Multiple-Objective Routing Path Optimisation

- **Group No**: 23
- **Authors**:
  - Amen Esenabhalu (s5633417)
  - Adetoro Odupitan (s5647385)
  - Chiderah Chukwudinma Onwumelu (s5644104)
  - Inaboya Uduzen (s5650545)
  - Chinduji Emereole (s5440921)
- **Unit Leader**:
  - Jiankang Zhang

Table of contents

Abstract

- *This research focusses on optimising routing paths in a wireless sensor network to maximise end-to-end transmission rates while minimising latency. To address this multi-objective problem, multiple optimisation strategies were employed and assessed, including a Discrete Genetic Algorithm (DGA), an Ant Colony Optimisation (ACO) algorithm, and a hybrid approach integrating Genetic Algorithm (GA) and Simulated Annealing (SA). The effectiveness of these approaches was evaluated using key metrics: convergence rates, fitness scores, optimal solution outputs, and computational effectiveness.*
- *The research identifies the Weighted Sum Method (WSM) as the cost function employed to calculate the fitness score of each solution (routing path) within the algorithms. The results underscore the benefits and drawbacks of each method, with the Ant Colony Optimisation (ACO) algorithm proving to be the most appropriate for this context due to its exceptional ability to balance transmission efficiency and latency, along with its fast convergence that reduces computational resources usage. Future research proposes approaches to improve performance and adaptability, including comparing WSM and Pareto front methodologies, exploring alternative approaches, and optimising hyperparameters.*

Import libraries and load data

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
import pandas as pd
import sys

from dataclasses import dataclass
import random
from typing import TYPE_CHECKING
if TYPE_CHECKING:
    from numpy.typing import NDArray
import numpy as np
```

In [8]:
```
# Load data
data = pd.read_csv('sub_data_file.csv', names=["x", "y"])
```

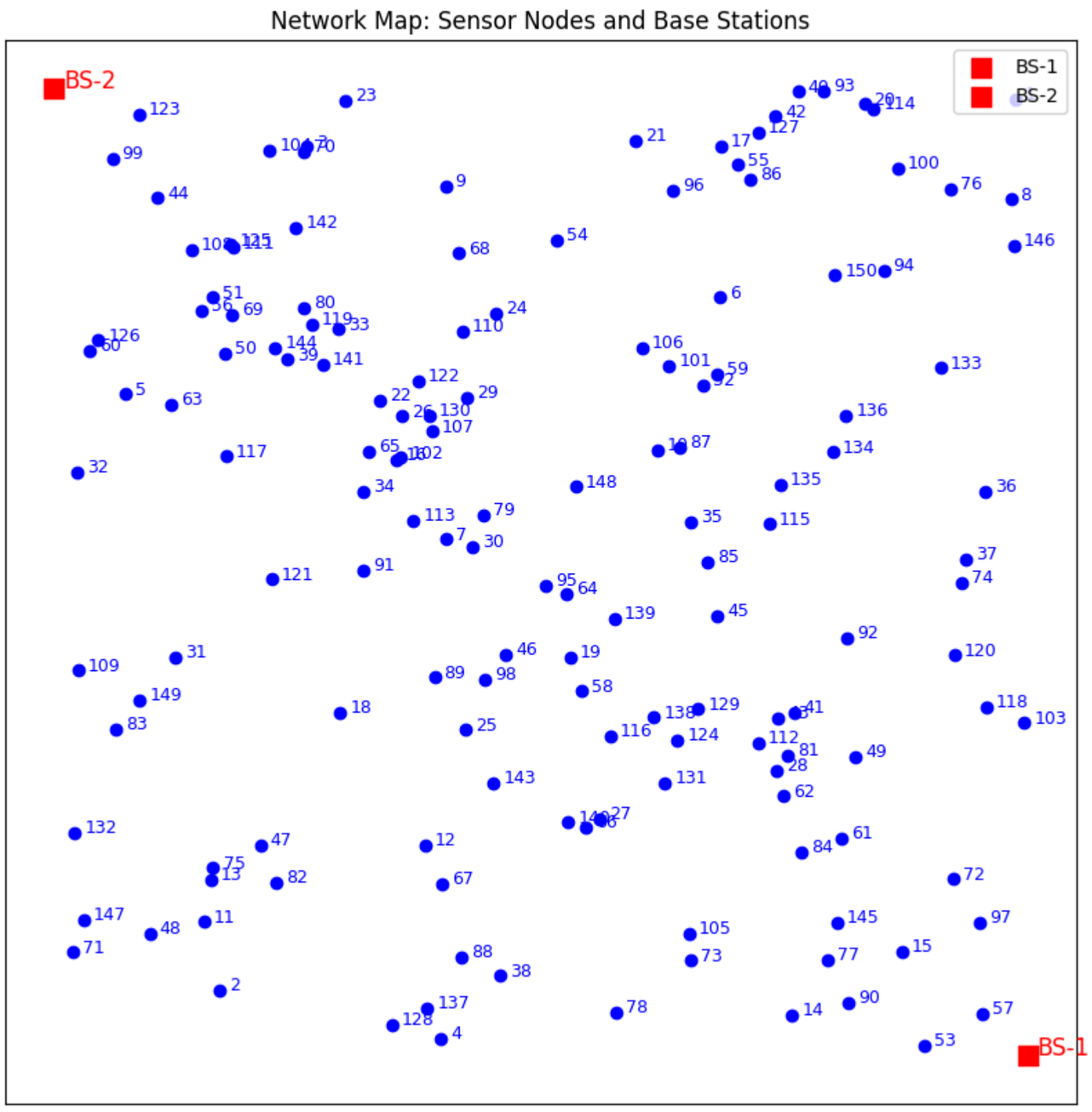
In [3]:
```
# Initializing nodes
NODES = data[['x', 'y']].to_numpy()
nodes_label = [f"node {index}" for index, _ in enumerate(NODES, start=1)]
```

In [ ]:
```
destination_coordinate = [(5000 , -5000), (-5000 ,5000)]
base_stations = {
    'BS-1': (5000, -5000),
    'BS-2': (-5000, 5000)
}
```

Problem statement

- In this problem, we aim to find the optimal routing path having the ***maximum end-to-end data transmission rate*** and ***minimum end-to-end latency*** for each node that can access any of two base stations. A routing path is a sequence of nodes that transmit data to a base station.

In [5]:
```
plt.figure(figsize=(10, 10))

# Plot sensor nodes
for index, node in enumerate(NODES):
    plt.scatter(node[0], node[1], color='blue', marker='o')  # Node positions (x, y)
    plt.text(node[0] + 100, node[1], f"{index + 1}", color="blue", fontsize=9)  # Node labels

# Plot base stations
for label, (x, y) in base_stations.items():
    plt.scatter(x, y, color='red', marker='s', s=100, label=label)  # Base station positions
    plt.text(x + 100, y, label, color="red", fontsize=12)


plt.xticks([])
plt.yticks([])

plt.title("Network Map: Sensor Nodes and Base Stations")
plt.legend(loc="upper right")
plt.grid(True)
plt.show()
```



- This is a multiple-objective optimization problem that seeks to balance two objectives.

1. **Maximising the transmission rate**: The minimum transmission rate of each link in a routing path

- $$T(r) = \min_{1<n<N} T(r_n -> r_{n+1})$$
- `max T(r)` for all possible routing path `r`

2. **Minimizing transmission latency**: The sum of all delays imposed by each link

- $$T(r) = \sum_{n=1}^{N} L(r_n -> r_{n+1})$$
- `min L(r)` for all possible rounting path `r`

Calculate distance and transmission rate

In [6]:
```
# Calculate distances between two nodes for every nodes
def calculate_distances(nodes, destination_coordinate):
    distances = cdist(nodes, nodes, metric='euclidean') # Calculate distance using
    base_distances = np.array([cdist(nodes, [bs], metric='euclidean').flatten() for bs in destination_coordinate]) # Distance between nodes and base stations
    return distances, base_distances

# Calculate distances between two nodes
def calculate_distance(node1, node2):
    if isinstance(node1, int):  # node1 is a regular node (index in the array)
        coord1 = NODES[node1 - 1]  # Accessing coordinates from the array (0-indexed)
    elif node1 in base_stations:  # if node1 is a base station
        coord1 = base_stations[node1]
    else:
        raise ValueError(f"Node number {node1} is not found.")

    if isinstance(node2, int):  # node2 is a node
        coord2 = NODES[node2 - 1]
    elif node2 in base_stations:  # if node2 is a base station
        coord2 = base_stations[node2]
    else:
        raise ValueError(f"Node number {node2} is not found.")

    return np.sqrt((coord1[0] - coord2[0]) ** 2 + (coord1[1] - coord2[1]) ** 2) # Calculate distance

# Calculate the latency of a path
def calculate_path_latency(path, delay=30):
    return (len(path) - 1) * delay # 30 ms transmission delay

# Get the transmission rate depending on the distance between two nodes
def transmission_rate(distance):
    # Return the transmission rate based on distance
    if distance >= 3000:
        return 0
    elif 2500 <= distance < 3000:
        return 1
    elif 2000 <= distance < 2500:
        return 2
    elif 1500 <= distance < 2000:
        return 3
    elif 1000 <= distance < 1500:
        return 4
    elif 500 <= distance < 1000:
        return 5
    else:
        return 7 # For distances less than 500m

# Calculate the transmission rate of a path
def calculate_path_transmission_rate(path_rates):
    return min(path_rates)
```

In [ ]:
```
# Calculate distance matrix for all pairs of nodes
distances, base_distances = calculate_distances(NODES, destination_coordinate)

# transmission rate matrix where rate value are stored for each link between nodes
transmission_matrix = np.vectorize(transmission_rate)(distances)
```

Objective function description

- The Weighted Sum Method (WSM) was employed to address this optimisation problem. WSM is a proficient and commonly employed method for addressing multi-objective optimisation challenges. Numerous multi-objective optimisation experiments have demonstrated the efficacy and feasibility of this method. Deb (2011) highlighted the simplicity of its application to solve optimisation tasks, remarking on its efficacy in situations where objectives can be articulated in comparable units.

Objective function

- Optimize the routing paths from sensor nodes to base stations to minimize transmission latency and maximize transmission rate.
  - **Multiple objective function**: Combine the two objectives into a single optimisation goal using a weighted sum:
    - $$\max F(r) = w1 * \frac{T(r)}{\max T} - w2 * \frac{L(r)}{\max L}$$
    - **Where**:
      - `max F(r)` maximum fitness score for all possible routing path `r`
      - **Normalization applied** to ensure both metrics are on the same scale:
        - $$\frac{T(r)}{\max T}$$
          - `T(r)` -> transmission rate for current path `r`
          - `max T` -> Maximum possible transmission rate in a path
        - $$\frac{L(r)}{\max L}$$
          - `L(r)` -> Latency of the current path `r`
          - `max L` -> Maximum possible latency in a path
    - `w1` and `w2` are weights that control the trade-off between latency and transmission rate. These can be tuned based on the relative importance of the two metrics.
      - $$w1, w2 \in [0, 1]$$
      - with
      $$w1 + w2 = 1$$

- By linearly combining multiple objectives such as minimizing end-to-end latency and maximizing end-to-end data transmission rate WSM provides a single scalar cost function that simplifies the decision-making process.
- This method guarantees computational efficiency by utilising proven optimisation methods tailored for single-objective situations. WSM offers considerable flexibility, allowing decision-makers to allocate weights to objectives according to their relative significance, hence facilitating tailored solutions that address specific system needs.

Variables & parameters

- `N` : Set of sensor nodes,

$$N = \{n_1, n_2, \ldots, n_k\}$$

- `BS` : Set of base stations,

$$BS = \{bs_1, bs_2\}$$

- `P_i` : Routing path for node `n_i` , where

$$P_i = (n_i, \ldots, bs_j), (bs_j \in BS)$$

- Euclidean distance between node `n_i` and `n_j` :

$$d(n_i, n_j)$$

.

- Transmission rate between node `n_i` and `n_j` :

$$T(n_i, n_j)$$

:, determined by distance between nodes:

$$d(n_i, n_j)$$

Constraints

1. **Path Constraints**: Each sensor node must have a valid path to exactly one of the base stations.

$$\sum_{j \in N \cup BS} x_{i,j} = 1 \quad \forall n_i \in N$$

This ensures that each node is connected to either another node or a base station.

2. **Transmission Rate Constraint**: The transmission rate between two nodes `n_i` and `n_j` is determined by the distance between them.

$$T(n_i, n_j) = \begin{cases} 7\,\text{Mbps} & \text{if } d(n_i, n_j) < 500\,\text{m} \\ 5\,\text{Mbps} & \text{if } 500\,\text{m} \leq d(n_i, n_j) < 1000\,\text{m} \\ 4\,\text{Mbps} & \text{if } 1000\,\text{m} \leq d(n_i, n_j) < 1500\,\text{m} \\ 3\,\text{Mbps} & \text{if } 1500\,\text{m} \leq d(n_i, n_j) < 2000\,\text{m} \\ 2\,\text{Mbps} & \text{if } 2000\,\text{m} \leq d(n_i, n_j) < 2500\,\text{m} \\ 1\,\text{Mbps} & \text{if } 2500\,\text{m} \leq d(n_i, n_j) < 3000\,\text{m} \\ 0\,\text{Mbps} & \text{if } d(n_i, n_j) \geq 3000\,\text{m} \end{cases}$$

- Transmission rate must be strictly greater than zero

$$7 \geqslant T(n_i, n_j) \geqslant 1$$

- The distance between any two nodes should be less than 3000 meters

$$d(n_i, n_j) < 3000$$

3. **Duplicates nodes**: The sequence of nodes in a path must be distinct.

4. **Node sequence constraint**: A path must start with a specific node and terminate to one of the two base stations.

Briefly review potential methods

- Routing path optimization in wireless sensor networks is a critical problem with smart cities, IoT systems, and environmental monitoring applications. One of the primary challenges of routing path optimization involves balancing different objectives such as maximizing the data transmission rate while also minimizing end-to-end latency.
- The complexity of multi-objective optimization problems can often be too high for traditional single-method optimization techniques to be effective. Integrating the advantages of several algorithms, and or implementing hybrid optimization techniques have proven effective ways to improve search effectiveness, convergence speed, and solution quality.
- Below are reviews of some potential methods that can be used to address this kind of routing path optimization problem:

Genetic Algorithm (GA)

- Genetic Algorithms (GA) are evolutionary algorithms which are metaheuristic inspired by natural selection processes. GA algorithms use crossover, mutation, and selection to evolve a population of better solutions towards optimality. GA handles constraints such as connectivity and valid paths through customized genetic operators.
- The application of GA in WSN routing path optimization problems is widely used due to its ability to explore a high solution space efficiently and discretely. (Alnajjar et al. 2022) demonstrated that GA can optimize routing paths by evolving solutions that balance latency and energy consumption. However, GA can suffer from premature or slow convergence by making it suboptimal for finding finetuned solutions.

Particle Swarm Optimization (PSO)

- PSO is a metaheuristic algorithm inspired by swarms' social behaviours. PSO optimizes by having solutions move the search space guided by their own experience and global best solution. Particle swarm optimization (PSO) is simple to implement, adapts well to continuous and discrete problems, and is also suitable for multi-objective optimization through Pareto-based extensions.
- PSO has proven to be effective in WNS's due to its simplicity and fast convergence. PSO was used by (Ghawy et al. 2022) to optimise WSN routing paths, resulting in high transmission rates and low latency paths.

Ant Colony Optimization (ACO)

- ACO is a bio-inspired algorithm based on the behavior of ants. It uses pheromone trails to guide the search for the shortest paths, making it more suitable for routing problems.
- Ant Colony Optimization (ACO) has successfully been applied to WSN routing optimization problems to minimize latency while also maintaining the robustness of the network. (Blum and López-Ibáñez 2007)demonstrated that ACO can dynamically adjust paths based on their pheromone evaporation which also prevents stagnation. This helps in discovering and finding high-quality paths. ACO has been proven to be effective in routing problems, such as network routing and traveling salesperson.

Hybrid Approaches

- This is the combination of two or more methods to leverage their strengths thereby overcoming the limitations of individual methods. Examples include combining Genetic Algorithm (GA) which is used to explore solutions globally, with local search techniques integrating Particle Swarm (PSO) with Ant Colony Optimization (ACO) or integrating Genetic Algorithm (GA) with Simulated Annealing (SA) which is also a local optimization technique. Hybrid Approaches increase the robustness and solution quality and address the weaknesses of individual methods.

Simulated Annealing (SA)

- SA is a metaheuristic by the annealing process in metallurgy, where a material is heated and then it slowly cools to reach a more stable configuration. SA explores solutions by accepting worse solutions with a certain probability, which decreases over time. It is simple and effective for discrete problems. SA has proven to be successful in optimizing routing in WSN's by focusing on minimizing latency and energy consumption. (Kirkpatrick et al. 1983) showed and demonstrated that SA can be effectively refined in solutions generated by other global search methods.

Selected methods and justification

Algorithm 1: Discrete genetic algorithm (DGA)

- GAs are well-suited for network routing optimization problems due to their robustness and flexibility in navigating large and complex search spaces (Amit Singh and Dr. Devendra Singh 2023). Studies have shown that genetic algorithms perform well in multi-objective optimization problems where trade-offs between objectives are crucial. Similarly, (Amit Singh and Dr. Devendra Singh 2023) highlighted the advantages of genetic algorithms in network optimization, particularly their ability to adapt to dynamic constraints and complex objective functions.

- A discrete genetic algorithm was developed for this optimal problem due to the nature of the optimization problem. The goal is to find the best routing path, which is a sequence of nodes that starts at a source sensor node and ends at one of the base stations, which makes the sequence of nodes a discrete representation, as nodes are labelled by integers example:

$$Node_1, Node_2, \ldots, Node_n$$

- Furthermore, the two target objectives are based on discrete choices:
  - The transmission rate between nodes are determined by fixed ranges of distances, leading to discrete values:

$$7\,\text{Mbps}, 5\,\text{Mbps}, \ldots, 0\,\text{Mbps}$$

  - The latency depends on the discrete number of hops in the routing path.

In [ ]:
```
@dataclass(frozen=False)
class Discrete_Genetic_algorithm:
nodes: 'NDArray'
pop_size: int
generations: int
mutation_rate: float

# Creates a population of path
def _create_population(self, size, nodes, starting_node):
    population = []
    n = len(nodes)
    available_nodes = [index for index, node in enumerate(nodes, start=1) if index not in [starting_node] ] # List that contains all the nodes except the starting node
    for _ in range(size):
        path = random.sample(available_nodes, random.randint(0, n-1)) # Creates random routing paths
        path.insert(0, starting_node) # Placing the starting node the beginning of each path
        path.append( # Placing one of the base stations at the end of each path
            random.choice(
                list(
                    base_stations.keys()
                )
            ) # type: ignore
        )
        population.append(path)

    return population

# Evaluate fitness of the population
def _evaluate_fitness(self, population, w1=.5, w2=.5):
    paths_rate = []
    paths_latency = []
    fitness_score = []
    paths_len = []
    max_rate = 7 # Maximum rate in the network
    max_latency = 30 * (len(self.nodes) + 2) # Max latency in the network

for index, pop_nodes in enumerate(population):
    distance_per_node = [ # Calculates distances between nodes in the current path
        calculate_distance(
            node1=pop_nodes[index - 1],
            node2=pop_nodes[index],
            )
        for index, node_id
        in enumerate(pop_nodes, start=1)
        if index < len(pop_nodes)
    ]

    rates = [ # Calculates rates between nodes in the current path
        transmission_rate(
            distance=dist
        )
        for dist in distance_per_node
    ]

    total_rate = calculate_path_transmission_rate(rates)
    total_latency = calculate_path_latency(path=pop_nodes)

    paths_len.append( len(pop_nodes) )
    paths_rate.append(total_rate)
    paths_latency.append(total_latency)

# Weighted sum cost function method
fitness_score = [
    w1 * (rate / max_rate) - w2 * (latency / max_latency)
    for rate, latency in zip(paths_rate, paths_latency)
]

return fitness_score, max(paths_rate), min(paths_latency)

# Selects the best route from the population
def _selection(self, population, fitnesses):
    best_indices = np.argsort(fitnesses)[::-1][:len(fitnesses) // 2] # Selects 50% of the routes with the highest fitness score
    return [population[i] for i in best_indices]


# Combines two path to generate two new offsprings (paths) using the two-points cross method
def _crossover(self, path1, path2):
    start, end = sorted( # Selects two random points in paths1
        random.sample(
            range(len(path1)),
            2
        )
    )

    offspring = path1.copy()
    sublist_to_replace = offspring[start:end]

    n = len(sublist_to_replace)

    # Two points crossover
    for i in range(len(offspring) - n + 1):
        if offspring[i:i+n] == sublist_to_replace:

            for node in path2:
                if node not in offspring and node not in base_stations.keys():
                    if i == 0:
                        continue
                    offspring[i] = node
                    i+=1
```

```
                    if i >= end:
                        break
            break

    return offspring

# Mutate random nodes based on a certain probability
def _mutation(self, path, mutation_rate):

    if len(path) > 2 and random.random() < mutation_rate:

        # Selects two random nodes in the current path.
        idx1 = random.randint(1, len(path) - 2)
        idx2 = random.randint(1, len(path) - 2)

        # Swaps the position the two selected nodes
        path[idx1], path[idx2] = path[idx2], path[idx1]


def _display_path_plot(self, best_latency_per_gen, best_transmission_rate_per_gen):
    fig = plt.figure(figsize=(12,6))
    ax = fig.add_subplot(1, 2, 1)
    plt.subplot(1,2,1)
    plt.plot(best_latency_per_gen, label="Latency", color='b')
    plt.xlabel("Generation")
    plt.ylabel("Latency")
    plt.title("Latency Over Generations (GA)")

    # ax.spines['left'].set_position('center')
    # ax.spines['bottom'].set_position('zero')
    ax.xaxis.set_ticks_position('bottom')
    ax.yaxis.set_ticks_position('left')

    plt.subplot(1,2,2)
    plt.plot(best_transmission_rate_per_gen, label="Transmission Rate", color='r')
    plt.xlabel("Generation")
    plt.ylabel("Transmission Rate")
    plt.title("Transmission Rate Over Generations (GA)")

    plt.tight_layout()
    plt.show()

def _display_fitness_plot(self, best_fitness, std_score):
    fig = plt.figure(figsize=(12,6))
    ax = fig.add_subplot(1, 2, 1)

    plt.subplot(1,2,1)
    plt.plot(best_fitness, label="Fitness score", color='b')
    # plt.plot(std_score, label="Fitness standard deviation", color='r')
    plt.xlabel("Generation")
    plt.ylabel("Fitness")
    plt.title("Fitness Over Generations (GA)")
    plt.legend()

    plt.subplot(1,2,2)
    plt.plot(std_score, label="Fitness standard deviation", color='r')
    plt.xlabel("Generation")
    plt.ylabel("Std")
    plt.title("Standard deviation Over Generations (GA)")

    plt.tight_layout()
    plt.legend()
    plt.show()

# Output one or more optimal paths
def _get_best_paths(self, best_fitness_score, last_population):
    fitnesses, _, _ = self._evaluate_fitness(last_population)
    best_indices = [index for index, value in enumerate(fitnesses) if value == best_fitness_score]
    best_paths = [last_population[i] for i in best_indices]
    best_paths = [list(x) for x in set(map(tuple, best_paths))]
    return best_paths

# Genetic algorithm process
def GA(self, starting_node: int, verbose=True):
    population = self._create_population(self.pop_size, self.nodes, starting_node)
    best_path = None
    best_fitness = 0
    best_latency = float('inf')
    best_transmission_rate = 0
    fitness_scores = []
    fitness_std = []
    all_best_paths = []
    iteration_num = 0


    best_latency_per_gen = []
    best_transmission_rate_per_gen = []
    convergence_counter = 0

    for generation in range(self.generations):
        fitnesses, _, _ = self._evaluate_fitness(population)
        best_gen_path = population[np.argmax(fitnesses)]

        if max(fitnesses) > best_fitness:
            best_fitness = max(fitnesses)
            best_path = best_gen_path

            best_distances = [
                calculate_distance(
                    node1=best_path[index - 1],
                    node2=best_path[index],
                    )
                for index, node_id
                in enumerate(best_path, start=1)
                if index < len(best_path)
            ]

            rates = [
                transmission_rate(
                    distance=dist
                )
                for dist in best_distances
            ]

            best_latency = calculate_path_latency(best_gen_path)
            best_transmission_rate = calculate_path_transmission_rate(rates)

        new_population = self._selection(population, fitnesses)

        # Recreates population with mutated paths
        while len(new_population) < self.pop_size:

            parent1, parent2 = random.sample(new_population, 2)

            if (parent1 is not None) and (parent2 is not None):
```

```
                child = self._crossover(parent1, parent2)
                self._mutation(child, self.mutation_rate)
                new_population.append(child)

        if verbose:
            print(f"Generation {generation}: Best latency = {best_latency}, Best rate = {best_transmission_rate}, Fitness std - ({np.std(fitnesses)}), Highest fitness for this gen - ({max

        iteration_num+=1 # Records generation number

        if np.std(fitnesses) == 0:
            convergence_counter+=1
            if convergence_counter >= 10:
                print("Algorithm converged") if verbose else None
                break
        else:
            convergence_counter = 0

        fitness_scores.append(max(fitnesses))
        fitness_std.append(np.std(fitnesses))
        best_latency_per_gen.append(sys.maxsize if best_latency == float('inf') else best_latency)
        best_transmission_rate_per_gen.append(best_transmission_rate)

        population = new_population

    if verbose:
        self._display_fitness_plot(fitness_scores, fitness_std)

    all_best_paths = self._get_best_paths(best_fitness ,population)

    return best_path, best_latency, best_transmission_rate, all_best_paths, fitness_scores, fitness_std, iteration_num
```

```
def plot_route(self, path):
    print("Best path:", path)
    path_coords = [NODES[node - 1] if isinstance(node, int) else base_stations[node] for node in path]
    # path_coords.append(path[0])
    x, y = zip(*path_coords)

    # print(x)

    plt.figure(figsize=(10,10))
    plt.plot(x, y)
    # Plot sensor nodes
    for index, node in enumerate(NODES, start=1):
        if node[0] in x and node[1] in y:
            plt.scatter(node[0], node[1], color='green', marker='D')  # Node positions (x, y)
            plt.text(node[0] + 100, node[1], f"{index}", color="green", fontsize=9)  # Node labels
        else:
            plt.scatter(node[0], node[1], color='blue', marker='o')  # Node positions (x, y)
            plt.text(node[0] + 100, node[1], f"{index}", color="blue", fontsize=9)  # Node labels

    # Plot base stations
    for label, (x, y) in base_stations.items():
        plt.scatter(x, y, color='red', marker='s', s=100, label=label)  # Base station positions
        plt.text(x + 100, y, label, color="red", fontsize=12)

    # for node, coord in enumerate(NODES, start=1):
    #     plt.annotate(f"{node}", coord, textcoords="offset points", xytext=(0,10), ha='center')

    plt.xticks([])
    plt.yticks([])

    plt.title(f"Best network path from node n°{path[0]}")
    plt.legend(loc="upper right")
    plt.grid(True)
    plt.show()

def plot_multiple_route(self, paths):
    plt.figure(figsize=(10, 10))

    for index, path in enumerate(paths, start=1):
        print(f"Best path n°{index}: {path}")
        path_coords = []

        for node in path:
            if isinstance(node, int):  # Check for nodes
                path_coords.append(NODES[node - 1])
            elif isinstance(node, str):  # Check for Base stations
                path_coords.append(base_stations[node])

        x, y = zip(*path_coords)
        plt.plot(x, y, label=f"Path {paths.index(path)+1}")

    # Plot sensor nodes
    for index, node in enumerate(NODES, start=1):

        is_part_of_any_path = False
        for path_coords in paths:
            path_points = []

            for node_in_path in path_coords:
                if isinstance(node_in_path, int):  # Check for sensor node
                    path_points.append(NODES[node_in_path - 1])
                elif isinstance(node_in_path, str):  # Check for base station
                    path_points.append(base_stations[node_in_path])

            # Check if the node is in any of the paths
            if any(node[0] == x and node[1] == y for x, y in path_points):
                is_part_of_any_path = True
                break

        if is_part_of_any_path:
            plt.scatter(node[0], node[1], color='green', marker='D')  # Node positions (x, y)
            plt.text(node[0] + 100, node[1], f"{index}", color="green", fontsize=9)  # Node labels
        else:
            plt.scatter(node[0], node[1], color='blue', marker='o')  # Node positions (x, y)
            plt.text(node[0] + 100, node[1], f"{index}", color="blue", fontsize=9)  # Node labels

    # Plot base stations
    for label, (x, y) in base_stations.items():
        plt.scatter(x, y, color='red', marker='s', s=100, label=f"Base station {label[-1]}")  # Base station positions
        plt.text(x + 100, y, label, color="red", fontsize=12)

    plt.xticks([])
    plt.yticks([])
    plt.title(f"Best network path(s) from node n°{paths[0][0]}")
    plt.legend(loc="upper right")
    plt.grid(True)
    plt.show()
```

Algorithm 2: Ant colony optimization (ACO)

- The distributed nature of ACA is perfectly aligned with the network's architecture for wireless sensor networks, which allows for localized decision-making and efficient resource utilization. Research by Di Caro and Dorigo (1998) established ACA as a leading approach for routing optimization in dynamic networks, underscoring its ability to manage real-time constraints and network variability.

- Additionally, ACA's pheromone evaporation mechanism prevents premature convergence to suboptimal solutions, promoting diversity and exploration of alternative routing paths.
- ACA provides a robust, adaptive, and scalable solution to the routing path optimization problem, making it a highly suitable algorithm for this optimization problem.

In [ ]:
```python
@dataclass(frozen=False)
class Ant_colony_algorithm:
    nodes: 'NDArray'
    alpha: int
    beta: int
    _evaporation_rate: float
    _pheromone_deposit: int
    BS_stations: dict
    dist_matrix: 'NDArray'
    rate_matrix: 'NDArray'

    # Initialise artificial ants that creates paths
    def _create_ant(self, starting_node, _pheromones, _heuristics, _min_rate_threshold=1,):
        path = [starting_node]
        visited = set( [path[-1] - 1] )
        prob = []
        rates = []

        while True:
            current_node = path[-1] - 1
            probabilities = self._calculate_transition_probabilities(current_node, visited, _pheromones, _heuristics)
            next_node = self._roulette_wheel_selection(probabilities)

            path.append((next_node + 1))
            visited.add(next_node)
            prob.append(probabilities)


            if self.nodes[next_node] in np.array(list(self.BS_stations.values())): # Check if the next_node is one of the base stations.
                node1 = path[-1] - 1
                node2 = path[-2] - 1
                rates.append(self.rate_matrix[node1, node2])
                break


            if self.rate_matrix[path[-2] - 1, path[-1] - 1] < _min_rate_threshold: # Checks if the path is valid (where rate > 0 )
                path.pop() # Removes the invalid node
            else:
                node1 = path[-1] - 1
                node2 = path[-2] - 1
                rates.append(self.rate_matrix[node1, node2])

        return path

    # Calculates probabilities that determines the movements of the ants
    def _calculate_transition_probabilities(self, current_node, visited, _pheromones, _heuristics ,_scaling_factor=1):
        n_nodes = len(self.nodes)
        probabilities = []
        for j in range(n_nodes):
            if j not in visited:
                pheromone = _pheromones[current_node, j] ** self.alpha
                heuristic = _heuristics[current_node, j] ** self.beta
                probabilities.append((pheromone * heuristic) ** _scaling_factor) # Normalize the probabilities to a scale between 0 and 1
            else:
                probabilities.append(0)

        probabilities = np.array(probabilities)
        return probabilities / probabilities.sum()

    # Selects node based on probabilities
    def _roulette_wheel_selection(self, probabilities):
        cumulative_probabilities = np.cumsum(probabilities)

        random_value = random.random()
        for i, cp in enumerate(cumulative_probabilities):

            if random_value <= cp:
                return i
        return len(probabilities) - 1

    def _evaluate_fitness(self, ant, w1=.5, w2=.5):
        max_rate = 7 # Maximum rate in the network
        max_latency = 30 * (len(self.nodes)) # Max latency in the network
        rates = []

        index = 1

        while True:

            node1 = ant[index - 1] - 1
            node2 = ant[index] - 1

            rates.append(
                self.rate_matrix[
                    node1,
                    node2
                ]
            )
            index+=1
            if index >= len(ant):
                break

        total_rate = min(rates)
        latency = (len(ant) - 1) * 30

        # Weighted sum cost function
        fitness_score = w1 * ( total_rate / max_rate ) - w2 * ( latency / max_latency )

        return fitness_score

    def _spread_pheromones(self, all_paths, all_fitnesses, pheromones):
        pheromones *= (1 - self._evaporation_rate)  # Evaporate pheromones
        for path, fitness in zip(all_paths, all_fitnesses):
            for i in range(len(path) - 1):
                node1 = path[i] - 1
                node2 = path[i + 1] - 1
                pheromones[node1, node2] += self._pheromone_deposit / fitness

    def _get_best_paths(self, best_fitness_score, last_population):
        # print("Last pop",last_population)
        fitnesses = [self._evaluate_fitness(path) for path in last_population]
        # print("fitness", max(fitnesses))
        best_indices = [index for index, value in enumerate(fitnesses) if value == best_fitness_score]
        best_paths = [last_population[i] for i in best_indices]
        # best_paths = list(set(best_paths))
        best_paths = [list(x) for x in set(map(tuple, best_paths))]
        # print("best paths:", best_paths)
        return best_paths

    def plot_route(self, path):
        display_best = []
```

```
        for node in path:
            if node == 151:
                display_best.append('BS-1')
            elif node == 152:
                display_best.append('BS-2')
            else:
                display_best.append(node)

        print("Best path:", display_best)

        path_coords = [self.nodes[node - 1] if isinstance(node, int) else base_stations[node] for node in path]
        # path_coords.append(path[0])
        x, y = zip(*path_coords)

        # print(x)

        plt.figure(figsize=(10,10))
        plt.plot(x, y)
        # Plot sensor nodes
        for index, node in enumerate(self.nodes, start=1):
            if node[0] in x and node[1] in y:
                plt.scatter(node[0], node[1], color='green', marker='D')  # Node positions (x, y)
                plt.text(node[0] + 100, node[1], f"{index}", color="green", fontsize=9)  # Node labels
            else:
                plt.scatter(node[0], node[1], color='blue', marker='o')  # Node positions (x, y)
                plt.text(node[0] + 100, node[1], f"{index}", color="blue", fontsize=9)  # Node labels

        # Plot base stations
        for label, (x, y) in self.BS_stations.items():
            plt.scatter(x, y, color='red', marker='s', s=100, label=label)  # Base station positions
            plt.text(x + 100, y, label, color="red", fontsize=12)

        # for node, coord in enumerate(NODES, start=1):
        #     plt.annotate(f"{node}", coord, textcoords="offset points", xytext=(0,10), ha='center')

        plt.xticks([])
        plt.yticks([])

        plt.title(f"Best network path from node n°{path[0]}")
        plt.legend(loc="upper right")
        plt.grid(True)
        plt.show()

    def _display_fitness_plot(self, best_fitness, std_score):
        fig = plt.figure(figsize=(12,6))
        ax = fig.add_subplot(1, 2, 1)

        plt.subplot(1,2,1)
        plt.plot(best_fitness, label="Fitness score", color='b')
        # plt.plot(std_score, label="Fitness standard deviation", color='r')
        plt.xlabel("Generation")
        plt.ylabel("Fitness")
        plt.title("Fitness Over Generations (ACO)")
        plt.legend()

        plt.subplot(1,2,2)
        plt.plot(std_score, label="Fitness standard deviation", color='r')
        plt.xlabel("Generation")
        plt.ylabel("Std")
        plt.title("Standard deviation Over Generations (ACO)")

        plt.tight_layout()
        plt.legend()
        plt.show()

    def plot_multiple_route(self, paths):
        plt.figure(figsize=(10, 10))

        for index, path in enumerate(paths, start=1):
            path_coords = []

            if path[-1] == 151:
                path[-1] = 'BS-1'
            elif path[-1] == 152:
                path[-1] = 'BS-2'

            print(f"Best path n°{index}: {path}")

            for node in path:
                if isinstance(node, int):  # Check for nodes
                    path_coords.append(NODES[node - 1])
                elif isinstance(node, str):  # Check for Base stations
                    path_coords.append(base_stations[node])

            x, y = zip(*path_coords)
            plt.plot(x, y, label=f"Path {paths.index(path)+1}")

        # Plot sensor nodes
        for index, node in enumerate(NODES, start=1):

            is_part_of_any_path = False
            for path_coords in paths:
                path_points = []

                for node_in_path in path_coords:
                    if isinstance(node_in_path, int):  # Check for sensor node
                        path_points.append(NODES[node_in_path - 1])
                    elif isinstance(node_in_path, str):  # Check for base station
                        path_points.append(base_stations[node_in_path])

                # Check if the node is in any of the paths
                if any(node[0] == x and node[1] == y for x, y in path_points):
                    is_part_of_any_path = True
                    break

            if is_part_of_any_path:
                plt.scatter(node[0], node[1], color='green', marker='D')  # Node positions (x, y)
                plt.text(node[0] + 100, node[1], f"{index}", color="green", fontsize=9)  # Node labels
            else:
                plt.scatter(node[0], node[1], color='blue', marker='o')  # Node positions (x, y)
                plt.text(node[0] + 100, node[1], f"{index}", color="blue", fontsize=9)  # Node labels

