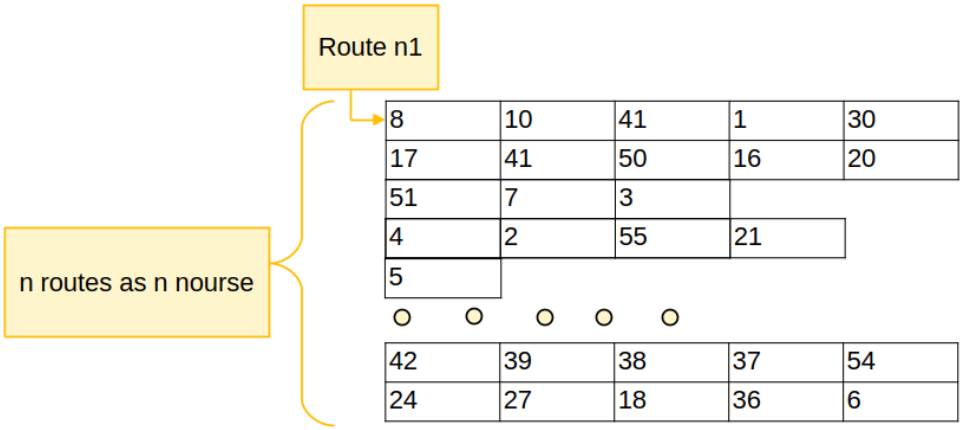
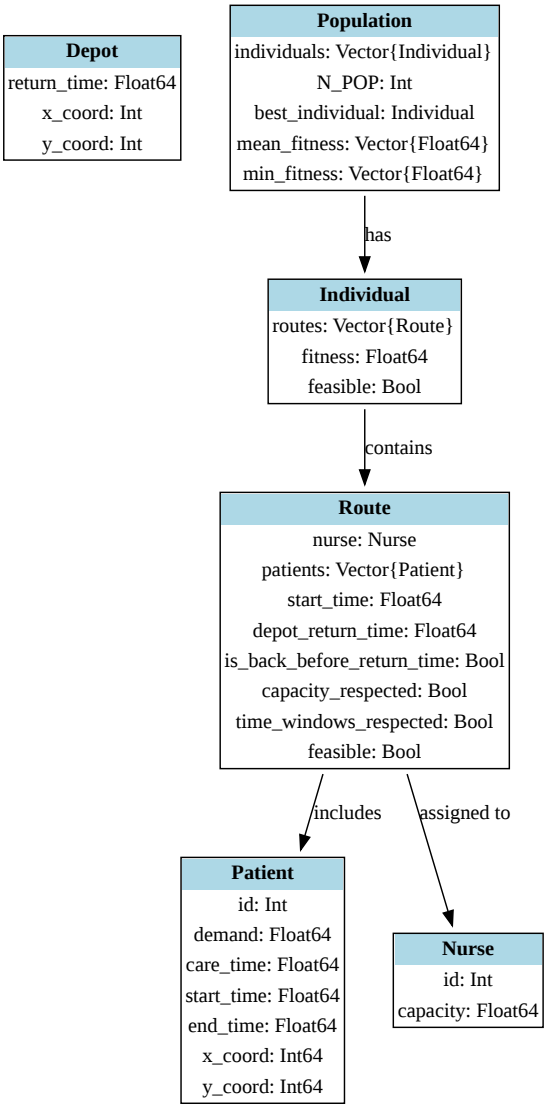
Bio-Inspired Artificial Intelligence Home Care Problem

An individual is represented as a list of lists, where each list corresponds to a nurse's route. Each route contains a sequence of patients, ordered according to the indices in the arrays.



As a result, a population is structured as a list (a set of individuals), where each individual is a list of lists (a set of routes), and each route is, in turn, a list of patients.

To ensure a clear and convenient representation in programming, we have chosen to use a series of structured data types (`struct`): **Patient**, **Nurse**, **Route**, **Individual**, and **Population**. This approach keeps the code readable and allows important attributes to be preserved throughout the iterations of the genetic algorithm.



