\_to\_insert]$
15     else // no station meets the conditions
16         Add $[v]$ to $stations$
```

The main idea behind this greedy approach is the concept of a **Weighted Centroid** for each station. In this context, the weight is calculated from the robot's depleted range, specifically defined as $r_{max} - r_i$ where r_i is the robot remaining range. This weight is the used to find the location of each station. The location can be calculated as follows.

$$x_s = \sum_{v \in (V|s \in S)} \left(\frac{R^{max} - R_v}{\sum_{v \in V}(R^{max} - R_v)} \times X_v \right)$$

$$y_s = \sum_{v \in (V|s \in S)} \left(\frac{R^{max} - R_v}{\sum_{v \in V}(R^{max} - R_v)} \times Y_v \right)$$

This weight reduces the chance of getting a penalty for robots, decreasing the creation of new stations.

An example is given below. Robot v will be added to station s only if $d_1 \leq R_1$, $d_2 \leq R_2$, $d_3 \leq R_3$, and $d_v \leq R_v$. Otherwise, if there is no other stations, a new station will be created for robot v.

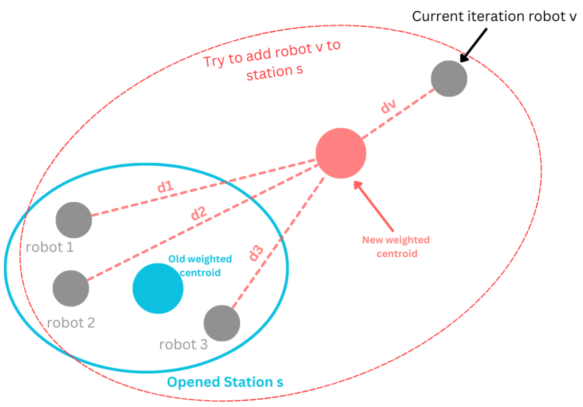

Figure 1: Example Construction Heuristic Iteration

To do the iterations, the starting robot needs to be defined. Initially, the first robot in the list was used to run the algorithm. Then to try to improve this, the random start method was applied to simply random the starting robot from all available robots. The amount of random start used was 10% of the robots amount, rounded up if needed. The construction heuristic algorithm is then run against all of those starting robots. After that, the results with the lowest cost (lowest objective value) will be used to process further in the improvement heuristics. A case where the robot starts at the first one was also tested.

1.2.1 Results and Analysis

The following shows a result comparison between the MINLP and construction heuristics using a sample of 100 robots. Noted that the %Gap of heuristic method was calculated comparing to the lower bound of the MINLP.

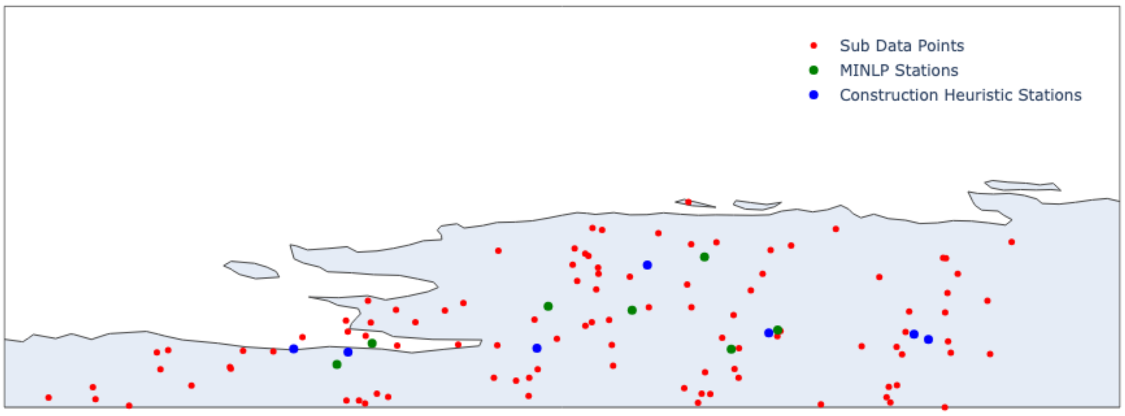

Figure 2: Stations Locations Comparison between MINLP and Construction Heuristics

Method	# Stations	Objective Value	%Gap	Runtime (s)	Lower Bound
MINLP	7	64,541.94	0.83	353	64,000.09
Construction Heuristics (Random Start)	7	64,245.57	0.38	0.08	-
Construction Heuristics (Starting Robot 1)	7	64,291.33	0.45	<0.01	-

Table 2: Performance of Each Method

Comparing to the MINLP, the runtime in the construction heuristics is significantly lower than that of the MINLP, with also a better objective value. This guarantees the performance of the proposed construction heuristic approach.

Comparing the results of with and without random start, the one with random start seems to run relatively longer than the one without. This is expected because the random start runs more iterations.