        # Plot base stations
        for label, (x, y) in base_stations.items():
            plt.scatter(x, y, color='red', marker='s', s=100, label=f"Base station {label[-1]}")  # Base station positions
            plt.text(x + 100, y, label, color="red", fontsize=12)

        plt.xticks([])
        plt.yticks([])
        plt.title(f"Best network path(s) from node n°{paths[0][0]}")
        plt.legend(loc="upper right")
        plt.grid(True)
        plt.show()

    def start_aco(self, starting_node, iteration, ants, verbose=True):
```

```
        n_nodes = len(self.nodes)
        _pheromones = np.ones((n_nodes, n_nodes))
        _heuristics = 1 / (self.dist_matrix + np.eye(n_nodes) * 1e-10)

        best_path = []
        best_fitness = -float('inf')
        best_rate = 0
        convergence_counter = 0
        fitness_std = []
        fitness_scores = []
        total_paths = []
        iteration_num = 0

        for iteration in range(iteration):
            all_paths = []
            all_fitnesses = []

            for _ in range(ants):
                # print("iter", i)
                path = self._create_ant(starting_node,_pheromones, _heuristics)
                fitness = self._evaluate_fitness(path)
                all_paths.append(path)
                total_paths.append(path)
                all_fitnesses.append(fitness)

                if fitness > best_fitness:
                    best_path = path
                    best_fitness = fitness
                    rates = [
                        self.rate_matrix[
                            best_path[index - 1] - 1,
                            best_path[index] - 1,
                        ]
                        for index, node_id
                        in enumerate(best_path, start=1)
                        if index < len(best_path)
                    ]
                    best_rate = min(rates)

            self._spread_pheromones(all_paths, all_fitnesses, _pheromones)

            fitness_std.append(np.std(all_fitnesses))
            fitness_scores.append(best_fitness)

            if verbose:
                print(f"Iteration {iteration}: Best rate - {best_rate}, Best latency - {(len(best_path) - 1) * 30}, Fitness std - ({np.std(all_fitnesses)}), Best Fitness = {best_fitness}")

            iteration_num+=1 # Generation number

            if np.std(all_fitnesses) == 0:
                convergence_counter+=1
                if convergence_counter > 4:
                    print("Algorithmm converged") if verbose else None
                    break
            else:
                convergence_counter = 0

        all_best_paths = self._get_best_paths(best_fitness, total_paths)

        if verbose:
            self._display_fitness_plot(fitness_scores, fitness_std)

        return best_path, all_best_paths, best_rate, (len(best_path) - 1) * 30, fitness_scores, fitness_std, iteration_num
```

Algorithm 3: Hybrid (Genetic algorithm + Simulated annealing)

- Implementing a hybrid algorithm combining Genetic Algorithm (GA) and Simulated Annealing (SA) leverages the strengths of both methods, making it a powerful choice for solving this optimization problem. This approach addresses the limitations of each algorithm individually while amplifying their advantages.
- The SA components ensure a diverse search across paths, while the GA components refines the best candidates to maximize solution quality.
- Hybrid approaches have been successfully utilized in multi-objective optimization problems. For example, Bean and Hadj-Alouane (1993) demonstrated the effectiveness of hybridizing GA with local search techniques to improve convergence rates and solution accuracy. Similarly, Shirazi A. (2017) showed that combining GA and SA significantly improved performance in network routing problems by integrating GA's diversity-driven exploration with SA's precision in refining solutions.

In [ ]:
```
@dataclass(frozen=False)
class Hybrid_GA_with_SA:
    nodes: 'NDArray'
    pop_size: int
    generations: int
    mutation_rate: float
    cooling_rate: float


    def _create_population(self, size, nodes, starting_node):
        population = []
        n = len(nodes)
        available_nodes = [index for index, node in enumerate(nodes, start=1) if index not in [starting_node] ] # List that contains all the nodes except the starting node
        for _ in range(size):
            path = random.sample(available_nodes, random.randint(0, n-1)) # Creates random routing paths
            path.insert(0, starting_node) # Placing the starting node the beginning of each path
            path.append( # Placing one of the base stations at the end of each path
                random.choice(
                    list(
                        base_stations.keys()
                    )
                ) # type: ignore
            )
            population.append(path)

        return population

    # Evaluate fitness of the population
    def _evaluate_fitness(self, population, w1=.5, w2=.5):
        paths_rate = []
        paths_latency = []
        fitness_score = []
        paths_len = []
        max_rate = 7 # Maximum rate in the network
        max_latency = 30 * (len(self.nodes) + 2) # Max latency in the network

        for index, pop_nodes in enumerate(population):
            distance_per_node = [
                calculate_distance(
                    node1=pop_nodes[index - 1],
                    node2=pop_nodes[index],
                    )
                for index, node_id
                in enumerate(pop_nodes, start=1)
                if index < len(pop_nodes)
            ]
```

```
            rates = [
                transmission_rate(
                    distance=dist
                )
                for dist in distance_per_node
            ]

            total_rate = calculate_path_transmission_rate(rates)
            total_latency = calculate_path_latency(path=pop_nodes)

            paths_len.append( len(pop_nodes) )
            paths_rate.append(total_rate)
            paths_latency.append(total_latency)

        # Promising 1
        fitness_score = [
            w1 * (rate / max_rate) - w2 * (latency / max_latency)
            for rate, latency in zip(paths_rate, paths_latency)
        ]

        return fitness_score, max(paths_rate), min(paths_latency)

    # Selects the best route from the population
    def _selection(self, population, fitnesses):
        best_indices = np.argsort(fitnesses)[::-1][:len(fitnesses) // 2] # Check for Highest
        return [population[i] for i in best_indices]

    # Combines two path to generate two new offsprings (paths) using the two-points cross method
    def _crossover(self, path1, path2):
        start, end = sorted(
            random.sample(
                range(len(path1)),
                2
            )
        )

        offspring = path1.copy()
        sublist_to_replace = offspring[start:end]
        n = len(sublist_to_replace)

        for i in range(len(offspring) - n + 1):
            if offspring[i:i+n] == sublist_to_replace:
                for node in path2:
                    if node not in offspring and node not in base_stations.keys():
                        if i == 0:
                            continue
                        offspring[i] = node
                        i+=1
                        if i >= end:
                            break
                break

        return offspring

    # Mutate random nodes based on a certain probability
    def _mutation(self, path, mutation_rate):
        if len(path) > 2 and random.random() < mutation_rate:
            idx1 = random.randint(1, len(path) - 2)
            idx2 = random.randint(1, len(path) - 2)

            path[idx1], path[idx2] = path[idx2], path[idx1]

    def _display_fitness_plot(self, best_fitness, std_score):
        fig = plt.figure(figsize=(12,6))
        ax = fig.add_subplot(1, 2, 1)

        plt.subplot(1,2,1)
        plt.plot(best_fitness, label="Fitness score", color='b')

        plt.xlabel("Generation")
        plt.ylabel("Fitness")
        plt.title("Fitness Over Generations (Hybrid)")
        plt.legend()

        plt.subplot(1,2,2)
        plt.plot(std_score, label="Fitness standard deviation", color='r')
        plt.xlabel("Generation")
        plt.ylabel("Std")
        plt.title("Standard deviation Over Generations (Hybrid)")

        plt.tight_layout()
        plt.legend()
        plt.show()

    # Accept worse solutions with a probability based on temperature
    def _accept_solution(self, current_fitness, best_fitness, temperature):
        if current_fitness >= best_fitness:
            return True
        else:
            probability = np.exp((current_fitness - best_fitness) / temperature)
            random_probability = random.uniform(.1, 1.0)
            return random_probability < probability

    def _evaluate_single_fitness(self, path, w1=.5, w2=.5):
        max_rate = 7 # Maximum rate in the network
        max_latency = 30 * (len(self.nodes) + 2) # Max Latency in the network

        distance_per_node = [
                calculate_distance(
                    node1=path[index - 1],
                    node2=path[index],
                    )
                for index, node_id
                in enumerate(path, start=1)
                if index < len(path)
        ]

        rates = [
                transmission_rate(
                    distance=dist
                )
                for dist in distance_per_node
        ]

        total_rate = calculate_path_transmission_rate(rates)
        total_latency = calculate_path_latency(path=path)

        fitness_score = w1 * ( total_rate / max_rate ) - w2 * ( total_latency / max_latency )

        return fitness_score

    def _get_best_paths(self, best_fitness_score, last_population):
        fitnesses, _, _ = self._evaluate_fitness(last_population)
        best_indices = [index for index, value in enumerate(fitnesses) if value == best_fitness_score]

        best_paths = [last_population[i] for i in best_indices]
```

```
        best_paths = [list(x) for x in set(map(tuple, best_paths))]
        return best_paths

    def GA(self, starting_node:int, temperature, verbose=True):
        population = self._create_population(self.pop_size, self.nodes, starting_node)
        best_path = None
        best_fitness = 0

        best_latency = float('inf')
        best_transmission_rate = 0
        final_paths = []
        fitness_scores = []
        fitness_std = []
        all_best_paths = []
        iteration_num = 0

        best_latency_per_gen = []
        best_transmission_rate_per_gen = []
        convergence_counter = 0

        for generation in range(self.generations):
            fitnesses, best_gen_rate, best_gen_latency = self._evaluate_fitness(population)
            best_gen_path = population[np.argmax(fitnesses)]

            if max(fitnesses) > best_fitness:
                best_fitness = max(fitnesses)
                best_path = best_gen_path
                final_paths.append((best_path, best_fitness))

                best_distances = [
                calculate_distance(
                    node1=best_path[index - 1],
                    node2=best_path[index],
                    )
                for index, node_id
                in enumerate(best_path, start=1)
                if index < len(best_path)
            ]

            rates = [
                transmission_rate(
                    distance=dist
                )
                for dist in best_distances
            ]

            best_latency = calculate_path_latency(best_gen_path)
            best_transmission_rate = calculate_path_transmission_rate(rates)

        new_population = self._selection(population, fitnesses)

        while len(new_population) < self.pop_size: # Recreates paths to repopulate
            parent1, parent2 = random.sample(new_population, 2)

            if (parent1 is not None) and (parent2 is not None):
                child = self._crossover(parent1, parent2)
                self._mutation(child, self.mutation_rate)
                child_fitness = self._evaluate_single_fitness(child)

                if self._accept_solution(child_fitness, best_fitness, temperature):
                    new_population.append(child)
                else:
                    new_population.append(
                        random.choice(
                            (parent1, parent2) # Chooses randomly between parent1 and parent2
                        )
                    )


        if verbose:
            print(f"Generation {generation}: Best latency = {best_latency}, Best rate = {best_transmission_rate}, Fitness std - ({np.std(fitnesses)}) Highest fitness for this gen - ({max(

        # Cool down the temperature
        temperature *= self.cooling_rate # type: ignore

        # Stop if the temperature is too low
        if temperature < 1e-3:
            break

        iteration_num+=1 # Generation number

        if np.std(fitnesses) == 0:
            convergence_counter+=1
            if convergence_counter >= 10:
                print("Algorithm converged") if verbose else None
                break
        else:
            convergence_counter = 0

        fitness_scores.append(max(fitnesses))
        fitness_std.append(np.std(fitnesses))
        best_latency_per_gen.append(sys.maxsize if best_latency == float('inf') else best_latency)
        best_transmission_rate_per_gen.append(best_transmission_rate)

        population = new_population

        if verbose:
            self._display_fitness_plot(fitness_scores, fitness_std)

        all_best_paths = self._get_best_paths(best_fitness ,population)

        return best_path, best_latency, best_transmission_rate, all_best_paths, fitness_scores, fitness_std, iteration_num

    def plot_route(self, path):
        print("Best path:", path)
        path_coords = [NODES[node - 1] if isinstance(node, int) else base_stations[node] for node in path]
        # path_coords.append(path[0])
        x, y = zip(*path_coords)

        # print(x)

        plt.figure(figsize=(10,10))
        plt.plot(x, y)
        # Plot sensor nodes
        for index, node in enumerate(NODES, start=1):
            if node[0] in x and node[1] in y:
                plt.scatter(node[0], node[1], color='green', marker='D')  # Node positions (x, y)
                plt.text(node[0] + 100, node[1], f"{index}", color="green", fontsize=9)  # Node labels
            else:
                plt.scatter(node[0], node[1], color='blue', marker='o')  # Node positions (x, y)
                plt.text(node[0] + 100, node[1], f"{index}", color="blue", fontsize=9)  # Node labels

        # Plot base stations
        for label, (x, y) in base_stations.items():
            plt.scatter(x, y, color='red', marker='s', s=100, label=label)  # Base station positions
            plt.text(x + 100, y, label, color="red", fontsize=12)
```

```
    plt.xticks([])
    plt.yticks([])

    plt.title(f"Best network path from node n°{path[0]}")
    plt.legend(loc="upper right")
    plt.grid(True)
    plt.show()

def plot_multiple_route(self, paths):
    plt.figure(figsize=(10, 10))

    for index, path in enumerate(paths, start=1):
        print(f"Best path n°{index}: {path}")
        path_coords = []

        for node in path:
            if isinstance(node, int):  # Check for nodes
                path_coords.append(NODES[node - 1])
            elif isinstance(node, str):  # Check for Base stations
                path_coords.append(base_stations[node])

        x, y = zip(*path_coords)
        plt.plot(x, y, label=f"Path {paths.index(path)+1}")

    # Plot sensor nodes
    for index, node in enumerate(NODES, start=1):

        is_part_of_any_path = False
        for path_coords in paths:
            path_points = []

            for node_in_path in path_coords:
                if isinstance(node_in_path, int):  # Check for sensor node
                    path_points.append(NODES[node_in_path - 1])
                elif isinstance(node_in_path, str):  # Check for base station
                    path_points.append(base_stations[node_in_path])

            # Check if the node is in any of the paths
            if any(node[0] == x and node[1] == y for x, y in path_points):
                is_part_of_any_path = True
                break

        if is_part_of_any_path:
            plt.scatter(node[0], node[1], color='green', marker='D')  # Node positions (x, y)
            plt.text(node[0] + 100, node[1], f"{index}", color="green", fontsize=9)  # Node labels
        else:
            plt.scatter(node[0], node[1], color='blue', marker='o')  # Node positions (x, y)
            plt.text(node[0] + 100, node[1], f"{index}", color="blue", fontsize=9)  # Node labels

    # Plot base stations
    for label, (x, y) in base_stations.items():
        plt.scatter(x, y, color='red', marker='s', s=100, label=f"Base station {label[-1]}")  # Base station positions
        plt.text(x + 100, y, label, color="red", fontsize=12)

    plt.xticks([])
    plt.yticks([])
    plt.title(f"Best network path(s) from node n°{paths[0][0]}")
    plt.legend(loc="upper right")
    plt.grid(True)
    plt.show()
```

Methods discussion

A. Discrete genetic algorithm

Initialization

- The initialisation process is initiated with a random approach, incorporating a circumvention for the constraint that requires a path to start from a node within the network and terminate at one of the two base stations.

```
# Creates a population of path
    def _create_population(self, size, nodes, starting_node):
        population = []
        n = len(nodes)
        available_nodes = [index for index, node in enumerate(nodes, start=1) if index not in [starting_node] ]
# List that contains all the nodes except the starting node
        for _ in range(size):
            path = random.sample(available_nodes, random.randint(0, n-1)) # Creates random routing paths
            path.insert(0, starting_node) # Placing the starting node the beginning of each path
            path.append( # Placing one of the base stations at the end of each path
                random.choice(
                    list(
                        base_stations.keys()
                    )
                ) # type: ignore
            )
            population.append(path)

        return population
```

- Also, the initialization function introduces diversity in the length of path.

Fitness

- The fitness function is used here as mechanism to evaluate the performance of paths based on the two primary objectives which are minimising the latency and maximising the transmission rate.

```python
# Evaluate fitness of the population
    def _evaluate_fitness(self, population, w1=.5, w2=.5):
        paths_rate = []
        paths_latency = []
        fitness_score = []
        paths_len = []
        max_rate = 7 # Maximum rate in the network
        max_latency = 30 * (len(self.nodes) + 2) # Max Latency in the network

        for index, pop_nodes in enumerate(population):
            distance_per_node = [ # Calculates distances between nodes in the current path
                calculate_distance(
                    node1=pop_nodes[index - 1],
                    node2=pop_nodes[index],
                    )
                for index, node_id
                in enumerate(pop_nodes, start=1)
                if index < len(pop_nodes)
            ]

            rates = [ # Calculates rates between nodes in the current path
                transmission_rate(
                    distance=dist
                )
                for dist in distance_per_node
            ]

            total_rate = calculate_path_transmission_rate(rates)
            total_latency = calculate_path_latency(path=pop_nodes)

            paths_len.append( len(pop_nodes) )
            paths_rate.append(total_rate)
            paths_latency.append(total_latency)

        # Weighted sum cost function method
        fitness_score = [
            w1 * (rate / max_rate) - w2 * (latency / max_latency)
            for rate, latency in zip(paths_rate, paths_latency)
        ]

        return fitness_score, max(paths_rate), min(paths_latency)
```

- The algorithm takes a list of multiple paths as input and calculates the transmission rate and latency for each of the paths. Then the fitness score is calculated by using the weighted sum method which enables the two objectives to be combined into one scalar.

$$fitness\ score = w_1 \cdot \left(\frac{rate}{max\ rate}\right) - w_2 \cdot \left(\frac{\text{latency}}{max\ latency}\right)$$

- Setting the weighted values to 0.5 indicates that both objectives are equally important.
- A higher fitness score indicates a more desirable path.

- If the transmission rate of a path is 7 and the latency is 30, the fitness can reach the maximum value of 0.497 in this optimization problem.

$$fitness\ score = 0.5 \cdot \left(\frac{7}{7}\right) - 0.5 \cdot \left(\frac{30}{30 * 152}\right) = 0.497$$

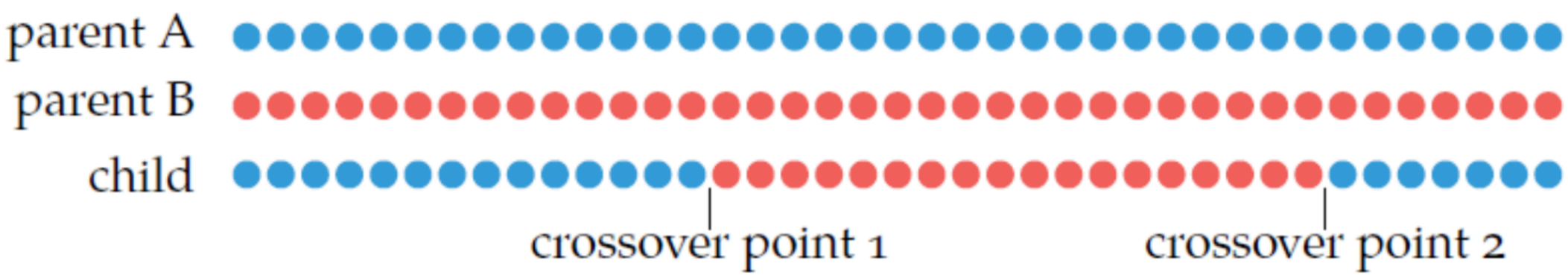
Selection

- The method used for the selection process was the “pairing from top to bottom” method. It only selects 50% of the best paths based on fitness scores.

```python
# Selects the best route from the population
    def _selection(self, population, fitnesses):
        best_indices = np.argsort(fitnesses)[::-1][:len(fitnesses) // 2] # Selects 50% of the routes with the highest fitness score
        return [population[i] for i in best_indices]
```

Crossover

- The “Two-point crossover” method was used to implement the crossover method. This method merges random sequence of nodes from path2 and merges it to path1 to create a new offspring. This method introduces diversity into the population and helps the algorithm to explore the search space.



```
# Combines two path to generate two new offsprings (paths) using the two-points cross method
    def _crossover(self, path1, path2):
        start, end = sorted( # Selects two random points in paths1
            random.sample(
                range(len(path1)),
                2
            )
        )

        offspring = path1.copy()
        sublist_to_replace = offspring[start:end]

        n = len(sublist_to_replace)

        # Two points crossover
        for i in range(len(offspring) - n + 1):
            if offspring[i:i+n] == sublist_to_replace:

                for node in path2:
                    if node not in offspring and node not in base_stations.keys():
                        if i == 0:
                            continue
                        offspring[i] = node
                        i+=1
                        if i >= end:
                            break
                break

        return offspring
```

Mutation

- The mutation here selects two random nodes in a path and swaps it. The mutation process slightly alters a path and introduces diversity into the population. Mutation only occurs if the path has more than two nodes and the mutation rate is met.

```
# Mutate random nodes based on a certain probability
    def _mutation(self, path, mutation_rate):

        if len(path) > 2 and random.random() < mutation_rate:

            # Selects two random nodes in the current path.
            idx1 = random.randint(1, len(path) - 2)
            idx2 = random.randint(1, len(path) - 2)

            # Swaps the position the two selected nodes
            path[idx1], path[idx2] = path[idx2], path[idx1]
```

Genetic algorithm process

- The function simulates the natural process of evolution, where the **selection** process ensures the survival of the fittest path, the **Crossover** creates new paths by combining two existing paths. Mutation maintains diversity and prevents stagnation.

```python
# Genetic algorithm process
    def GA(self, starting_node: int, verbose=True):
        population = self._create_population(self.pop_size, self.nodes, starting_node)
        best_path = None
        best_fitness = 0
        best_latency = float('inf')
        best_transmission_rate = 0
        fitness_scores = []
        fitness_std = []
        all_best_paths = []
        iteration_num = 0


        best_latency_per_gen = []
        best_transmission_rate_per_gen = []
        convergence_counter = 0

        for generation in range(self.generations):
            fitnesses, _, _ = self._evaluate_fitness(population)
            best_gen_path = population[np.argmax(fitnesses)]

            if max(fitnesses) > best_fitness:
                best_fitness = max(fitnesses)
                best_path = best_gen_path

                best_distances = [
                    calculate_distance(
                        node1=best_path[index - 1],
                        node2=best_path[index],
                        )
                    for index, node_id
                    in enumerate(best_path, start=1)
                    if index < len(best_path)
                ]

                rates = [
                    transmission_rate(
                        distance=dist
                    )
                    for dist in best_distances
                ]

                best_latency = calculate_path_latency(best_gen_path)
                best_transmission_rate = calculate_path_transmission_rate(rates)

            new_population = self._selection(population, fitnesses)

            # Recreates population with mutated paths
            while len(new_population) < self.pop_size:

                parent1, parent2 = random.sample(new_population, 2)

                if (parent1 is not None) and (parent2 is not None):

                    child = self._crossover(parent1, parent2)
                    self._mutation(child, self.mutation_rate)
                    new_population.append(child)

            if verbose:
                print(f"Generation {generation}: Best latency = {best_latency}, Best rate = {best_transmission_rate}, Fitness std - ({np.std(
fitnesses)}), Highest fitness for this gen - ({max(fitnesses)})")

            iteration_num+=1 # Records generation number

            if np.std(fitnesses) == 0:
                convergence_counter+=1
                if convergence_counter >= 10:
                    print("Algorithm converged") if verbose else None
                    break
            else:
                convergence_counter = 0

            fitness_scores.append(max(fitnesses))
            fitness_std.append(np.std(fitnesses))
            best_latency_per_gen.append(sys.maxsize if best_latency == float('inf') else best_latency)
            best_transmission_rate_per_gen.append(best_transmission_rate)

            population = new_population

        if verbose:
            self._display_fitness_plot(fitness_scores, fitness_std)

        all_best_paths = self._get_best_paths(best_fitness ,population)

        return best_path, best_latency, best_transmission_rate, all_best_paths, fitness_scores, fitness_std, iteration_num
```

- The algorithm populations evolve over generations to find the most optimal path from a node to a base station. The algorithm either stops after a set number of iterations or it stops if it converges to the optimal point.

B. Ant colony algorithm

Initialization

- The process generates a random valid path (with a transmission rate that is always higher than 0) from a specified starting node to a base station.

```python
# Initialise artificial ants that creates paths
    def _create_ant(self, starting_node, _pheromones, _heuristics, _min_rate_threshold=1,):
        path = [starting_node]
        visited = set( [path[-1] - 1] )
        prob = []
        rates = []

        while True:
            current_node = path[-1] - 1
            probabilities = self._calculate_transition_probabilities(current_node, visited, _pheromones, _heuristics)
            next_node = self._roulette_wheel_selection(probabilities)

            path.append((next_node + 1))
            visited.add(next_node)
            prob.append(probabilities)


            if self.nodes[next_node] in np.array(list(self.BS_stations.values())): # Check if the next_node is one of the base stations.
                node1 = path[-1] - 1
                node2 = path[-2] - 1
                rates.append(self.rate_matrix[node1, node2])
                break


        if self.rate_matrix[path[-2] - 1, path[-1] - 1] < _min_rate_threshold: # Checks if the path is valid (where rate > 0 )
            path.pop() # Removes the invalid node
        else:
            node1 = path[-1] - 1
            node2 = path[-2] - 1
            rates.append(self.rate_matrix[node1, node2])

    return path
```

Path construction

- For each iteration, each ant constructs a solution by iteratively adding components based on the transition probability and the distance (heuristic information).
- The transition probability for ant `k` to move from node `i` to node `j` is given by:

$$Pj = \frac{(\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta})^p}{\sum_{k \notin visited} (\tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta})^p}$$

where:

- Pheromone trail between nodes i and j

τ_{ij}

- η_{ij} is the heuristic information, typically $\frac{1}{d_{ij}}$, where d_{ij} is the distance between nodes i and j.
- α and β are parameters that control the influence of pheromone trails and heuristic information, respectively.
- N_i^k `p` adjusts the probabilities:
- Flattens probabilities (reduces dominance of high values): $p < 1$
- No change: $p = 1$
- Amplifies dominance: $p > 1$

```python
# Calculates probabilities that determines the movements of the ants
    def _calculate_transition_probabilities(self, current_node, visited, _pheromones, _heuristics ,_scaling_factor=1):
        n_nodes = len(self.nodes)
        probabilities = []
        for j in range(n_nodes):
            if j not in visited:
                pheromone = _pheromones[current_node, j] ** self.alpha
                heuristic = _heuristics[current_node, j] ** self.beta
                probabilities.append((pheromone * heuristic) ** _scaling_factor) # Normalize the probabilities to a scale between 0 and 1
            else:
                probabilities.append(0)

        probabilities = np.array(probabilities)
        return probabilities / probabilities.sum()
```

Selection

- The total fitness of the population is computed by summing the fitness values of all individuals:

$$F_{\text{total}} = \sum_{i=1}^{N} f(i)$$

- where `N` is the number of individuals in the population. The probability of selecting an individual `i` is proportional to its fitness value relative to the total fitness. This means that the probability `P(i)` of selecting individual `i` is given by:

$$P(i) = \frac{f(i)}{F_{\text{total}}}$$

- Ants are selected by choosing a random number `r` between 0 and 1. The "wheel" is divided into segments corresponding to the cumulative fitness values of each ant. The individual selected corresponds to the segment that contains the value `r`.

$$C(i) = \sum_{j=1}^{i} P(j)$$

- where `C(i)` is the cumulative probability for individual `i`.
- The ant `i` is selected such that:

$$C(i-1) \leq r < C(i)$$

- where

$$C(0) = 0 \text{ and } C(N) = 1$$

```python
# Selects node based on probabilities
    def _roulette_wheel_selection(self, probabilities):
        cumulative_probabilities = np.cumsum(probabilities)

        random_value = random.random()
        for i, cp in enumerate(cumulative_probabilities):

            if random_value <= cp:
                return i
        return len(probabilities) - 1
```

Pheromone update

- After all ants have constructed their solutions, the pheromone trails are updated:

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

- where:
- ρ is the pheromone evaporation rate

$$0 < \rho < 1$$

- `m` is the number of ants.
- $\Delta\tau_{ij}^{k}$ is the amount of pheromone deposited by ant k, given by:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{L_k} & \text{if edge } (i,j) \text{ is used by ant } k \\ 0 & \text{otherwise} \end{cases}$$

- where L_k is the length of the tour constructed by ant

```
def _spread_pheromones(self, all_paths, all_fitnesses, pheromones):
        pheromones *= (1 - self._evaporation_rate)  # Evaporate pheromones
        for path, fitness in zip(all_paths, all_fitnesses):
            for i in range(len(path) - 1):
                node1 = path[i] - 1
                node2 = path[i + 1] - 1
                pheromones[node1, node2] += self._pheromone_deposit / fitness
```

Fitness

- The same fitness is used as the Discrete Genetic algorithm:

```
def _evaluate_fitness(self, ant, w1=.5, w2=.5):
        max_rate = 7 # Maximum rate in the network
        max_latency = 30 * (len(self.nodes)) # Max latency in the network
        rates = []

        index = 1

        while True:

            node1 = ant[index - 1] - 1
            node2 = ant[index] - 1

            rates.append(
                self.rate_matrix[
                    node1,
                    node2
                ]
            )
            index+=1
            if index >= len(ant):
                break

        total_rate = min(rates)
        latency = (len(ant) - 1) * 30

        # Weighted sum cost function
        fitness_score = w1 * ( total_rate / max_rate ) - w2 * ( latency / max_latency )

        return fitness_score
```

ACO process

- The ACO process is designed to solve optimization problems by mimicking the natural behavior of ants searching for food.
- This is how the process of ACO works:
  - **Artificial Ants**:
    - Creates artificial ants that constructs a solution (a path).
      - Each ants maintains a memory of its path and the corresponding solution quality.
  - **Transition Probability**:
    - Initialize pheromone trails with a small positive value to determines the decision-making of the ants.
    - Initialize distance matrix (heuristic information) to determines the decision-making of the ants.
  - **Pheromone Update:**
    - The pheromone update rule adjusts the pheromone trails based on the quality of the solutions found by the ants.
    - It includes pheromone deposit, which reinforces good paths, and pheromone evaporation, which reduces the influence of old or poor paths.

```
def start_aco(self, starting_node, iteration, ants, verbose=True):
        n_nodes = len(self.nodes)
        _pheromones = np.ones((n_nodes, n_nodes))
        _heuristics = 1 / (self.dist_matrix + np.eye(n_nodes) * 1e-10)

        best_path = []
        best_fitness = -float('inf')
        best_rate = 0
        convergence_counter = 0
        fitness_std = []
        fitness_scores = []
        total_paths = []
        iteration_num = 0

        for iteration in range(iteration):
            all_paths = []
            all_fitnesses = []

            for _ in range(ants):
                # print("iter", i)
                path = self._create_ant(starting_node,_pheromones, _heuristics)
                fitness = self._evaluate_fitness(path)
                all_paths.append(path)
                total_paths.append(path)
                all_fitnesses.append(fitness)

                if fitness > best_fitness:
                    best_path = path
                    best_fitness = fitness
                    rates = [
                        self.rate_matrix[
                            best_path[index - 1] - 1,
                            best_path[index] - 1,
                        ]
                        for index, node_id
                        in enumerate(best_path, start=1)
                        if index < len(best_path)
                    ]
                    best_rate = min(rates)

            self._spread_pheromones(all_paths, all_fitnesses, _pheromones)

            fitness_std.append(np.std(all_fitnesses))
            fitness_scores.append(best_fitness)

            if verbose:
                print(f"Iteration {iteration}: Best rate - {best_rate}, Best latency - {(len(best_path) - 1) * 30}, Fitness std - ({np.std(all_fitnesses)}
), Best Fitness = {best_fitness}")

            iteration_num+=1 # Generation number

            if np.std(all_fitnesses) == 0:
                convergence_counter+=1
                if convergence_counter > 4:
                    print("Algorithmm converged") if verbose else None
                    break
            else:
                convergence_counter = 0

        all_best_paths = self._get_best_paths(best_fitness, total_paths)

        if verbose:
            self._display_fitness_plot(fitness_scores, fitness_std)

        return best_path, all_best_paths, best_rate, (len(best_path) - 1) * 30, fitness_scores, fitness_std, iteration_num
```

C. Hybrid (GA + SA)

The hybrid algorithm has the exact same properties as the genetic algorithm. However, it introduces a temperature-based acceptance mechanism into the GA with three concepts:

- Temperature
- Acceptance criteria
- Cooling schedule

Temperature

- The temperature starts with a high value `T_0`, which allows a broader exploration of the search space. The temperature gradually reduces using a cooling schedule.

$$T = T_0 \cdot \alpha^k$$

  Where:
- `T_0` is the initial temperature.
- `α` is the cooling factor (e.g., 0.95).
- `k` is the current generation.

```
# Cool down the temperature
temperature *= self.cooling_rate # type: ignore
```

Acceptance criteria

- This mechanism computes and compares an individual's current fitness (path) in a population with the current best fitness score.
- If the individual’s current score is better than the best, then the solution is selected in the population for the next generation. The acceptance criteria is determined by probabilities if the solution is worse.

$$P = \exp\left(-\frac{\Delta}{T}\right)$$

- Where:
  - `Δ` is the fitness difference

$$(\text{current fitness} - \text{best fitness})$$

.
  - `T` is the current temperature.

```
# Accept worse solutions with a probability based on temperature
    def _accept_solution(self, current_fitness, best_fitness, temperature):
        if current_fitness >= best_fitness:
            return True
        else:
            probability = np.exp((current_fitness - best_fitness) / temperature)
            random_probability = random.uniform(.1, 1.0)
            return random_probability < probability
```

Cooling schedule

The cooling system uses the temperature to influence the mutation and crossover process.

- A higher temperature increases the probability of selecting a less-fit individual.
- Gradually the temperature decreases which reduces the probability of selecting a worse individual.

```
while len(new_population) < self.pop_size: # Recreates paths to repopulate
    parent1, parent2 = random.sample(new_population, 2)

    if (parent1 is not None) and (parent2 is not None):
        child = self._crossover(parent1, parent2)
        self._mutation(child, self.mutation_rate)
        child_fitness = self._evaluate_single_fitness(child)

        if self._accept_solution(child_fitness, best_fitness, temperature):
            new_population.append(child)
        else:
            new_population.append(
                random.choice(
                    (parent1, parent2) # Chooses randomly between parent1 and parent2
                )
            )
```

Path visualization with discussion

Discrete genetic algorithm

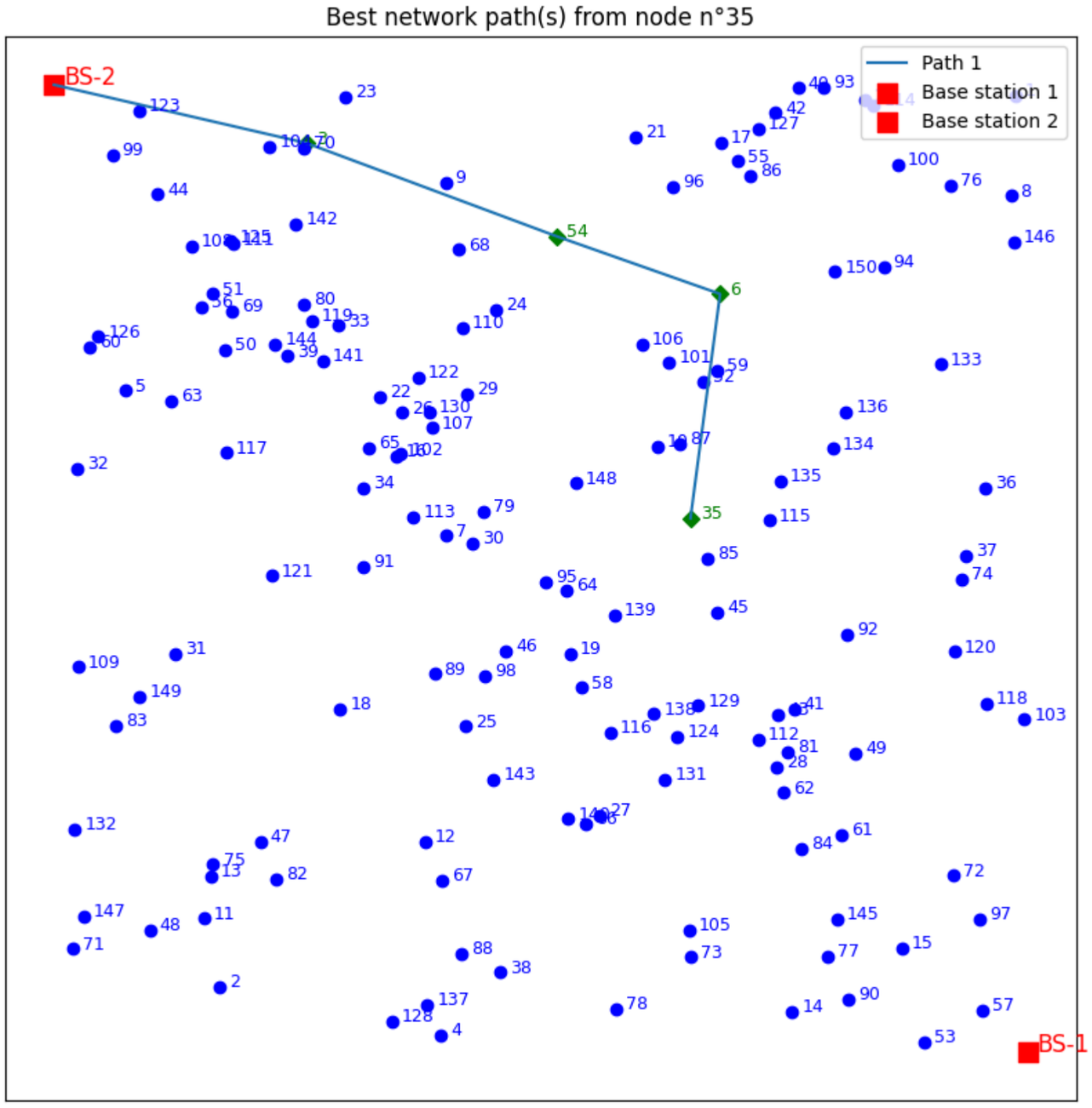
In [22]:
```
DGA = Discrete_Genetic_algorithm(nodes=NODES, pop_size=9000, generations=300, mutation_rate=.9)
```

In [23]:
```
best_path, best_latency, best_transmission_rate, all_best_paths, _, _, _ = DGA.GA(starting_node=35, verbose=False)
print("Total best paths", all_best_paths)
print("Best latency", best_latency)
print("Best transmission rate", best_transmission_rate)
```

```
Total best paths [[35, 6, 54, 3, 'BS-2']]
Best latency 120
Best transmission rate 1
```

In [ ]:
```
DGA.plot_multiple_route(all_best_paths)
```

```
Best path n°1: [35, 6, 54, 3, 'BS-2']
```



Ant colony optimization

In [25]:
```
BS_stations = base_stations.copy()
modified_nodes = NODES.copy()
modified_nodes = np.insert(modified_nodes, len(modified_nodes), list(BS_stations.values()), axis=0)

dist_matrix, base_stations_dist_matrix = calculate_distances(modified_nodes, destination_coordinate)
rate_matrix = np.vectorize(transmission_rate)(dist_matrix)
```

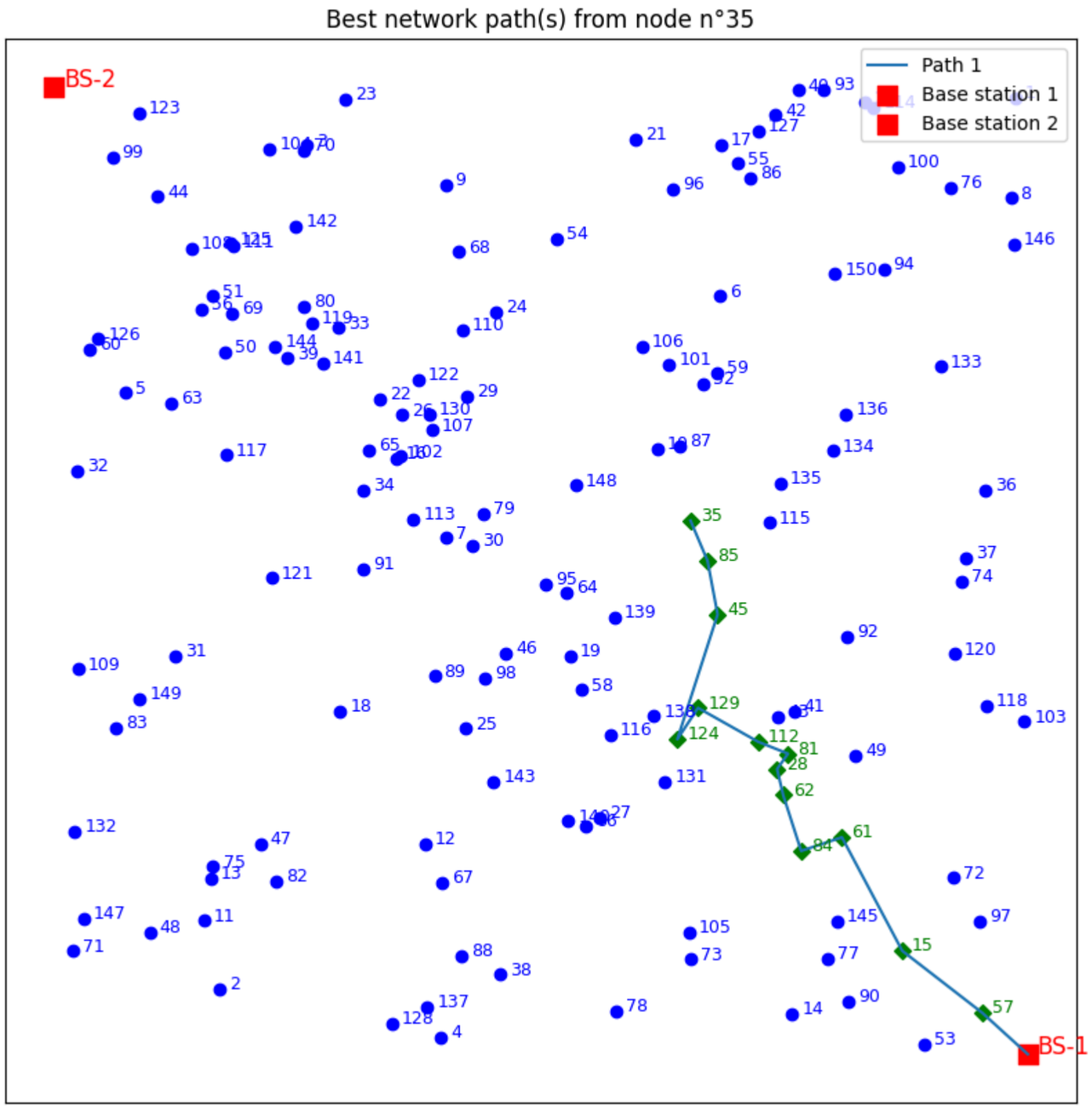
In [26]:
```
ACO = Ant_colony_algorithm(nodes=modified_nodes, alpha=1, beta=2, _evaporation_rate=.5, _pheromone_deposit=1, BS_stations=BS_stations, dist_matrix=dist_matrix, rate_matrix=rate_matrix)
```

In [27]:
```
best_path_ACO, all_best_paths_ACO, best_rate_ACO, best_latency_ACO, _, _, _ = ACO.start_aco(starting_node=35, iteration=300, ants=70, verbose=False)

print("Best path", all_best_paths_ACO)
print("Best latency", best_latency_ACO)
print("Best transmission rate", best_rate_ACO)
```