Fitness function and Penalty

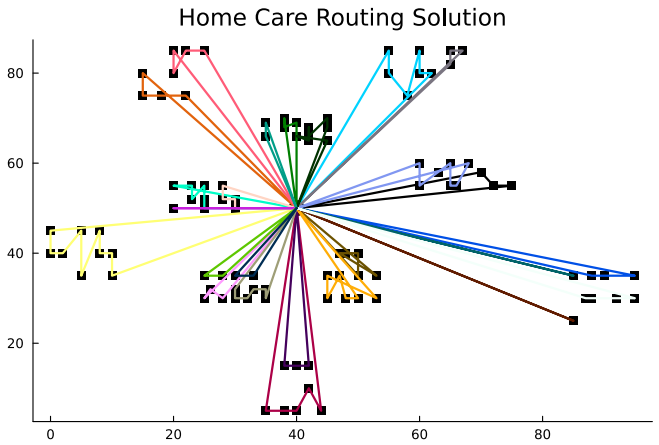
Genetic Algorithm Process

The algorithm follows the standard genetic algorithm process:

1. **Initialization**: A population is created.
2. **Selection**: A subset of the population is chosen for crossover.
3. **Crossover**: Offspring are generated and added to the population.
4. **Mutation**: Individuals undergo random modifications.
5. **Survivor selection**: The next generation is selected from the current population.

K-NN Initialization

In the **K-NN Initialization** phase, we use a clustering-based approach to generate the initial population. This method ensures that routes are structured based on proximity and other relevant factors before genetic operations are applied. To promote exploration of the solution space, we initialize the population with 10 different cluster counts (e.g., one with 9 clusters, another with 10, 11, and so on).



Exmple of initialization with k-nn (25 clusters)

Challenges with Crossover

A major challenge is performing **crossover on a list-of-lists structure**. Traditional crossover operators such as **Order One Crossover (OX1)** work well for **linear sequences** (e.g., Traveling Salesman Problem) but are not directly applicable to **nested route structures**.

Issues with Flattening Routes for Crossover

A possible solution is **flattening** all routes into a single sequence, applying crossover, and then reconstructing the original structure. However, this introduces several issues:

- **Loss of route integrity**, leading to arbitrary patient assignments.
- **Violation of constraints**, such as nurse workload limits.
- **Suboptimal solutions**, as crossover disrupts balanced assignments.

Crossover Can Be Detrimental

Given the structured nature of the problem, crossover risks degrading solutions instead of improving them:

- **Increased nurse count**, reducing efficiency.

- **Invalid assignments**, requiring extensive repairs.
- **Disruption of optimized routes**, leading to longer travel times.

Due to these challenges, we prioritize a **mutation-based approach** over crossover.

Mutation Strategies

To optimize solutions while maintaining valid assignments, we implement three mutation types, divided into two categories:

- **Intra-route mutations**: Modify a single route.
  - **Mutation Swap**: Swap the positions of two random patients in a route.
  - **Mutation Inversion**: Select a subset of a route and shuffle it.
- **Inter-route mutations**: Modify multiple routes.
  - **Mutation Shift**: Move a patient from one route to another.

Current Algorithm Implementation

The algorithm starts with an initial population of 500, generated using K-NN initialization, and runs for a maximum of 50,000 iterations. At each iteration, the population is **doubled** by creating copies of the existing individuals, after which **mutations** are applied **only** to the copied individuals. **Survivor selection** is then performed: the **top 50 individuals** with the best fitness are retained, while the **remaining 450 individuals** are selected through **tournament selection**.

Initial Observations and Adjustments

A major issue emerged: the algorithm tends to minimize the number of nurses, leading to a few nurses handling too many patients and as a result, the constraints are often violated, making the solution impractical.

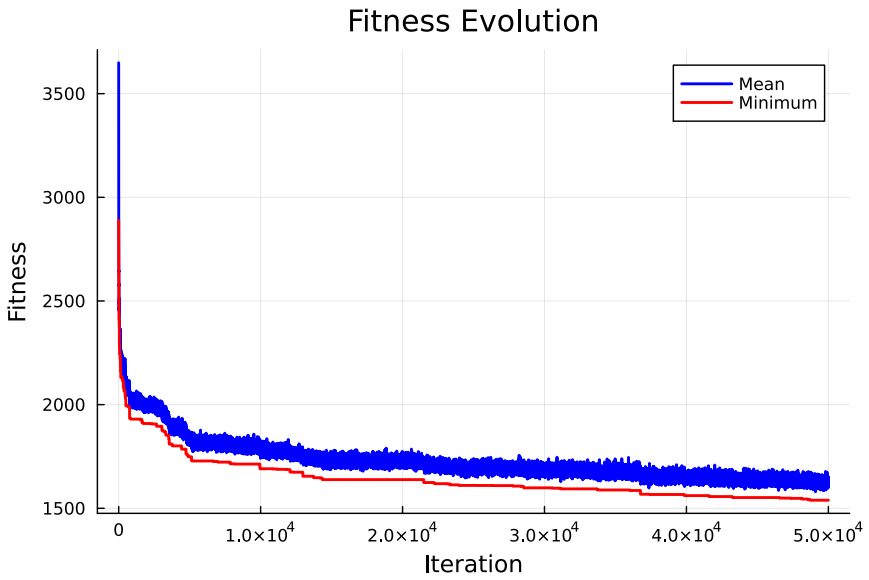
To mitigate this issue, we modified the **Mutation Shift** function, responsible for redistributing patients between routes. Specifically:

- A check was added to prevent assigning a patient to a route where the nurse’s capacity is already zero, as this would lead to an inherently infeasible solution.
- This avoids exploring invalid search spaces, improving computational efficiency.