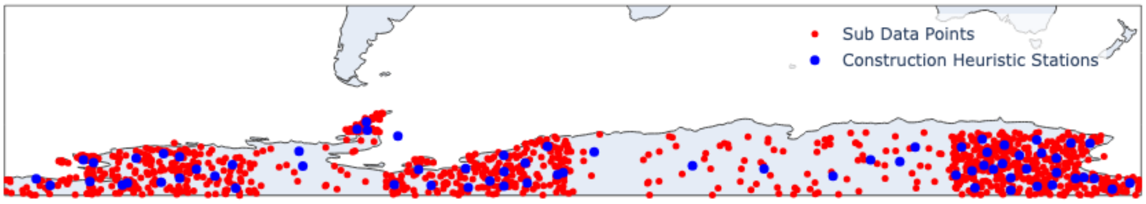

Figure 3: Locations from Running All Robots

Method	# Stations	Objective Value	Runtime (s)
Construction Heuristics (Random Start)	69	660,133.15	493.22
Construction Heuristics (Starting Robot 1)	70	665,141.04	4.15

Table 3: Performance of Each Method

Running with random start seems to still yield better objective value in the case of large amount of robots. It also aligns with the number of stations opened, which impacts directly to the cost. However, since the random start method ran the construction heuristics for 10% of robots as a number of times, it is fair that it took significantly longer runtime than the fixed starting one. In this case, there are 1,072 robots, meaning that the random start ran $\lceil 0.1 \times 1,072 \rceil = 108$ times. The runtime, thus, looks reasonable, which should be around $4.15 \times 108 = 448.2$ seconds, where it is a bit higher in the results due to some differences during the process.

Comparing to the MINLP in the case of using 100 robots, the random start in this case of using all robots still runs in the close amount of time, implying that the performance is still greatly improved. In some cases, one might consider to decrease the number of random starts in order to receive better runtime, but in this case, the runtime is acceptable.

1.3 Question 1c: Improvement Heuristics

1.3.1 Improvement Heuristic Approaches

To improve the results, three improvement heuristic approaches were applied sequentially as follows.

1. **Improved Centroid**
   After the construction heuristic approach, the candidate locations for each station were retrieved through the weighted centroid calculation. The main goal of this approach is to find more optimal locations for those locations.

   For each station, initially, a new target station is built using a **Regular Centroid**. Then, the middle point (average) between this new target centroid and the current location (weighted centroid) is calculated. If this new average point satisfies the cost improvement condition, which means to have no penalty (no need for the human-operated vehicle), the weighted centroid is moved to that location. Otherwise, the new regular centroid is moved to the location instead. This iterates until either:

   - The new regular centroids have been updated 20 times, or
   - The improvement in cost is less than £100.

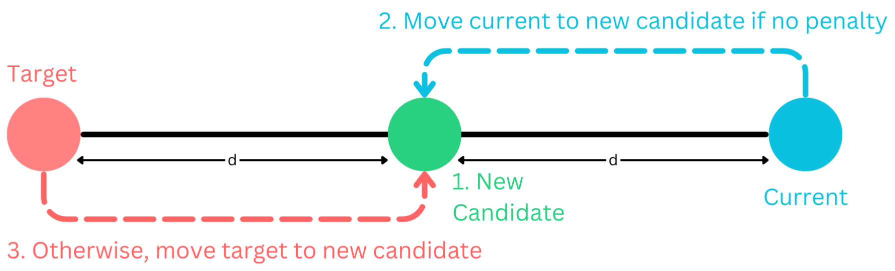
2. Move current to new candidate if no penalty
Target
d
d
1. New Candidate
Current
3. Otherwise, move target to new candidate

Figure 4: Improved Centroid

The reason behind this is that in normal situation, disregarding the amount of fuel left, the normal centroid is the point of minimum sum distance among all robots. Assuming that the robots all have sufficient fuel to travel to the station, the minimum sum distance would be the best location. Therefore, the objective to move to that best location in normal situation is aimed here with an expectation that it could improve the cost.

After running, however, this method does not improve much, or not at all, in terms of the cost. The results and reasons will be explained further in the later section.

2. **Station Reduction**
A major weakness of the previous model is the lack of assessment regarding whether a station is worth opening. The prior heuristic assigns every robot to its own respective station by avoiding any penalty of using the human-operated vehicle. However, using a rough benchmark cost of £1000 per robot for manual collection, it may be cheaper to avoid opening a station if it serves not more than 5 robots. For instance, serving 5 robots manually costs £5000, which is equivalent to the base cost of opening a station—excluding charger and charging costs. The calculation can be shown below.

The cost from having a station with 5 robots is as follows.

$$\begin{aligned} C_s^{total} &= C_b + \lceil \frac{5}{Q} \rceil \cdot C_v^{charge} \\ &= 5000 + \lceil \frac{5}{2} \rceil \cdot + C_v^{charge} \\ &= 6500 + C_v^{charge} \end{aligned}$$

The cost of having 5 robots incurring penalty in the opened station is as follows.

$$\begin{aligned} C_s^{total} &= 5 \cdot C_h + \lceil \frac{5}{Q} \rceil \cdot C_m + C_v^{charge} \\ &= 5 \cdot 1000 + \lceil \frac{5}{2} \rceil \cdot 500 + C^{charging} \\ &= 6500 + C_v^{charge} \end{aligned}$$

Looking at the number roughly, both cases incur the same cost. If the number of robots is less than 5 (e.g. 4), it is obvious that the cost of having penalty is less than building a new station, hence, those 4 robots should be allocated to the opened station, if available. However, looking at the cost of having 5 robots closely, in the penalty case, the *charging_cost* would be lower since the robot does not need to travel, thus, burning more fuel. In addition, if the already opened station has an odd number of robots, one charger can be assigned one of the 5 robots, hence, only 2 more chargers will be required to build, making the cost of having penalty less than the cost of building the new one. Therefore, with 5 robots, if possible, having them as a penalty in the opened stations would have less cost than opening a new station for them.