```
Best path [[35, 85, 45, 124, 129, 112, 81, 28, 62, 84, 61, 15, 57, 151]]
Best latency 390
Best transmission rate 4
```

In [ ]:
```
ACO.plot_multiple_route(all_best_paths_ACO)
```

```
Best path n°1: [35, 85, 45, 124, 129, 112, 81, 28, 62, 84, 61, 15, 57, 'BS-1']
```



Hybrid (GA + SA)

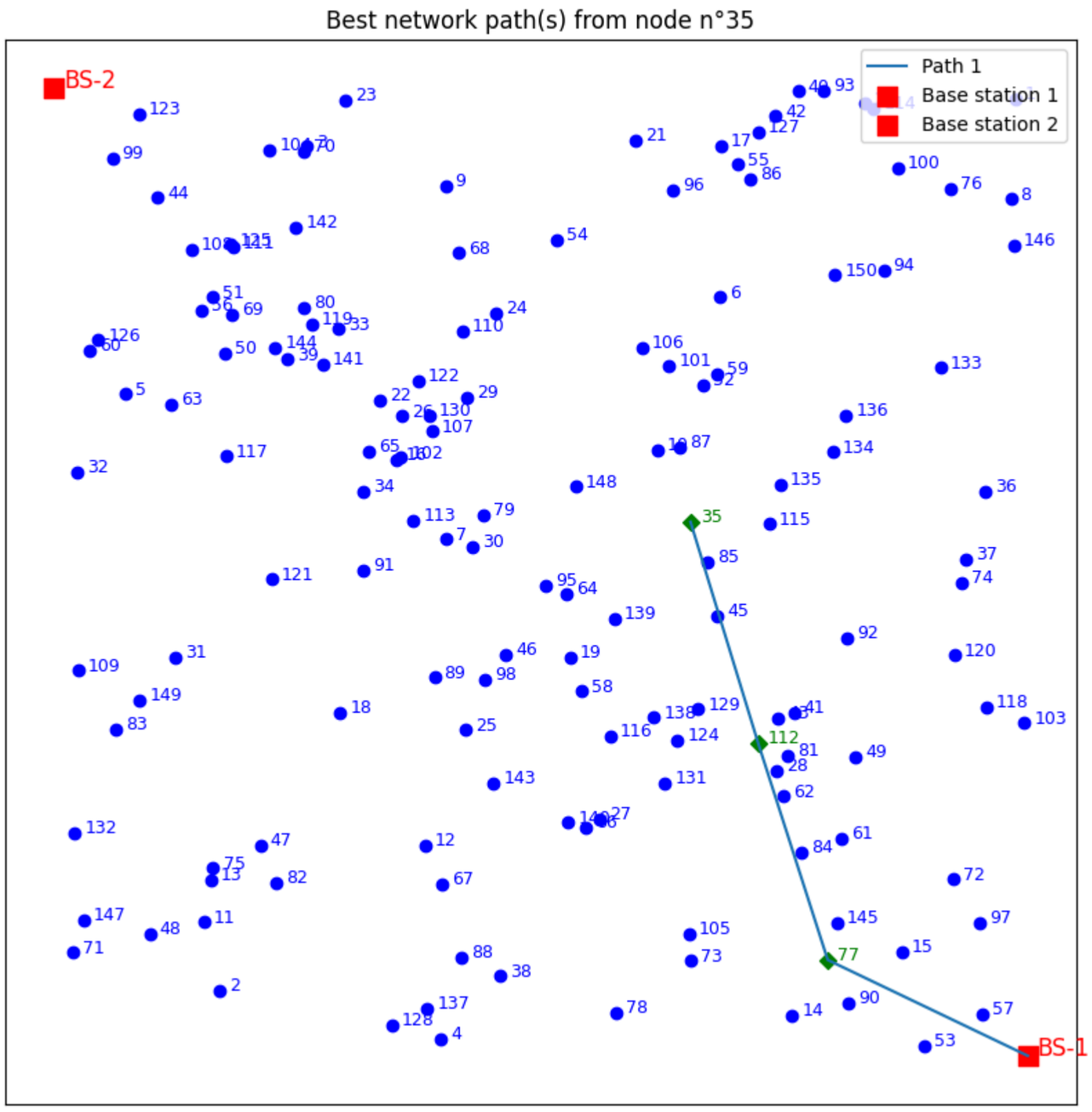
In [29]:
```
GA_SA = Hybrid_GA_with_SA(nodes=NODES, pop_size=9000, generations=300, mutation_rate=.6, cooling_rate=.96)
```

In [30]:
```
best_path_SA, best_latency_SA, best_transmission_rate_SA, all_best_paths_SA, _, _, _ = GA_SA.GA(starting_node=35, temperature=1000, verbose=False)
print("Best path", all_best_paths_SA)
print("Best latency", best_latency_SA)
print("Best transmission rate", best_transmission_rate_SA)
```

```
Best path [[35, 112, 77, 'BS-1']]
Best latency 90
Best transmission rate 2
```

In [ ]:
```
GA_SA.plot_multiple_route(all_best_paths_SA)
```

```
Best path n°1: [35, 112, 77, 'BS-1']
```



Discussion

- The 3 algorithms start from node 35.
- It appears that all the algorithms are finding a valid path.
- In comparison to other algorithms, ACO appears to find a more optimal path.
- It looks like the hybrid algorithm has a slightly better path than the Genetic algorithm.

Algorithms performance evaluation & comparison

- To compare the performances of the 3 algorithms the following metrics will be used:
  - **Solution**:
    - Transmission rate
    - Latency
  - **Fitness score over generations**
  - **Fitness standard deviation**
  - **Execution time**
  - **Number of generations after convergence.**

In [39]:
```
import time

GENERATIONS = 300
STARTING_NODE_1 = 14
STARTING_NODE_2 = 115
STARTING_NODE_3 = 29
```

- All the algorithms will have the same amount of iterations and the same starting nodes.
- 3 tests will be conducted with 3 different starting node.

In [ ]:
```
GA_metrics = {
    "Test1": {
        "paths": [],
        "latency": 0,
        "transmission": 0,
        "Fitness": [],
        "Fitness_std": [],
        "gen": 0,
        "Execution_time": 0
    },
    "Test2": {
        "paths": [],
        "latency": 0,
        "transmission": 0,
        "Fitness": [],
        "Fitness_std": [],
        "gen": 0,
        "Execution_time": 0
    },
    "Test3": {
        "paths": [],
        "latency": 0,
        "transmission": 0,
        "Fitness": [],
        "Fitness_std": [],
```

```
        "gen": 0,
        "Execution_time": 0
    }
}

ACO_metrics = {
    "Test1": {
        "paths": [],
        "latency": 0,
        "transmission": 0,
        "Fitness": [],
        "Fitness_std": [],
        "gen": 0,
        "Execution_time": 0
    },
    "Test2": {
        "paths": [],
        "latency": 0,
        "transmission": 0,
        "Fitness": [],
        "Fitness_std": [],
        "gen": 0,
        "Execution_time": 0
    },
    "Test3": {
        "paths": [],
        "latency": 0,
        "transmission": 0,
        "Fitness": [],
        "Fitness_std": [],
        "gen": 0,
        "Execution_time": 0
    }
}

HYBRID_metrics = {
    "Test1": {
        "paths": [],
        "latency": 0,
        "transmission": 0,
        "Fitness": [],
        "Fitness_std": [],
        "gen": 0,
        "Execution_time": 0
    },
    "Test2": {
        "paths": [],
        "latency": 0,
        "transmission": 0,
        "Fitness": [],
        "Fitness_std": [],
        "gen": 0,
        "Execution_time": 0
    },
    "Test3": {
        "paths": [],
        "latency": 0,
        "transmission": 0,
        "Fitness": [],
        "Fitness_std": [],
        "gen": 0,
        "Execution_time": 0
    }
}
```

Test 1 - (Starting Node : 14)

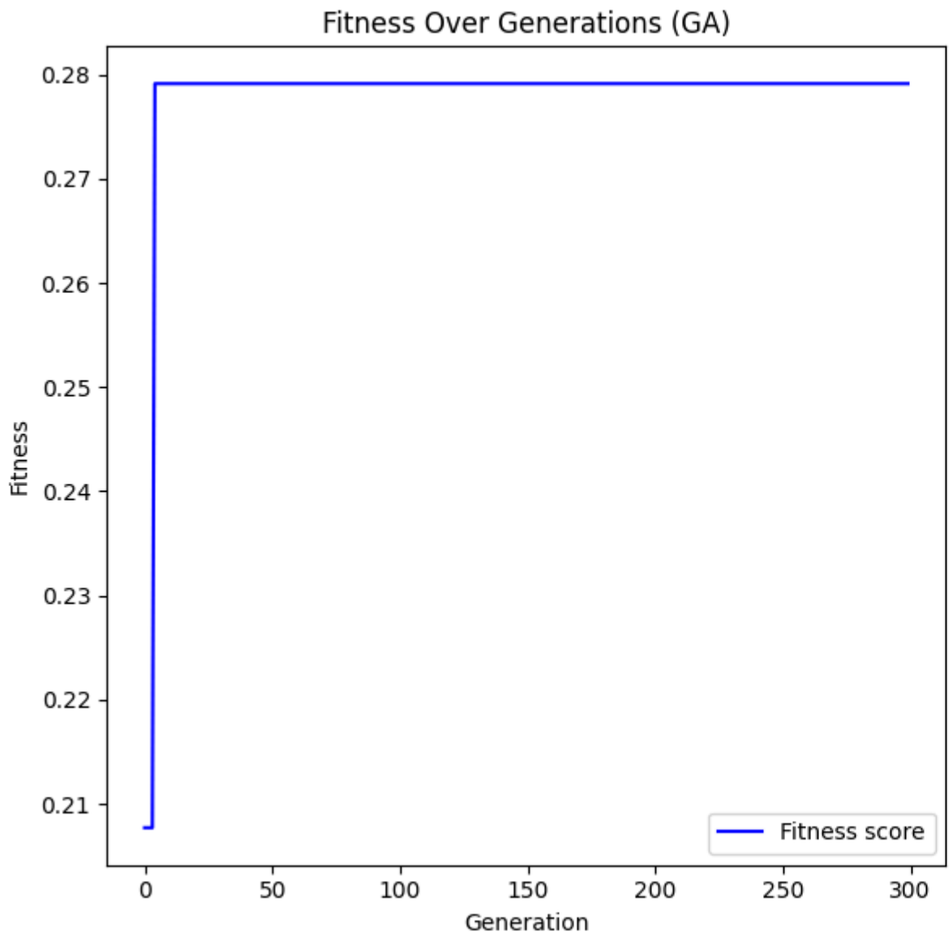
Discrete Genetic algorithm

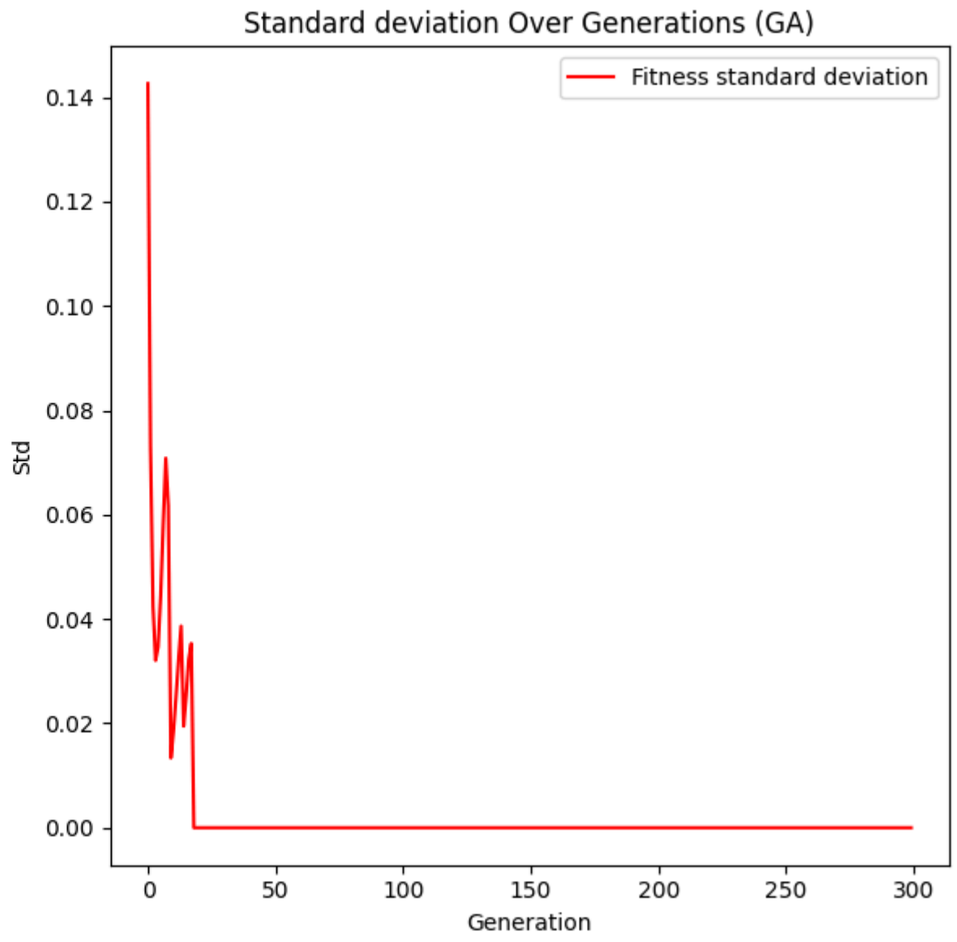
In [ ]:
```
DGA = Discrete_Genetic_algorithm(nodes=NODES, pop_size=9000, generations=GENERATIONS, mutation_rate=.6)

start_time = time.time()

best_path, best_latency, best_transmission_rate, all_best_paths, ga_fitness_score, ga_fitness_std, gen_num_GA = DGA.GA(starting_node=STARTING_NODE_1)

end_time = time.time()
execution_time = end_time - start_time

print("Total best paths", all_best_paths)
print("Best latency", best_latency)
print("Best transmission rate", best_transmission_rate)
print("Execution time", execution_time, "seconds")

GA_metrics["Test1"]["paths"] = all_best_paths
GA_metrics["Test1"]["latency"] = best_latency
GA_metrics["Test1"]["transmission"] = best_transmission_rate

GA_metrics["Test1"]["gen"] = gen_num_GA

GA_metrics["Test1"]["Fitness"] = ga_fitness_score
GA_metrics["Test1"]["Fitness_std"] = ga_fitness_std
GA_metrics["Test1"]["Execution_time"] = execution_time
```

```
Generation 0: Best latency = 60, Best rate = 3, Fitness std - (0.14257940789596182), Highest fitness for this gen - (0.20770676691729323)
Generation 1: Best latency = 60, Best rate = 3, Fitness std - (0.07307018593300713), Highest fitness for this gen - (0.20770676691729323)
Generation 2: Best latency = 60, Best rate = 3, Fitness std - (0.04241752154001251), Highest fitness for this gen - (0.20770676691729323)
Generation 3: Best latency = 60, Best rate = 3, Fitness std - (0.03206789139630084), Highest fitness for this gen - (0.20770676691729323)
Generation 4: Best latency = 60, Best rate = 4, Fitness std - (0.03469951340055798), Highest fitness for this gen - (0.2791353383458646)
Generation 5: Best latency = 60, Best rate = 4, Fitness std - (0.044445894500381285), Highest fitness for this gen - (0.2791353383458646)
Generation 6: Best latency = 60, Best rate = 4, Fitness std - (0.058325997952352264), Highest fitness for this gen - (0.2791353383458646)
Generation 7: Best latency = 60, Best rate = 4, Fitness std - (0.07080567049837454), Highest fitness for this gen - (0.2791353383458646)
Generation 8: Best latency = 60, Best rate = 4, Fitness std - (0.061603207962691975), Highest fitness for this gen - (0.2791353383458646)
Generation 9: Best latency = 60, Best rate = 4, Fitness std - (0.013371066513365419), Highest fitness for this gen - (0.2791353383458646)
Generation 10: Best latency = 60, Best rate = 4, Fitness std - (0.018402922950498623), Highest fitness for this gen - (0.2791353383458646)
Generation 11: Best latency = 60, Best rate = 4, Fitness std - (0.025005785202940816), Highest fitness for this gen - (0.2791353383458646)
Generation 12: Best latency = 60, Best rate = 4, Fitness std - (0.03216388063795429), Highest fitness for this gen - (0.2791353383458646)
Generation 13: Best latency = 60, Best rate = 4, Fitness std - (0.03863535770311139), Highest fitness for this gen - (0.2791353383458646)
Generation 14: Best latency = 60, Best rate = 4, Fitness std - (0.01947556877761435), Highest fitness for this gen - (0.2791353383458646)
Generation 15: Best latency = 60, Best rate = 4, Fitness std - (0.02537290167925968), Highest fitness for this gen - (0.2791353383458646)
Generation 16: Best latency = 60, Best rate = 4, Fitness std - (0.03228974346165835), Highest fitness for this gen - (0.2791353383458646)
Generation 17: Best latency = 60, Best rate = 4, Fitness std - (0.035298520331685825), Highest fitness for this gen - (0.2791353383458646)
Generation 18: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 19: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 20: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 21: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 22: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 23: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 24: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 25: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 26: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 27: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 28: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 29: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 30: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 31: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 32: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 33: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 34: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 35: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 36: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 37: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 38: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 39: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 40: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 41: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 42: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 43: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 44: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 45: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 46: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 47: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 48: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 49: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 50: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 51: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 52: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 53: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 54: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 55: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 56: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 57: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 58: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 59: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 60: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 61: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 62: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 63: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 64: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 65: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 66: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 67: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 68: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 69: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 70: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 71: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 72: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 73: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 74: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 75: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 76: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 77: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 78: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 79: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 80: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 81: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 82: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 83: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 84: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 85: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 86: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 87: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 88: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 89: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 90: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 91: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 92: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 93: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 94: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 95: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 96: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 97: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 98: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 99: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 100: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 101: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 102: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 103: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 104: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 105: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 106: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 107: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 108: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 109: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 110: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 111: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 112: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 113: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 114: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 115: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 116: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 117: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 118: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 119: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 120: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 121: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 122: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 123: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 124: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 125: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
```

```
Generation 126: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 127: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 128: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 129: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 130: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 131: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 132: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 133: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 134: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 135: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 136: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 137: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 138: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 139: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 140: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 141: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 142: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 143: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 144: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 145: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 146: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 147: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 148: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 149: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 150: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 151: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 152: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 153: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 154: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 155: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 156: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 157: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 158: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 159: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 160: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 161: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 162: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 163: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 164: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 165: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 166: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 167: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 168: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 169: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 170: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 171: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 172: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 173: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 174: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 175: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 176: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 177: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 178: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 179: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 180: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 181: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 182: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 183: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 184: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 185: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 186: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 187: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 188: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 189: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 190: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 191: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 192: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 193: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 194: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 195: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 196: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 197: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 198: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 199: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 200: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 201: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 202: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 203: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 204: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 205: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 206: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 207: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 208: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 209: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 210: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 211: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 212: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 213: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 214: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 215: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 216: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 217: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 218: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 219: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 220: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 221: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 222: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 223: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 224: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 225: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 226: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 227: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 228: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 229: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 230: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 231: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 232: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 233: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 234: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 235: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 236: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 237: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 238: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 239: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 240: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 241: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 242: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 243: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 244: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 245: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 246: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 247: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 248: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 249: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 250: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 251: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
```

```
Generation 252: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 253: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 254: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 255: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 256: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 257: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 258: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 259: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 260: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 261: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 262: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 263: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 264: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 265: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 266: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 267: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 268: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 269: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 270: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 271: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 272: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 273: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 274: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 275: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 276: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 277: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 278: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 279: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 280: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 281: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 282: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 283: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 284: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 285: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 286: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 287: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 288: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 289: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 290: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 291: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 292: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 293: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 294: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 295: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 296: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 297: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 298: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
Generation 299: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17), Highest fitness for this gen - (0.2791353383458646)
```





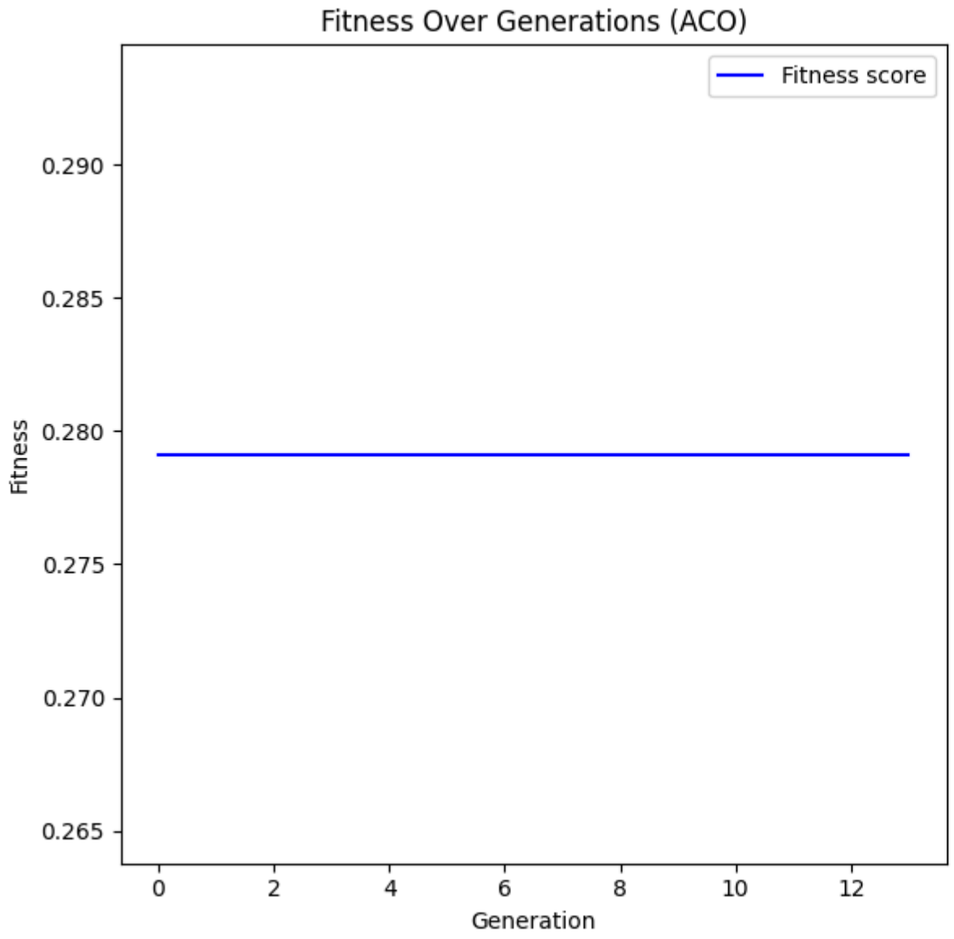

```
Total best paths [[14, 53, 'BS-1']]
Best latency 60
Best transmission rate 4
Execution time 19.552256107330322 seconds
```

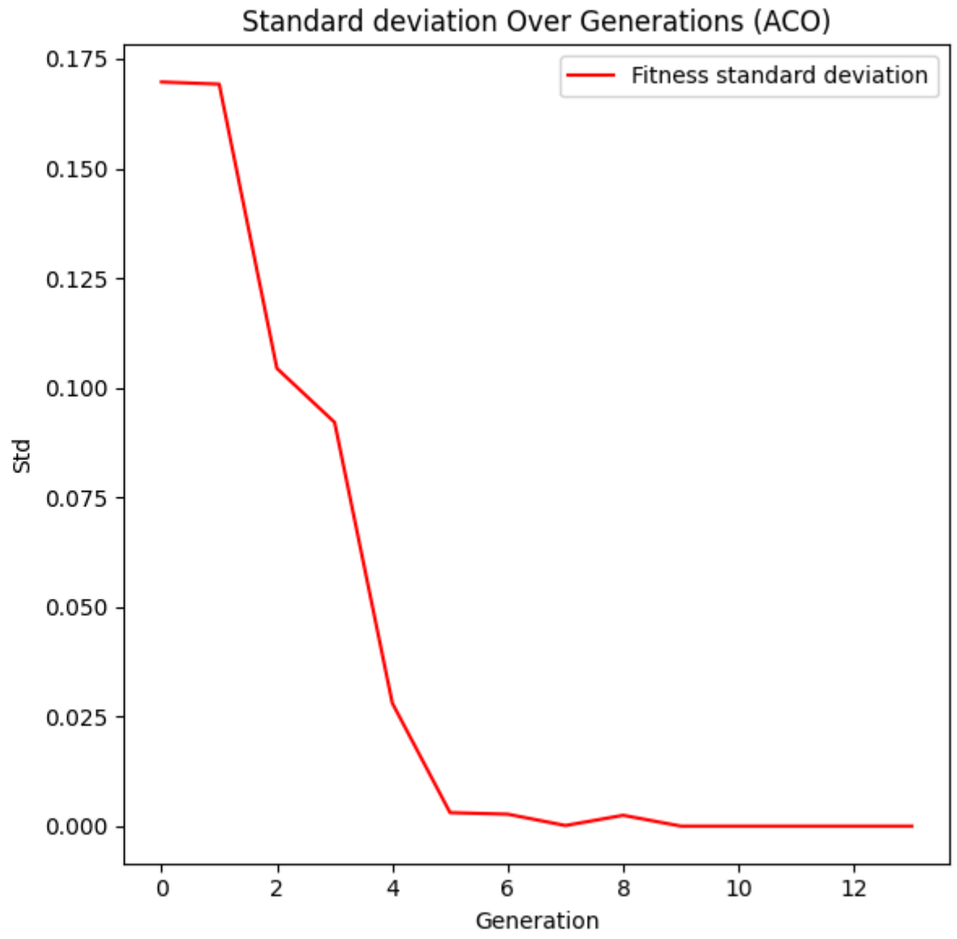
Ant colony algorithm

In [42]:
```
BS_stations = base_stations.copy()
modified_nodes = NODES.copy()
modified_nodes = np.insert(modified_nodes, len(modified_nodes), list(BS_stations.values()), axis=0)

dist_matrix, base_stations_dist_matrix = calculate_distances(modified_nodes, destination_coordinate)
rate_matrix = np.vectorize(transmission_rate)(dist_matrix)
```

In [ ]:
```
ACO = Ant_colony_algorithm(nodes=modified_nodes, alpha=1, beta=2, _evaporation_rate=.5, _pheromone_deposit=1, BS_stations=BS_stations, dist_matrix=dist_matrix, rate_matrix=rate_matrix)

start_time = time.time()

best_path_ACO, all_best_paths_ACO, best_rate_ACO, best_latency_ACO, aco_fitness_score, aco_fitness_std, gen_num_ACO = ACO.start_aco(starting_node=STARTING_NODE_1, iteration=GENERATIONS, ants=

end_time = time.time()
execution_time = end_time - start_time

print("Best path", all_best_paths_ACO)
print("Best latency", best_latency_ACO)
print("Best transmission rate", best_rate_ACO)
print("Execution time", execution_time, "seconds")

ACO_metrics["Test1"]["paths"] = all_best_paths_ACO
ACO_metrics["Test1"]["latency"] = best_latency_ACO
ACO_metrics["Test1"]["transmission"] = best_rate_ACO

ACO_metrics["Test1"]["gen"] = gen_num_ACO

ACO_metrics["Test1"]["Fitness"] = aco_fitness_score
ACO_metrics["Test1"]["Fitness_std"] = aco_fitness_std
ACO_metrics["Test1"]["Execution_time"] = execution_time
```

```
Iteration 0: Best rate - 4, Best latency - 60, Fitness std - (0.16971711764761066), Best Fitness = 0.2791353383458646
Iteration 1: Best rate - 4, Best latency - 60, Fitness std - (0.16922832870162766), Best Fitness = 0.2791353383458646
Iteration 2: Best rate - 4, Best latency - 60, Fitness std - (0.10440773737599791), Best Fitness = 0.2791353383458646
Iteration 3: Best rate - 4, Best latency - 60, Fitness std - (0.09208392993697236), Best Fitness = 0.2791353383458646
Iteration 4: Best rate - 4, Best latency - 60, Fitness std - (0.02805163805641759), Best Fitness = 0.2791353383458646
Iteration 5: Best rate - 4, Best latency - 60, Fitness std - (0.003071798967184109), Best Fitness = 0.2791353383458646
Iteration 6: Best rate - 4, Best latency - 60, Fitness std - (0.002715888973106316), Best Fitness = 0.2791353383458646
Iteration 7: Best rate - 4, Best latency - 60, Fitness std - (0.0001315789473684203), Best Fitness = 0.2791353383458646
Iteration 8: Best rate - 4, Best latency - 60, Fitness std - (0.002462352206265093), Best Fitness = 0.2791353383458646
Iteration 9: Best rate - 4, Best latency - 60, Fitness std - (0.0), Best Fitness = 0.2791353383458646
Iteration 10: Best rate - 4, Best latency - 60, Fitness std - (0.0), Best Fitness = 0.2791353383458646
Iteration 11: Best rate - 4, Best latency - 60, Fitness std - (0.0), Best Fitness = 0.2791353383458646
Iteration 12: Best rate - 4, Best latency - 60, Fitness std - (0.0), Best Fitness = 0.2791353383458646
Iteration 13: Best rate - 4, Best latency - 60, Fitness std - (0.0), Best Fitness = 0.2791353383458646
Algorithmm converged
```






```
Best path [[14, 53, 151]]
Best latency 60
Best transmission rate 4
Execution time 1.809568166732788 seconds
```

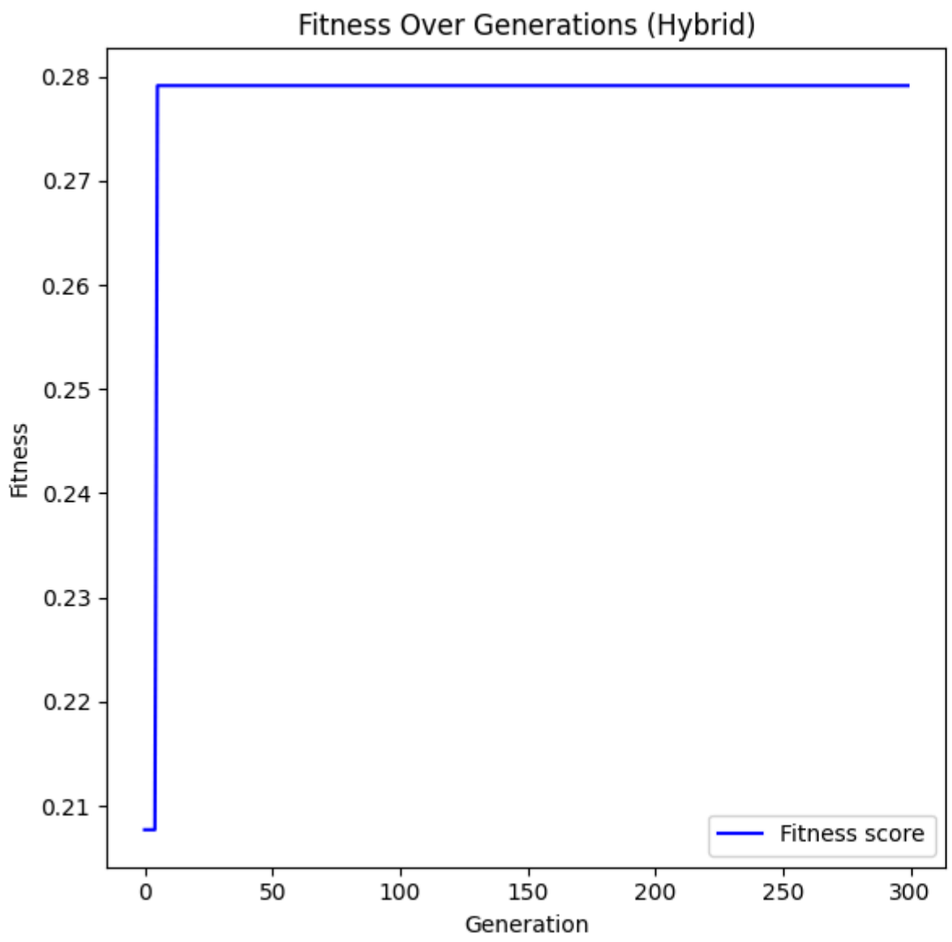
Hybrid algorithm (GA + SA)

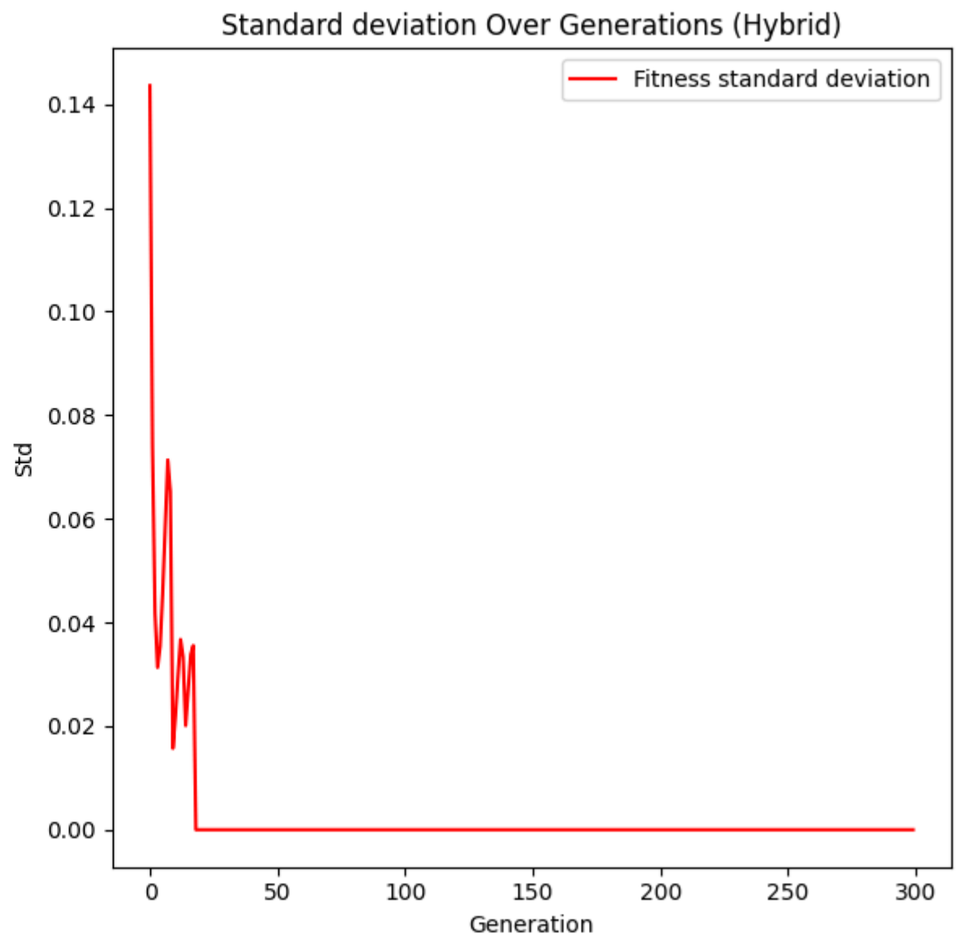
In [ ]:
```
GA_SA = Hybrid_GA_with_SA(nodes=NODES, pop_size=9000, generations=GENERATIONS, mutation_rate=.6, cooling_rate=.96)

start_time = time.time()

best_path_SA, best_latency_SA, best_transmission_rate_SA, all_best_paths_SA, hybrid_fitness_score, hybrid_fitness_std, gen_num_SA = GA_SA.GA(starting_node=STARTING_NODE_1, temperature=1000)

end_time = time.time()
execution_time = end_time - start_time

print("Best path", all_best_paths_SA)
print("Best latency", best_latency_SA)
print("Best transmission rate", best_transmission_rate_SA)
print("Execution time", execution_time, "seconds")

HYBRID_metrics["Test1"]["paths"] = all_best_paths_SA
HYBRID_metrics["Test1"]["latency"] = best_latency_SA
HYBRID_metrics["Test1"]["transmission"] = best_transmission_rate_SA

HYBRID_metrics["Test1"]["gen"] = gen_num_SA

HYBRID_metrics["Test1"]["Fitness"] = hybrid_fitness_score
HYBRID_metrics["Test1"]["Fitness_std"] = hybrid_fitness_std
HYBRID_metrics["Test1"]["Execution_time"] = execution_time
```