- However, the mutation does not enforce checks on whether the route finishes within the expected time. This is because excessive workload violations can stem not only from patient count but also from the sequence of visits.

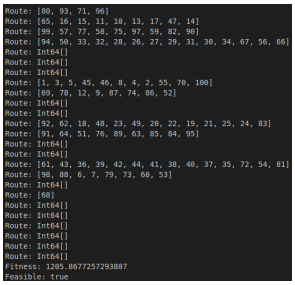
Experimental Results and Improvements

The results show that acceptable solutions are achieved.



Exploration Issues and Solution

One remaining challenge is limited exploration. Once the algorithm reaches a local minimum, it struggles to escape. In particular, it can construct individuals with fewer routes but not with more routes, leading to stagnation in these local optima. This issue arises because **Mutation Shift** is the only operator capable of adding elements to an empty route, doing so one patient at a time. Routes created in this manner have a low probability of surviving the next iteration since the start and return times to the depot accumulate, making them less competitive.



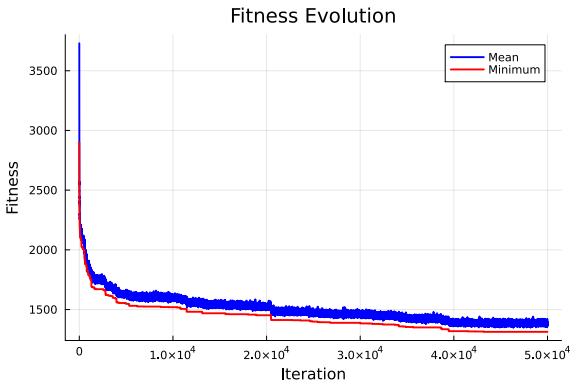
14/25 routes are empty

To enhance exploration, we introduced a fourth mutation: **Mutation Split**. This mutation takes an existing route and splits it into two separate routes. This allows the algorithm to explore solutions with more routes, preventing stagnation.

Another refinement involved applying **Mutation Inversion** only when the route's `time_window_respected` flag is set to `false`. This is because inversion drastically alters the route structure, making it suitable only when schedule violations need to be addressed.

Results

The results indicate that the algorithm successfully produces acceptable solutions, with **travel times under 30% from the benchmark values at each test**.



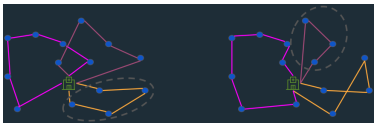
This approach is not effective because it lacks true exploration; instead, it simply applies random modifications to a limited set of high-performing individuals, causing the algorithm to get stuck in a local minimum.

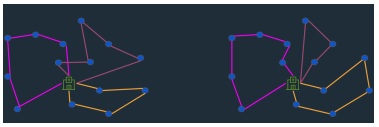
To overcome this limitation, we needed to implement crossover, an idea we derived from Visma slides.

Crossover

The crossover process follows these steps:

1. Select two random individuals, **A** and **B**.
2. Pick a random route from each, denoted as $\mathbf{R_a}$ (from A) and $\mathbf{R_\beta}$ (from B).
3. Remove the patients in $\mathbf{R_a}$ from B and the patients in $\mathbf{R_\beta}$ from A.
4. For each patient left without a route, find the best possible insertion point.





Finding the Best Insertion

To determine the optimal insertion point, we evaluate the fitness impact of placing the patient in every possible position across all routes, selecting the placement that yields the best result. While this approach is computationally expensive, it proves to be highly effective.

Adaptive mutation

We introduced an adaptive mutation mechanism to enhance exploration when needed. If the average fitness over the last 12 generations does not show significant improvement, the algorithm dynamically adjusts the mutation hyperparameters.