With the reasons above, this heuristic approach will try to close stations serving not more than 5 robots and will re-assign them to the opened stations, if possible. The steps are as follows.

(i) First, determine if the affected robots can be reassigned to other nearby stations without incurring penalty costs.

(ii) If not, apply a penalty for manual transport by assigning to any available stations.

To assign penalized robots efficiently, the algorithm proceeds as follows.

(i) Identify stations that have more than 5 robots and an odd number of unassigned robot slots. Then assign one robot to each of those stations.

(ii) If there are still robots left unassigned, reassign robots in pairs (two at a time) to stations with fewer than 16 robots until all unallocated robots are reassigned. Noted that after the first step, if there are still robots left, all stations that have less than 16 robots assigned will have at most 14 robots (since all stations with odd number of robots were already assigned one robot each, making them all have even number of robots).

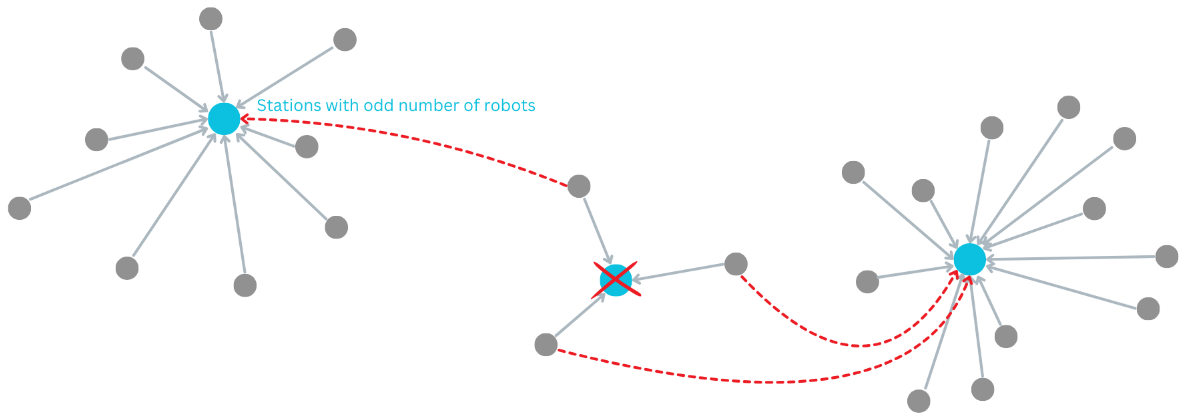

Figure 5: Station Reduction: All Moved Robots Penalized

From the example above, the station with three robots will be demolished and the three robots will be distributed out to other stations. Assuming that there are only these other two stations opened and moving any robots into those stations will always cause the penalty, one robot will go to the station with nine (odd number) robots, to share the charger that was assigned only one robot, while the other two would get transferred together in a pair. This approach helps in reducing both operational and computational costs.

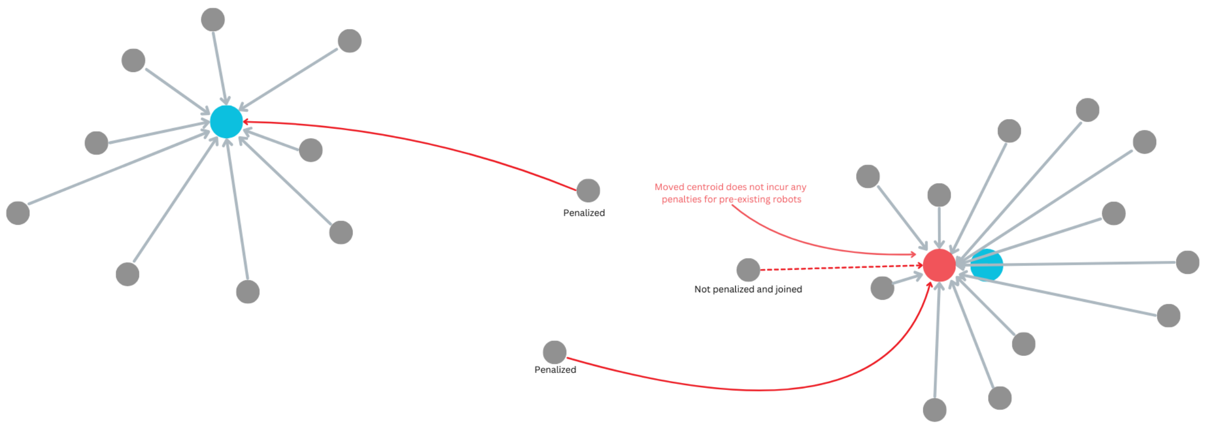
Penalized
Moved centroid does not incur any penalties for pre-existing robots
Not penalized and joined
Penalized

Figure 6: Station Reduction: One Robot Merged Peacefully

On the other hand, assuming that one of the robots can join in with the new station that it was moved to and cause the station to have a new weighted centroid without incurring any penalty for any existing robots, including itself, the behavior will be changed to move the robot to that station and change the centroid of that station to the new weighted one instead.

3. **Local Search Interchange Heuristics (1-Neighborhood)**
   This heuristic focuses on optimizing robot-to-station assignments by swapping robots between stations. The steps are as follows.