```
Generation 0: Best latency = 60, Best rate = 3, Fitness std - (0.14359513692589687) Highest fitness for this gen - (0.20770676691729323), Temperature - 1000.000
Generation 1: Best latency = 60, Best rate = 3, Fitness std - (0.07409426576621099) Highest fitness for this gen - (0.20770676691729323), Temperature - 960.000
Generation 2: Best latency = 60, Best rate = 3, Fitness std - (0.04179794490720078) Highest fitness for this gen - (0.20770676691729323), Temperature - 921.600
Generation 3: Best latency = 60, Best rate = 3, Fitness std - (0.031306829771457906) Highest fitness for this gen - (0.20770676691729323), Temperature - 884.736
Generation 4: Best latency = 60, Best rate = 3, Fitness std - (0.03537549647271452) Highest fitness for this gen - (0.20770676691729323), Temperature - 849.347
Generation 5: Best latency = 60, Best rate = 4, Fitness std - (0.0456661068353646) Highest fitness for this gen - (0.2791353383458646), Temperature - 815.373
Generation 6: Best latency = 60, Best rate = 4, Fitness std - (0.05889753823978558) Highest fitness for this gen - (0.2791353383458646), Temperature - 782.758
Generation 7: Best latency = 60, Best rate = 4, Fitness std - (0.07136637104484615) Highest fitness for this gen - (0.2791353383458646), Temperature - 751.447
Generation 8: Best latency = 60, Best rate = 4, Fitness std - (0.06528612080522718) Highest fitness for this gen - (0.2791353383458646), Temperature - 721.390
Generation 9: Best latency = 60, Best rate = 4, Fitness std - (0.01570813830244534) Highest fitness for this gen - (0.2791353383458646), Temperature - 692.534
Generation 10: Best latency = 60, Best rate = 4, Fitness std - (0.021887308464063292) Highest fitness for this gen - (0.2791353383458646), Temperature - 664.833
Generation 11: Best latency = 60, Best rate = 4, Fitness std - (0.02963144483404505) Highest fitness for this gen - (0.2791353383458646), Temperature - 638.239
Generation 12: Best latency = 60, Best rate = 4, Fitness std - (0.03672629166927833) Highest fitness for this gen - (0.2791353383458646), Temperature - 612.710
Generation 13: Best latency = 60, Best rate = 4, Fitness std - (0.033317442924179874) Highest fitness for this gen - (0.2791353383458646), Temperature - 588.201
Generation 14: Best latency = 60, Best rate = 4, Fitness std - (0.02009809912909099) Highest fitness for this gen - (0.2791353383458646), Temperature - 564.673
Generation 15: Best latency = 60, Best rate = 4, Fitness std - (0.026738181295075592) Highest fitness for this gen - (0.2791353383458646), Temperature - 542.086
Generation 16: Best latency = 60, Best rate = 4, Fitness std - (0.03381807809670181) Highest fitness for this gen - (0.2791353383458646), Temperature - 520.403
Generation 17: Best latency = 60, Best rate = 4, Fitness std - (0.035540421549978364) Highest fitness for this gen - (0.2791353383458646), Temperature - 499.587
Generation 18: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 479.603
Generation 19: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 460.419
Generation 20: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 442.002
Generation 21: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 424.322
Generation 22: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 407.349
Generation 23: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 391.055
Generation 24: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 375.413
Generation 25: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 360.397
Generation 26: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 345.981
Generation 27: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 332.142
Generation 28: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 318.856
Generation 29: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 306.102
Generation 30: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 293.858
Generation 31: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 282.103
Generation 32: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 270.819
Generation 33: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 259.986
Generation 34: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 249.587
Generation 35: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 239.603
Generation 36: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 230.019
Generation 37: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 220.819
Generation 38: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 211.986
Generation 39: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 203.506
Generation 40: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 195.366
Generation 41: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 187.552
Generation 42: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 180.049
Generation 43: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 172.847
Generation 44: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 165.934
Generation 45: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 159.296
Generation 46: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 152.924
Generation 47: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 146.807
Generation 48: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 140.935
Generation 49: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 135.298
Generation 50: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 129.886
Generation 51: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 124.690
Generation 52: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 119.703
Generation 53: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 114.915
Generation 54: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 110.318
Generation 55: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 105.905
Generation 56: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 101.669
Generation 57: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 97.602
Generation 58: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 93.698
Generation 59: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 89.950
Generation 60: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 86.352
Generation 61: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 82.898
Generation 62: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 79.582
Generation 63: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 76.399
Generation 64: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 73.343
Generation 65: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 70.409
Generation 66: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 67.593
Generation 67: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 64.889
Generation 68: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 62.294
Generation 69: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 59.802
Generation 70: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 57.410
Generation 71: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 55.113
Generation 72: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 52.909
Generation 73: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 50.793
Generation 74: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 48.761
Generation 75: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 46.810
Generation 76: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 44.938
Generation 77: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 43.140
Generation 78: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 41.415
Generation 79: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 39.758
Generation 80: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 38.168
Generation 81: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 36.641
Generation 82: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 35.176
Generation 83: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 33.769
Generation 84: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 32.418
Generation 85: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 31.121
Generation 86: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 29.876
Generation 87: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 28.681
Generation 88: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 27.534
Generation 89: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 26.433
Generation 90: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 25.375
Generation 91: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 24.360
Generation 92: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 23.386
Generation 93: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 22.450
Generation 94: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 21.552
Generation 95: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 20.690
Generation 96: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 19.863
Generation 97: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 19.068
Generation 98: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 18.305
Generation 99: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 17.573
Generation 100: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 16.870
Generation 101: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 16.196
Generation 102: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 15.548
Generation 103: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 14.926
Generation 104: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 14.329
Generation 105: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 13.756
Generation 106: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 13.205
Generation 107: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 12.677
Generation 108: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 12.170
Generation 109: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 11.683
Generation 110: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 11.216
Generation 111: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 10.767
Generation 112: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 10.337
Generation 113: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 9.923
Generation 114: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 9.526
Generation 115: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 9.145
Generation 116: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 8.779
Generation 117: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 8.428
Generation 118: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 8.091
Generation 119: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 7.767
Generation 120: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 7.457
Generation 121: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 7.158
Generation 122: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 6.872
Generation 123: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 6.597
Generation 124: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 6.333
Generation 125: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 6.080
```

```
Generation 126: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 5.837
Generation 127: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 5.603
Generation 128: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 5.379
Generation 129: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 5.164
Generation 130: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 4.957
Generation 131: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 4.759
Generation 132: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 4.569
Generation 133: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 4.386
Generation 134: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 4.211
Generation 135: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 4.042
Generation 136: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 3.881
Generation 137: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 3.725
Generation 138: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 3.576
Generation 139: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 3.433
Generation 140: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 3.296
Generation 141: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 3.164
Generation 142: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 3.037
Generation 143: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.916
Generation 144: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.799
Generation 145: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.687
Generation 146: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.580
Generation 147: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.477
Generation 148: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.378
Generation 149: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.283
Generation 150: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.191
Generation 151: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.104
Generation 152: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 2.019
Generation 153: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.939
Generation 154: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.861
Generation 155: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.787
Generation 156: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.715
Generation 157: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.647
Generation 158: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.581
Generation 159: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.517
Generation 160: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.457
Generation 161: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.399
Generation 162: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.343
Generation 163: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.289
Generation 164: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.237
Generation 165: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.188
Generation 166: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.140
Generation 167: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.095
Generation 168: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.051
Generation 169: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 1.009
Generation 170: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.969
Generation 171: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.930
Generation 172: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.893
Generation 173: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.857
Generation 174: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.823
Generation 175: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.790
Generation 176: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.758
Generation 177: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.728
Generation 178: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.699
Generation 179: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.671
Generation 180: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.644
Generation 181: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.618
Generation 182: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.593
Generation 183: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.570
Generation 184: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.547
Generation 185: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.525
Generation 186: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.504
Generation 187: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.484
Generation 188: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.465
Generation 189: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.446
Generation 190: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.428
Generation 191: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.411
Generation 192: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.395
Generation 193: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.379
Generation 194: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.364
Generation 195: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.349
Generation 196: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.335
Generation 197: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.322
Generation 198: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.309
Generation 199: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.296
Generation 200: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.285
Generation 201: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.273
Generation 202: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.262
Generation 203: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.252
Generation 204: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.242
Generation 205: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.232
Generation 206: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.223
Generation 207: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.214
Generation 208: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.205
Generation 209: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.197
Generation 210: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.189
Generation 211: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.182
Generation 212: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.174
Generation 213: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.167
Generation 214: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.161
Generation 215: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.154
Generation 216: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.148
Generation 217: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.142
Generation 218: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.136
Generation 219: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.131
Generation 220: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.126
Generation 221: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.121
Generation 222: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.116
Generation 223: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.111
Generation 224: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.107
Generation 225: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.103
Generation 226: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.098
Generation 227: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.095
Generation 228: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.091
Generation 229: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.087
Generation 230: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.084
Generation 231: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.080
Generation 232: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.077
Generation 233: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.074
Generation 234: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.071
Generation 235: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.068
Generation 236: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.065
Generation 237: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.063
Generation 238: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.060
Generation 239: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.058
Generation 240: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.056
Generation 241: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.053
Generation 242: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.051
Generation 243: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.049
Generation 244: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.047
Generation 245: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.045
Generation 246: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.044
Generation 247: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.042
Generation 248: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.040
Generation 249: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.039
Generation 250: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.037
Generation 251: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.035
```

```
Generation 252: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.034
Generation 253: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.033
Generation 254: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.031
Generation 255: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.030
Generation 256: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.029
Generation 257: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.028
Generation 258: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.027
Generation 259: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.026
Generation 260: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.025
Generation 261: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.024
Generation 262: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.023
Generation 263: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.022
Generation 264: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.021
Generation 265: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.020
Generation 266: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.019
Generation 267: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.018
Generation 268: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.018
Generation 269: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.017
Generation 270: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.016
Generation 271: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.016
Generation 272: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.015
Generation 273: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.014
Generation 274: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.014
Generation 275: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.013
Generation 276: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.013
Generation 277: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.012
Generation 278: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.012
Generation 279: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.011
Generation 280: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.011
Generation 281: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.010
Generation 282: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.010
Generation 283: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.010
Generation 284: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.009
Generation 285: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.009
Generation 286: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.009
Generation 287: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.008
Generation 288: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.008
Generation 289: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.008
Generation 290: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.007
Generation 291: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.007
Generation 292: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.007
Generation 293: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.006
Generation 294: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.006
Generation 295: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.006
Generation 296: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.006
Generation 297: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.005
Generation 298: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.005
Generation 299: Best latency = 60, Best rate = 4, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.2791353383458646), Temperature - 0.005
```

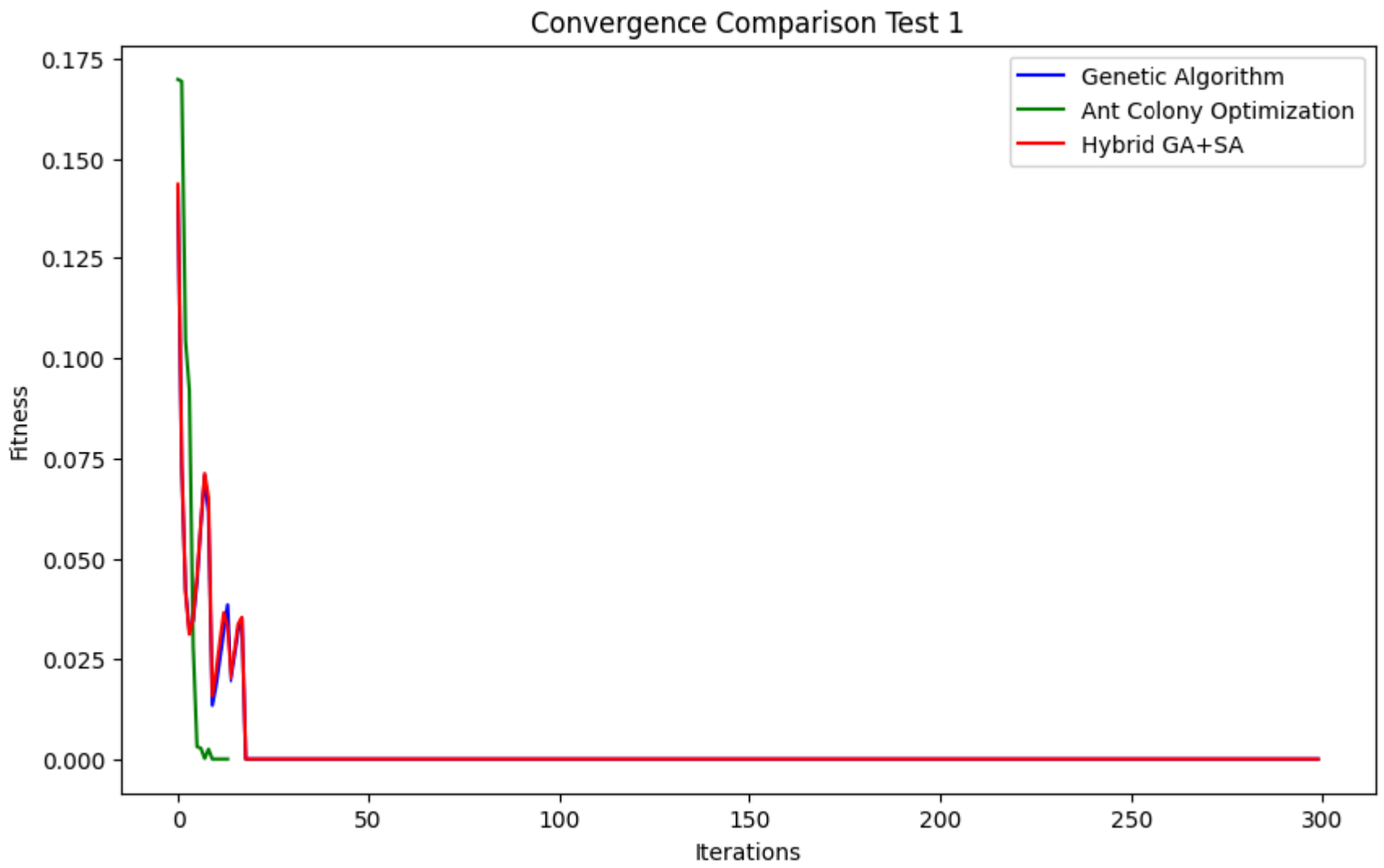





```
Best path [[14, 53, 'BS-1']]
Best latency 60
Best transmission rate 4
Execution time 28.017407178878784 seconds
```

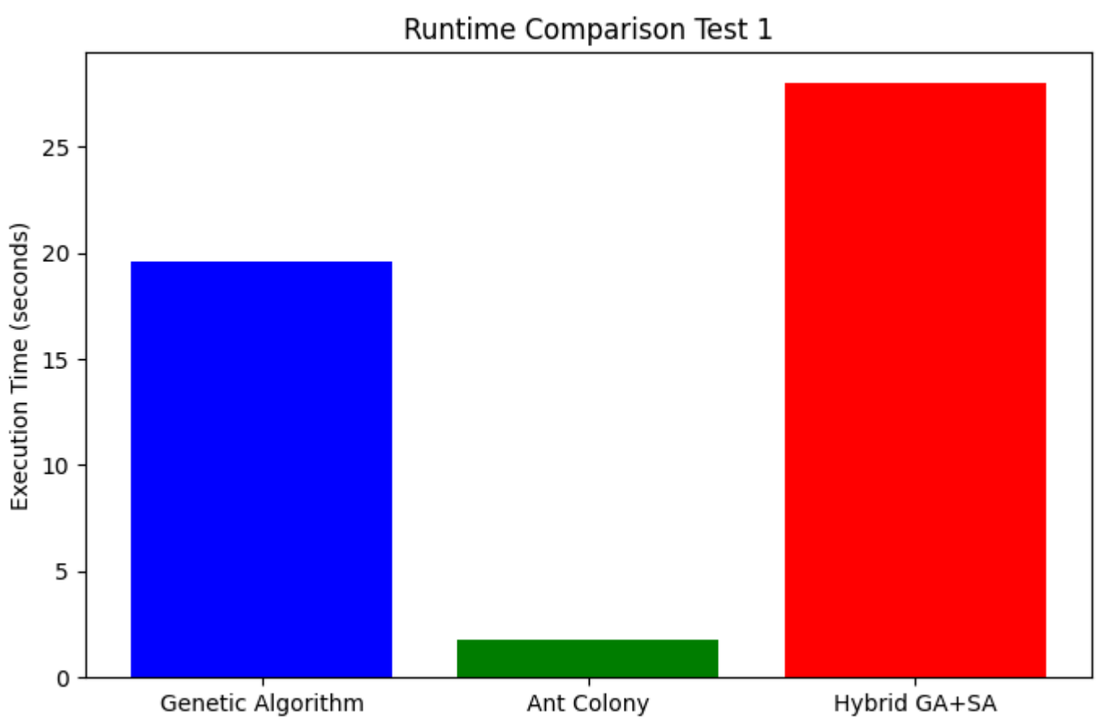
Convergence Curve (Test 1)

In [ ]:
```
plt.figure(figsize=(10, 6))
plt.plot(GA_metrics["Test1"]["Fitness_std"], label='Genetic Algorithm', color='blue')
plt.plot(ACO_metrics["Test1"]["Fitness_std"], label='Ant Colony Optimization', color='green')
plt.plot(HYBRID_metrics["Test1"]["Fitness_std"], label='Hybrid GA+SA', color='red')
plt.xlabel('Iterations')
plt.ylabel('Fitness')
plt.title('Convergence Comparison Test 1')
plt.legend()
plt.show()
```



Runtime Comparison (Test 1)

In [ ]:
```
algorithms = ['Genetic Algorithm', 'Ant Colony', 'Hybrid GA+SA']
runtimes = [
    GA_metrics["Test1"]["Execution_time"],
    ACO_metrics["Test1"]["Execution_time"],
    HYBRID_metrics["Test1"]["Execution_time"]
]

plt.figure(figsize=(8, 5))
plt.bar(algorithms, runtimes, color=['blue', 'green', 'red'])
plt.ylabel('Execution Time (seconds)')
plt.title('Runtime Comparison Test 1')
plt.show()
```



Results Comparison (Test 1)

In [ ]:
```
algorithms = ["Genetic algorithm", "Ant colony algorithm", "Hybrid GA + SA"]
best_paths = [
    GA_metrics["Test1"]["paths"],
    ACO_metrics["Test1"]["paths"],
    HYBRID_metrics["Test1"]["paths"]
]
best_latencies = [
    GA_metrics["Test1"]["latency"],
    ACO_metrics["Test1"]["latency"],
    HYBRID_metrics["Test1"]["latency"]
]
best_transmissions = [
    GA_metrics["Test1"]["transmission"],
    ACO_metrics["Test1"]["transmission"],
    HYBRID_metrics["Test1"]["transmission"]
]
final_score = [
    f"{GA_metrics['Test1']['Fitness'][-1]:.3f}",
    f"{ACO_metrics['Test1']['Fitness'][-1]:.3f}",
    f"{HYBRID_metrics['Test1']['Fitness'][-1]:.3f}"
]
final_fitness_std = [
    f"{GA_metrics['Test1']['Fitness_std'][-1]:.3f}",
    f"{ACO_metrics['Test1']['Fitness_std'][-1]:.3f}",
    f"{HYBRID_metrics['Test1']['Fitness_std'][-1]:.3f}",
]
all_gen_num = [
    GA_metrics["Test1"]["gen"],
    ACO_metrics["Test1"]["gen"],
    HYBRID_metrics["Test1"]["gen"],
]
all_execution_time = [
    f"{GA_metrics['Test1']['Execution_time']:.2f}",
    f"{ACO_metrics['Test1']['Execution_time']:.2f}",
    f"{HYBRID_metrics['Test1']['Execution_time']:.2f}",
]

index = [x for x in range(len(algorithms))]

pd.DataFrame({
    'Algorithms':algorithms,
    'Best path(s)':best_paths,
    'Best latency':best_latencies,
    'Best transmission rate':best_transmissions,
    'Final fitness score':final_score,
    'Final fitness std':final_fitness_std,
    "Number of iteration to converge": all_gen_num,
```

```
    "Execution time (seconds)": all_execution_time
    },index=index)
```

Out[ ]:

	Algorithms	Best path(s)	Best latency	Best transmission rate	Final fitness score	Final fitness std	Number of iteration to converge	Execution time (seconds)
0	Genetic algorithm	[[14, 53, BS-1]]	60	4	0.279	0.000	300	19.55
1	Ant colony algorithm	[[14, 53, 151]]	60	4	0.279	0.000	14	1.81
2	Hybrid GA + SA	[[14, 53, BS-1]]	60	4	0.279	0.000	300	28.02

Test 2 - (Starting Node : 115)

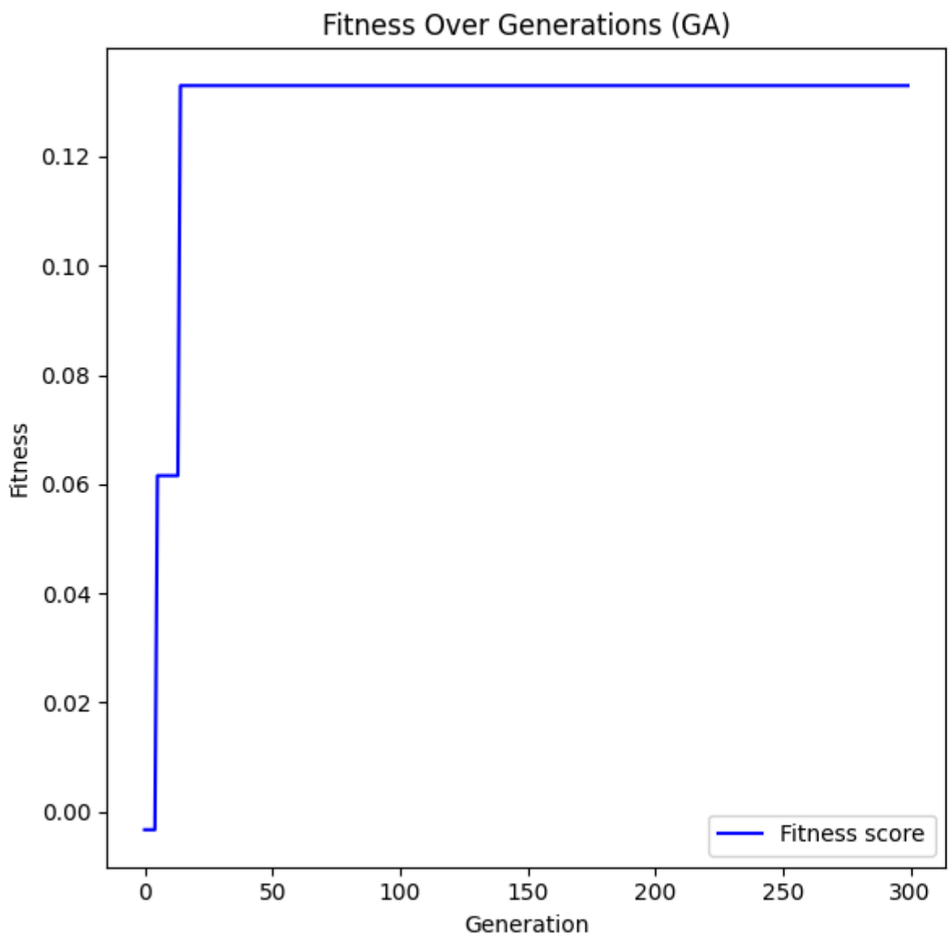
Discrete Genetic Algorithm

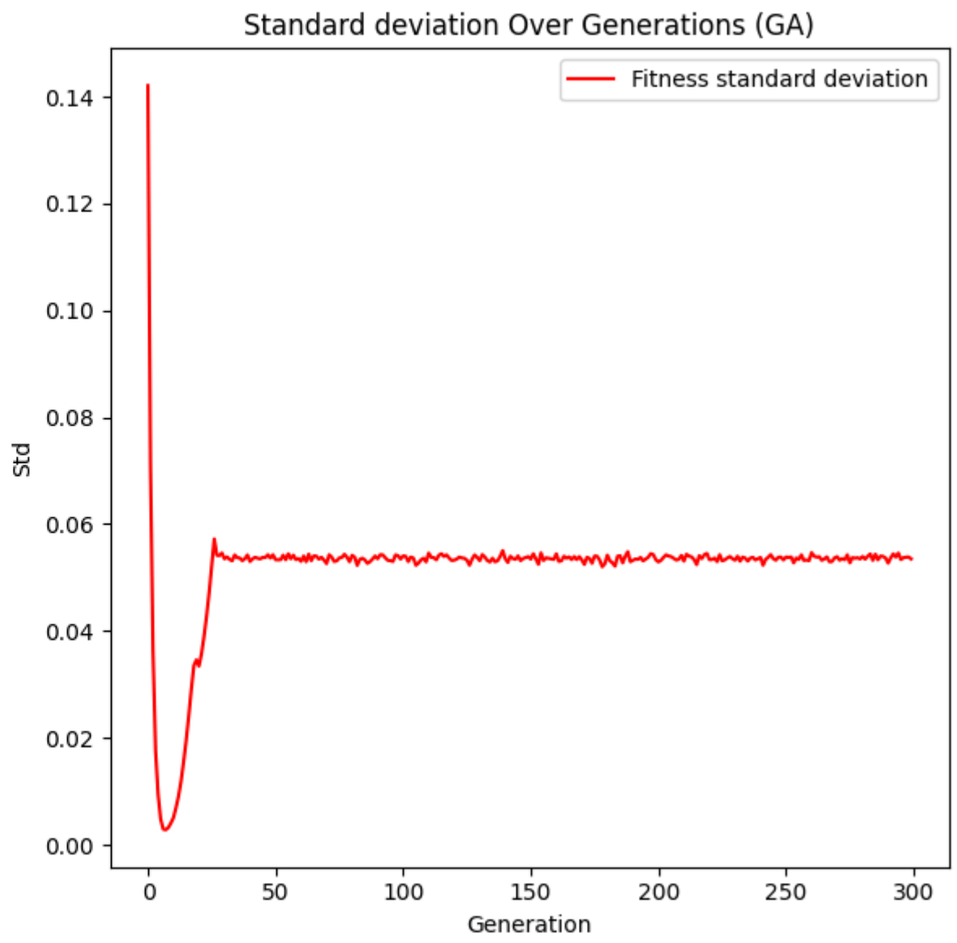
In [ ]:
```
DGA = Discrete_Genetic_algorithm(nodes=NODES, pop_size=9000, generations=GENERATIONS, mutation_rate=.6)

start_time = time.time()

best_path, best_latency, best_transmission_rate, all_best_paths, ga_fitness_score, ga_fitness_std, gen_num_GA = DGA.GA(starting_node=STARTING_NODE_2)

end_time = time.time()
execution_time = end_time - start_time

print("Total best paths", all_best_paths)
print("Best latency", best_latency)
print("Best transmission rate", best_transmission_rate)
print("Execution time", execution_time, "seconds")

GA_metrics["Test2"]["paths"] = all_best_paths
GA_metrics["Test2"]["latency"] = best_latency
GA_metrics["Test2"]["transmission"] = best_transmission_rate

GA_metrics["Test2"]["gen"] = gen_num_GA

GA_metrics["Test2"]["Fitness"] = ga_fitness_score
GA_metrics["Test2"]["Fitness_std"] = ga_fitness_std
GA_metrics["Test2"]["Execution_time"] = execution_time
```

```
Generation 0: Best latency = inf, Best rate = 0, Fitness std - (0.1420839473711371), Highest fitness for this gen - (-0.003289473684210526)
Generation 1: Best latency = inf, Best rate = 0, Fitness std - (0.07094439378053315), Highest fitness for this gen - (-0.003289473684210526)
Generation 2: Best latency = inf, Best rate = 0, Fitness std - (0.036244175340930475), Highest fitness for this gen - (-0.003289473684210526)
Generation 3: Best latency = inf, Best rate = 0, Fitness std - (0.017633983290503332), Highest fitness for this gen - (-0.003289473684210526)
Generation 4: Best latency = inf, Best rate = 0, Fitness std - (0.009342482077895998), Highest fitness for this gen - (-0.003289473684210526)
Generation 5: Best latency = 90, Best rate = 1, Fitness std - (0.004766374156794099), Highest fitness for this gen - (0.06156015037593984)
Generation 6: Best latency = 90, Best rate = 1, Fitness std - (0.002919415458792397), Highest fitness for this gen - (0.06156015037593984)
Generation 7: Best latency = 90, Best rate = 1, Fitness std - (0.0028013946278714723), Highest fitness for this gen - (0.06156015037593984)
Generation 8: Best latency = 90, Best rate = 1, Fitness std - (0.0032054754845368817), Highest fitness for this gen - (0.06156015037593984)
Generation 9: Best latency = 90, Best rate = 1, Fitness std - (0.004041976793245253), Highest fitness for this gen - (0.06156015037593984)
Generation 10: Best latency = 90, Best rate = 1, Fitness std - (0.0050598686313055455), Highest fitness for this gen - (0.06156015037593984)
Generation 11: Best latency = 90, Best rate = 1, Fitness std - (0.0069362595333713846), Highest fitness for this gen - (0.06156015037593984)
Generation 12: Best latency = 90, Best rate = 1, Fitness std - (0.009090359616784938), Highest fitness for this gen - (0.06156015037593984)
Generation 13: Best latency = 90, Best rate = 1, Fitness std - (0.01193908399294004), Highest fitness for this gen - (0.06156015037593984)
Generation 14: Best latency = 90, Best rate = 2, Fitness std - (0.015574454605613564), Highest fitness for this gen - (0.13298872180451127)
Generation 15: Best latency = 90, Best rate = 2, Fitness std - (0.019550411391098966), Highest fitness for this gen - (0.13298872180451127)
Generation 16: Best latency = 90, Best rate = 2, Fitness std - (0.024111479371338496), Highest fitness for this gen - (0.13298872180451127)
Generation 17: Best latency = 90, Best rate = 2, Fitness std - (0.0290277132764763), Highest fitness for this gen - (0.13298872180451127)
Generation 18: Best latency = 90, Best rate = 2, Fitness std - (0.03349864852491557), Highest fitness for this gen - (0.13298872180451127)
Generation 19: Best latency = 90, Best rate = 2, Fitness std - (0.03460270304646003), Highest fitness for this gen - (0.13298872180451127)
Generation 20: Best latency = 90, Best rate = 2, Fitness std - (0.03343144197717498), Highest fitness for this gen - (0.13298872180451127)
Generation 21: Best latency = 90, Best rate = 2, Fitness std - (0.035760794232399584), Highest fitness for this gen - (0.13298872180451127)
Generation 22: Best latency = 90, Best rate = 2, Fitness std - (0.03901636379765456), Highest fitness for this gen - (0.13298872180451127)
Generation 23: Best latency = 90, Best rate = 2, Fitness std - (0.04286751123844585), Highest fitness for this gen - (0.13298872180451127)
Generation 24: Best latency = 90, Best rate = 2, Fitness std - (0.04722771476660806), Highest fitness for this gen - (0.13298872180451127)
Generation 25: Best latency = 90, Best rate = 2, Fitness std - (0.05217834734804073), Highest fitness for this gen - (0.13298872180451127)
Generation 26: Best latency = 90, Best rate = 2, Fitness std - (0.0572290085798144), Highest fitness for this gen - (0.13298872180451127)
Generation 27: Best latency = 90, Best rate = 2, Fitness std - (0.05418578010356683), Highest fitness for this gen - (0.13298872180451127)
Generation 28: Best latency = 90, Best rate = 2, Fitness std - (0.05411751301953464), Highest fitness for this gen - (0.13298872180451127)
Generation 29: Best latency = 90, Best rate = 2, Fitness std - (0.05459116700752178), Highest fitness for this gen - (0.13298872180451127)
Generation 30: Best latency = 90, Best rate = 2, Fitness std - (0.05350793179923981), Highest fitness for this gen - (0.13298872180451127)
Generation 31: Best latency = 90, Best rate = 2, Fitness std - (0.053870067077962006), Highest fitness for this gen - (0.13298872180451127)
Generation 32: Best latency = 90, Best rate = 2, Fitness std - (0.05340946946536193), Highest fitness for this gen - (0.13298872180451127)
Generation 33: Best latency = 90, Best rate = 2, Fitness std - (0.05314012386060445), Highest fitness for this gen - (0.13298872180451127)
Generation 34: Best latency = 90, Best rate = 2, Fitness std - (0.05422664334602011), Highest fitness for this gen - (0.13298872180451127)
Generation 35: Best latency = 90, Best rate = 2, Fitness std - (0.0537175408972748), Highest fitness for this gen - (0.13298872180451127)
Generation 36: Best latency = 90, Best rate = 2, Fitness std - (0.05366182848855128), Highest fitness for this gen - (0.13298872180451127)
Generation 37: Best latency = 90, Best rate = 2, Fitness std - (0.05316862084329847), Highest fitness for this gen - (0.13298872180451127)
Generation 38: Best latency = 90, Best rate = 2, Fitness std - (0.05353598827719841), Highest fitness for this gen - (0.13298872180451127)
Generation 39: Best latency = 90, Best rate = 2, Fitness std - (0.05422664334602011), Highest fitness for this gen - (0.13298872180451127)
Generation 40: Best latency = 90, Best rate = 2, Fitness std - (0.05301146260525643), Highest fitness for this gen - (0.13298872180451127)
Generation 41: Best latency = 90, Best rate = 2, Fitness std - (0.0535079317992398), Highest fitness for this gen - (0.13298872180451127)
Generation 42: Best latency = 90, Best rate = 2, Fitness std - (0.05380086084688573), Highest fitness for this gen - (0.13298872180451127)
Generation 43: Best latency = 90, Best rate = 2, Fitness std - (0.05361995685603104), Highest fitness for this gen - (0.13298872180451127)
Generation 44: Best latency = 90, Best rate = 2, Fitness std - (0.05349389097721426), Highest fitness for this gen - (0.13298872180451127)
Generation 45: Best latency = 90, Best rate = 2, Fitness std - (0.053786994862387784), Highest fitness for this gen - (0.13298872180451127)
Generation 46: Best latency = 90, Best rate = 2, Fitness std - (0.053745347317480825), Highest fitness for this gen - (0.13298872180451127)
Generation 47: Best latency = 90, Best rate = 2, Fitness std - (0.05426743403343706), Highest fitness for this gen - (0.13298872180451127)
Generation 48: Best latency = 90, Best rate = 2, Fitness std - (0.05364787960527988), Highest fitness for this gen - (0.13298872180451127)
Generation 49: Best latency = 90, Best rate = 2, Fitness std - (0.05426743403343707), Highest fitness for this gen - (0.13298872180451127)
Generation 50: Best latency = 90, Best rate = 2, Fitness std - (0.05335301979824891), Highest fitness for this gen - (0.13298872180451127)
Generation 51: Best latency = 90, Best rate = 2, Fitness std - (0.053324744206336344), Highest fitness for this gen - (0.13298872180451127)
Generation 52: Best latency = 90, Best rate = 2, Fitness std - (0.05336714489454173), Highest fitness for this gen - (0.13298872180451127)
Generation 53: Best latency = 90, Best rate = 2, Fitness std - (0.054185780103566825), Highest fitness for this gen - (0.13298872180451127)
Generation 54: Best latency = 90, Best rate = 2, Fitness std - (0.05333888623775719), Highest fitness for this gen - (0.13298872180451127)
Generation 55: Best latency = 90, Best rate = 2, Fitness std - (0.054443359237645914), Highest fitness for this gen - (0.13298872180451127)
Generation 56: Best latency = 90, Best rate = 2, Fitness std - (0.05353598827719841), Highest fitness for this gen - (0.13298872180451127)
Generation 57: Best latency = 90, Best rate = 2, Fitness std - (0.054185780103566825), Highest fitness for this gen - (0.13298872180451127)
Generation 58: Best latency = 90, Best rate = 2, Fitness std - (0.05336714489454173), Highest fitness for this gen - (0.13298872180451127)
Generation 59: Best latency = 90, Best rate = 2, Fitness std - (0.053564011244299994), Highest fitness for this gen - (0.13298872180451127)
Generation 60: Best latency = 90, Best rate = 2, Fitness std - (0.05304011421840393), Highest fitness for this gen - (0.13298872180451127)
Generation 61: Best latency = 90, Best rate = 2, Fitness std - (0.05402159828606239), Highest fitness for this gen - (0.13298872180451127)
Generation 62: Best latency = 90, Best rate = 2, Fitness std - (0.05296842045152711), Highest fitness for this gen - (0.13298872180451127)
Generation 63: Best latency = 90, Best rate = 2, Fitness std - (0.05437585572721609), Highest fitness for this gen - (0.13298872180451127)
Generation 64: Best latency = 90, Best rate = 2, Fitness std - (0.05316862084329847), Highest fitness for this gen - (0.13298872180451127)
Generation 65: Best latency = 90, Best rate = 2, Fitness std - (0.05404904317169045), Highest fitness for this gen - (0.13298872180451127)
Generation 66: Best latency = 90, Best rate = 2, Fitness std - (0.05400786361563955), Highest fitness for this gen - (0.13298872180451127)
Generation 67: Best latency = 90, Best rate = 2, Fitness std - (0.0534517180996126), Highest fitness for this gen - (0.13298872180451127)
Generation 68: Best latency = 90, Best rate = 2, Fitness std - (0.05380086084688572), Highest fitness for this gen - (0.13298872180451127)
Generation 69: Best latency = 90, Best rate = 2, Fitness std - (0.05326809123636526), Highest fitness for this gen - (0.13298872180451127)
Generation 70: Best latency = 90, Best rate = 2, Fitness std - (0.05262126452750513), Highest fitness for this gen - (0.13298872180451127)
Generation 71: Best latency = 90, Best rate = 2, Fitness std - (0.05426743403343706), Highest fitness for this gen - (0.13298872180451127)
Generation 72: Best latency = 90, Best rate = 2, Fitness std - (0.05382856805850426), Highest fitness for this gen - (0.13298872180451127)
Generation 73: Best latency = 90, Best rate = 2, Fitness std - (0.05301146260525643), Highest fitness for this gen - (0.13298872180451127)
Generation 74: Best latency = 90, Best rate = 2, Fitness std - (0.05314012386060446), Highest fitness for this gen - (0.13298872180451127)
Generation 75: Best latency = 90, Best rate = 2, Fitness std - (0.05367576905291173), Highest fitness for this gen - (0.13298872180451127)
Generation 76: Best latency = 90, Best rate = 2, Fitness std - (0.053773120616920064), Highest fitness for this gen - (0.13298872180451127)
Generation 77: Best latency = 90, Best rate = 2, Fitness std - (0.05436233106249908), Highest fitness for this gen - (0.13298872180451127)
Generation 78: Best latency = 90, Best rate = 2, Fitness std - (0.05380086084688573), Highest fitness for this gen - (0.13298872180451127)
Generation 79: Best latency = 90, Best rate = 2, Fitness std - (0.0529684204515271), Highest fitness for this gen - (0.13298872180451127)
Generation 80: Best latency = 90, Best rate = 2, Fitness std - (0.0541721428710376), Highest fitness for this gen - (0.13298872180451127)
Generation 81: Best latency = 90, Best rate = 2, Fitness std - (0.05381471857679947), Highest fitness for this gen - (0.13298872180451127)
Generation 82: Best latency = 90, Best rate = 2, Fitness std - (0.05228380601947496), Highest fitness for this gen - (0.13298872180451127)
Generation 83: Best latency = 90, Best rate = 2, Fitness std - (0.05338126153335472), Highest fitness for this gen - (0.13298872180451127)
Generation 84: Best latency = 90, Best rate = 2, Fitness std - (0.05361995685603104), Highest fitness for this gen - (0.13298872180451127)
Generation 85: Best latency = 90, Best rate = 2, Fitness std - (0.05331059369724495), Highest fitness for this gen - (0.13298872180451127)
Generation 86: Best latency = 90, Best rate = 2, Fitness std - (0.05266493486250177), Highest fitness for this gen - (0.13298872180451127)
Generation 87: Best latency = 90, Best rate = 2, Fitness std - (0.052982776476649486), Highest fitness for this gen - (0.13298872180451127)
Generation 88: Best latency = 90, Best rate = 2, Fitness std - (0.05346578413402597), Highest fitness for this gen - (0.13298872180451127)
Generation 89: Best latency = 90, Best rate = 2, Fitness std - (0.05403532480267526), Highest fitness for this gen - (0.13298872180451127)
Generation 90: Best latency = 90, Best rate = 2, Fitness std - (0.0535219642303685), Highest fitness for this gen - (0.13298872180451127)
Generation 91: Best latency = 90, Best rate = 2, Fitness std - (0.05426743403343706), Highest fitness for this gen - (0.13298872180451127)
Generation 92: Best latency = 90, Best rate = 2, Fitness std - (0.0542538451891942), Highest fitness for this gen - (0.13298872180451127)
Generation 93: Best latency = 90, Best rate = 2, Fitness std - (0.05370362525080933), Highest fitness for this gen - (0.13298872180451127)
Generation 94: Best latency = 90, Best rate = 2, Fitness std - (0.053225512232681634), Highest fitness for this gen - (0.13298872180451127)
Generation 95: Best latency = 90, Best rate = 2, Fitness std - (0.05316862084329847), Highest fitness for this gen - (0.13298872180451127)
Generation 96: Best latency = 90, Best rate = 2, Fitness std - (0.05301146260525643), Highest fitness for this gen - (0.13298872180451127)
Generation 97: Best latency = 90, Best rate = 2, Fitness std - (0.054240248295406905), Highest fitness for this gen - (0.13298872180451127)
Generation 98: Best latency = 90, Best rate = 2, Fitness std - (0.05409014945512355), Highest fitness for this gen - (0.13298872180451127)
Generation 99: Best latency = 90, Best rate = 2, Fitness std - (0.05346578413402597), Highest fitness for this gen - (0.13298872180451127)
Generation 100: Best latency = 90, Best rate = 2, Fitness std - (0.05404904317169045), Highest fitness for this gen - (0.13298872180451127)
Generation 101: Best latency = 90, Best rate = 2, Fitness std - (0.05398036978846449), Highest fitness for this gen - (0.13298872180451127)
Generation 102: Best latency = 90, Best rate = 2, Fitness std - (0.05299712385888676), Highest fitness for this gen - (0.13298872180451127)
Generation 103: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 104: Best latency = 90, Best rate = 2, Fitness std - (0.05364787960527988), Highest fitness for this gen - (0.13298872180451127)
Generation 105: Best latency = 90, Best rate = 2, Fitness std - (0.052313337794118064), Highest fitness for this gen - (0.13298872180451127)
Generation 106: Best latency = 90, Best rate = 2, Fitness std - (0.05270852599458692), Highest fitness for this gen - (0.13298872180451127)
Generation 107: Best latency = 90, Best rate = 2, Fitness std - (0.053310593697244946), Highest fitness for this gen - (0.13298872180451127)
Generation 108: Best latency = 90, Best rate = 2, Fitness std - (0.05361995685603104), Highest fitness for this gen - (0.13298872180451127)
Generation 109: Best latency = 90, Best rate = 2, Fitness std - (0.05293968244449775), Highest fitness for this gen - (0.13298872180451127)
Generation 110: Best latency = 90, Best rate = 2, Fitness std - (0.05457776956059234), Highest fitness for this gen - (0.13298872180451127)
Generation 111: Best latency = 90, Best rate = 2, Fitness std - (0.053786994862387784), Highest fitness for this gen - (0.13298872180451127)
Generation 112: Best latency = 90, Best rate = 2, Fitness std - (0.05373144825068602), Highest fitness for this gen - (0.13298872180451127)
Generation 113: Best latency = 90, Best rate = 2, Fitness std - (0.053324744206336344), Highest fitness for this gen - (0.13298872180451127)
Generation 114: Best latency = 90, Best rate = 2, Fitness std - (0.054185780103566825), Highest fitness for this gen - (0.13298872180451127)
Generation 115: Best latency = 90, Best rate = 2, Fitness std - (0.05445683601868581), Highest fitness for this gen - (0.13298872180451127)
Generation 116: Best latency = 90, Best rate = 2, Fitness std - (0.053952843271250814), Highest fitness for this gen - (0.13298872180451127)
Generation 117: Best latency = 90, Best rate = 2, Fitness std - (0.0542538451891942), Highest fitness for this gen - (0.13298872180451127)
Generation 118: Best latency = 90, Best rate = 2, Fitness std - (0.05366182848855128), Highest fitness for this gen - (0.13298872180451127)
Generation 119: Best latency = 90, Best rate = 2, Fitness std - (0.05319708361569663), Highest fitness for this gen - (0.13298872180451127)
Generation 120: Best latency = 90, Best rate = 2, Fitness std - (0.05326809123636526), Highest fitness for this gen - (0.13298872180451127)
Generation 121: Best latency = 90, Best rate = 2, Fitness std - (0.05346578413402597), Highest fitness for this gen - (0.13298872180451127)
Generation 122: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 123: Best latency = 90, Best rate = 2, Fitness std - (0.05367576905291173), Highest fitness for this gen - (0.13298872180451127)
Generation 124: Best latency = 90, Best rate = 2, Fitness std - (0.05360598297704039), Highest fitness for this gen - (0.13298872180451127)
Generation 125: Best latency = 90, Best rate = 2, Fitness std - (0.053239713749994984), Highest fitness for this gen - (0.13298872180451127)
```