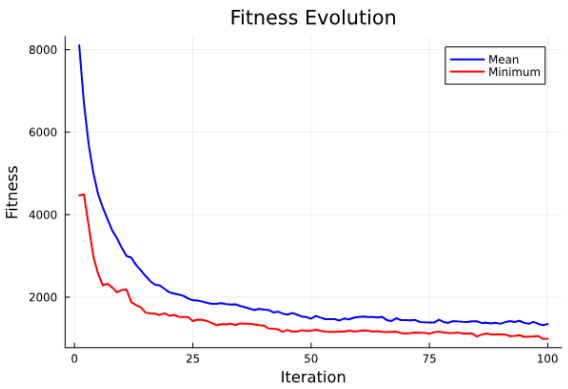
To detect stagnation, we perform a one-sample t-test on the fitness values of the last 12 generations:

- If the p-value > 0.05, indicating no significant improvement, the mutation parameters are doubled to increase exploration.
- Otherwise, the mutation parameters are reset to their original values.

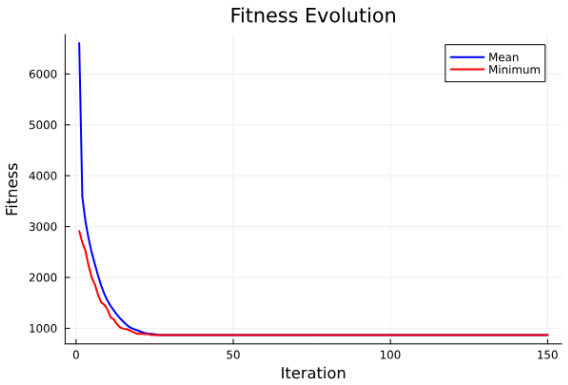
This adaptive strategy helps the algorithm escape local optima while maintaining efficiency.

Results

The results are promising but never consistently better than 5% of the benchmark. We attempted to fine-tune the hyperparameters, but even after optimization, we couldn't achieve a consistent improvement beyond 15%.



In our experimentation, we tested removing all types of mutation, and surprisingly, we achieved very good results. The fitness plot showed a rapid decrease, and in less than 50 generations, the algorithm consistently produced solutions within 5-10% of the benchmark.

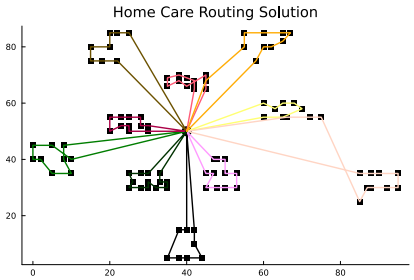


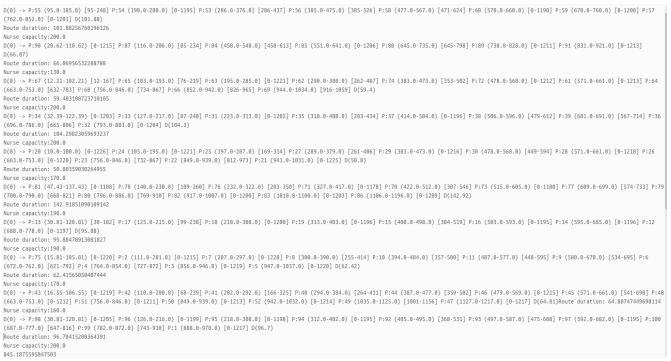
Results using GA without mutation and just crossover.

Results on test data

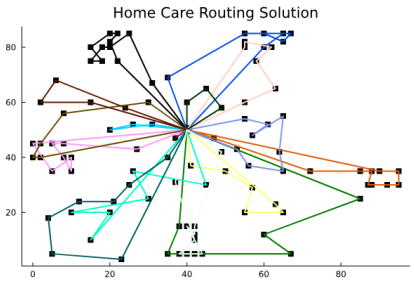
The obtained results for the test data are as follows:

- **test_0** = 845 (**<5%**)

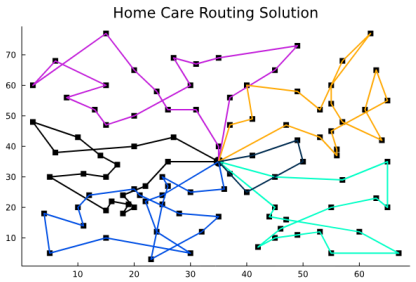




- **test_1** = 1560 (**<5%**)