   (i) For each pair of stations (e.g., Station 1 and Station 2), find all possible pairs of robots (i.e. find a Cartesian of all robots) between the two stations.

   (ii) Attempt to swap each pair of the robots in the Cartesian product retrieved from the previous step and keep the change in the cost.

   (iii) Perform the swap for the pair with most improving cost.

   (iv) Iterate the pairing on all robots in the selected two stations until no improving pair exists.

   (v) Selected another pair of stations. Iterate the whole process for every pairs of stations.

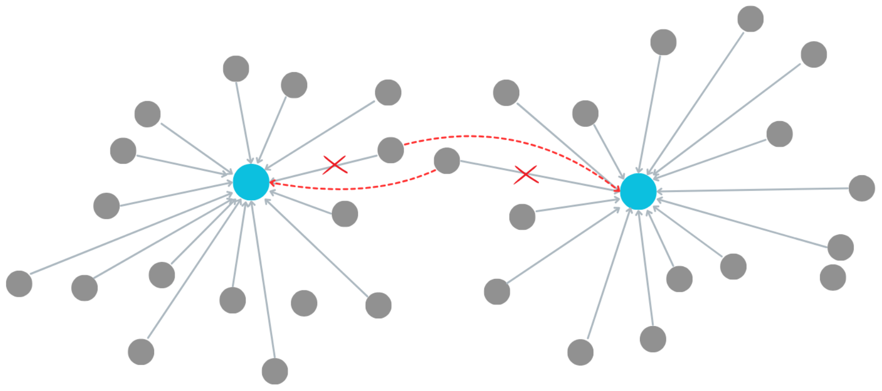

Figure 7: Local Search

1.3.2 Results and Analysis

The table below shows the performance of improvement heuristic methods using random start and fixed start construction. Each row represents the results after running that heuristic method.

Method	# Stations	Objective Value	Runtime (s)
Construction Heuristics (Random Start)	69	660,133.15	493.22
Improved Centroid (Random Start)	69	660,124.37	<0.01
Station Reduction (Random Start)	68	658,611.40	<0.01
Local Search (Random Start)	68	652,541.96	55.36
Construction Heuristics (Starting Robot 1)	70	665,141.04	4.15
Improved Centroid (Fixed Starting Robot 1)	70	665,133.63	<0.01
Station Reduction (Fixed Starting Robot 1)	67	655,628.03	<0.01
Local Search (Fixed Starting Robot 1)	67	648,065.76	60.47

Table 4: Results After Sequentially Running Each Method On Top of Previous Methods

From the table, the runtime for *Improved Centroid* and *Station Reduction* seems to be extremely low. This is due to that the process only runs in the station level, not the level of every robots.

Local Search, on the other hand, takes longer time comparing to the others. This is because it runs on nearly every single pair of robots in multiple iterations. In addition, in each iteration, it needs to compute the cost of both stations in the pair, twice for each – cost before and after switching the robots of each station. With this, in addition to the number of iterations, it takes significant amount of time to process.

One notice is that while looking at the objective value, using random start, despite having better objective value when construction, does not guarantee the better cost after running improvement heuristics.

This can be because random start makes the construction reaching the local minimum value, and after improving, it still gets stuck in that local minimum. In addition, the fixed start in this case starts from the first robot in the list, and the list of robots is already an ordering list, ordering robots geographically from left to right. With this, having the robots cumulatively construct a group from the corner or edge geographically might be a better choice than randomly pick a robot to start.

From the previous deduction, only the analysis from the Fixed Start Construction Heuristics will be made below.

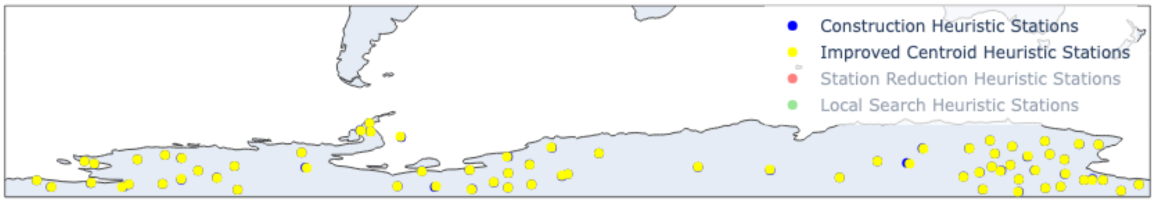
Construction Heuristic Stations
Improved Centroid Heuristic Stations
Station Reduction Heuristic Stations
Local Search Heuristic Stations

Figure 8: Locations Difference between before and after Improved Centroid

Looking at Figure 8 above, together with the results in Table 4, it seems that *Improved Centroid* does not improve the objective value as much as expected. It can be seen in Figure 8 that some of the location does not even move. The reason behind this is that moving the location towards the normal centroid does not guarantee an improvement in the cost. This is because while it minimize the sum distance between each robot and the centroid, it does not take the range of each robot into account. With the range of each robot, the feasible region (a region where the centroid would not cause a penalty to a robot) could be constructed, and within this feasible region, there might be other directions that could improve the minimum sum distance given the range of each robot.

Another reason could be that this weighted centroid used in the construction already took the range into account and is proved to already be "good enough", causing the improvement to be very minute.