```
Generation 126: Best latency = 90, Best rate = 2, Fitness std - (0.05231333779411806), Highest fitness for this gen - (0.13298872180451127)
Generation 127: Best latency = 90, Best rate = 2, Fitness std - (0.0534517180996126), Highest fitness for this gen - (0.13298872180451127)
Generation 128: Best latency = 90, Best rate = 2, Fitness std - (0.054240248295406905), Highest fitness for this gen - (0.13298872180451127)
Generation 129: Best latency = 90, Best rate = 2, Fitness std - (0.053773120616920064), Highest fitness for this gen - (0.13298872180451127)
Generation 130: Best latency = 90, Best rate = 2, Fitness std - (0.05316862084329847), Highest fitness for this gen - (0.13298872180451127)
Generation 131: Best latency = 90, Best rate = 2, Fitness std - (0.05429458759746161), Highest fitness for this gen - (0.13298872180451127)
Generation 132: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 133: Best latency = 90, Best rate = 2, Fitness std - (0.053182856502350065), Highest fitness for this gen - (0.13298872180451127)
Generation 134: Best latency = 90, Best rate = 2, Fitness std - (0.05292530044849644), Highest fitness for this gen - (0.13298872180451127)
Generation 135: Best latency = 90, Best rate = 2, Fitness std - (0.05361995685603104), Highest fitness for this gen - (0.13298872180451127)
Generation 136: Best latency = 90, Best rate = 2, Fitness std - (0.05295405577649026), Highest fitness for this gen - (0.13298872180451127)
Generation 137: Best latency = 90, Best rate = 2, Fitness std - (0.05315437663167578), Highest fitness for this gen - (0.13298872180451127)
Generation 138: Best latency = 90, Best rate = 2, Fitness std - (0.05396661061923747), Highest fitness for this gen - (0.13298872180451127)
Generation 139: Best latency = 90, Best rate = 2, Fitness std - (0.05504201652162137), Highest fitness for this gen - (0.13298872180451127)
Generation 140: Best latency = 90, Best rate = 2, Fitness std - (0.05353598827719841), Highest fitness for this gen - (0.13298872180451127)
Generation 141: Best latency = 90, Best rate = 2, Fitness std - (0.05283882613182436), Highest fitness for this gen - (0.13298872180451127)
Generation 142: Best latency = 90, Best rate = 2, Fitness std - (0.053994120785184375), Highest fitness for this gen - (0.13298872180451127)
Generation 143: Best latency = 90, Best rate = 2, Fitness std - (0.05356401124429994), Highest fitness for this gen - (0.13298872180451127)
Generation 144: Best latency = 90, Best rate = 2, Fitness std - (0.05342356077193878), Highest fitness for this gen - (0.13298872180451127)
Generation 145: Best latency = 90, Best rate = 2, Fitness std - (0.054049043171690445), Highest fitness for this gen - (0.13298872180451127)
Generation 146: Best latency = 90, Best rate = 2, Fitness std - (0.05316862084329847), Highest fitness for this gen - (0.13298872180451127)
Generation 147: Best latency = 90, Best rate = 2, Fitness std - (0.05357801017771353), Highest fitness for this gen - (0.13298872180451127)
Generation 148: Best latency = 90, Best rate = 2, Fitness std - (0.053745347317480825), Highest fitness for this gen - (0.13298872180451127)
Generation 149: Best latency = 90, Best rate = 2, Fitness std - (0.05392528401393797), Highest fitness for this gen - (0.13298872180451127)
Generation 150: Best latency = 90, Best rate = 2, Fitness std - (0.05346578413402597), Highest fitness for this gen - (0.13298872180451127)
Generation 151: Best latency = 90, Best rate = 2, Fitness std - (0.05445683601868581), Highest fitness for this gen - (0.13298872180451127)
Generation 152: Best latency = 90, Best rate = 2, Fitness std - (0.05349389097721426), Highest fitness for this gen - (0.13298872180451127)
Generation 153: Best latency = 90, Best rate = 2, Fitness std - (0.052489776326401574), Highest fitness for this gen - (0.13298872180451127)
Generation 154: Best latency = 90, Best rate = 2, Fitness std - (0.054617938203445245), Highest fitness for this gen - (0.13298872180451127)
Generation 155: Best latency = 90, Best rate = 2, Fitness std - (0.053211302190192766), Highest fitness for this gen - (0.13298872180451127)
Generation 156: Best latency = 90, Best rate = 2, Fitness std - (0.05368970130484126), Highest fitness for this gen - (0.13298872180451127)
Generation 157: Best latency = 90, Best rate = 2, Fitness std - (0.05360598297704039), Highest fitness for this gen - (0.13298872180451127)
Generation 158: Best latency = 90, Best rate = 2, Fitness std - (0.0535079317992398), Highest fitness for this gen - (0.13298872180451127)
Generation 159: Best latency = 90, Best rate = 2, Fitness std - (0.05339536972139584), Highest fitness for this gen - (0.13298872180451127)
Generation 160: Best latency = 90, Best rate = 2, Fitness std - (0.054443359237645914), Highest fitness for this gen - (0.13298872180451127)
Generation 161: Best latency = 90, Best rate = 2, Fitness std - (0.05302579272276418), Highest fitness for this gen - (0.13298872180451127)
Generation 162: Best latency = 90, Best rate = 2, Fitness std - (0.05314012386060445), Highest fitness for this gen - (0.13298872180451127)
Generation 163: Best latency = 90, Best rate = 2, Fitness std - (0.05399412078518437), Highest fitness for this gen - (0.13298872180451127)
Generation 164: Best latency = 90, Best rate = 2, Fitness std - (0.05361995685603104), Highest fitness for this gen - (0.13298872180451127)
Generation 165: Best latency = 90, Best rate = 2, Fitness std - (0.05387006707796201), Highest fitness for this gen - (0.13298872180451127)
Generation 166: Best latency = 90, Best rate = 2, Fitness std - (0.05336714489454173), Highest fitness for this gen - (0.13298872180451127)
Generation 167: Best latency = 90, Best rate = 2, Fitness std - (0.0539390677382419), Highest fitness for this gen - (0.13298872180451127)
Generation 168: Best latency = 90, Best rate = 2, Fitness std - (0.05336714489454173), Highest fitness for this gen - (0.13298872180451127)
Generation 169: Best latency = 90, Best rate = 2, Fitness std - (0.05392528401393797), Highest fitness for this gen - (0.13298872180451127)
Generation 170: Best latency = 90, Best rate = 2, Fitness std - (0.053310593697244946), Highest fitness for this gen - (0.13298872180451127)
Generation 171: Best latency = 90, Best rate = 2, Fitness std - (0.05246045914060064), Highest fitness for this gen - (0.13298872180451127)
Generation 172: Best latency = 90, Best rate = 2, Fitness std - (0.05380086084688572), Highest fitness for this gen - (0.13298872180451127)
Generation 173: Best latency = 90, Best rate = 2, Fitness std - (0.05357801017771352), Highest fitness for this gen - (0.13298872180451127)
Generation 174: Best latency = 90, Best rate = 2, Fitness std - (0.05267947403047019), Highest fitness for this gen - (0.13298872180451127)
Generation 175: Best latency = 90, Best rate = 2, Fitness std - (0.0540215982860624), Highest fitness for this gen - (0.13298872180451127)
Generation 176: Best latency = 90, Best rate = 2, Fitness std - (0.05399412078518437), Highest fitness for this gen - (0.13298872180451127)
Generation 177: Best latency = 90, Best rate = 2, Fitness std - (0.05326809123636526), Highest fitness for this gen - (0.13298872180451127)
Generation 178: Best latency = 90, Best rate = 2, Fitness std - (0.052031327172258696), Highest fitness for this gen - (0.13298872180451127)
Generation 179: Best latency = 90, Best rate = 2, Fitness std - (0.0525775147922417), Highest fitness for this gen - (0.13298872180451127)
Generation 180: Best latency = 90, Best rate = 2, Fitness std - (0.05419940925617936), Highest fitness for this gen - (0.13298872180451127)
Generation 181: Best latency = 90, Best rate = 2, Fitness std - (0.05314012386060445), Highest fitness for this gen - (0.13298872180451127)
Generation 182: Best latency = 90, Best rate = 2, Fitness std - (0.052752038120106266), Highest fitness for this gen - (0.13298872180451127)
Generation 183: Best latency = 90, Best rate = 2, Fitness std - (0.052135606270274255), Highest fitness for this gen - (0.13298872180451127)
Generation 184: Best latency = 90, Best rate = 2, Fitness std - (0.05399412078518437), Highest fitness for this gen - (0.13298872180451127)
Generation 185: Best latency = 90, Best rate = 2, Fitness std - (0.0540764554917272), Highest fitness for this gen - (0.13298872180451127)
Generation 186: Best latency = 90, Best rate = 2, Fitness std - (0.052752038120106266), Highest fitness for this gen - (0.13298872180451127)
Generation 187: Best latency = 90, Best rate = 2, Fitness std - (0.05396661061923747), Highest fitness for this gen - (0.13298872180451127)
Generation 188: Best latency = 90, Best rate = 2, Fitness std - (0.0548442238425241), Highest fitness for this gen - (0.13298872180451127)
Generation 189: Best latency = 90, Best rate = 2, Fitness std - (0.05295405577649026), Highest fitness for this gen - (0.13298872180451127)
Generation 190: Best latency = 90, Best rate = 2, Fitness std - (0.053239713749994984), Highest fitness for this gen - (0.13298872180451127)
Generation 191: Best latency = 90, Best rate = 2, Fitness std - (0.05346578413402598), Highest fitness for this gen - (0.13298872180451127)
Generation 192: Best latency = 90, Best rate = 2, Fitness std - (0.05340946946536193), Highest fitness for this gen - (0.13298872180451127)
Generation 193: Best latency = 90, Best rate = 2, Fitness std - (0.053773120616920064), Highest fitness for this gen - (0.13298872180451127)
Generation 194: Best latency = 90, Best rate = 2, Fitness std - (0.05312586252319542), Highest fitness for this gen - (0.13298872180451127)
Generation 195: Best latency = 90, Best rate = 2, Fitness std - (0.0534517180996126), Highest fitness for this gen - (0.13298872180451127)
Generation 196: Best latency = 90, Best rate = 2, Fitness std - (0.05380086084688573), Highest fitness for this gen - (0.13298872180451127)
Generation 197: Best latency = 90, Best rate = 2, Fitness std - (0.05448376570707629), Highest fitness for this gen - (0.13298872180451127)
Generation 198: Best latency = 90, Best rate = 2, Fitness std - (0.05429458759746161), Highest fitness for this gen - (0.13298872180451127)
Generation 199: Best latency = 90, Best rate = 2, Fitness std - (0.05333888623775719), Highest fitness for this gen - (0.13298872180451127)
Generation 200: Best latency = 90, Best rate = 2, Fitness std - (0.05292530044849644), Highest fitness for this gen - (0.13298872180451127)
Generation 201: Best latency = 90, Best rate = 2, Fitness std - (0.05322551223268164), Highest fitness for this gen - (0.13298872180451127)
Generation 202: Best latency = 90, Best rate = 2, Fitness std - (0.053786994862387784), Highest fitness for this gen - (0.13298872180451127)
Generation 203: Best latency = 90, Best rate = 2, Fitness std - (0.0542538451891942), Highest fitness for this gen - (0.13298872180451127)
Generation 204: Best latency = 90, Best rate = 2, Fitness std - (0.0540215982860624), Highest fitness for this gen - (0.13298872180451127)
Generation 205: Best latency = 90, Best rate = 2, Fitness std - (0.05392528401393797), Highest fitness for this gen - (0.13298872180451127)
Generation 206: Best latency = 90, Best rate = 2, Fitness std - (0.053592000753112316), Highest fitness for this gen - (0.13298872180451127)
Generation 207: Best latency = 90, Best rate = 2, Fitness std - (0.05328226721902888), Highest fitness for this gen - (0.13298872180451127)
Generation 208: Best latency = 90, Best rate = 2, Fitness std - (0.053423560771938786), Highest fitness for this gen - (0.13298872180451127)
Generation 209: Best latency = 90, Best rate = 2, Fitness std - (0.05414484414180483), Highest fitness for this gen - (0.13298872180451127)
Generation 210: Best latency = 90, Best rate = 2, Fitness std - (0.053083027043878094), Highest fitness for this gen - (0.13298872180451127)
Generation 211: Best latency = 90, Best rate = 2, Fitness std - (0.05436233106249908), Highest fitness for this gen - (0.13298872180451127)
Generation 212: Best latency = 90, Best rate = 2, Fitness std - (0.054007863615639556), Highest fitness for this gen - (0.13298872180451127)
Generation 213: Best latency = 90, Best rate = 2, Fitness std - (0.05388388363037656), Highest fitness for this gen - (0.13298872180451127)
Generation 214: Best latency = 90, Best rate = 2, Fitness std - (0.053353019798248906), Highest fitness for this gen - (0.13298872180451127)
Generation 215: Best latency = 90, Best rate = 2, Fitness std - (0.05243110634030652), Highest fitness for this gen - (0.13298872180451127)
Generation 216: Best latency = 90, Best rate = 2, Fitness std - (0.05409014945512353), Highest fitness for this gen - (0.13298872180451127)
Generation 217: Best latency = 90, Best rate = 2, Fitness std - (0.05355000394631677), Highest fitness for this gen - (0.13298872180451127)
Generation 218: Best latency = 90, Best rate = 2, Fitness std - (0.05428101483418094), Highest fitness for this gen - (0.13298872180451127)
Generation 219: Best latency = 90, Best rate = 2, Fitness std - (0.05452410064384746), Highest fitness for this gen - (0.13298872180451127)
Generation 220: Best latency = 90, Best rate = 2, Fitness std - (0.053353019798248906), Highest fitness for this gen - (0.13298872180451127)
Generation 221: Best latency = 90, Best rate = 2, Fitness std - (0.05338126153335472), Highest fitness for this gen - (0.13298872180451127)
Generation 222: Best latency = 90, Best rate = 2, Fitness std - (0.05364787960527988), Highest fitness for this gen - (0.13298872180451127)
Generation 223: Best latency = 90, Best rate = 2, Fitness std - (0.052982776476649486), Highest fitness for this gen - (0.13298872180451127)
Generation 224: Best latency = 90, Best rate = 2, Fitness std - (0.05428101483418094), Highest fitness for this gen - (0.13298872180451127)
Generation 225: Best latency = 90, Best rate = 2, Fitness std - (0.05312586252319542), Highest fitness for this gen - (0.13298872180451127)
Generation 226: Best latency = 90, Best rate = 2, Fitness std - (0.05331059369724495), Highest fitness for this gen - (0.13298872180451127)
Generation 227: Best latency = 90, Best rate = 2, Fitness std - (0.05366182848855128), Highest fitness for this gen - (0.13298872180451127)
Generation 228: Best latency = 90, Best rate = 2, Fitness std - (0.05395284327125081), Highest fitness for this gen - (0.13298872180451127)
Generation 229: Best latency = 90, Best rate = 2, Fitness std - (0.05355000394631677), Highest fitness for this gen - (0.13298872180451127)
Generation 230: Best latency = 90, Best rate = 2, Fitness std - (0.05325390674895319), Highest fitness for this gen - (0.13298872180451127)
Generation 231: Best latency = 90, Best rate = 2, Fitness std - (0.05399412078518437), Highest fitness for this gen - (0.13298872180451127)
Generation 232: Best latency = 90, Best rate = 2, Fitness std - (0.05306873137199632), Highest fitness for this gen - (0.13298872180451127)
Generation 233: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 234: Best latency = 90, Best rate = 2, Fitness std - (0.05364787960527988), Highest fitness for this gen - (0.13298872180451127)
Generation 235: Best latency = 90, Best rate = 2, Fitness std - (0.053083027043878094), Highest fitness for this gen - (0.13298872180451127)
Generation 236: Best latency = 90, Best rate = 2, Fitness std - (0.05364787960527988), Highest fitness for this gen - (0.13298872180451127)
Generation 237: Best latency = 90, Best rate = 2, Fitness std - (0.05380086084688572), Highest fitness for this gen - (0.13298872180451127)
Generation 238: Best latency = 90, Best rate = 2, Fitness std - (0.05329643470373041), Highest fitness for this gen - (0.13298872180451127)
Generation 239: Best latency = 90, Best rate = 2, Fitness std - (0.05355000394631677), Highest fitness for this gen - (0.13298872180451127)
Generation 240: Best latency = 90, Best rate = 2, Fitness std - (0.053773120616920064), Highest fitness for this gen - (0.13298872180451127)
Generation 241: Best latency = 90, Best rate = 2, Fitness std - (0.05231333779411806), Highest fitness for this gen - (0.13298872180451127)
Generation 242: Best latency = 90, Best rate = 2, Fitness std - (0.05339536972139583), Highest fitness for this gen - (0.13298872180451127)
Generation 243: Best latency = 90, Best rate = 2, Fitness std - (0.05378699486238779), Highest fitness for this gen - (0.13298872180451127)
Generation 244: Best latency = 90, Best rate = 2, Fitness std - (0.05429458759746161), Highest fitness for this gen - (0.13298872180451127)
Generation 245: Best latency = 90, Best rate = 2, Fitness std - (0.05355000394631677), Highest fitness for this gen - (0.13298872180451127)
Generation 246: Best latency = 90, Best rate = 2, Fitness std - (0.0532539067489532), Highest fitness for this gen - (0.13298872180451127)
Generation 247: Best latency = 90, Best rate = 2, Fitness std - (0.05381471857679947), Highest fitness for this gen - (0.13298872180451127)
Generation 248: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 249: Best latency = 90, Best rate = 2, Fitness std - (0.05329643470373041), Highest fitness for this gen - (0.13298872180451127)
Generation 250: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 251: Best latency = 90, Best rate = 2, Fitness std - (0.05375923810408655), Highest fitness for this gen - (0.13298872180451127)
```

```
Generation 252: Best latency = 90, Best rate = 2, Fitness std - (0.053856242302736126), Highest fitness for this gen - (0.13298872180451127)
Generation 253: Best latency = 90, Best rate = 2, Fitness std - (0.052752038120106266), Highest fitness for this gen - (0.13298872180451127)
Generation 254: Best latency = 90, Best rate = 2, Fitness std - (0.05367576905291173), Highest fitness for this gen - (0.13298872180451127)
Generation 255: Best latency = 90, Best rate = 2, Fitness std - (0.05338126153335472), Highest fitness for this gen - (0.13298872180451127)
Generation 256: Best latency = 90, Best rate = 2, Fitness std - (0.0542538451891942), Highest fitness for this gen - (0.13298872180451127)
Generation 257: Best latency = 90, Best rate = 2, Fitness std - (0.05319708361569663), Highest fitness for this gen - (0.13298872180451127)
Generation 258: Best latency = 90, Best rate = 2, Fitness std - (0.05331059369724495), Highest fitness for this gen - (0.13298872180451127)
Generation 259: Best latency = 90, Best rate = 2, Fitness std - (0.053154376631675784), Highest fitness for this gen - (0.13298872180451127)
Generation 260: Best latency = 90, Best rate = 2, Fitness std - (0.05465803583951965), Highest fitness for this gen - (0.13298872180451127)
Generation 261: Best latency = 90, Best rate = 2, Fitness std - (0.05400786361563955), Highest fitness for this gen - (0.13298872180451127)
Generation 262: Best latency = 90, Best rate = 2, Fitness std - (0.05382856805850425), Highest fitness for this gen - (0.13298872180451127)
Generation 263: Best latency = 90, Best rate = 2, Fitness std - (0.05391149209205619), Highest fitness for this gen - (0.13298872180451127)
Generation 264: Best latency = 90, Best rate = 2, Fitness std - (0.053239713749994984), Highest fitness for this gen - (0.13298872180451127)
Generation 265: Best latency = 90, Best rate = 2, Fitness std - (0.05347984175768293), Highest fitness for this gen - (0.13298872180451127)
Generation 266: Best latency = 90, Best rate = 2, Fitness std - (0.05388388363037656), Highest fitness for this gen - (0.13298872180451127)
Generation 267: Best latency = 90, Best rate = 2, Fitness std - (0.05296842045152711), Highest fitness for this gen - (0.13298872180451127)
Generation 268: Best latency = 90, Best rate = 2, Fitness std - (0.05299712385888676), Highest fitness for this gen - (0.13298872180451127)
Generation 269: Best latency = 90, Best rate = 2, Fitness std - (0.05368970130484126), Highest fitness for this gen - (0.13298872180451127)
Generation 270: Best latency = 90, Best rate = 2, Fitness std - (0.05391149209205619), Highest fitness for this gen - (0.13298872180451127)
Generation 271: Best latency = 90, Best rate = 2, Fitness std - (0.053253906748953195), Highest fitness for this gen - (0.13298872180451127)
Generation 272: Best latency = 90, Best rate = 2, Fitness std - (0.05355000394631677), Highest fitness for this gen - (0.13298872180451127)
Generation 273: Best latency = 90, Best rate = 2, Fitness std - (0.05333888623775719), Highest fitness for this gen - (0.13298872180451127)
Generation 274: Best latency = 90, Best rate = 2, Fitness std - (0.054253845189194204), Highest fitness for this gen - (0.13298872180451127)
Generation 275: Best latency = 90, Best rate = 2, Fitness std - (0.05276652463872317), Highest fitness for this gen - (0.13298872180451127)
Generation 276: Best latency = 90, Best rate = 2, Fitness std - (0.05373144825068602), Highest fitness for this gen - (0.13298872180451127)
Generation 277: Best latency = 90, Best rate = 2, Fitness std - (0.05361995685603104), Highest fitness for this gen - (0.13298872180451127)
Generation 278: Best latency = 90, Best rate = 2, Fitness std - (0.0537175408972748), Highest fitness for this gen - (0.13298872180451127)
Generation 279: Best latency = 90, Best rate = 2, Fitness std - (0.05349389097721426), Highest fitness for this gen - (0.13298872180451127)
Generation 280: Best latency = 90, Best rate = 2, Fitness std - (0.053952843271250814), Highest fitness for this gen - (0.13298872180451127)
Generation 281: Best latency = 90, Best rate = 2, Fitness std - (0.0534517180996126), Highest fitness for this gen - (0.13298872180451127)
Generation 282: Best latency = 90, Best rate = 2, Fitness std - (0.05398036978846449), Highest fitness for this gen - (0.13298872180451127)
Generation 283: Best latency = 90, Best rate = 2, Fitness std - (0.05441638177231396), Highest fitness for this gen - (0.13298872180451127)
Generation 284: Best latency = 90, Best rate = 2, Fitness std - (0.053225512232681634), Highest fitness for this gen - (0.13298872180451127)
Generation 285: Best latency = 90, Best rate = 2, Fitness std - (0.054375855727216094), Highest fitness for this gen - (0.13298872180451127)
Generation 286: Best latency = 90, Best rate = 2, Fitness std - (0.053324744206336344), Highest fitness for this gen - (0.13298872180451127)
Generation 287: Best latency = 90, Best rate = 2, Fitness std - (0.0540215982860624), Highest fitness for this gen - (0.13298872180451127)
Generation 288: Best latency = 90, Best rate = 2, Fitness std - (0.05396661061923747), Highest fitness for this gen - (0.13298872180451127)
Generation 289: Best latency = 90, Best rate = 2, Fitness std - (0.053661828488551284), Highest fitness for this gen - (0.13298872180451127)
Generation 290: Best latency = 90, Best rate = 2, Fitness std - (0.05270852599458692), Highest fitness for this gen - (0.13298872180451127)
Generation 291: Best latency = 90, Best rate = 2, Fitness std - (0.05368970130484126), Highest fitness for this gen - (0.13298872180451127)
Generation 292: Best latency = 90, Best rate = 2, Fitness std - (0.054429874490829276), Highest fitness for this gen - (0.13298872180451127)
Generation 293: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 294: Best latency = 90, Best rate = 2, Fitness std - (0.05459116700752178), Highest fitness for this gen - (0.13298872180451127)
Generation 295: Best latency = 90, Best rate = 2, Fitness std - (0.05336714489454173), Highest fitness for this gen - (0.13298872180451127)
Generation 296: Best latency = 90, Best rate = 2, Fitness std - (0.05373144825068602), Highest fitness for this gen - (0.13298872180451127)
Generation 297: Best latency = 90, Best rate = 2, Fitness std - (0.05381471857679947), Highest fitness for this gen - (0.13298872180451127)
Generation 298: Best latency = 90, Best rate = 2, Fitness std - (0.05385624230273613), Highest fitness for this gen - (0.13298872180451127)
Generation 299: Best latency = 90, Best rate = 2, Fitness std - (0.05349389097721426), Highest fitness for this gen - (0.13298872180451127)
```



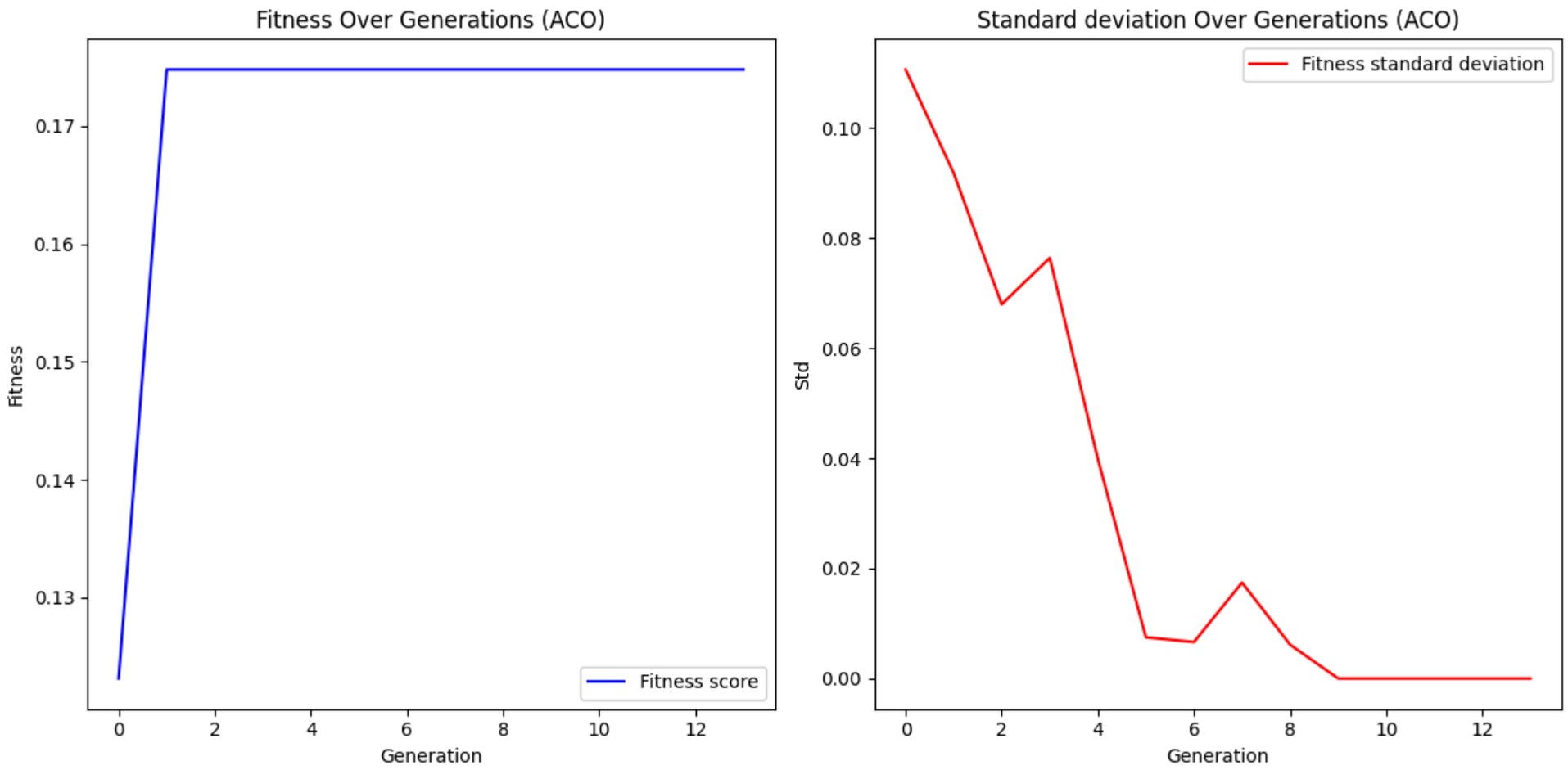



```
Total best paths [[115, 112, 77, 'BS-1']]
Best latency 90
Best transmission rate 2
Execution time 23.152353286743164 seconds
```

Ant colony algorithm

In [ ]:
```
ACO = Ant_colony_algorithm(nodes=modified_nodes, alpha=1, beta=2, _evaporation_rate=.5, _pheromone_deposit=1, BS_stations=BS_stations, dist_matrix=dist_matrix, rate_matrix=rate_matrix)

start_time = time.time()

best_path_ACO, all_best_paths_ACO, best_rate_ACO, best_latency_ACO, aco_fitness_score, aco_fitness_std, gen_num_ACO = ACO.start_aco(starting_node=STARTING_NODE_2, iteration=GENERATIONS, ants=
end_time = time.time()
execution_time = end_time - start_time

print("Best path", all_best_paths_ACO)
print("Best latency", best_latency_ACO)
print("Best transmission rate", best_rate_ACO)
print("Execution time", execution_time, "seconds")

ACO_metrics["Test2"]["paths"] = all_best_paths_ACO
ACO_metrics["Test2"]["latency"] = best_latency_ACO
ACO_metrics["Test2"]["transmission"] = best_rate_ACO

ACO_metrics["Test2"]["gen"] = gen_num_ACO

ACO_metrics["Test2"]["Fitness"] = aco_fitness_score
ACO_metrics["Test2"]["Fitness_std"] = aco_fitness_std
ACO_metrics["Test2"]["Execution_time"] = execution_time
```

```
Iteration 0: Best rate - 2, Best latency - 180, Fitness std - (0.11064357335634444), Best Fitness = 0.12312030075187969
Iteration 1: Best rate - 3, Best latency - 360, Fitness std - (0.09182823152699948), Best Fitness = 0.17481203007518797
Iteration 2: Best rate - 3, Best latency - 360, Fitness std - (0.06798019129602233), Best Fitness = 0.17481203007518797
Iteration 3: Best rate - 3, Best latency - 360, Fitness std - (0.07639738140395093), Best Fitness = 0.17481203007518797
Iteration 4: Best rate - 3, Best latency - 360, Fitness std - (0.039903638780855115), Best Fitness = 0.17481203007518797
Iteration 5: Best rate - 3, Best latency - 360, Fitness std - (0.007472537439817456), Best Fitness = 0.17481203007518797
Iteration 6: Best rate - 3, Best latency - 360, Fitness std - (0.00661993809128227), Best Fitness = 0.17481203007518797
Iteration 7: Best rate - 3, Best latency - 360, Fitness std - (0.017424135264371657), Best Fitness = 0.17481203007518797
Iteration 8: Best rate - 3, Best latency - 360, Fitness std - (0.006137844099837159), Best Fitness = 0.17481203007518797
Iteration 9: Best rate - 3, Best latency - 360, Fitness std - (0.0), Best Fitness = 0.17481203007518797
Iteration 10: Best rate - 3, Best latency - 360, Fitness std - (0.0), Best Fitness = 0.17481203007518797
Iteration 11: Best rate - 3, Best latency - 360, Fitness std - (0.0), Best Fitness = 0.17481203007518797
Iteration 12: Best rate - 3, Best latency - 360, Fitness std - (0.0), Best Fitness = 0.17481203007518797
Iteration 13: Best rate - 3, Best latency - 360, Fitness std - (0.0), Best Fitness = 0.17481203007518797
Algorithmm converged
```