- **test_2** = 937 (**<5%**)




```
D(0) -> P:52 (11.31-21.31) [11-758] P:18 (229.0-239.0) [229-592] P:45 (253.0-263.0) [29-319] P:46 (338.0-348.0) [338-815] P:8 (393.0-403.0) [393-502] P:83 (410.0-420.0) [21-852] P:60
(449.0-459.0) [449-981] P:5 (463.0-473.0) [28-916] P:84 (478.0-488.0) [425-532] P:17 (673.0-683.0) [673-940] P:91 (698.0-708.0) [222-974] P:85 (711.0-721.0) [110-737] P:98
(724.0-734.0) [21-758] P:100 (821.0-831.0) [821-975] P:37 (834.0-844.0) [339-978] P:93 (848.0-858.0) [691-979] P:96 (863.0-873.0) [344-963] P:89 (882.0-892.0) [489-991] D(134.01)
Route duration: 134.00541204143656
Nurse capacity:221.0
D(0) -> P:95 (14.87-24.87) [14-803] P:92 (28.0-38.0) [18-169] P:42 (49.0-59.0) [25-302] P:15 (176.0-186.0) [176-309] P:14 (282.0-212.0) [32-704] P:38 (298.0-308.0) [298-419] P:86
(321.0-331.0) [230-633] P:44 (339.0-349.0) [230-353] P:16 (355.0-365.0) [102-553] P:61 (370.0-380.0) [126-551] P:99 (388.0-398.0) [328-447] P:59 (400.0-410.0) [17-902] P:87
(419.0-429.0) [397-480] P:2 (436.0-446.0) [18-802] P:57 (452.0-462.0) [246-715] P:43 (475.0-485.0) [458-877] P:97 (503.0-513.0) [462-833] P:94 (519.0-529.0) [190-759] P:6
(532.0-542.0) [355-504] P:13 (694.0-704.0) [694-833] P:58 (854.0-864.0) [854-990] D(191.43)
Route duration: 191.43328414074264
Nurse capacity:323.0
D(0) -> P:21 (18.03-28.03) [18-757] P:73 (143.0-153.0) [143-574] P:72 (156.0-166.0) [22-925] P:39 (180.0-190.0) [33-354] P:67 (291.0-301.0) [291-414] P:23 (313.0-323.0) [286-335] P:56
(525.0-535.0) [525-762] P:75 (539.0-549.0) [27-721] P:22 (553.0-563.0) [349-564] P:41 (567.0-577.0) [536-581] P:74 (681.0-691.0) [681-812] P:4 (702.0-712.0) [679-818] P:55
(721.0-731.0) [620-749] P:25 (826.0-836.0) [826-966] P:24 (851.0-861.0) [435-970] P:54 (871.0-881.0) [475-910] P:26 (893.0-903.0) [328-937] D(158.61)
Route duration: 158.60658525963103
Nurse capacity:258.0
D(0) -> P:27 (44.0-54.0) [44-325] P:31 (67.0-77.0) [17-754] P:88 (219.0-229.0) [219-458] P:62 (235.0-245.0) [168-307] P:11 (253.0-263.0) [206-335] P:64 (276.0-286.0) [154-403] P:36
(308.0-318.0) [41-338] P:49 (441.0-451.0) [441-618] P:19 (463.0-473.0) [95-562] P:47 (481.0-491.0) [34-612] P:48 (498.0-508.0) [280-972] P:82 (513.0-523.0) [23-771] P:7 (529.0-539.0)
[72-679] P:10 (553.0-563.0) [517-702] P:63 (572.0-582.0) [34-598] P:90 (587.0-597.0) [225-664] P:32 (601.0-611.0) [482-965] P:66 (626.0-636.0) [532-763] P:20 (645.0-655.0) [418-839]
P:70 (668.0-678.0) [668-978] D(207.32)
Route duration: 207.31857333616094
Nurse capacity:319.0
D(0) -> P:69 (165.0-175.0) [165-300] P:1 (190.0-200.0) [190-984] P:30 (211.0-221.0) [199-424] P:51 (230.0-240.0) [176-635] P:33 (247.0-257.0) [24-309] P:71 (274.0-284.0) [98-547] P:65
(294.0-304.0) [49-318] P:9 (394.0-404.0) [394-513] P:81 (410.0-420.0) [268-689] P:79 (426.0-436.0) [377-486] P:29 (446.0-456.0) [29-523] P:78 (466.0-476.0) [324-569] P:35
(626.0-636.0) [626-883] P:34 (646.0-656.0) [265-886] P:3 (670.0-680.0) [289-832] P:68 (686.0-696.0) [577-822] P:88 (719.0-729.0) [719-978] P:77 (821.0-831.0) [821-958] P:50
(838.0-848.0) [272-927] D(197.74)
Route duration: 197.73665711293755
Nurse capacity:274.0
D(0) -> P:28 (6.32-16.32) [6-536] P:76 (281.0-291.0) [281-392] P:12 (298.0-308.0) [15-735] P:40 (322.0-332.0) [300-507] P:53 (403.0-413.0) [403-510] D(48.15)
Route duration: 48.15207977548874
Nurse capacity:71.0
```