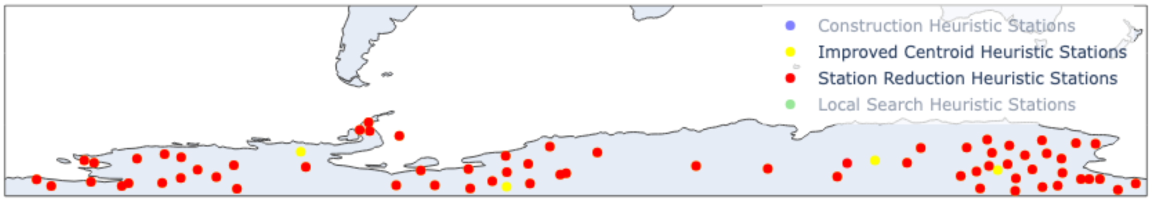
Construction Heuristic Stations
Improved Centroid Heuristic Stations
Station Reduction Heuristic Stations
Local Search Heuristic Stations

Figure 9: Locations Difference between before and after Station Reduction

In *Station Reduction*, stations with small amount of robots (less than 6) are aimed to be closed. With this main idea, from Table 4, it can be seen that the number of locations reduces by three. This means there were three stations removed from the process and the robots got reassigned to other stations. With this, it also improves the objective function by approximately £10,000.

From Figure 9, some locations after *Station Reduction* are exactly the same location as *Improved Centroid*. This is because the process mainly tries to close the station, while trying to find a new suitable station for each robot, which could result in a shift in some station locations for a robot not to be penalized. While this is rare, it is possible. From Figure 9, it can also be seen that there are four obvious locations that is shown as the stations from *Improved Centroid*. Combining with the results on Table 4, the three number of stations reduced aligns with three of the stations shown explicitly in yellow in Figure 9. One additional yellow point implies that there is a robot moved from the closed station to the new station without incurring any penalty, causing the centroid of one station to shift, exposing the station in yellow point from *Improved Centroid*.

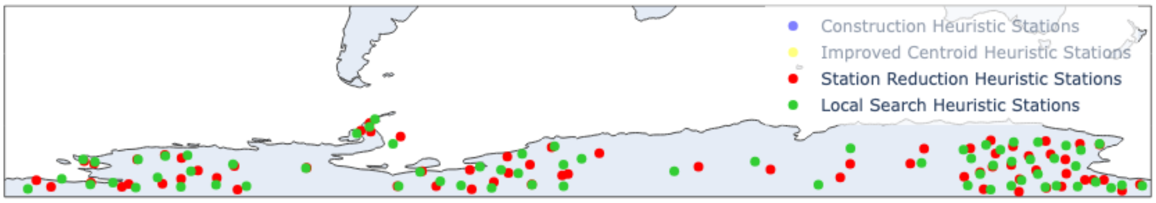
Construction Heuristic Stations
Improved Centroid Heuristic Stations
Station Reduction Heuristic Stations
Local Search Heuristic Stations

Figure 10: Locations Difference between before and Local Search

From Figure 10, it is clearly seen that there are many shifts in the station locations. This happens from the *Local Search* trying to find improving pairs between stations, which actually exist by observing from the figure. Swapping pairs of robots is what caused the locations to shift. From Table 4, this also improves the objective value by £7,000 in estimate. Trading off with the increased in runtime, this seems to be acceptable. The runtime could be reduced by setting how many stations each station can look forward into. In this case, each station checks with all other stations while in some cases, it can be set to check with only 5-10 nearby stations to improve the performance further, with a trade-off of a small acceptable increase in the objective function.

2 Question 2: Stochastic Version

2.1 Model Formulation

In this section, the deterministic model is extended into a two-stage stochastic version. Consequently, some input and output variables, constraints, and the objective function are modified to account for uncertainty across different scenarios.

2.1.1 Additional Sets and Notation

- Ξ : Set of 100 scenarios, $\Xi = \{0, 1, \ldots, 99\}$.

2.1.2 Adjusted Input Variables

- R_v^ξ: Current range of robot v at scenario ξ ; $v \in V$, $\xi \in \Xi$
- R^{min} [km]: Minimum travel range per robot ; $R^{min} = 10$ km
- λ: Constant for calculating probability ; $\lambda = 0.012$
- P_v^ξ; Probability that robot v requires charging at scenario ξ, determined by

$$P_v^\xi = e^{-\lambda^2 (R_v^\xi - R^{min})^2} \quad ; v \in V, \xi \in \Xi$$

- A_v^ξ: Binary Matrix represented with size $|V| \times |\Xi|$, determining whether robot v require charging at scenario ξ. Each element of this matrix is obtained by

$$A_{v,\xi} = \begin{cases} 1, & \text{if } U_{v,\xi} \leq P_v^\xi \\ 0, & \text{otherwise} \end{cases} \quad ; v \in V, \xi \in \Xi,$$

  where $U_{v,\xi} \sim \mathcal{U}(0, 1)$ is a uniform random variable.
- σ: The probability that scenario ξ occurs; $\xi \in \Xi$. In this case,

$$\sigma = \frac{1}{|\Xi|} \quad ; \forall \xi \in \Xi.$$

2.1.3 Adjusted Output Variables

With the introduction of scenarios, the following variables are scenario-dependent:

- $\beta_{v,s}^\xi$:Decision whether robot v is assigned to station s at scenario ξ ; $v \in V$, $s \in S$, $\xi \in \Xi$.