```
Best path [[115, 135, 35, 87, 10, 45, 62, 84, 77, 90, 53, 57, 151], [115, 85, 35, 148, 10, 45, 62, 84, 61, 90, 53, 57, 151]]
Best latency 360
Best transmission rate 3
Execution time 2.475212335586548 seconds
```

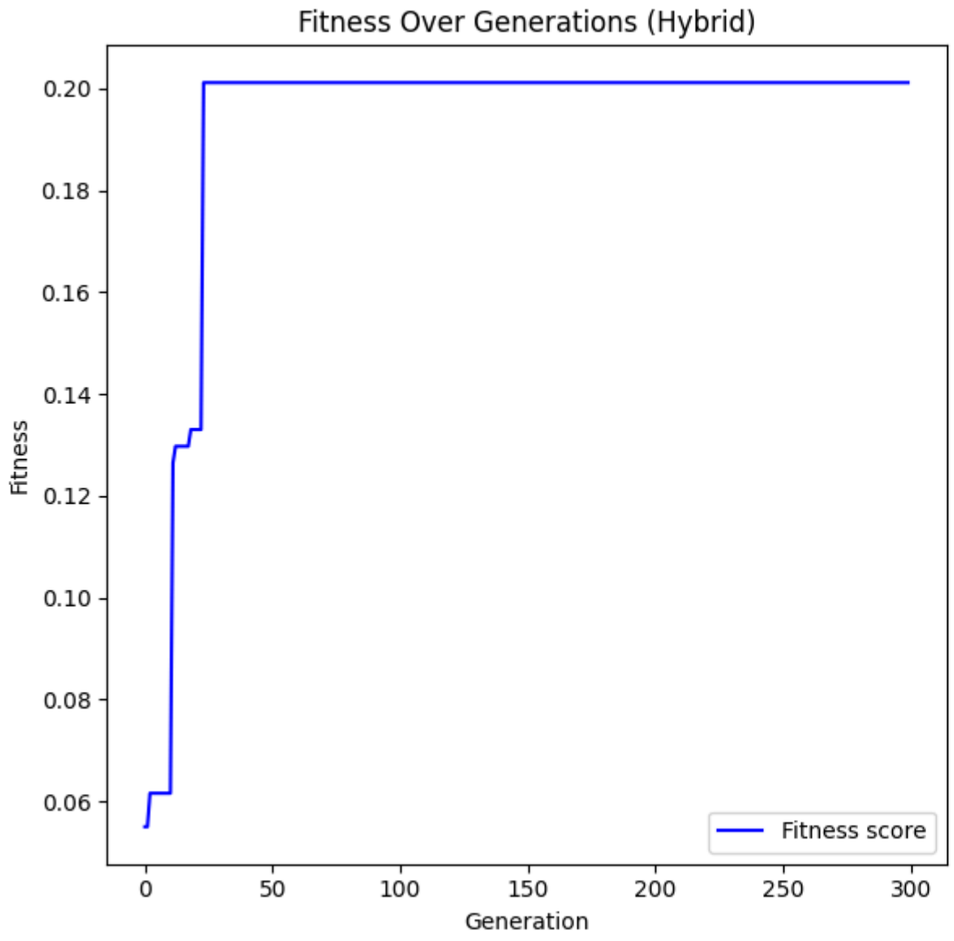
Hybrid algorithm (GA + SA)

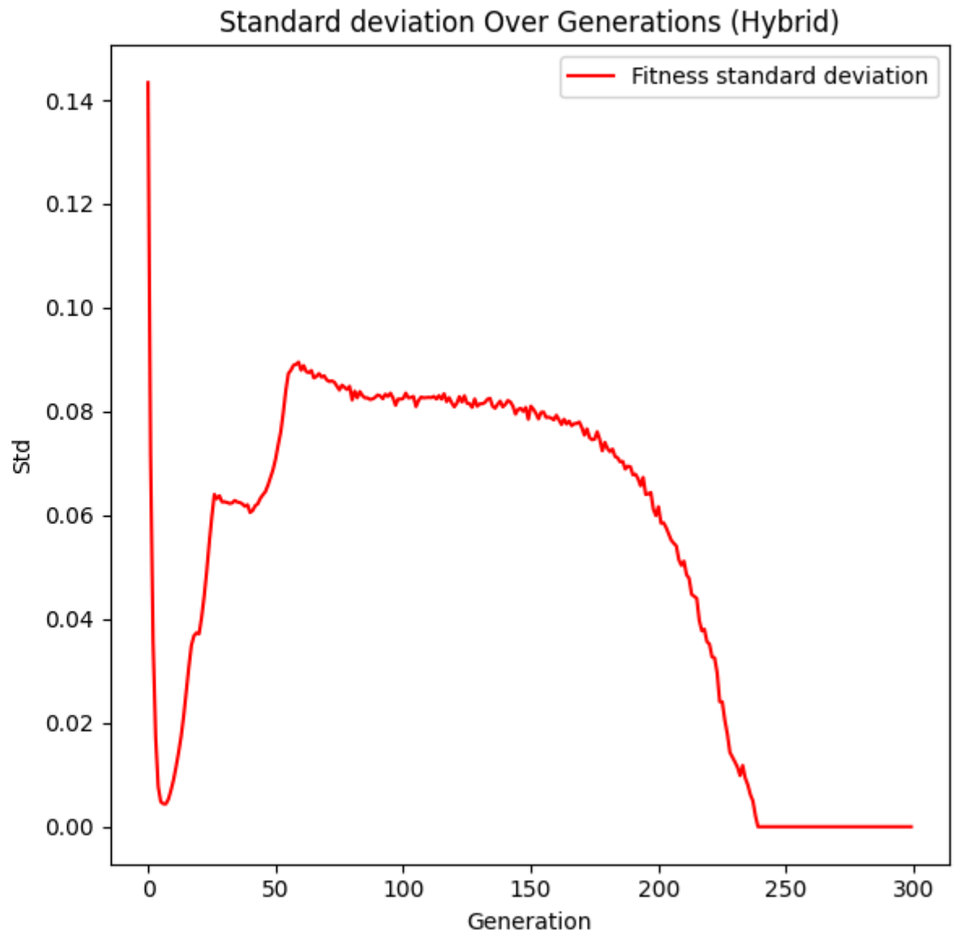
In [ ]:
```
GA_SA = Hybrid_GA_with_SA(nodes=NODES, pop_size=9000, generations=GENERATIONS, mutation_rate=.6, cooling_rate=.96)

start_time = time.time()

best_path_SA, best_latency_SA, best_transmission_rate_SA, all_best_paths_SA, hybrid_fitness_score, hybrid_fitness_std, gen_num_SA = GA_SA.GA(starting_node=STARTING_NODE_2, temperature=1000)

end_time = time.time()
execution_time = end_time - start_time

print("Best path", all_best_paths_SA)
print("Best latency", best_latency_SA)
print("Best transmission rate", best_transmission_rate_SA)
print("Execution time", execution_time, "seconds")

HYBRID_metrics["Test2"]["paths"] = all_best_paths_SA
HYBRID_metrics["Test2"]["latency"] = best_latency_SA
HYBRID_metrics["Test2"]["transmission"] = best_transmission_rate_SA

HYBRID_metrics["Test2"]["gen"] = gen_num_SA

HYBRID_metrics["Test2"]["Fitness"] = hybrid_fitness_score
HYBRID_metrics["Test2"]["Fitness_std"] = hybrid_fitness_std
HYBRID_metrics["Test2"]["Execution_time"] = execution_time
```

```
Generation 0: Best latency = 150, Best rate = 1, Fitness std - (0.14336480983409028) Highest fitness for this gen - (0.0549812030075188), Temperature - 1000.000
Generation 1: Best latency = 150, Best rate = 1, Fitness std - (0.0714583881169365) Highest fitness for this gen - (0.0549812030075188), Temperature - 960.000
Generation 2: Best latency = 90, Best rate = 1, Fitness std - (0.035464159439758156) Highest fitness for this gen - (0.06156015037593984), Temperature - 921.600
Generation 3: Best latency = 90, Best rate = 1, Fitness std - (0.017208689320075115) Highest fitness for this gen - (0.06156015037593984), Temperature - 884.736
Generation 4: Best latency = 90, Best rate = 1, Fitness std - (0.007706722696512824) Highest fitness for this gen - (0.06156015037593984), Temperature - 849.347
Generation 5: Best latency = 90, Best rate = 1, Fitness std - (0.004797320057025409) Highest fitness for this gen - (0.06156015037593984), Temperature - 815.373
Generation 6: Best latency = 90, Best rate = 1, Fitness std - (0.0044273763076176081) Highest fitness for this gen - (0.06156015037593984), Temperature - 782.758
Generation 7: Best latency = 90, Best rate = 1, Fitness std - (0.0043983428686776698) Highest fitness for this gen - (0.06156015037593984), Temperature - 751.447
Generation 8: Best latency = 90, Best rate = 1, Fitness std - (0.0053055975905536696) Highest fitness for this gen - (0.06156015037593984), Temperature - 721.390
Generation 9: Best latency = 90, Best rate = 1, Fitness std - (0.0070327694779158765) Highest fitness for this gen - (0.06156015037593984), Temperature - 692.534
Generation 10: Best latency = 90, Best rate = 1, Fitness std - (0.00902163560304256) Highest fitness for this gen - (0.06156015037593984), Temperature - 664.833
Generation 11: Best latency = 150, Best rate = 2, Fitness std - (0.011563805803485957) Highest fitness for this gen - (0.12640977443609022), Temperature - 638.239
Generation 12: Best latency = 120, Best rate = 2, Fitness std - (0.014366871288227515) Highest fitness for this gen - (0.12969924812030076), Temperature - 612.710
Generation 13: Best latency = 120, Best rate = 2, Fitness std - (0.01747203609670743) Highest fitness for this gen - (0.12969924812030076), Temperature - 588.201
Generation 14: Best latency = 120, Best rate = 2, Fitness std - (0.02140414166561384) Highest fitness for this gen - (0.12969924812030076), Temperature - 564.673
Generation 15: Best latency = 120, Best rate = 2, Fitness std - (0.026062048617883093) Highest fitness for this gen - (0.12969924812030076), Temperature - 542.086
Generation 16: Best latency = 120, Best rate = 2, Fitness std - (0.030772076689668647) Highest fitness for this gen - (0.12969924812030076), Temperature - 520.403
Generation 17: Best latency = 120, Best rate = 2, Fitness std - (0.034945394916799266) Highest fitness for this gen - (0.12969924812030076), Temperature - 499.587
Generation 18: Best latency = 90, Best rate = 2, Fitness std - (0.036783878971792194) Highest fitness for this gen - (0.13298872180451127), Temperature - 479.603
Generation 19: Best latency = 90, Best rate = 2, Fitness std - (0.037350286148055634) Highest fitness for this gen - (0.13298872180451127), Temperature - 460.419
Generation 20: Best latency = 90, Best rate = 2, Fitness std - (0.037145948227158734) Highest fitness for this gen - (0.13298872180451127), Temperature - 442.002
Generation 21: Best latency = 90, Best rate = 2, Fitness std - (0.040331013583965396) Highest fitness for this gen - (0.13298872180451127), Temperature - 424.322
Generation 22: Best latency = 90, Best rate = 2, Fitness std - (0.0443754950906496) Highest fitness for this gen - (0.13298872180451127), Temperature - 407.349
Generation 23: Best latency = 120, Best rate = 3, Fitness std - (0.049204065898076035) Highest fitness for this gen - (0.20112781954887216), Temperature - 391.055
Generation 24: Best latency = 120, Best rate = 3, Fitness std - (0.05472078434996678) Highest fitness for this gen - (0.20112781954887216), Temperature - 375.413
Generation 25: Best latency = 120, Best rate = 3, Fitness std - (0.059709700142233445) Highest fitness for this gen - (0.20112781954887216), Temperature - 360.397
Generation 26: Best latency = 120, Best rate = 3, Fitness std - (0.06402730612768467) Highest fitness for this gen - (0.20112781954887216), Temperature - 345.981
Generation 27: Best latency = 120, Best rate = 3, Fitness std - (0.06319638163855791) Highest fitness for this gen - (0.20112781954887216), Temperature - 332.142
Generation 28: Best latency = 120, Best rate = 3, Fitness std - (0.06376485410301651) Highest fitness for this gen - (0.20112781954887216), Temperature - 318.856
Generation 29: Best latency = 120, Best rate = 3, Fitness std - (0.06258216260425017) Highest fitness for this gen - (0.20112781954887216), Temperature - 306.102
Generation 30: Best latency = 120, Best rate = 3, Fitness std - (0.06261071915855193) Highest fitness for this gen - (0.20112781954887216), Temperature - 293.858
Generation 31: Best latency = 120, Best rate = 3, Fitness std - (0.06246683052090257) Highest fitness for this gen - (0.20112781954887216), Temperature - 282.103
Generation 32: Best latency = 120, Best rate = 3, Fitness std - (0.06227720815562608) Highest fitness for this gen - (0.20112781954887216), Temperature - 270.819
Generation 33: Best latency = 120, Best rate = 3, Fitness std - (0.0623833139087498) Highest fitness for this gen - (0.20112781954887216), Temperature - 259.986
Generation 34: Best latency = 120, Best rate = 3, Fitness std - (0.06285800097543225) Highest fitness for this gen - (0.20112781954887216), Temperature - 249.587
Generation 35: Best latency = 120, Best rate = 3, Fitness std - (0.06255584634790402) Highest fitness for this gen - (0.20112781954887216), Temperature - 239.603
Generation 36: Best latency = 120, Best rate = 3, Fitness std - (0.062456028111127) Highest fitness for this gen - (0.20112781954887216), Temperature - 230.019
Generation 37: Best latency = 120, Best rate = 3, Fitness std - (0.062188442886405376) Highest fitness for this gen - (0.20112781954887216), Temperature - 220.819
Generation 38: Best latency = 120, Best rate = 3, Fitness std - (0.06171864890065998) Highest fitness for this gen - (0.20112781954887216), Temperature - 211.986
Generation 39: Best latency = 120, Best rate = 3, Fitness std - (0.06203650339359795) Highest fitness for this gen - (0.20112781954887216), Temperature - 203.506
Generation 40: Best latency = 120, Best rate = 3, Fitness std - (0.060543348545107865) Highest fitness for this gen - (0.20112781954887216), Temperature - 195.366
Generation 41: Best latency = 120, Best rate = 3, Fitness std - (0.060920385567950305) Highest fitness for this gen - (0.20112781954887216), Temperature - 187.552
Generation 42: Best latency = 120, Best rate = 3, Fitness std - (0.06183981647271754) Highest fitness for this gen - (0.20112781954887216), Temperature - 180.049
Generation 43: Best latency = 120, Best rate = 3, Fitness std - (0.06220288761284298) Highest fitness for this gen - (0.20112781954887216), Temperature - 172.847
Generation 44: Best latency = 120, Best rate = 3, Fitness std - (0.06326298956872345) Highest fitness for this gen - (0.20112781954887216), Temperature - 165.934
Generation 45: Best latency = 120, Best rate = 3, Fitness std - (0.06394596917433375) Highest fitness for this gen - (0.20112781954887216), Temperature - 159.296
Generation 46: Best latency = 120, Best rate = 3, Fitness std - (0.0645575789103073) Highest fitness for this gen - (0.20112781954887216), Temperature - 152.924
Generation 47: Best latency = 120, Best rate = 3, Fitness std - (0.06579442297571966) Highest fitness for this gen - (0.20112781954887216), Temperature - 146.807
Generation 48: Best latency = 120, Best rate = 3, Fitness std - (0.06725858386233419) Highest fitness for this gen - (0.20112781954887216), Temperature - 140.935
Generation 49: Best latency = 120, Best rate = 3, Fitness std - (0.06881375050405482) Highest fitness for this gen - (0.20112781954887216), Temperature - 135.298
Generation 50: Best latency = 120, Best rate = 3, Fitness std - (0.07088380014976371) Highest fitness for this gen - (0.20112781954887216), Temperature - 129.886
Generation 51: Best latency = 120, Best rate = 3, Fitness std - (0.07355811417702066) Highest fitness for this gen - (0.20112781954887216), Temperature - 124.690
Generation 52: Best latency = 120, Best rate = 3, Fitness std - (0.07602434365022852) Highest fitness for this gen - (0.20112781954887216), Temperature - 119.703
Generation 53: Best latency = 120, Best rate = 3, Fitness std - (0.0797718843482117) Highest fitness for this gen - (0.20112781954887216), Temperature - 114.915
Generation 54: Best latency = 120, Best rate = 3, Fitness std - (0.0840614011109855) Highest fitness for this gen - (0.20112781954887216), Temperature - 110.318
Generation 55: Best latency = 120, Best rate = 3, Fitness std - (0.08732167460847716) Highest fitness for this gen - (0.20112781954887216), Temperature - 105.905
Generation 56: Best latency = 120, Best rate = 3, Fitness std - (0.08795648061847754) Highest fitness for this gen - (0.20112781954887216), Temperature - 101.669
Generation 57: Best latency = 120, Best rate = 3, Fitness std - (0.08890606268337052) Highest fitness for this gen - (0.20112781954887216), Temperature - 97.602
Generation 58: Best latency = 120, Best rate = 3, Fitness std - (0.08909312562714142) Highest fitness for this gen - (0.20112781954887216), Temperature - 93.698
Generation 59: Best latency = 120, Best rate = 3, Fitness std - (0.08950570992313024) Highest fitness for this gen - (0.20112781954887216), Temperature - 89.950
Generation 60: Best latency = 120, Best rate = 3, Fitness std - (0.08796047142113247) Highest fitness for this gen - (0.20112781954887216), Temperature - 86.352
Generation 61: Best latency = 120, Best rate = 3, Fitness std - (0.08884255595179241) Highest fitness for this gen - (0.20112781954887216), Temperature - 82.898
Generation 62: Best latency = 120, Best rate = 3, Fitness std - (0.0876775454596453) Highest fitness for this gen - (0.20112781954887216), Temperature - 79.582
Generation 63: Best latency = 120, Best rate = 3, Fitness std - (0.08749393528995049) Highest fitness for this gen - (0.20112781954887216), Temperature - 76.399
Generation 64: Best latency = 120, Best rate = 3, Fitness std - (0.08789162082839166) Highest fitness for this gen - (0.20112781954887216), Temperature - 73.343
Generation 65: Best latency = 120, Best rate = 3, Fitness std - (0.08646323258801487) Highest fitness for this gen - (0.20112781954887216), Temperature - 70.409
Generation 66: Best latency = 120, Best rate = 3, Fitness std - (0.08676862969715085) Highest fitness for this gen - (0.20112781954887216), Temperature - 67.593
Generation 67: Best latency = 120, Best rate = 3, Fitness std - (0.08731961393973306) Highest fitness for this gen - (0.20112781954887216), Temperature - 64.889
Generation 68: Best latency = 120, Best rate = 3, Fitness std - (0.08662844053484176) Highest fitness for this gen - (0.20112781954887216), Temperature - 62.294
Generation 69: Best latency = 120, Best rate = 3, Fitness std - (0.08688815199680565) Highest fitness for this gen - (0.20112781954887216), Temperature - 59.802
Generation 70: Best latency = 120, Best rate = 3, Fitness std - (0.08612178121296916) Highest fitness for this gen - (0.20112781954887216), Temperature - 57.410
Generation 71: Best latency = 120, Best rate = 3, Fitness std - (0.08576949684218363) Highest fitness for this gen - (0.20112781954887216), Temperature - 55.113
Generation 72: Best latency = 120, Best rate = 3, Fitness std - (0.08589971088460047) Highest fitness for this gen - (0.20112781954887216), Temperature - 52.909
Generation 73: Best latency = 120, Best rate = 3, Fitness std - (0.08565865919853992) Highest fitness for this gen - (0.20112781954887216), Temperature - 50.793
Generation 74: Best latency = 120, Best rate = 3, Fitness std - (0.08492954670639713) Highest fitness for this gen - (0.20112781954887216), Temperature - 48.761
Generation 75: Best latency = 120, Best rate = 3, Fitness std - (0.08414793988774755) Highest fitness for this gen - (0.20112781954887216), Temperature - 46.810
Generation 76: Best latency = 120, Best rate = 3, Fitness std - (0.08508872216668721) Highest fitness for this gen - (0.20112781954887216), Temperature - 44.938
Generation 77: Best latency = 120, Best rate = 3, Fitness std - (0.08466473496644734) Highest fitness for this gen - (0.20112781954887216), Temperature - 43.140
Generation 78: Best latency = 120, Best rate = 3, Fitness std - (0.08423084334746653) Highest fitness for this gen - (0.20112781954887216), Temperature - 41.415
Generation 79: Best latency = 120, Best rate = 3, Fitness std - (0.08485573992747605) Highest fitness for this gen - (0.20112781954887216), Temperature - 39.758
Generation 80: Best latency = 120, Best rate = 3, Fitness std - (0.08221910271706873) Highest fitness for this gen - (0.20112781954887216), Temperature - 38.168
Generation 81: Best latency = 120, Best rate = 3, Fitness std - (0.08392432575849891) Highest fitness for this gen - (0.20112781954887216), Temperature - 36.641
Generation 82: Best latency = 120, Best rate = 3, Fitness std - (0.08264242346525562) Highest fitness for this gen - (0.20112781954887216), Temperature - 35.176
Generation 83: Best latency = 120, Best rate = 3, Fitness std - (0.08383826467270665) Highest fitness for this gen - (0.20112781954887216), Temperature - 33.769
Generation 84: Best latency = 120, Best rate = 3, Fitness std - (0.08308484456155397) Highest fitness for this gen - (0.20112781954887216), Temperature - 32.418
Generation 85: Best latency = 120, Best rate = 3, Fitness std - (0.08262057963296049) Highest fitness for this gen - (0.20112781954887216), Temperature - 31.121
Generation 86: Best latency = 120, Best rate = 3, Fitness std - (0.08263320365740894) Highest fitness for this gen - (0.20112781954887216), Temperature - 29.876
Generation 87: Best latency = 120, Best rate = 3, Fitness std - (0.08230154291223489) Highest fitness for this gen - (0.20112781954887216), Temperature - 28.681
Generation 88: Best latency = 120, Best rate = 3, Fitness std - (0.08243277531698218) Highest fitness for this gen - (0.20112781954887216), Temperature - 27.534
Generation 89: Best latency = 120, Best rate = 3, Fitness std - (0.082723976184826) Highest fitness for this gen - (0.20112781954887216), Temperature - 26.433
Generation 90: Best latency = 120, Best rate = 3, Fitness std - (0.08318712760120205) Highest fitness for this gen - (0.20112781954887216), Temperature - 25.375
Generation 91: Best latency = 120, Best rate = 3, Fitness std - (0.08299545068840235) Highest fitness for this gen - (0.20112781954887216), Temperature - 24.360
Generation 92: Best latency = 120, Best rate = 3, Fitness std - (0.08245707060842783) Highest fitness for this gen - (0.20112781954887216), Temperature - 23.386
Generation 93: Best latency = 120, Best rate = 3, Fitness std - (0.08334176223434156) Highest fitness for this gen - (0.20112781954887216), Temperature - 22.450
Generation 94: Best latency = 120, Best rate = 3, Fitness std - (0.08290407203503289) Highest fitness for this gen - (0.20112781954887216), Temperature - 21.552
Generation 95: Best latency = 120, Best rate = 3, Fitness std - (0.08353580844474426) Highest fitness for this gen - (0.20112781954887216), Temperature - 20.690
Generation 96: Best latency = 120, Best rate = 3, Fitness std - (0.08261625111532057) Highest fitness for this gen - (0.20112781954887216), Temperature - 19.863
Generation 97: Best latency = 120, Best rate = 3, Fitness std - (0.08120033455020834) Highest fitness for this gen - (0.20112781954887216), Temperature - 19.068
Generation 98: Best latency = 120, Best rate = 3, Fitness std - (0.08240928497100983) Highest fitness for this gen - (0.20112781954887216), Temperature - 18.305
Generation 99: Best latency = 120, Best rate = 3, Fitness std - (0.08236766415751651) Highest fitness for this gen - (0.20112781954887216), Temperature - 17.573
Generation 100: Best latency = 120, Best rate = 3, Fitness std - (0.0825206323406744) Highest fitness for this gen - (0.20112781954887216), Temperature - 16.870
Generation 101: Best latency = 120, Best rate = 3, Fitness std - (0.08354478237828752) Highest fitness for this gen - (0.20112781954887216), Temperature - 16.196
Generation 102: Best latency = 120, Best rate = 3, Fitness std - (0.08263168864762398) Highest fitness for this gen - (0.20112781954887216), Temperature - 15.548
Generation 103: Best latency = 120, Best rate = 3, Fitness std - (0.08268553619706118) Highest fitness for this gen - (0.20112781954887216), Temperature - 14.926
Generation 104: Best latency = 120, Best rate = 3, Fitness std - (0.08292006026662457) Highest fitness for this gen - (0.20112781954887216), Temperature - 14.329
Generation 105: Best latency = 120, Best rate = 3, Fitness std - (0.08094859896238658) Highest fitness for this gen - (0.20112781954887216), Temperature - 13.756
Generation 106: Best latency = 120, Best rate = 3, Fitness std - (0.08211290766033294) Highest fitness for this gen - (0.20112781954887216), Temperature - 13.205
Generation 107: Best latency = 120, Best rate = 3, Fitness std - (0.08278058708651141) Highest fitness for this gen - (0.20112781954887216), Temperature - 12.677
Generation 108: Best latency = 120, Best rate = 3, Fitness std - (0.08263417934711931) Highest fitness for this gen - (0.20112781954887216), Temperature - 12.170
Generation 109: Best latency = 120, Best rate = 3, Fitness std - (0.08266350259409354) Highest fitness for this gen - (0.20112781954887216), Temperature - 11.683
Generation 110: Best latency = 120, Best rate = 3, Fitness std - (0.08272430512005749) Highest fitness for this gen - (0.20112781954887216), Temperature - 11.216
Generation 111: Best latency = 120, Best rate = 3, Fitness std - (0.08267077158937532) Highest fitness for this gen - (0.20112781954887216), Temperature - 10.767
Generation 112: Best latency = 120, Best rate = 3, Fitness std - (0.08291635321294305) Highest fitness for this gen - (0.20112781954887216), Temperature - 10.337
Generation 113: Best latency = 120, Best rate = 3, Fitness std - (0.08239757264218565) Highest fitness for this gen - (0.20112781954887216), Temperature - 9.923
Generation 114: Best latency = 120, Best rate = 3, Fitness std - (0.08305776204988123) Highest fitness for this gen - (0.20112781954887216), Temperature - 9.526
Generation 115: Best latency = 120, Best rate = 3, Fitness std - (0.08238122419443185) Highest fitness for this gen - (0.20112781954887216), Temperature - 9.145
Generation 116: Best latency = 120, Best rate = 3, Fitness std - (0.0834273558776901) Highest fitness for this gen - (0.20112781954887216), Temperature - 8.779
Generation 117: Best latency = 120, Best rate = 3, Fitness std - (0.08190293118109035) Highest fitness for this gen - (0.20112781954887216), Temperature - 8.428
Generation 118: Best latency = 120, Best rate = 3, Fitness std - (0.08266257601794721) Highest fitness for this gen - (0.20112781954887216), Temperature - 8.091
Generation 119: Best latency = 120, Best rate = 3, Fitness std - (0.08166784633636703) Highest fitness for this gen - (0.20112781954887216), Temperature - 7.767
Generation 120: Best latency = 120, Best rate = 3, Fitness std - (0.08089668796968513) Highest fitness for this gen - (0.20112781954887216), Temperature - 7.457
Generation 121: Best latency = 120, Best rate = 3, Fitness std - (0.08158714718038627) Highest fitness for this gen - (0.20112781954887216), Temperature - 7.158
Generation 122: Best latency = 120, Best rate = 3, Fitness std - (0.08286814376888552) Highest fitness for this gen - (0.20112781954887216), Temperature - 6.872
Generation 123: Best latency = 120, Best rate = 3, Fitness std - (0.08183227651093396) Highest fitness for this gen - (0.20112781954887216), Temperature - 6.597
Generation 124: Best latency = 120, Best rate = 3, Fitness std - (0.08303217540011514) Highest fitness for this gen - (0.20112781954887216), Temperature - 6.333
Generation 125: Best latency = 120, Best rate = 3, Fitness std - (0.08159401954924807) Highest fitness for this gen - (0.20112781954887216), Temperature - 6.080
```

```
Generation 126: Best latency = 120, Best rate = 3, Fitness std - (0.0814932039513077) Highest fitness for this gen - (0.20112781954887216), Temperature - 5.837
Generation 127: Best latency = 120, Best rate = 3, Fitness std - (0.08079498548808679) Highest fitness for this gen - (0.20112781954887216), Temperature - 5.603
Generation 128: Best latency = 120, Best rate = 3, Fitness std - (0.08242222067851133) Highest fitness for this gen - (0.20112781954887216), Temperature - 5.379
Generation 129: Best latency = 120, Best rate = 3, Fitness std - (0.0810172460325511) Highest fitness for this gen - (0.20112781954887216), Temperature - 5.164
Generation 130: Best latency = 120, Best rate = 3, Fitness std - (0.08147310889437748) Highest fitness for this gen - (0.20112781954887216), Temperature - 4.957
Generation 131: Best latency = 120, Best rate = 3, Fitness std - (0.0814932039513077) Highest fitness for this gen - (0.20112781954887216), Temperature - 4.759
Generation 132: Best latency = 120, Best rate = 3, Fitness std - (0.08173707918306528) Highest fitness for this gen - (0.20112781954887216), Temperature - 4.569
Generation 133: Best latency = 120, Best rate = 3, Fitness std - (0.08236201268659507) Highest fitness for this gen - (0.20112781954887216), Temperature - 4.386
Generation 134: Best latency = 120, Best rate = 3, Fitness std - (0.0825514506262846) Highest fitness for this gen - (0.20112781954887216), Temperature - 4.211
Generation 135: Best latency = 120, Best rate = 3, Fitness std - (0.08103803591713044) Highest fitness for this gen - (0.20112781954887216), Temperature - 4.042
Generation 136: Best latency = 120, Best rate = 3, Fitness std - (0.08062490212415517) Highest fitness for this gen - (0.20112781954887216), Temperature - 3.881
Generation 137: Best latency = 120, Best rate = 3, Fitness std - (0.08139133984375799) Highest fitness for this gen - (0.20112781954887216), Temperature - 3.725
Generation 138: Best latency = 120, Best rate = 3, Fitness std - (0.08200381261530898) Highest fitness for this gen - (0.20112781954887216), Temperature - 3.576
Generation 139: Best latency = 120, Best rate = 3, Fitness std - (0.08089469427661077) Highest fitness for this gen - (0.20112781954887216), Temperature - 3.433
Generation 140: Best latency = 120, Best rate = 3, Fitness std - (0.08135669303528563) Highest fitness for this gen - (0.20112781954887216), Temperature - 3.296
Generation 141: Best latency = 120, Best rate = 3, Fitness std - (0.08212470430505821) Highest fitness for this gen - (0.20112781954887216), Temperature - 3.164
Generation 142: Best latency = 120, Best rate = 3, Fitness std - (0.08182609694402802) Highest fitness for this gen - (0.20112781954887216), Temperature - 3.037
Generation 143: Best latency = 120, Best rate = 3, Fitness std - (0.0810094476727554) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.916
Generation 144: Best latency = 120, Best rate = 3, Fitness std - (0.07950424465470629) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.799
Generation 145: Best latency = 120, Best rate = 3, Fitness std - (0.08040764393905393) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.687
Generation 146: Best latency = 120, Best rate = 3, Fitness std - (0.0806107446416139) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.580
Generation 147: Best latency = 120, Best rate = 3, Fitness std - (0.08002692251382978) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.477
Generation 148: Best latency = 120, Best rate = 3, Fitness std - (0.08078187926843858) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.378
Generation 149: Best latency = 120, Best rate = 3, Fitness std - (0.07850538189285244) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.283
Generation 150: Best latency = 120, Best rate = 3, Fitness std - (0.08103434653725591) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.191
Generation 151: Best latency = 120, Best rate = 3, Fitness std - (0.08043554865873824) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.104
Generation 152: Best latency = 120, Best rate = 3, Fitness std - (0.07963632972041007) Highest fitness for this gen - (0.20112781954887216), Temperature - 2.019
Generation 153: Best latency = 120, Best rate = 3, Fitness std - (0.07857683964051464) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.939
Generation 154: Best latency = 120, Best rate = 3, Fitness std - (0.07983230368728024) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.861
Generation 155: Best latency = 120, Best rate = 3, Fitness std - (0.07989016419180207) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.787
Generation 156: Best latency = 120, Best rate = 3, Fitness std - (0.07885336461822677) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.715
Generation 157: Best latency = 120, Best rate = 3, Fitness std - (0.07891618088479813) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.647
Generation 158: Best latency = 120, Best rate = 3, Fitness std - (0.07873789924506383) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.581
Generation 159: Best latency = 120, Best rate = 3, Fitness std - (0.07837260558462024) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.517
Generation 160: Best latency = 120, Best rate = 3, Fitness std - (0.07927891155651856) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.457
Generation 161: Best latency = 120, Best rate = 3, Fitness std - (0.07842082086329345) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.399
Generation 162: Best latency = 120, Best rate = 3, Fitness std - (0.07748768544625387) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.343
Generation 163: Best latency = 120, Best rate = 3, Fitness std - (0.078512223565386) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.289
Generation 164: Best latency = 120, Best rate = 3, Fitness std - (0.07759210750709307) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.237
Generation 165: Best latency = 120, Best rate = 3, Fitness std - (0.07815125386534512) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.188
Generation 166: Best latency = 120, Best rate = 3, Fitness std - (0.07726034557088407) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.140
Generation 167: Best latency = 120, Best rate = 3, Fitness std - (0.07764802263741899) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.095
Generation 168: Best latency = 120, Best rate = 3, Fitness std - (0.07770056055205328) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.051
Generation 169: Best latency = 120, Best rate = 3, Fitness std - (0.07798098494893124) Highest fitness for this gen - (0.20112781954887216), Temperature - 1.009
Generation 170: Best latency = 120, Best rate = 3, Fitness std - (0.07687673084901402) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.969
Generation 171: Best latency = 120, Best rate = 3, Fitness std - (0.07544696703626344) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.930
Generation 172: Best latency = 120, Best rate = 3, Fitness std - (0.07665984885390692) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.893
Generation 173: Best latency = 120, Best rate = 3, Fitness std - (0.07497348821750344) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.857
Generation 174: Best latency = 120, Best rate = 3, Fitness std - (0.07455804781732864) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.823
Generation 175: Best latency = 120, Best rate = 3, Fitness std - (0.07468435299655803) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.790
Generation 176: Best latency = 120, Best rate = 3, Fitness std - (0.0761075106320209) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.758
Generation 177: Best latency = 120, Best rate = 3, Fitness std - (0.07462063245582841) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.728
Generation 178: Best latency = 120, Best rate = 3, Fitness std - (0.07242281536182664) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.699
Generation 179: Best latency = 120, Best rate = 3, Fitness std - (0.07405613520500268) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.671
Generation 180: Best latency = 120, Best rate = 3, Fitness std - (0.07290354325268192) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.644
Generation 181: Best latency = 120, Best rate = 3, Fitness std - (0.07225552999609357) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.618
Generation 182: Best latency = 120, Best rate = 3, Fitness std - (0.07275900252875649) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.593
Generation 183: Best latency = 120, Best rate = 3, Fitness std - (0.07134220130875367) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.570
Generation 184: Best latency = 120, Best rate = 3, Fitness std - (0.07106676444596546) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.547
Generation 185: Best latency = 120, Best rate = 3, Fitness std - (0.07030863644518039) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.525
Generation 186: Best latency = 120, Best rate = 3, Fitness std - (0.07038634296788004) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.504
Generation 187: Best latency = 120, Best rate = 3, Fitness std - (0.06891282409147835) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.484
Generation 188: Best latency = 120, Best rate = 3, Fitness std - (0.06945774747864529) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.465
Generation 189: Best latency = 120, Best rate = 3, Fitness std - (0.06936207132702647) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.446
Generation 190: Best latency = 120, Best rate = 3, Fitness std - (0.06780281868935996) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.428
Generation 191: Best latency = 120, Best rate = 3, Fitness std - (0.06792184560289348) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.411
Generation 192: Best latency = 120, Best rate = 3, Fitness std - (0.06707516522808876) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.395
Generation 193: Best latency = 120, Best rate = 3, Fitness std - (0.06568870706466724) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.379
Generation 194: Best latency = 120, Best rate = 3, Fitness std - (0.06731756651799559) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.364
Generation 195: Best latency = 120, Best rate = 3, Fitness std - (0.06402376525171573) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.349
Generation 196: Best latency = 120, Best rate = 3, Fitness std - (0.0641352732148634) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.335
Generation 197: Best latency = 120, Best rate = 3, Fitness std - (0.06440454625437005) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.322
Generation 198: Best latency = 120, Best rate = 3, Fitness std - (0.06121280962743216) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.309
Generation 199: Best latency = 120, Best rate = 3, Fitness std - (0.059941751771550485) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.296
Generation 200: Best latency = 120, Best rate = 3, Fitness std - (0.06167094579747869) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.285
Generation 201: Best latency = 120, Best rate = 3, Fitness std - (0.05848715008115689) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.273
Generation 202: Best latency = 120, Best rate = 3, Fitness std - (0.05856703859317285) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.262
Generation 203: Best latency = 120, Best rate = 3, Fitness std - (0.057656574752970695) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.252
Generation 204: Best latency = 120, Best rate = 3, Fitness std - (0.056468011912609056) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.242
Generation 205: Best latency = 120, Best rate = 3, Fitness std - (0.05522002475358778) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.232
Generation 206: Best latency = 120, Best rate = 3, Fitness std - (0.05461880313438499) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.223
Generation 207: Best latency = 120, Best rate = 3, Fitness std - (0.054105031506934166) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.214
Generation 208: Best latency = 120, Best rate = 3, Fitness std - (0.05143419954996802) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.205
Generation 209: Best latency = 120, Best rate = 3, Fitness std - (0.05035790210442429) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.197
Generation 210: Best latency = 120, Best rate = 3, Fitness std - (0.05117037811441187) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.189
Generation 211: Best latency = 120, Best rate = 3, Fitness std - (0.04857014211388995) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.182
Generation 212: Best latency = 120, Best rate = 3, Fitness std - (0.04786697298288999) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.174
Generation 213: Best latency = 120, Best rate = 3, Fitness std - (0.044768388588154975) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.167
Generation 214: Best latency = 120, Best rate = 3, Fitness std - (0.04437379234088538) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.161
Generation 215: Best latency = 120, Best rate = 3, Fitness std - (0.044010763216644676) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.154
Generation 216: Best latency = 120, Best rate = 3, Fitness std - (0.039761698031968995) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.148
Generation 217: Best latency = 120, Best rate = 3, Fitness std - (0.03774222600195656) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.142
Generation 218: Best latency = 120, Best rate = 3, Fitness std - (0.03805841391519737) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.136
Generation 219: Best latency = 120, Best rate = 3, Fitness std - (0.035666436199903874) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.131
Generation 220: Best latency = 120, Best rate = 3, Fitness std - (0.03517658635031652) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.126
Generation 221: Best latency = 120, Best rate = 3, Fitness std - (0.03271764633124281) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.121
Generation 222: Best latency = 120, Best rate = 3, Fitness std - (0.03254452054626544) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.116
Generation 223: Best latency = 120, Best rate = 3, Fitness std - (0.02968541599679031) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.111
Generation 224: Best latency = 120, Best rate = 3, Fitness std - (0.024090970145048213) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.107
Generation 225: Best latency = 120, Best rate = 3, Fitness std - (0.024059067943603733) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.103
Generation 226: Best latency = 120, Best rate = 3, Fitness std - (0.020552950745052305) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.098
Generation 227: Best latency = 120, Best rate = 3, Fitness std - (0.01787556545293129) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.095
Generation 228: Best latency = 120, Best rate = 3, Fitness std - (0.014333254231707058) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.091
Generation 229: Best latency = 120, Best rate = 3, Fitness std - (0.013408706083773253) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.087
Generation 230: Best latency = 120, Best rate = 3, Fitness std - (0.012460459002330242) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.084
Generation 231: Best latency = 120, Best rate = 3, Fitness std - (0.011431172543502469) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.080
Generation 232: Best latency = 120, Best rate = 3, Fitness std - (0.009850878565691437) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.077
Generation 233: Best latency = 120, Best rate = 3, Fitness std - (0.01181613571520363) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.074
Generation 234: Best latency = 120, Best rate = 3, Fitness std - (0.00938361093582909) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.071
Generation 235: Best latency = 120, Best rate = 3, Fitness std - (0.008096156188212742) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.068
Generation 236: Best latency = 120, Best rate = 3, Fitness std - (0.006202897590929386) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.065
Generation 237: Best latency = 120, Best rate = 3, Fitness std - (0.004991274826035416) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.063
Generation 238: Best latency = 120, Best rate = 3, Fitness std - (0.0021293519159027306) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.060
Generation 239: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.058
Generation 240: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.056
Generation 241: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.053
Generation 242: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.051
Generation 243: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.049
Generation 244: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.047
Generation 245: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.045
Generation 246: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.044
Generation 247: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.042
Generation 248: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.040
Generation 249: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.039
Generation 250: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.037
Generation 251: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.035
```

```
Generation 252: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.034
Generation 253: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.033
Generation 254: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.031
Generation 255: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.030
Generation 256: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.029
Generation 257: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.028
Generation 258: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.027
Generation 259: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.026
Generation 260: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.025
Generation 261: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.024
Generation 262: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.023
Generation 263: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.022
Generation 264: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.021
Generation 265: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.020
Generation 266: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.019
Generation 267: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.018
Generation 268: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.018
Generation 269: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.017
Generation 270: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.016
Generation 271: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.016
Generation 272: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.015
Generation 273: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.014
Generation 274: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.014
Generation 275: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.013
Generation 276: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.013
Generation 277: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.012
Generation 278: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.012
Generation 279: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.011
Generation 280: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.011
Generation 281: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.010
Generation 282: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.010
Generation 283: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.010
Generation 284: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.009
Generation 285: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.009
Generation 286: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.009
Generation 287: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.008
Generation 288: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.008
Generation 289: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.008
Generation 290: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.007
Generation 291: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.007
Generation 292: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.007
Generation 293: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.006
Generation 294: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.006
Generation 295: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.006
Generation 296: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.006
Generation 297: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.005
Generation 298: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.005
Generation 299: Best latency = 120, Best rate = 3, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.20112781954887216), Temperature - 0.005
```

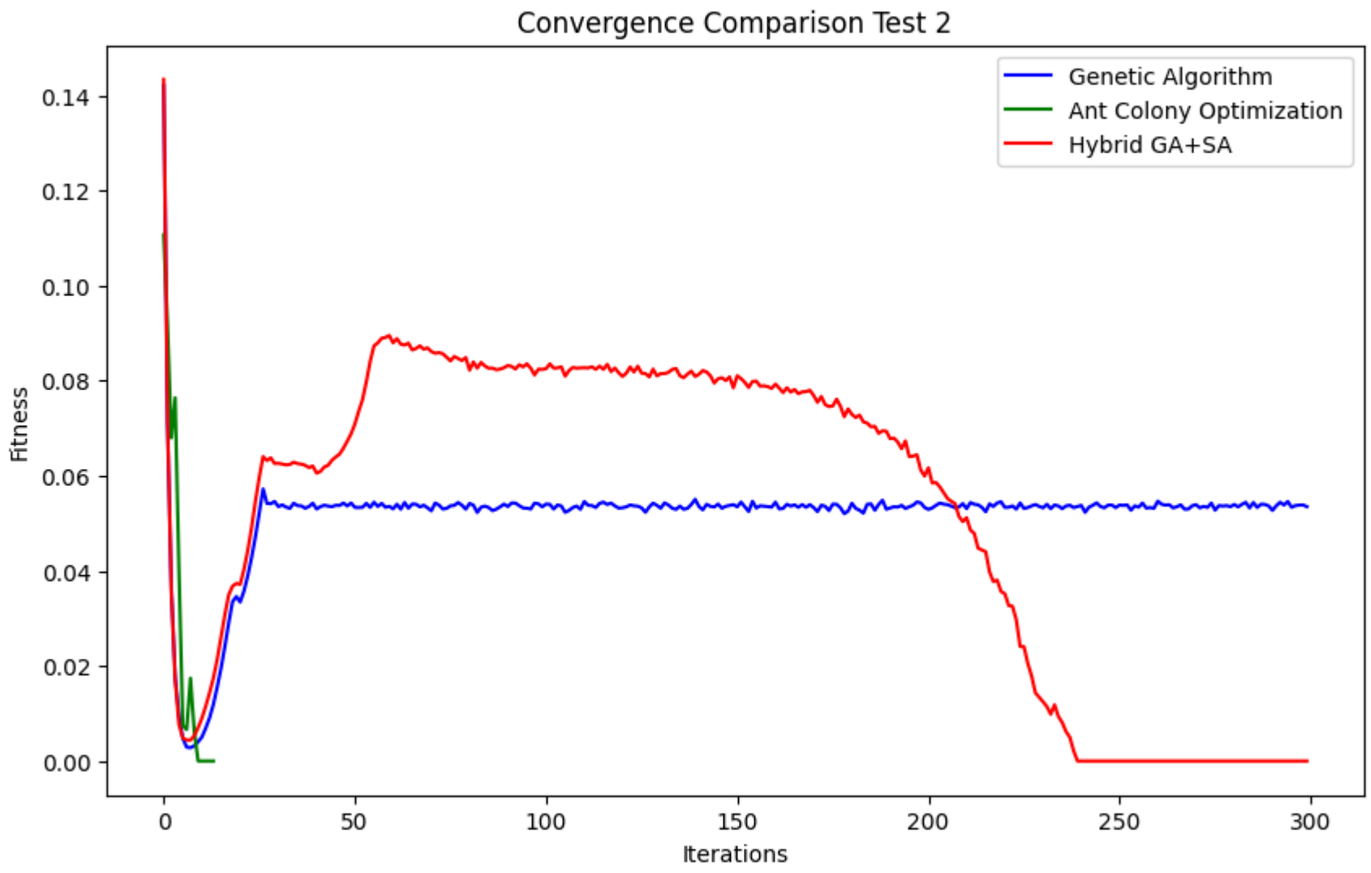





```
Best path [[115, 41, 61, 72, 'BS-1']]
Best latency 120
Best transmission rate 3
Execution time 40.776482820510864 seconds
```

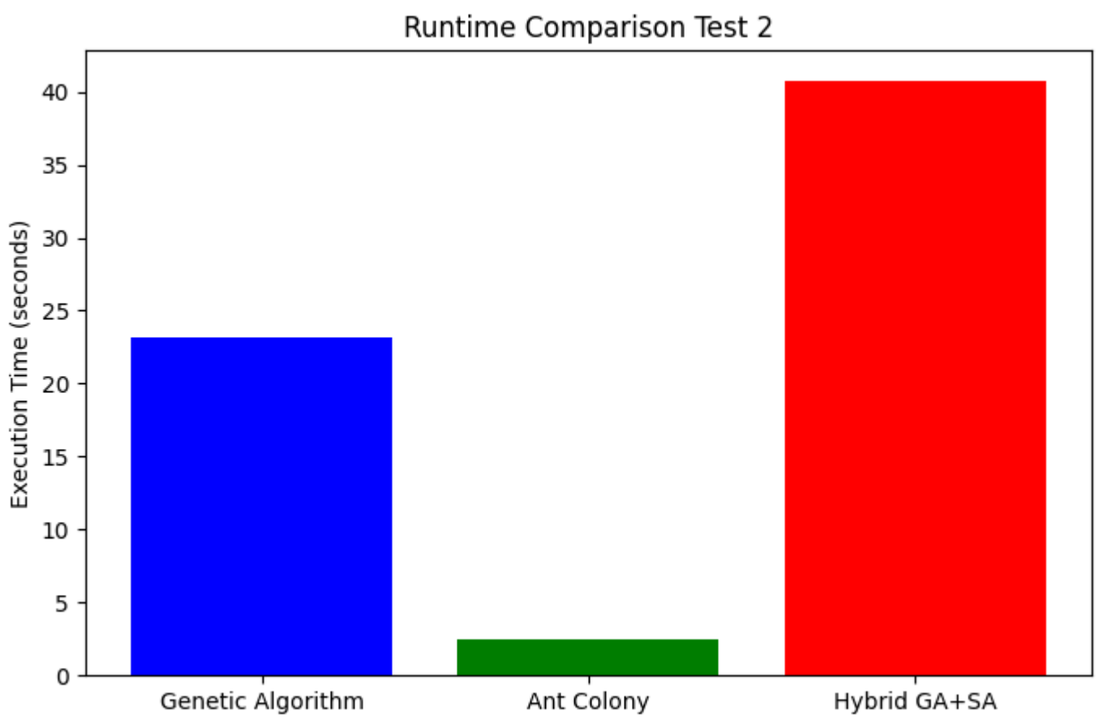
Convergence Curve (Test 2)

In [ ]:
```
plt.figure(figsize=(10, 6))
plt.plot(GA_metrics["Test2"]["Fitness_std"], label='Genetic Algorithm', color='blue')
plt.plot(ACO_metrics["Test2"]["Fitness_std"], label='Ant Colony Optimization', color='green')
plt.plot(HYBRID_metrics["Test2"]["Fitness_std"], label='Hybrid GA+SA', color='red')
plt.xlabel('Iterations')
plt.ylabel('Fitness')
plt.title('Convergence Comparison Test 2')
plt.legend()
plt.show()
```



Runtime Comparison (Test 2)

In [ ]:
```
algorithms = ['Genetic Algorithm', 'Ant Colony', 'Hybrid GA+SA']
runtimes = [
    GA_metrics["Test2"]["Execution_time"],
    ACO_metrics["Test2"]["Execution_time"],
    HYBRID_metrics["Test2"]["Execution_time"]
]

plt.figure(figsize=(8, 5))
plt.bar(algorithms, runtimes, color=['blue', 'green', 'red'])
plt.ylabel('Execution Time (seconds)')
plt.title('Runtime Comparison Test 2')
plt.show()
```



Results Comparison (Test 2)

In [ ]:
```
algorithms = ["Genetic algorithm", "Ant colony algorithm", "Hybrid GA + SA"]
best_paths = [
    GA_metrics["Test2"]["paths"],
    ACO_metrics["Test2"]["paths"],
    HYBRID_metrics["Test2"]["paths"]
]
best_latencies = [
    GA_metrics["Test2"]["latency"],
    ACO_metrics["Test2"]["latency"],
    HYBRID_metrics["Test2"]["latency"]
]
best_transmissions = [
    GA_metrics["Test2"]["transmission"],
    ACO_metrics["Test2"]["transmission"],
    HYBRID_metrics["Test2"]["transmission"]
]
final_score = [
    f"{GA_metrics['Test2']['Fitness'][-1]:.3f}",
    f"{ACO_metrics['Test2']['Fitness'][-1]:.3f}",
    f"{HYBRID_metrics['Test2']['Fitness'][-1]:.3f}"
]
final_fitness_std = [
    f"{GA_metrics['Test2']['Fitness_std'][-1]:.3f}",
    f"{ACO_metrics['Test2']['Fitness_std'][-1]:.3f}",
    f"{HYBRID_metrics['Test2']['Fitness_std'][-1]:.3f}",
]
all_gen_num = [
    GA_metrics["Test2"]["gen"],
    ACO_metrics["Test2"]["gen"],
    HYBRID_metrics["Test2"]["gen"],
]
all_execution_time = [
    f"{GA_metrics['Test2']['Execution_time']:.2f}",
    f"{ACO_metrics['Test2']['Execution_time']:.2f}",
    f"{HYBRID_metrics['Test2']['Execution_time']:.2f}",
]

index = [x for x in range(len(algorithms))]

pd.DataFrame({
    'Algorithms':algorithms,
    'Best path(s)':best_paths,
    'Best latency':best_latencies,
    'Best transmission rate':best_transmissions,
    'Final fitness score':final_score,
    'Final fitness std':final_fitness_std,
    "Number of iteration to converge": all_gen_num,
```

```
    "Execution time (seconds)": all_execution_time
    },index=index)
```

Out[ ]:

	Algorithms	Best path(s)	Best latency	Best transmission rate	Final fitness score	Final fitness std	Number of iteration to converge	Execution time (seconds)
0	Genetic algorithm	[[115, 112, 77, BS-1]]	90	2	0.133	0.053	300	23.15
1	Ant colony algorithm	[[115, 135, 35, 87, 10, 45, 62, 84, 77, 90, 53...	360	3	0.175	0.000	14	2.48
2	Hybrid GA + SA	[[115, 41, 61, 72, BS-1]]	120	3	0.201	0.000	300	40.78

Test 3 - (Starting Node: 29)

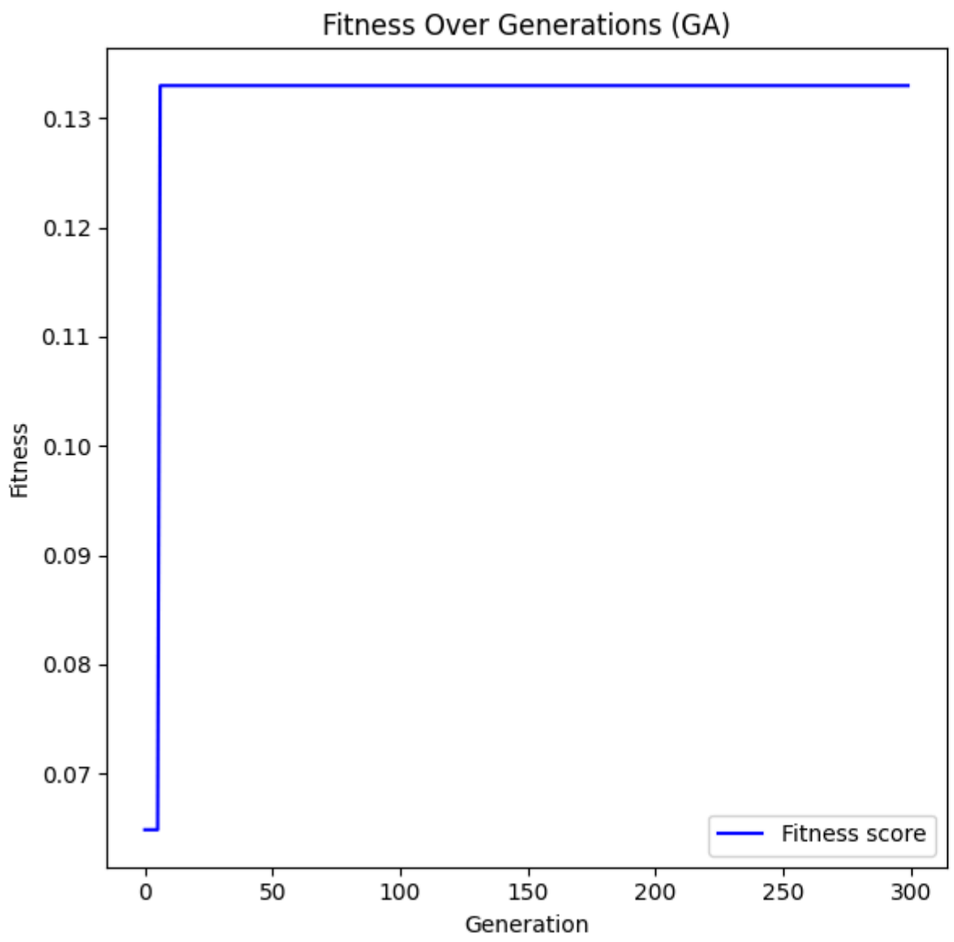
Discret Genetic Algorithm

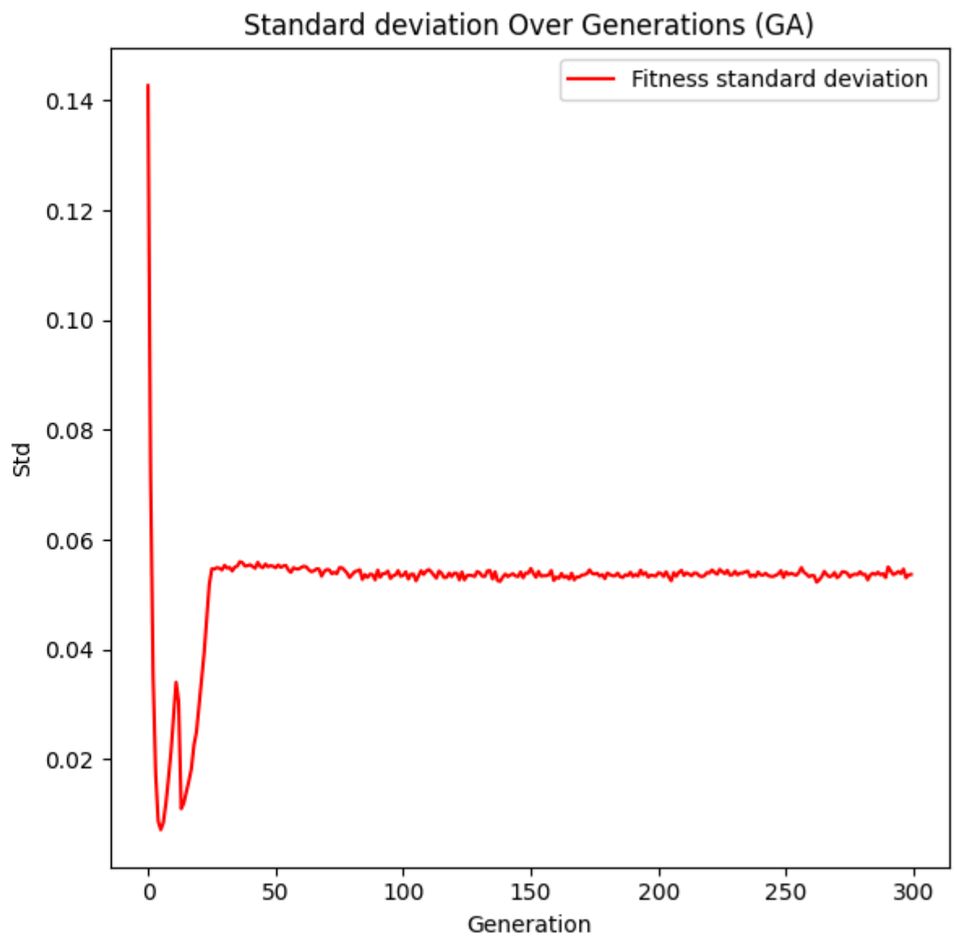
In [ ]:
```
DGA = Discrete_Genetic_algorithm(nodes=NODES, pop_size=9000, generations=GENERATIONS, mutation_rate=.6)

start_time = time.time()

best_path, best_latency, best_transmission_rate, all_best_paths, ga_fitness_score, ga_fitness_std, gen_num_GA = DGA.GA(starting_node=STARTING_NODE_3)

end_time = time.time()
execution_time = end_time - start_time

print("Total best paths", all_best_paths)
print("Best latency", best_latency)
print("Best transmission rate", best_transmission_rate)
print("Execution time", execution_time, "seconds")

GA_metrics["Test3"]["paths"] = all_best_paths
GA_metrics["Test3"]["latency"] = best_latency
GA_metrics["Test3"]["transmission"] = best_transmission_rate

GA_metrics["Test3"]["gen"] = gen_num_GA

GA_metrics["Test3"]["Fitness"] = ga_fitness_score
GA_metrics["Test3"]["Fitness_std"] = ga_fitness_std
GA_metrics["Test3"]["Execution_time"] = execution_time
```

```
Generation 0: Best latency = 60, Best rate = 1, Fitness std - (0.1426942330605486), Highest fitness for this gen - (0.06484962406015038)
Generation 1: Best latency = 60, Best rate = 1, Fitness std - (0.07205504185457265), Highest fitness for this gen - (0.06484962406015038)
Generation 2: Best latency = 60, Best rate = 1, Fitness std - (0.035273525490018255), Highest fitness for this gen - (0.06484962406015038)
Generation 3: Best latency = 60, Best rate = 1, Fitness std - (0.017321787140640026), Highest fitness for this gen - (0.06484962406015038)
Generation 4: Best latency = 60, Best rate = 1, Fitness std - (0.008706206321600019), Highest fitness for this gen - (0.06484962406015038)
Generation 5: Best latency = 60, Best rate = 1, Fitness std - (0.007168663089725876), Highest fitness for this gen - (0.06484962406015038)
Generation 6: Best latency = 90, Best rate = 2, Fitness std - (0.008461751907259538), Highest fitness for this gen - (0.13298872180451127)
Generation 7: Best latency = 90, Best rate = 2, Fitness std - (0.011923392513994413), Highest fitness for this gen - (0.13298872180451127)
Generation 8: Best latency = 90, Best rate = 2, Fitness std - (0.016421186823457293), Highest fitness for this gen - (0.13298872180451127)
Generation 9: Best latency = 90, Best rate = 2, Fitness std - (0.021287585314922777), Highest fitness for this gen - (0.13298872180451127)
Generation 10: Best latency = 90, Best rate = 2, Fitness std - (0.027887520576662297), Highest fitness for this gen - (0.13298872180451127)
Generation 11: Best latency = 90, Best rate = 2, Fitness std - (0.03402326228053313), Highest fitness for this gen - (0.13298872180451127)
Generation 12: Best latency = 90, Best rate = 2, Fitness std - (0.030523212271699726), Highest fitness for this gen - (0.13298872180451127)
Generation 13: Best latency = 90, Best rate = 2, Fitness std - (0.011000365598069243), Highest fitness for this gen - (0.13298872180451127)
Generation 14: Best latency = 90, Best rate = 2, Fitness std - (0.012001863188582934), Highest fitness for this gen - (0.13298872180451127)
Generation 15: Best latency = 90, Best rate = 2, Fitness std - (0.014013328224374087), Highest fitness for this gen - (0.13298872180451127)
Generation 16: Best latency = 90, Best rate = 2, Fitness std - (0.016002377766355685), Highest fitness for this gen - (0.13298872180451127)
Generation 17: Best latency = 90, Best rate = 2, Fitness std - (0.01829352715593208), Highest fitness for this gen - (0.13298872180451127)
Generation 18: Best latency = 90, Best rate = 2, Fitness std - (0.022583613579616156), Highest fitness for this gen - (0.13298872180451127)
Generation 19: Best latency = 90, Best rate = 2, Fitness std - (0.025007442116586804), Highest fitness for this gen - (0.13298872180451127)
Generation 20: Best latency = 90, Best rate = 2, Fitness std - (0.029770703691886172), Highest fitness for this gen - (0.13298872180451127)
Generation 21: Best latency = 90, Best rate = 2, Fitness std - (0.03451488131312266), Highest fitness for this gen - (0.13298872180451127)
Generation 22: Best latency = 90, Best rate = 2, Fitness std - (0.03937322169844985), Highest fitness for this gen - (0.13298872180451127)
Generation 23: Best latency = 90, Best rate = 2, Fitness std - (0.04568482044596139), Highest fitness for this gen - (0.13298872180451127)
Generation 24: Best latency = 90, Best rate = 2, Fitness std - (0.05181288114302824), Highest fitness for this gen - (0.13298872180451127)
Generation 25: Best latency = 90, Best rate = 2, Fitness std - (0.05468990801580523), Highest fitness for this gen - (0.13298872180451127)
Generation 26: Best latency = 90, Best rate = 2, Fitness std - (0.054638598236352656), Highest fitness for this gen - (0.13298872180451127)
Generation 27: Best latency = 90, Best rate = 2, Fitness std - (0.054908118584782464), Highest fitness for this gen - (0.13298872180451127)
Generation 28: Best latency = 90, Best rate = 2, Fitness std - (0.054791663756757376), Highest fitness for this gen - (0.13298872180451127)
Generation 29: Best latency = 90, Best rate = 2, Fitness std - (0.054484033919387796), Highest fitness for this gen - (0.13298872180451127)
Generation 30: Best latency = 90, Best rate = 2, Fitness std - (0.055319068091686324), Highest fitness for this gen - (0.13298872180451127)
Generation 31: Best latency = 90, Best rate = 2, Fitness std - (0.05482191099294255), Highest fitness for this gen - (0.13298872180451127)
Generation 32: Best latency = 90, Best rate = 2, Fitness std - (0.054928029596683595), Highest fitness for this gen - (0.13298872180451127)
Generation 33: Best latency = 90, Best rate = 2, Fitness std - (0.05429470592942475), Highest fitness for this gen - (0.13298872180451127)
Generation 34: Best latency = 90, Best rate = 2, Fitness std - (0.05499608944159096), Highest fitness for this gen - (0.13298872180451127)
Generation 35: Best latency = 90, Best rate = 2, Fitness std - (0.05519575039509839), Highest fitness for this gen - (0.13298872180451127)
Generation 36: Best latency = 90, Best rate = 2, Fitness std - (0.055987431889705695), Highest fitness for this gen - (0.13298872180451127)
Generation 37: Best latency = 90, Best rate = 2, Fitness std - (0.05580317604798203), Highest fitness for this gen - (0.13298872180451127)
Generation 38: Best latency = 90, Best rate = 2, Fitness std - (0.05518327648416082), Highest fitness for this gen - (0.13298872180451127)
Generation 39: Best latency = 90, Best rate = 2, Fitness std - (0.055263766612734216), Highest fitness for this gen - (0.13298872180451127)
Generation 40: Best latency = 90, Best rate = 2, Fitness std - (0.05543466154564554), Highest fitness for this gen - (0.13298872180451127)
Generation 41: Best latency = 90, Best rate = 2, Fitness std - (0.0550921574962017), Highest fitness for this gen - (0.13298872180451127)
Generation 42: Best latency = 90, Best rate = 2, Fitness std - (0.05478467435531298), Highest fitness for this gen - (0.13298872180451127)
Generation 43: Best latency = 90, Best rate = 2, Fitness std - (0.05586330577150654), Highest fitness for this gen - (0.13298872180451127)
Generation 44: Best latency = 90, Best rate = 2, Fitness std - (0.055150093534194804), Highest fitness for this gen - (0.13298872180451127)
Generation 45: Best latency = 90, Best rate = 2, Fitness std - (0.054887080308411926), Highest fitness for this gen - (0.13298872180451127)
Generation 46: Best latency = 90, Best rate = 2, Fitness std - (0.05556151838748471), Highest fitness for this gen - (0.13298872180451127)
Generation 47: Best latency = 90, Best rate = 2, Fitness std - (0.05507186205243013), Highest fitness for this gen - (0.13298872180451127)
Generation 48: Best latency = 90, Best rate = 2, Fitness std - (0.05526421568153699), Highest fitness for this gen - (0.13298872180451127)
Generation 49: Best latency = 90, Best rate = 2, Fitness std - (0.0551603853811257), Highest fitness for this gen - (0.13298872180451127)
Generation 50: Best latency = 90, Best rate = 2, Fitness std - (0.05488342624317906), Highest fitness for this gen - (0.13298872180451127)
Generation 51: Best latency = 90, Best rate = 2, Fitness std - (0.05540340559909497), Highest fitness for this gen - (0.13298872180451127)
Generation 52: Best latency = 90, Best rate = 2, Fitness std - (0.05495740981363693), Highest fitness for this gen - (0.13298872180451127)
Generation 53: Best latency = 90, Best rate = 2, Fitness std - (0.05520672286902214), Highest fitness for this gen - (0.13298872180451127)
Generation 54: Best latency = 90, Best rate = 2, Fitness std - (0.05530712651991738), Highest fitness for this gen - (0.13298872180451127)
Generation 55: Best latency = 90, Best rate = 2, Fitness std - (0.05447055635108082), Highest fitness for this gen - (0.13298872180451127)
Generation 56: Best latency = 90, Best rate = 2, Fitness std - (0.054036546426564434), Highest fitness for this gen - (0.13298872180451127)
Generation 57: Best latency = 90, Best rate = 2, Fitness std - (0.05489714092861812), Highest fitness for this gen - (0.13298872180451127)
Generation 58: Best latency = 90, Best rate = 2, Fitness std - (0.05466291893482438), Highest fitness for this gen - (0.13298872180451127)
Generation 59: Best latency = 90, Best rate = 2, Fitness std - (0.0546506783949658), Highest fitness for this gen - (0.13298872180451127)
Generation 60: Best latency = 90, Best rate = 2, Fitness std - (0.054933060687190775), Highest fitness for this gen - (0.13298872180451127)
Generation 61: Best latency = 90, Best rate = 2, Fitness std - (0.05516454462901858), Highest fitness for this gen - (0.13298872180451127)
Generation 62: Best latency = 90, Best rate = 2, Fitness std - (0.05503158750738313), Highest fitness for this gen - (0.13298872180451127)
Generation 63: Best latency = 90, Best rate = 2, Fitness std - (0.054505125492594425), Highest fitness for this gen - (0.13298872180451127)
Generation 64: Best latency = 90, Best rate = 2, Fitness std - (0.05416676609096055), Highest fitness for this gen - (0.13298872180451127)
Generation 65: Best latency = 90, Best rate = 2, Fitness std - (0.05432201457348341), Highest fitness for this gen - (0.13298872180451127)
Generation 66: Best latency = 90, Best rate = 2, Fitness std - (0.05467785993751894), Highest fitness for this gen - (0.13298872180451127)
Generation 67: Best latency = 90, Best rate = 2, Fitness std - (0.05465019143641018), Highest fitness for this gen - (0.13298872180451127)
Generation 68: Best latency = 90, Best rate = 2, Fitness std - (0.05335535259740723), Highest fitness for this gen - (0.13298872180451127)
Generation 69: Best latency = 90, Best rate = 2, Fitness std - (0.054180092527642054), Highest fitness for this gen - (0.13298872180451127)
Generation 70: Best latency = 90, Best rate = 2, Fitness std - (0.05456014362020036), Highest fitness for this gen - (0.13298872180451127)
Generation 71: Best latency = 90, Best rate = 2, Fitness std - (0.054514164716144335), Highest fitness for this gen - (0.13298872180451127)
Generation 72: Best latency = 90, Best rate = 2, Fitness std - (0.05379827573593073), Highest fitness for this gen - (0.13298872180451127)
Generation 73: Best latency = 90, Best rate = 2, Fitness std - (0.054047616716779656), Highest fitness for this gen - (0.13298872180451127)
Generation 74: Best latency = 90, Best rate = 2, Fitness std - (0.05380317364318867), Highest fitness for this gen - (0.13298872180451127)
Generation 75: Best latency = 90, Best rate = 2, Fitness std - (0.054902281546602635), Highest fitness for this gen - (0.13298872180451127)
Generation 76: Best latency = 90, Best rate = 2, Fitness std - (0.0548730663801302), Highest fitness for this gen - (0.13298872180451127)
Generation 77: Best latency = 90, Best rate = 2, Fitness std - (0.0544393392719626), Highest fitness for this gen - (0.13298872180451127)
Generation 78: Best latency = 90, Best rate = 2, Fitness std - (0.053897918101831775), Highest fitness for this gen - (0.13298872180451127)
Generation 79: Best latency = 90, Best rate = 2, Fitness std - (0.053111763390334146), Highest fitness for this gen - (0.13298872180451127)
Generation 80: Best latency = 90, Best rate = 2, Fitness std - (0.05362968261273655), Highest fitness for this gen - (0.13298872180451127)
Generation 81: Best latency = 90, Best rate = 2, Fitness std - (0.05418222696439176), Highest fitness for this gen - (0.13298872180451127)
Generation 82: Best latency = 90, Best rate = 2, Fitness std - (0.05429548552179423), Highest fitness for this gen - (0.13298872180451127)
Generation 83: Best latency = 90, Best rate = 2, Fitness std - (0.05451842814412963), Highest fitness for this gen - (0.13298872180451127)
Generation 84: Best latency = 90, Best rate = 2, Fitness std - (0.05274690289179299), Highest fitness for this gen - (0.13298872180451127)
Generation 85: Best latency = 90, Best rate = 2, Fitness std - (0.05363900441054717), Highest fitness for this gen - (0.13298872180451127)
Generation 86: Best latency = 90, Best rate = 2, Fitness std - (0.05311349189153419), Highest fitness for this gen - (0.13298872180451127)
Generation 87: Best latency = 90, Best rate = 2, Fitness std - (0.053724809820270544), Highest fitness for this gen - (0.13298872180451127)
Generation 88: Best latency = 90, Best rate = 2, Fitness std - (0.053640695959677374), Highest fitness for this gen - (0.13298872180451127)
Generation 89: Best latency = 90, Best rate = 2, Fitness std - (0.0526767597512284), Highest fitness for this gen - (0.13298872180451127)
Generation 90: Best latency = 90, Best rate = 2, Fitness std - (0.054463201336532104), Highest fitness for this gen - (0.13298872180451127)
Generation 91: Best latency = 90, Best rate = 2, Fitness std - (0.05316331969836408), Highest fitness for this gen - (0.13298872180451127)
Generation 92: Best latency = 90, Best rate = 2, Fitness std - (0.05378973977267686), Highest fitness for this gen - (0.13298872180451127)
Generation 93: Best latency = 90, Best rate = 2, Fitness std - (0.053849303435361824), Highest fitness for this gen - (0.13298872180451127)
Generation 94: Best latency = 90, Best rate = 2, Fitness std - (0.054272622110120926), Highest fitness for this gen - (0.13298872180451127)
Generation 95: Best latency = 90, Best rate = 2, Fitness std - (0.052952494551855606), Highest fitness for this gen - (0.13298872180451127)
Generation 96: Best latency = 90, Best rate = 2, Fitness std - (0.05343223480541811), Highest fitness for this gen - (0.13298872180451127)
Generation 97: Best latency = 90, Best rate = 2, Fitness std - (0.05358823016949162), Highest fitness for this gen - (0.13298872180451127)
Generation 98: Best latency = 90, Best rate = 2, Fitness std - (0.05435809478942213), Highest fitness for this gen - (0.13298872180451127)
Generation 99: Best latency = 90, Best rate = 2, Fitness std - (0.05333001158873323), Highest fitness for this gen - (0.13298872180451127)
Generation 100: Best latency = 90, Best rate = 2, Fitness std - (0.0535079317992398), Highest fitness for this gen - (0.13298872180451127)
Generation 101: Best latency = 90, Best rate = 2, Fitness std - (0.054135966647485344), Highest fitness for this gen - (0.13298872180451127)
Generation 102: Best latency = 90, Best rate = 2, Fitness std - (0.05282624043034423), Highest fitness for this gen - (0.13298872180451127)
Generation 103: Best latency = 90, Best rate = 2, Fitness std - (0.053691385391004216), Highest fitness for this gen - (0.13298872180451127)
Generation 104: Best latency = 90, Best rate = 2, Fitness std - (0.05371921942049597), Highest fitness for this gen - (0.13298872180451127)
Generation 105: Best latency = 90, Best rate = 2, Fitness std - (0.052504421582513276), Highest fitness for this gen - (0.13298872180451127)
Generation 106: Best latency = 90, Best rate = 2, Fitness std - (0.05343764364780 11), Highest fitness for this gen - (0.13298872180451127)
Generation 107: Best latency = 90, Best rate = 2, Fitness std - (0.054335257722738575), Highest fitness for this gen - (0.13298872180451127)
Generation 108: Best latency = 90, Best rate = 2, Fitness std - (0.053661828488551284), Highest fitness for this gen - (0.13298872180451127)
Generation 109: Best latency = 90, Best rate = 2, Fitness std - (0.05426743403343707), Highest fitness for this gen - (0.13298872180451127)
Generation 110: Best latency = 90, Best rate = 2, Fitness std - (0.05456436420659404), Highest fitness for this gen - (0.13298872180451127)
Generation 111: Best latency = 90, Best rate = 2, Fitness std - (0.05411751301953464), Highest fitness for this gen - (0.13298872180451127)
Generation 112: Best latency = 90, Best rate = 2, Fitness std - (0.0535219642303685), Highest fitness for this gen - (0.13298872180451127)
Generation 113: Best latency = 90, Best rate = 2, Fitness std - (0.05299712385888677), Highest fitness for this gen - (0.13298872180451127)
Generation 114: Best latency = 90, Best rate = 2, Fitness std - (0.05428101483418094), Highest fitness for this gen - (0.13298872180451127)
Generation 115: Best latency = 90, Best rate = 2, Fitness std - (0.05389769196630346), Highest fitness for this gen - (0.13298872180451127)
Generation 116: Best latency = 90, Best rate = 2, Fitness std - (0.05309731412175021), Highest fitness for this gen - (0.13298872180451127)
Generation 117: Best latency = 90, Best rate = 2, Fitness std - (0.05314012386060445), Highest fitness for this gen - (0.13298872180451127)
Generation 118: Best latency = 90, Best rate = 2, Fitness std - (0.05391149209205619), Highest fitness for this gen - (0.13298872180451127)
Generation 119: Best latency = 90, Best rate = 2, Fitness std - (0.05328226721902888), Highest fitness for this gen - (0.13298872180451127)
Generation 120: Best latency = 90, Best rate = 2, Fitness std - (0.05391149209205618), Highest fitness for this gen - (0.13298872180451127)
Generation 121: Best latency = 90, Best rate = 2, Fitness std - (0.053870067077962006), Highest fitness for this gen - (0.13298872180451127)
Generation 122: Best latency = 90, Best rate = 2, Fitness std - (0.05267947403047019), Highest fitness for this gen - (0.13298872180451127)
Generation 123: Best latency = 90, Best rate = 2, Fitness std - (0.05411751301953463), Highest fitness for this gen - (0.13298872180451127)
Generation 124: Best latency = 90, Best rate = 2, Fitness std - (0.0526940044054079), Highest fitness for this gen - (0.13298872180451127)
Generation 125: Best latency = 90, Best rate = 2, Fitness std - (0.0534517180996126), Highest fitness for this gen - (0.13298872180451127)
```