- $\alpha_{v,s}^\xi$:Decision whether robot v is brought manually by human to station s at scenario ξ; $v \in V$, $s \in S$, $\xi \in \Xi$
- $d_{v,s}^\xi \in R^+$: Distance robot v will travel to station s at scenario ξ $v \in V$, $s \in S$, $\xi \in \Xi$

2.1.4 Adjusted Constraints

Constraints

- If Robot v requires charging at scenario ξ, it can only be assigned to built stations, so

$$\beta^{\xi}_{v,s} \leq A_{v,\xi} \cdot \mu_s \quad \forall v \in V,\ s \in S,\ \xi \in \Xi$$

- Robot v must be assigned to only one station if it requires charging in scenario ξ, so

$$\sum_{s \in S} \beta^{\xi}_{v,s} = A_{v,\xi} \quad \forall v \in V,\ \xi \in \Xi.$$

- Number of chargers at each stations is determined by

$$\eta_s \geq \frac{1}{Q} \sum_{v \in V} A_{v,\xi} \cdot \beta^{\xi}_{v,s} \quad ;\forall s \in S,\ \forall \xi \in \Xi$$

- If a robot v is assigned to station s and requires charging at scenario ξ, it must either go by itself or be transported (by human manually)

$$\alpha^{\xi}_{v,s} \leq A_{v,\xi} \cdot \beta^{\xi}_{v,s} \quad ;\forall v \in V,\ s \in S,\ \xi \in \Xi$$

- Distance that robot v will travel to station s is determined by constraints below:

$$\begin{aligned}
&d^{\xi}_{v,s} \geq A_{v,\xi} \left[\sqrt{(X_v - x_s)^2 + (Y_v - y_s)^2} - D^{max}(\alpha^{\xi}_{v,s} + (1 - \beta^{\xi}_{v,s})) \right] && ;\forall v \in V,\ \forall s \in S,\ \forall \xi \in \Xi \\
&d^{\xi}_{v,s} \leq A_{v,\xi} \cdot D^{max} \cdot \beta^{\xi}_{v,s} && ;\forall v \in V,\ \forall s \in S,\ \forall \xi \in \Xi \\
&d^{\xi}_{v,s} \leq A_{v,\xi} \cdot R_v \cdot (1 - \alpha^{\xi}_{v,s}) && ;\forall v \in V,\ \forall s \in S,\ \forall \xi \in \Xi
\end{aligned}$$

  The constraints work the same way as in deterministic model. The only change is the inclusion of scenarios.

2.1.5 Objective Function

The objective function for the stochastic model is formulated as:

$$\min \sum_{s \in S} C^b \mu_s + \sum_{s \in S} C^m \eta_s + \sum_{\xi \in \Xi} \sigma \cdot A_{v,\xi} \left(\sum_{v \in V} \sum_{s \in S} \left[C^h \alpha^{\xi}_{v,s} + \ C^c \left[R^{max} \beta^{\xi}_{v,s} - (R_v \beta^{\xi}_{v,s} - d^{\xi}_{v,s}) \right] \right] \right)$$

The first three terms correspond to first-stage decisions, which are made before the uncertainty is revealed and are therefore scenario-independent. The last term represents the second-stage cost, which depends on scenario-specific outcomes. By taking the expectation over all scenarios (weighted by their probabilities σ), the model captures uncertainty through a recourse function.

2.2 Modified Heuristics Approach

The concept of fixing the station locations for all scenarios works the same way as the mathematical model in the previous section. The heuristics used in the stochastic model are also the same one as in deterministic. However, some aspects were modified to suit the incorporation of the scenarios.

1. **Fixed Start Construction Heuristics**
   From the result of deterministic heuristics, it was found that random start does not guarantee the better result after running improvement heuristics. Therefore, in this stochastic version, only fixed start will be used in the construction heuristic.

2. **Weighted Centroid Function**
   Previously in deterministic heuristic approach, the weighted centroid is calculated from the range of each robot (R_v). However, in stochastic approach, R_v are given in multiple scenarios, with the notation changed to R^{ξ}_v. This raised a question of which value of the robot range to be used for

each robot.

A solution was come up with to pre-calculate the expected range ($\mathbb{E}_v^R$) for each robot, using A_v^ξ and R_v^ξ. For each robot, the expected value for its range is calculated from the scenario that the robot needs to charge, excluding scenarios that the robot would not charge. The value can be calculated as follows.

$$\mathbb{E}_v^R = \frac{1}{\sum_{\xi \in \Xi} A_v^\xi} \cdot \sum_{\xi \in \Xi} A_v^\xi \cdot R_v^\xi$$

Once the weighted centroid is calculated, it uses this expected values instead of originally using the actual deterministic values.

3. **Cost Function**
The cost function is modified in the stochastic version. The changes focused on taking scenarios into account and utilized the expected values of the cost over scenarios instead of using just one deterministic value. The function is changed to the following function.

$$C_s^{total} = C_b + \lceil \frac{|v \in (V|s)|}{Q} \rceil \cdot C_m + \frac{1}{|\Xi|} \sum_{\xi \in \Xi} \sum_{v \in (V|s)} C_{v,\xi}^{charge}$$

The condition for C_v^{charge} is also slightly changed as follows.

$$C_{v,\xi}^{charge} = \begin{cases} C_c \cdot (R^{max} - R_v^\xi + d_{v,s}), & \text{if } R_v^\xi \geq d_{v,s}, \\ C_c \cdot (R^{max} - R_v^\xi) + C_h, & \text{otherwise} \end{cases}$$

2.3 Results

Below shows the results comparing between MINLP and the construction heuristics method. The table also compares the results from the deterministic version.

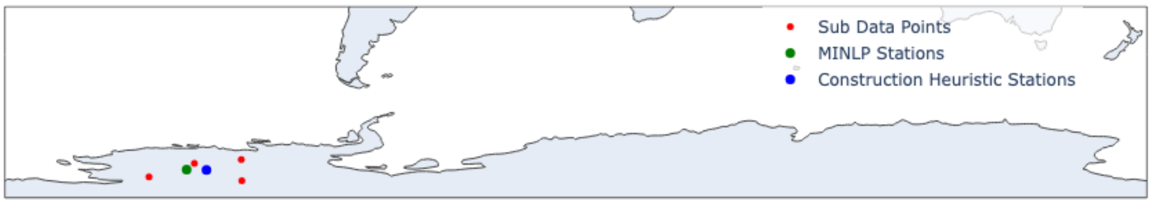

Figure 11: Stations Locations Comparison between MINLP and Construction Heuristics with 4 sampled robots

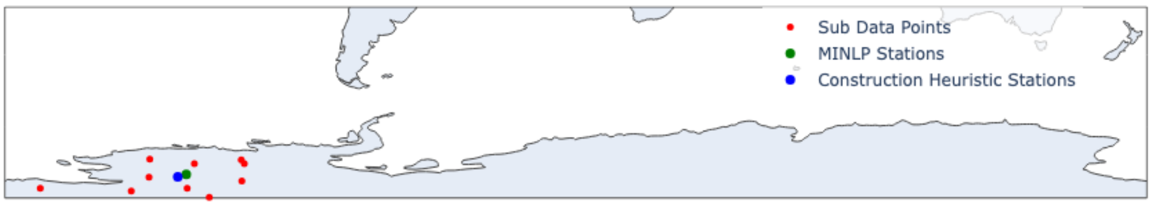

Figure 12: Stations Locations Comparison between MINLP and Construction Heuristics with 10 sampled robots

Method	Version	# Samples	# Stations	Objective Value	%Gap	Runtime (s)	Lower Bound
MINLP	Deterministic	4	1	6,153.38	0.31	0.18	6,134.26
Construction Heuristics	Deterministic	4	1	6,153.57	0.31	<0.01	-
MINLP	Stochastic	4	1	6,141.50	0.52	10.42	6,109.45
Construction Heuristics	Stochastic	4	1	6159.95	0.82	<0.01	-
MINLP	Deterministic	10	1	7,986.57	0.94	1.07	7,911.27
Construction Heuristics	Deterministic	10	2	13,459.29	41.22	<0.01	-
MINLP	Stochastic	10	1	8,013.64	6.39	1,415.65	7,501.42
Construction Heuristics	Stochastic	10	1	8,063.87	6.97	<0.01	-

Table 5: Performance of Each Method with 4 and 10 Sampled Robots

From Table 5, it can be seen that the runtime for MINLP in stochastic version increases drastically compared to the deterministic version. This is due to the rise in complexity of scenarios. On the other hand, the runtime for the heuristic method seems to still be significantly low, as same as in the deterministic version. The results from heuristics are also close to that of the MINLP. This can guarantee the performance of the heuristic method.

Comparing to the deterministic version, for 4 samples case, the value seems to be close to the stochastic version. As for the 10 samples case, the value for the MINLP looks fairly close to the deterministic version. In contrast, with the greedy algorithm implemented, the objective value in the deterministic version for the heuristics becomes higher due to two stations being built. This is expected to be reduced by further improvement heuristics. To prove this, a quick run was done to run all construction and improvement heuristics using the same 10 samples. The results give the objective value of 7,900.56 with one station opened, which aligns with the MINLP.

One notice is that the %Gap for the 10 samples case is relatively higher comparing to the 4 samples case in stochastic version. Looking at the lower bound value, this might be the reason of this higher %Gap. Intuitively, the cost of 10 robots should be from opening 1 station, having 5 chargers, and adding the charging cost. Looking at the cost of opening station and the chargers, it is already $1 \times 5,000 + 5 \times 500 = 7,500$. The charging cost, arbitrarily assuming each robot needs to charge for 50 km, is $0.42 \times 50 \times 10 = 210$. Summing up all these cost is already $7,500 + 210 = 7,710$, which is already higher than the lower bound. Therefore, it could be said that the lower bound does not converge as much as expected, which could be due to the complexity and possibly additional runtime required. With the arbitrary assumption, the %Gap could be around 3.78% for MINLP and 4.39% for the heuristics. Moreover, using the lower bound from the deterministic version, the %Gap is 1.28% for the MINLP and 1.90% for the heuristics.

The table below shows the results after running heuristics with all robots.

Method	# Stations	Objective Value	Runtime (s)
Construction Heuristics	69	643,618.72	4.81
Improved Centroid	69	643,618.47	0.10
Station Reduction	67	636,988.35	<0.01
Local Search	67	631,218.59	1,344.07

Table 6: Results After Sequentially Running Each Method On Top of Previous Methods in Stochastic Version

The runtime of each heuristics seems to be in the same proportion as in deterministic, which is expected since the algorithm remains the same, only the data size is increased in the dimension of scenarios. However, *Local Search* runtime seems to drastically rise compared to the others. This is due to the fact that the cost function is changed. The cost function in stochastic version has 100 more iterations than the deterministic version following to the number of scenarios. This cause 400 more iterations (same as what explained in the deterministic results that the cost function gets called 4 times, twice for each station in the pair) in total in one iteration of the *Local Search.*

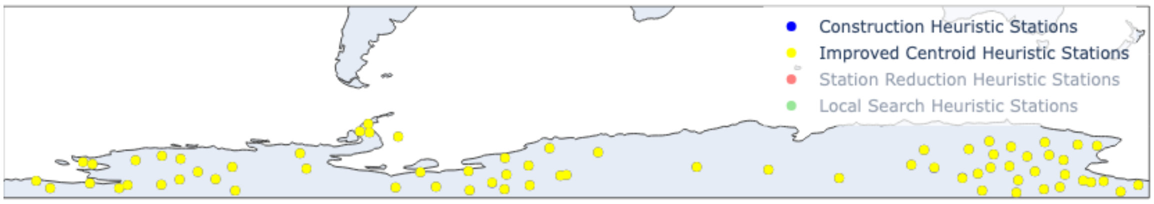
Construction Heuristic Stations
Improved Centroid Heuristic Stations
Station Reduction Heuristic Stations
Local Search Heuristic Stations

Figure 13: Locations Difference between before and after Improved Centroid (Stochastic)

The *Improved Centroid* does not seem to improve much in stochastic version. This could be due to the complexity of the cost function that utilizes the marge amount of scenarios, causing the process to hardly find the improvement from the centroid alone.

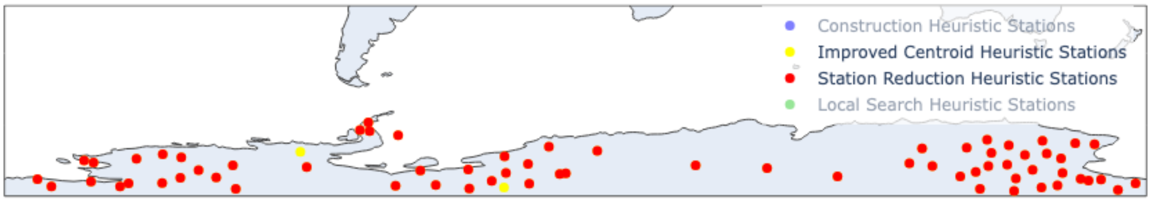
Construction Heuristic Stations
Improved Centroid Heuristic Stations
Station Reduction Heuristic Stations
Local Search Heuristic Stations

Figure 14: Locations Difference between before and after Improved Centroid (Stochastic)

The *Station Reduction* seems to work the same way as deterministic version. The only difference is that in this stochastic version, two stations got removed and there were no robot peacefully joined with the other stations since there are two yellow points shown in Figure 14 that corresponds to the deduction in the number of stations on Table 6.

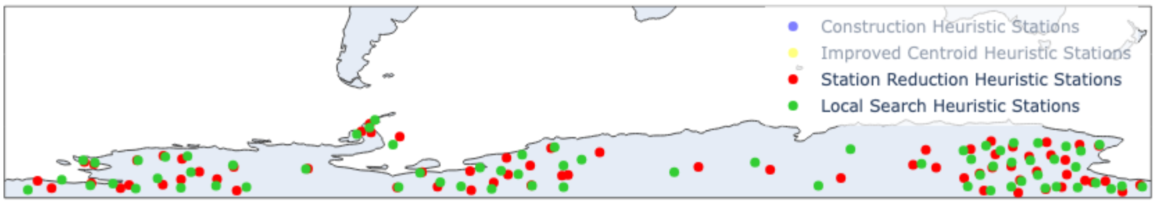
Construction Heuristic Stations
Improved Centroid Heuristic Stations
Station Reduction Heuristic Stations
Local Search Heuristic Stations

Figure 15: Locations Difference between before and after Local Search (Stochastic)

Lastly, *Local Search* still seems to be the only one changing the station locations the most, while also improving the cost. From Table 6, the cost decreases further by around £6,000. The station locations also changed a lot in the same sense as in deterministic version.

3 Conclusion

In conclusion, the construction heuristic seems to be working fine. If the random start is desired to be used, a careful random logic needs to be implemented in order to pick the starting robots. Otherwise, it could result in the too-optimized construction heuristic, causing it to be stuck in the local minimum.

As for improvement heuristics, *Stations Reduction* and *Local Search* come in the top at reducing the objective value. These two shows the best performance in terms of both objective value and runtime, making it a strong suggestion to implement these algorithm.

However, *Station Reduction* might be too specific with this case since it is derived from the mathematical calculation from the given parameters. If those parameters are changed, this algorithm might need to be adjusted, or in worst case, it might not be able to apply this method at all. Therefore, *Local Search* seems to be more general in terms of the logic, with some tradeoff in the longer runtime, which is still in an acceptable range.

Improved Centroid, on the other hand, does not improve much, or not improve at all, in terms of objective value. However, it runs very fast, thus, there is no harm in adding it as an intermediate layer between other improvement heuristics.