```
Generation 126: Best latency = 90, Best rate = 2, Fitness std - (0.05353598827719842), Highest fitness for this gen - (0.13298872180451127)
Generation 127: Best latency = 90, Best rate = 2, Fitness std - (0.0534517180996126), Highest fitness for this gen - (0.13298872180451127)
Generation 128: Best latency = 90, Best rate = 2, Fitness std - (0.05329643470373042), Highest fitness for this gen - (0.13298872180451127)
Generation 129: Best latency = 90, Best rate = 2, Fitness std - (0.05419940925617936), Highest fitness for this gen - (0.13298872180451127)
Generation 130: Best latency = 90, Best rate = 2, Fitness std - (0.053952843271250814), Highest fitness for this gen - (0.13298872180451127)
Generation 131: Best latency = 90, Best rate = 2, Fitness std - (0.053324744206336344), Highest fitness for this gen - (0.13298872180451127)
Generation 132: Best latency = 90, Best rate = 2, Fitness std - (0.05331059369724495), Highest fitness for this gen - (0.13298872180451127)
Generation 133: Best latency = 90, Best rate = 2, Fitness std - (0.05430815232930529), Highest fitness for this gen - (0.13298872180451127)
Generation 134: Best latency = 90, Best rate = 2, Fitness std - (0.05436233106249908), Highest fitness for this gen - (0.13298872180451127)
Generation 135: Best latency = 90, Best rate = 2, Fitness std - (0.05283882613182437), Highest fitness for this gen - (0.13298872180451127)
Generation 136: Best latency = 90, Best rate = 2, Fitness std - (0.054158497552488076), Highest fitness for this gen - (0.13298872180451127)
Generation 137: Best latency = 90, Best rate = 2, Fitness std - (0.052533685458339266), Highest fitness for this gen - (0.13298872180451127)
Generation 138: Best latency = 90, Best rate = 2, Fitness std - (0.05240171786567086), Highest fitness for this gen - (0.13298872180451127)
Generation 139: Best latency = 90, Best rate = 2, Fitness std - (0.053211302190192766), Highest fitness for this gen - (0.13298872180451127)
Generation 140: Best latency = 90, Best rate = 2, Fitness std - (0.05373144825068602), Highest fitness for this gen - (0.13298872180451127)
Generation 141: Best latency = 90, Best rate = 2, Fitness std - (0.05326809123636526), Highest fitness for this gen - (0.13298872180451127)
Generation 142: Best latency = 90, Best rate = 2, Fitness std - (0.05357801017771353), Highest fitness for this gen - (0.13298872180451127)
Generation 143: Best latency = 90, Best rate = 2, Fitness std - (0.053786994862387784), Highest fitness for this gen - (0.13298872180451127)
Generation 144: Best latency = 90, Best rate = 2, Fitness std - (0.05339536972139583), Highest fitness for this gen - (0.13298872180451127)
Generation 145: Best latency = 90, Best rate = 2, Fitness std - (0.053282267219028887), Highest fitness for this gen - (0.13298872180451127)
Generation 146: Best latency = 90, Best rate = 2, Fitness std - (0.05411751301953464), Highest fitness for this gen - (0.13298872180451127)
Generation 147: Best latency = 90, Best rate = 2, Fitness std - (0.05305442709915784), Highest fitness for this gen - (0.13298872180451127)
Generation 148: Best latency = 90, Best rate = 2, Fitness std - (0.0539390677382419), Highest fitness for this gen - (0.13298872180451127)
Generation 149: Best latency = 90, Best rate = 2, Fitness std - (0.05389769196630346), Highest fitness for this gen - (0.13298872180451127)
Generation 150: Best latency = 90, Best rate = 2, Fitness std - (0.054751321728708655), Highest fitness for this gen - (0.13298872180451127)
Generation 151: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 152: Best latency = 90, Best rate = 2, Fitness std - (0.053154376631675784), Highest fitness for this gen - (0.13298872180451127)
Generation 153: Best latency = 90, Best rate = 2, Fitness std - (0.05422664334602011), Highest fitness for this gen - (0.13298872180451127)
Generation 154: Best latency = 90, Best rate = 2, Fitness std - (0.053353019798248906), Highest fitness for this gen - (0.13298872180451127)
Generation 155: Best latency = 90, Best rate = 2, Fitness std - (0.0534376436478011), Highest fitness for this gen - (0.13298872180451127)
Generation 156: Best latency = 90, Best rate = 2, Fitness std - (0.05342356077193878), Highest fitness for this gen - (0.13298872180451127)
Generation 157: Best latency = 90, Best rate = 2, Fitness std - (0.053592000753112316), Highest fitness for this gen - (0.13298872180451127)
Generation 158: Best latency = 90, Best rate = 2, Fitness std - (0.05437585572721609), Highest fitness for this gen - (0.13298872180451127)
Generation 159: Best latency = 90, Best rate = 2, Fitness std - (0.052562913868393034), Highest fitness for this gen - (0.13298872180451127)
Generation 160: Best latency = 90, Best rate = 2, Fitness std - (0.053182856502350065), Highest fitness for this gen - (0.13298872180451127)
Generation 161: Best latency = 90, Best rate = 2, Fitness std - (0.0529684204515271), Highest fitness for this gen - (0.13298872180451127)
Generation 162: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 163: Best latency = 90, Best rate = 2, Fitness std - (0.05311159261254806), Highest fitness for this gen - (0.13298872180451127)
Generation 164: Best latency = 90, Best rate = 2, Fitness std - (0.05338126153335472), Highest fitness for this gen - (0.13298872180451127)
Generation 165: Best latency = 90, Best rate = 2, Fitness std - (0.05280993172584953), Highest fitness for this gen - (0.13298872180451127)
Generation 166: Best latency = 90, Best rate = 2, Fitness std - (0.05389769196630346), Highest fitness for this gen - (0.13298872180451127)
Generation 167: Best latency = 90, Best rate = 2, Fitness std - (0.05267947403047019), Highest fitness for this gen - (0.13298872180451127)
Generation 168: Best latency = 90, Best rate = 2, Fitness std - (0.053211302190192766), Highest fitness for this gen - (0.13298872180451127)
Generation 169: Best latency = 90, Best rate = 2, Fitness std - (0.05319708361569663), Highest fitness for this gen - (0.13298872180451127)
Generation 170: Best latency = 90, Best rate = 2, Fitness std - (0.053564011244299936), Highest fitness for this gen - (0.13298872180451127)
Generation 171: Best latency = 90, Best rate = 2, Fitness std - (0.053619956856031034), Highest fitness for this gen - (0.13298872180451127)
Generation 172: Best latency = 90, Best rate = 2, Fitness std - (0.05381471857679947), Highest fitness for this gen - (0.13298872180451127)
Generation 173: Best latency = 90, Best rate = 2, Fitness std - (0.05451066360318929), Highest fitness for this gen - (0.13298872180451127)
Generation 174: Best latency = 90, Best rate = 2, Fitness std - (0.053870067077962006), Highest fitness for this gen - (0.13298872180451127)
Generation 175: Best latency = 90, Best rate = 2, Fitness std - (0.0537175408972748), Highest fitness for this gen - (0.13298872180451127)
Generation 176: Best latency = 90, Best rate = 2, Fitness std - (0.05388388363037656), Highest fitness for this gen - (0.13298872180451127)
Generation 177: Best latency = 90, Best rate = 2, Fitness std - (0.053353019798248906), Highest fitness for this gen - (0.13298872180451127)
Generation 178: Best latency = 90, Best rate = 2, Fitness std - (0.05364787960527988), Highest fitness for this gen - (0.13298872180451127)
Generation 179: Best latency = 90, Best rate = 2, Fitness std - (0.05291090978142133), Highest fitness for this gen - (0.13298872180451127)
Generation 180: Best latency = 90, Best rate = 2, Fitness std - (0.0539390677382419), Highest fitness for this gen - (0.13298872180451127)
Generation 181: Best latency = 90, Best rate = 2, Fitness std - (0.053633922396606894), Highest fitness for this gen - (0.13298872180451127)
Generation 182: Best latency = 90, Best rate = 2, Fitness std - (0.053423560771938786), Highest fitness for this gen - (0.13298872180451127)
Generation 183: Best latency = 90, Best rate = 2, Fitness std - (0.053409469465361924), Highest fitness for this gen - (0.13298872180451127)
Generation 184: Best latency = 90, Best rate = 2, Fitness std - (0.05391149209205619), Highest fitness for this gen - (0.13298872180451127)
Generation 185: Best latency = 90, Best rate = 2, Fitness std - (0.05326809123636526), Highest fitness for this gen - (0.13298872180451127)
Generation 186: Best latency = 90, Best rate = 2, Fitness std - (0.05311159261254805), Highest fitness for this gen - (0.13298872180451127)
Generation 187: Best latency = 90, Best rate = 2, Fitness std - (0.05360598297704039), Highest fitness for this gen - (0.13298872180451127)
Generation 188: Best latency = 90, Best rate = 2, Fitness std - (0.05346578413402597), Highest fitness for this gen - (0.13298872180451127)
Generation 189: Best latency = 90, Best rate = 2, Fitness std - (0.054062753399310305), Highest fitness for this gen - (0.13298872180451127)
Generation 190: Best latency = 90, Best rate = 2, Fitness std - (0.05328226721902888), Highest fitness for this gen - (0.13298872180451127)
Generation 191: Best latency = 90, Best rate = 2, Fitness std - (0.05370362525080933), Highest fitness for this gen - (0.13298872180451127)
Generation 192: Best latency = 90, Best rate = 2, Fitness std - (0.05333888623775719), Highest fitness for this gen - (0.13298872180451127)
Generation 193: Best latency = 90, Best rate = 2, Fitness std - (0.054443359237645914), Highest fitness for this gen - (0.13298872180451127)
Generation 194: Best latency = 90, Best rate = 2, Fitness std - (0.05361995685603104), Highest fitness for this gen - (0.13298872180451127)
Generation 195: Best latency = 90, Best rate = 2, Fitness std - (0.05400786361563955), Highest fitness for this gen - (0.13298872180451127)
Generation 196: Best latency = 90, Best rate = 2, Fitness std - (0.05316862084329847), Highest fitness for this gen - (0.13298872180451127)
Generation 197: Best latency = 90, Best rate = 2, Fitness std - (0.05355000394631676), Highest fitness for this gen - (0.13298872180451127)
Generation 198: Best latency = 90, Best rate = 2, Fitness std - (0.0534376436478011), Highest fitness for this gen - (0.13298872180451127)
Generation 199: Best latency = 90, Best rate = 2, Fitness std - (0.053353019798248906), Highest fitness for this gen - (0.13298872180451127)
Generation 200: Best latency = 90, Best rate = 2, Fitness std - (0.053980369788464495), Highest fitness for this gen - (0.13298872180451127)
Generation 201: Best latency = 90, Best rate = 2, Fitness std - (0.05385624230273613), Highest fitness for this gen - (0.13298872180451127)
Generation 202: Best latency = 90, Best rate = 2, Fitness std - (0.05329643470373041), Highest fitness for this gen - (0.13298872180451127)
Generation 203: Best latency = 90, Best rate = 2, Fitness std - (0.054144844141804836), Highest fitness for this gen - (0.13298872180451127)
Generation 204: Best latency = 90, Best rate = 2, Fitness std - (0.053225512232681634), Highest fitness for this gen - (0.13298872180451127)
Generation 205: Best latency = 90, Best rate = 2, Fitness std - (0.052504421582513276), Highest fitness for this gen - (0.13298872180451127)
Generation 206: Best latency = 90, Best rate = 2, Fitness std - (0.05399412078518437), Highest fitness for this gen - (0.13298872180451127)
Generation 207: Best latency = 90, Best rate = 2, Fitness std - (0.05333888623775719), Highest fitness for this gen - (0.13298872180451127)
Generation 208: Best latency = 90, Best rate = 2, Fitness std - (0.054049043171690445), Highest fitness for this gen - (0.13298872180451127)
Generation 209: Best latency = 90, Best rate = 2, Fitness std - (0.05441638177231396), Highest fitness for this gen - (0.13298872180451127)
Generation 210: Best latency = 90, Best rate = 2, Fitness std - (0.05347984175768293), Highest fitness for this gen - (0.13298872180451127)
Generation 211: Best latency = 90, Best rate = 2, Fitness std - (0.05396661061923747), Highest fitness for this gen - (0.13298872180451127)
Generation 212: Best latency = 90, Best rate = 2, Fitness std - (0.0537175408972748), Highest fitness for this gen - (0.13298872180451127)
Generation 213: Best latency = 90, Best rate = 2, Fitness std - (0.05340946946536193), Highest fitness for this gen - (0.13298872180451127)
Generation 214: Best latency = 90, Best rate = 2, Fitness std - (0.05393906773824189), Highest fitness for this gen - (0.13298872180451127)
Generation 215: Best latency = 90, Best rate = 2, Fitness std - (0.05388388363037656), Highest fitness for this gen - (0.13298872180451127)
Generation 216: Best latency = 90, Best rate = 2, Fitness std - (0.05342356077193878), Highest fitness for this gen - (0.13298872180451127)
Generation 217: Best latency = 90, Best rate = 2, Fitness std - (0.05326809123636526), Highest fitness for this gen - (0.13298872180451127)
Generation 218: Best latency = 90, Best rate = 2, Fitness std - (0.0534376436478011), Highest fitness for this gen - (0.13298872180451127)
Generation 219: Best latency = 90, Best rate = 2, Fitness std - (0.053994120785184375), Highest fitness for this gen - (0.13298872180451127)
Generation 220: Best latency = 90, Best rate = 2, Fitness std - (0.05392528401393797), Highest fitness for this gen - (0.13298872180451127)
Generation 221: Best latency = 90, Best rate = 2, Fitness std - (0.054497218626224), Highest fitness for this gen - (0.13298872180451127)
Generation 222: Best latency = 90, Best rate = 2, Fitness std - (0.05402159828606624), Highest fitness for this gen - (0.13298872180451127)
Generation 223: Best latency = 90, Best rate = 2, Fitness std - (0.05368970130484126), Highest fitness for this gen - (0.13298872180451127)
Generation 224: Best latency = 90, Best rate = 2, Fitness std - (0.05460455655320243), Highest fitness for this gen - (0.13298872180451127)
Generation 225: Best latency = 90, Best rate = 2, Fitness std - (0.054158497552488076), Highest fitness for this gen - (0.13298872180451127)
Generation 226: Best latency = 90, Best rate = 2, Fitness std - (0.05380086084688572), Highest fitness for this gen - (0.13298872180451127)
Generation 227: Best latency = 90, Best rate = 2, Fitness std - (0.05440288107616887), Highest fitness for this gen - (0.13298872180451127)
Generation 228: Best latency = 90, Best rate = 2, Fitness std - (0.05355000394631677), Highest fitness for this gen - (0.13298872180451127)
Generation 229: Best latency = 90, Best rate = 2, Fitness std - (0.05361995685603104), Highest fitness for this gen - (0.13298872180451127)
Generation 230: Best latency = 90, Best rate = 2, Fitness std - (0.0542538451891942), Highest fitness for this gen - (0.13298872180451127)
Generation 231: Best latency = 90, Best rate = 2, Fitness std - (0.053564011244299936), Highest fitness for this gen - (0.13298872180451127)
Generation 232: Best latency = 90, Best rate = 2, Fitness std - (0.05392528401393798), Highest fitness for this gen - (0.13298872180451127)
Generation 233: Best latency = 90, Best rate = 2, Fitness std - (0.05389769196630346), Highest fitness for this gen - (0.13298872180451127)
Generation 234: Best latency = 90, Best rate = 2, Fitness std - (0.05413118263286469), Highest fitness for this gen - (0.13298872180451127)
Generation 235: Best latency = 90, Best rate = 2, Fitness std - (0.05426743403343706), Highest fitness for this gen - (0.13298872180451127)
Generation 236: Best latency = 90, Best rate = 2, Fitness std - (0.0532539067489532), Highest fitness for this gen - (0.13298872180451127)
Generation 237: Best latency = 90, Best rate = 2, Fitness std - (0.053605982977040396), Highest fitness for this gen - (0.13298872180451127)
Generation 238: Best latency = 90, Best rate = 2, Fitness std - (0.0532539067489532), Highest fitness for this gen - (0.13298872180451127)
Generation 239: Best latency = 90, Best rate = 2, Fitness std - (0.05411751301953463), Highest fitness for this gen - (0.13298872180451127)
Generation 240: Best latency = 90, Best rate = 2, Fitness std - (0.053592000753112316), Highest fitness for this gen - (0.13298872180451127)
Generation 241: Best latency = 90, Best rate = 2, Fitness std - (0.05353598827719841), Highest fitness for this gen - (0.13298872180451127)
Generation 242: Best latency = 90, Best rate = 2, Fitness std - (0.05388388363037656), Highest fitness for this gen - (0.13298872180451127)
Generation 243: Best latency = 90, Best rate = 2, Fitness std - (0.05346578413402597), Highest fitness for this gen - (0.13298872180451127)
Generation 244: Best latency = 90, Best rate = 2, Fitness std - (0.05331059369724495), Highest fitness for this gen - (0.13298872180451127)
Generation 245: Best latency = 90, Best rate = 2, Fitness std - (0.05329643470373041), Highest fitness for this gen - (0.13298872180451127)
Generation 246: Best latency = 90, Best rate = 2, Fitness std - (0.0535219642303685), Highest fitness for this gen - (0.13298872180451127)
Generation 247: Best latency = 90, Best rate = 2, Fitness std - (0.05375923810408655), Highest fitness for this gen - (0.13298872180451127)
Generation 248: Best latency = 90, Best rate = 2, Fitness std - (0.05436233106249908), Highest fitness for this gen - (0.13298872180451127)
Generation 249: Best latency = 90, Best rate = 2, Fitness std - (0.05309731412175021), Highest fitness for this gen - (0.13298872180451127)
Generation 250: Best latency = 90, Best rate = 2, Fitness std - (0.05411751301953463), Highest fitness for this gen - (0.13298872180451127)
Generation 251: Best latency = 90, Best rate = 2, Fitness std - (0.053633922396606894), Highest fitness for this gen - (0.13298872180451127)
```

```
Generation 252: Best latency = 90, Best rate = 2, Fitness std - (0.05388388363037656), Highest fitness for this gen - (0.13298872180451127)
Generation 253: Best latency = 90, Best rate = 2, Fitness std - (0.05366182848855128), Highest fitness for this gen - (0.13298872180451127)
Generation 254: Best latency = 90, Best rate = 2, Fitness std - (0.0535219642303685), Highest fitness for this gen - (0.13298872180451127)
Generation 255: Best latency = 90, Best rate = 2, Fitness std - (0.054007863615639556), Highest fitness for this gen - (0.13298872180451127)
Generation 256: Best latency = 90, Best rate = 2, Fitness std - (0.05491034861007274), Highest fitness for this gen - (0.13298872180451127)
Generation 257: Best latency = 90, Best rate = 2, Fitness std - (0.054062753399310305), Highest fitness for this gen - (0.13298872180451127)
Generation 258: Best latency = 90, Best rate = 2, Fitness std - (0.0537175408972748), Highest fitness for this gen - (0.13298872180451127)
Generation 259: Best latency = 90, Best rate = 2, Fitness std - (0.0532539067489532), Highest fitness for this gen - (0.13298872180451127)
Generation 260: Best latency = 90, Best rate = 2, Fitness std - (0.05347984175768293), Highest fitness for this gen - (0.13298872180451127)
Generation 261: Best latency = 90, Best rate = 2, Fitness std - (0.0534376436478011), Highest fitness for this gen - (0.13298872180451127)
Generation 262: Best latency = 90, Best rate = 2, Fitness std - (0.05225423826812144), Highest fitness for this gen - (0.13298872180451127)
Generation 263: Best latency = 90, Best rate = 2, Fitness std - (0.05267947403047019), Highest fitness for this gen - (0.13298872180451127)
Generation 264: Best latency = 90, Best rate = 2, Fitness std - (0.053282267219028887), Highest fitness for this gen - (0.13298872180451127)
Generation 265: Best latency = 90, Best rate = 2, Fitness std - (0.05422664334602011), Highest fitness for this gen - (0.13298872180451127)
Generation 266: Best latency = 90, Best rate = 2, Fitness std - (0.05359200075311232), Highest fitness for this gen - (0.13298872180451127)
Generation 267: Best latency = 90, Best rate = 2, Fitness std - (0.05323971374999499), Highest fitness for this gen - (0.13298872180451127)
Generation 268: Best latency = 90, Best rate = 2, Fitness std - (0.05342356077193878), Highest fitness for this gen - (0.13298872180451127)
Generation 269: Best latency = 90, Best rate = 2, Fitness std - (0.0540215982860624), Highest fitness for this gen - (0.13298872180451127)
Generation 270: Best latency = 90, Best rate = 2, Fitness std - (0.05305442709915784), Highest fitness for this gen - (0.13298872180451127)
Generation 271: Best latency = 90, Best rate = 2, Fitness std - (0.053423560771938786), Highest fitness for this gen - (0.13298872180451127)
Generation 272: Best latency = 90, Best rate = 2, Fitness std - (0.054144844141804836), Highest fitness for this gen - (0.13298872180451127)
Generation 273: Best latency = 90, Best rate = 2, Fitness std - (0.05411751301953463), Highest fitness for this gen - (0.13298872180451127)
Generation 274: Best latency = 90, Best rate = 2, Fitness std - (0.05387006707796201), Highest fitness for this gen - (0.13298872180451127)
Generation 275: Best latency = 90, Best rate = 2, Fitness std - (0.05319708361569663), Highest fitness for this gen - (0.13298872180451127)
Generation 276: Best latency = 90, Best rate = 2, Fitness std - (0.053592000753112316), Highest fitness for this gen - (0.13298872180451127)
Generation 277: Best latency = 90, Best rate = 2, Fitness std - (0.0537175408972748), Highest fitness for this gen - (0.13298872180451127)
Generation 278: Best latency = 90, Best rate = 2, Fitness std - (0.053633922396606894), Highest fitness for this gen - (0.13298872180451127)
Generation 279: Best latency = 90, Best rate = 2, Fitness std - (0.0541721428710376), Highest fitness for this gen - (0.13298872180451127)
Generation 280: Best latency = 90, Best rate = 2, Fitness std - (0.05364787960527988), Highest fitness for this gen - (0.13298872180451127)
Generation 281: Best latency = 90, Best rate = 2, Fitness std - (0.05368970130484126), Highest fitness for this gen - (0.13298872180451127)
Generation 282: Best latency = 90, Best rate = 2, Fitness std - (0.05267947403047019), Highest fitness for this gen - (0.13298872180451127)
Generation 283: Best latency = 90, Best rate = 2, Fitness std - (0.05356401124429994), Highest fitness for this gen - (0.13298872180451127)
Generation 284: Best latency = 90, Best rate = 2, Fitness std - (0.05387006707796201), Highest fitness for this gen - (0.13298872180451127)
Generation 285: Best latency = 90, Best rate = 2, Fitness std - (0.053633922396606894), Highest fitness for this gen - (0.13298872180451127)
Generation 286: Best latency = 90, Best rate = 2, Fitness std - (0.0540764554917272), Highest fitness for this gen - (0.13298872180451127)
Generation 287: Best latency = 90, Best rate = 2, Fitness std - (0.05360598297704039), Highest fitness for this gen - (0.13298872180451127)
Generation 288: Best latency = 90, Best rate = 2, Fitness std - (0.05380086084688573), Highest fitness for this gen - (0.13298872180451127)
Generation 289: Best latency = 90, Best rate = 2, Fitness std - (0.053083027043878094), Highest fitness for this gen - (0.13298872180451127)
Generation 290: Best latency = 90, Best rate = 2, Fitness std - (0.05504201652162137), Highest fitness for this gen - (0.13298872180451127)
Generation 291: Best latency = 90, Best rate = 2, Fitness std - (0.054348798396332734), Highest fitness for this gen - (0.13298872180451127)
Generation 292: Best latency = 90, Best rate = 2, Fitness std - (0.05361995685603104), Highest fitness for this gen - (0.13298872180451127)
Generation 293: Best latency = 90, Best rate = 2, Fitness std - (0.053842409298364965), Highest fitness for this gen - (0.13298872180451127)
Generation 294: Best latency = 90, Best rate = 2, Fitness std - (0.05419940925617936), Highest fitness for this gen - (0.13298872180451127)
Generation 295: Best latency = 90, Best rate = 2, Fitness std - (0.05382856805850425), Highest fitness for this gen - (0.13298872180451127)
Generation 296: Best latency = 90, Best rate = 2, Fitness std - (0.05464467784081566), Highest fitness for this gen - (0.13298872180451127)
Generation 297: Best latency = 90, Best rate = 2, Fitness std - (0.05305442709915784), Highest fitness for this gen - (0.13298872180451127)
Generation 298: Best latency = 90, Best rate = 2, Fitness std - (0.053592000753112316), Highest fitness for this gen - (0.13298872180451127)
Generation 299: Best latency = 90, Best rate = 2, Fitness std - (0.053661828488551284), Highest fitness for this gen - (0.13298872180451127)
```



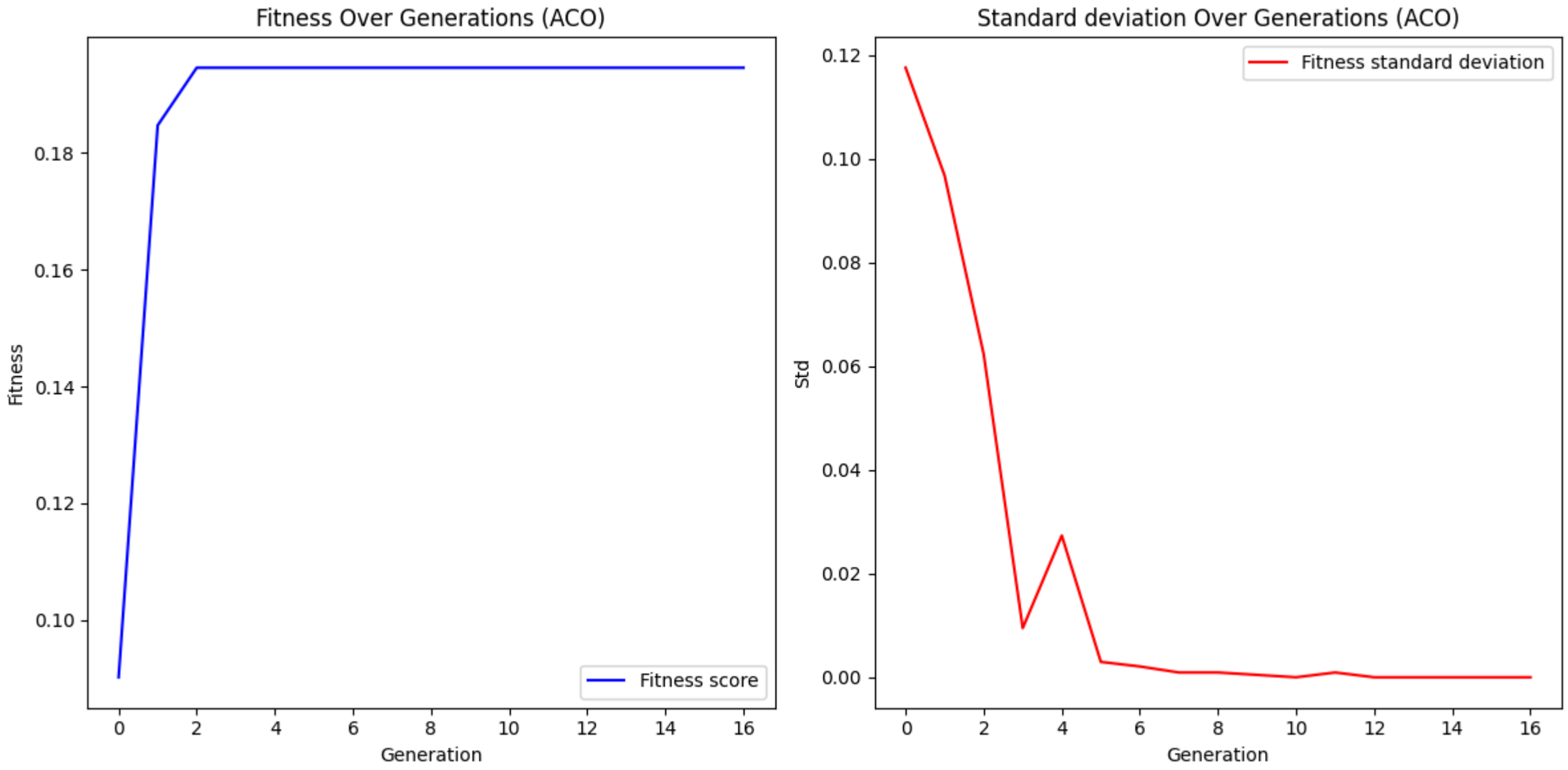



```
Total best paths [[29, 9, 104, 'BS-2']]
Best latency 90
Best transmission rate 2
Execution time 23.04551672935486 seconds
```

Ant colony algorithm

In [ ]:
```
ACO = Ant_colony_algorithm(nodes=modified_nodes, alpha=1, beta=2, _evaporation_rate=.5, _pheromone_deposit=1, BS_stations=BS_stations, dist_matrix=dist_matrix, rate_matrix=rate_matrix)

start_time = time.time()

best_path_ACO, all_best_paths_ACO, best_rate_ACO, best_latency_ACO, aco_fitness_score, aco_fitness_std, gen_num_ACO = ACO.start_aco(starting_node=STARTING_NODE_3, iteration=GENERATIONS, ants=
end_time = time.time()
execution_time = end_time - start_time

print("Best path", all_best_paths_ACO)
print("Best latency", best_latency_ACO)
print("Best transmission rate", best_rate_ACO)
print("Execution time", execution_time, "seconds")

ACO_metrics["Test3"]["paths"] = all_best_paths_ACO
ACO_metrics["Test3"]["latency"] = best_latency_ACO
ACO_metrics["Test3"]["transmission"] = best_rate_ACO

ACO_metrics["Test3"]["gen"] = gen_num_ACO

ACO_metrics["Test3"]["Fitness"] = aco_fitness_score
ACO_metrics["Test3"]["Fitness_std"] = aco_fitness_std
ACO_metrics["Test3"]["Execution_time"] = execution_time
```

```
Iteration 0: Best rate - 2, Best latency - 480, Fitness std - (0.11750560836729017), Best Fitness = 0.09022556390977443
Iteration 1: Best rate - 3, Best latency - 270, Fitness std - (0.09668780650434192), Best Fitness = 0.18468045112781953
Iteration 2: Best rate - 3, Best latency - 180, Fitness std - (0.06226737830754231), Best Fitness = 0.1945488721804511
Iteration 3: Best rate - 3, Best latency - 180, Fitness std - (0.009488292830168393), Best Fitness = 0.1945488721804511
Iteration 4: Best rate - 3, Best latency - 180, Fitness std - (0.027306960044088655), Best Fitness = 0.1945488721804511
Iteration 5: Best rate - 3, Best latency - 180, Fitness std - (0.0029605263157894725), Best Fitness = 0.1945488721804511
Iteration 6: Best rate - 3, Best latency - 180, Fitness std - (0.002081485792046891), Best Fitness = 0.1945488721804511
Iteration 7: Best rate - 3, Best latency - 180, Fitness std - (0.0009210526315789463), Best Fitness = 0.1945488721804511
Iteration 8: Best rate - 3, Best latency - 180, Fitness std - (0.0009210526315789463), Best Fitness = 0.1945488721804511
Iteration 9: Best rate - 3, Best latency - 180, Fitness std - (0.0004605263157894732), Best Fitness = 0.1945488721804511
Iteration 10: Best rate - 3, Best latency - 180, Fitness std - (0.0), Best Fitness = 0.1945488721804511
Iteration 11: Best rate - 3, Best latency - 180, Fitness std - (0.0009210526315789464), Best Fitness = 0.1945488721804511
Iteration 12: Best rate - 3, Best latency - 180, Fitness std - (0.0), Best Fitness = 0.1945488721804511
Iteration 13: Best rate - 3, Best latency - 180, Fitness std - (0.0), Best Fitness = 0.1945488721804511
Iteration 14: Best rate - 3, Best latency - 180, Fitness std - (0.0), Best Fitness = 0.1945488721804511
Iteration 15: Best rate - 3, Best latency - 180, Fitness std - (0.0), Best Fitness = 0.1945488721804511
Iteration 16: Best rate - 3, Best latency - 180, Fitness std - (0.0), Best Fitness = 0.1945488721804511
Algorithmm converged
```




```
Best path [[29, 122, 22, 142, 104, 44, 152]]
Best latency 180
Best transmission rate 3
Execution time 1.9085125923156738 seconds
```

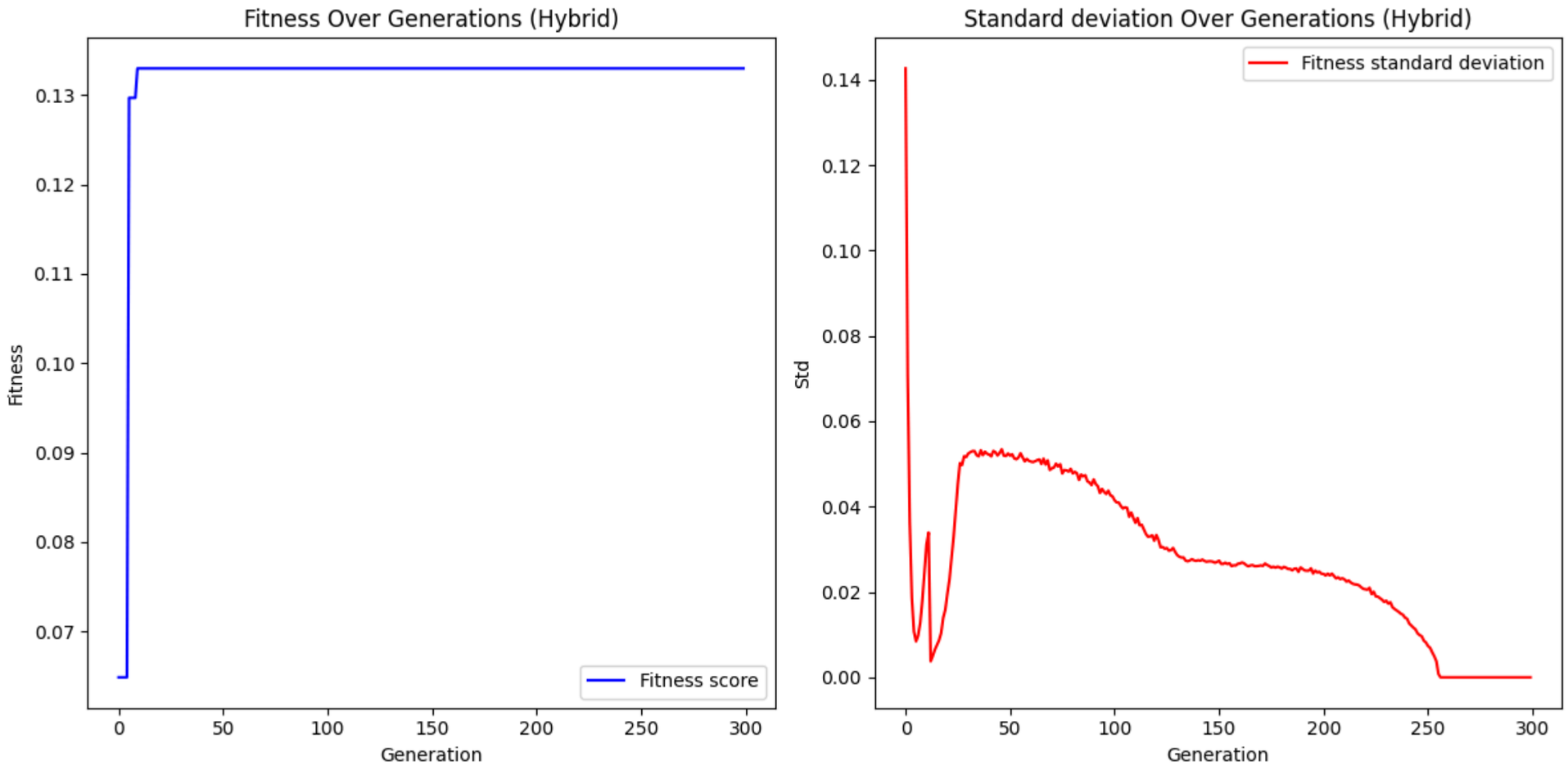
Hybrid algorithm (GA + SA)

In [ ]:
```
GA_SA = Hybrid_GA_with_SA(nodes=NODES, pop_size=9000, generations=GENERATIONS, mutation_rate=.6, cooling_rate=.96)

start_time = time.time()

best_path_SA, best_latency_SA, best_transmission_rate_SA, all_best_paths_SA, hybrid_fitness_score, hybrid_fitness_std, gen_num_SA = GA_SA.GA(starting_node=STARTING_NODE_3, temperature=1000)

end_time = time.time()
execution_time = end_time - start_time

print("Best path", all_best_paths_SA)
print("Best latency", best_latency_SA)
print("Best transmission rate", best_transmission_rate_SA)
print("Execution time", execution_time, "seconds")

HYBRID_metrics["Test3"]["paths"] = all_best_paths_SA
HYBRID_metrics["Test3"]["latency"] = best_latency_SA
HYBRID_metrics["Test3"]["transmission"] = best_transmission_rate_SA

HYBRID_metrics["Test3"]["gen"] = gen_num_SA

HYBRID_metrics["Test3"]["Fitness"] = hybrid_fitness_score
HYBRID_metrics["Test3"]["Fitness_std"] = hybrid_fitness_std
HYBRID_metrics["Test3"]["Execution_time"] = execution_time
```

```
Generation 0: Best latency = 60, Best rate = 1, Fitness std - (0.14267912723114715) Highest fitness for this gen - (0.06484962406015038), Temperature - 1000.000
Generation 1: Best latency = 60, Best rate = 1, Fitness std - (0.07084361096468286) Highest fitness for this gen - (0.06484962406015038), Temperature - 960.000
Generation 2: Best latency = 60, Best rate = 1, Fitness std - (0.03639008394406272) Highest fitness for this gen - (0.06484962406015038), Temperature - 921.600
Generation 3: Best latency = 60, Best rate = 1, Fitness std - (0.018566117128336144) Highest fitness for this gen - (0.06484962406015038), Temperature - 884.736
Generation 4: Best latency = 60, Best rate = 1, Fitness std - (0.010728036462823639) Highest fitness for this gen - (0.06484962406015038), Temperature - 849.347
Generation 5: Best latency = 120, Best rate = 2, Fitness std - (0.008421111488523583) Highest fitness for this gen - (0.12969924812030076), Temperature - 815.373
Generation 6: Best latency = 120, Best rate = 2, Fitness std - (0.009798928467227578) Highest fitness for this gen - (0.12969924812030076), Temperature - 782.758
Generation 7: Best latency = 120, Best rate = 2, Fitness std - (0.012714605024462047) Highest fitness for this gen - (0.12969924812030076), Temperature - 751.447
Generation 8: Best latency = 120, Best rate = 2, Fitness std - (0.017909886636614053) Highest fitness for this gen - (0.12969924812030076), Temperature - 721.390
Generation 9: Best latency = 90, Best rate = 2, Fitness std - (0.024342136954110374) Highest fitness for this gen - (0.13298872180451127), Temperature - 692.534
Generation 10: Best latency = 90, Best rate = 2, Fitness std - (0.031065493364988928) Highest fitness for this gen - (0.13298872180451127), Temperature - 664.833
Generation 11: Best latency = 90, Best rate = 2, Fitness std - (0.03392873939863425) Highest fitness for this gen - (0.13298872180451127), Temperature - 638.239
Generation 12: Best latency = 90, Best rate = 2, Fitness std - (0.0037615543746471326) Highest fitness for this gen - (0.13298872180451127), Temperature - 612.710
Generation 13: Best latency = 90, Best rate = 2, Fitness std - (0.00501353751883112) Highest fitness for this gen - (0.13298872180451127), Temperature - 588.201
Generation 14: Best latency = 90, Best rate = 2, Fitness std - (0.006460711158137622) Highest fitness for this gen - (0.13298872180451127), Temperature - 564.673
Generation 15: Best latency = 90, Best rate = 2, Fitness std - (0.007534803670784664) Highest fitness for this gen - (0.13298872180451127), Temperature - 542.086
Generation 16: Best latency = 90, Best rate = 2, Fitness std - (0.008666964988586415) Highest fitness for this gen - (0.13298872180451127), Temperature - 520.403
Generation 17: Best latency = 90, Best rate = 2, Fitness std - (0.01036717905302973) Highest fitness for this gen - (0.13298872180451127), Temperature - 499.587
Generation 18: Best latency = 90, Best rate = 2, Fitness std - (0.01393516896261359) Highest fitness for this gen - (0.13298872180451127), Temperature - 479.603
Generation 19: Best latency = 90, Best rate = 2, Fitness std - (0.015839488125167014) Highest fitness for this gen - (0.13298872180451127), Temperature - 460.419
Generation 20: Best latency = 90, Best rate = 2, Fitness std - (0.01959467935841162) Highest fitness for this gen - (0.13298872180451127), Temperature - 442.002
Generation 21: Best latency = 90, Best rate = 2, Fitness std - (0.022964024236905203) Highest fitness for this gen - (0.13298872180451127), Temperature - 424.322
Generation 22: Best latency = 90, Best rate = 2, Fitness std - (0.027905505865036657) Highest fitness for this gen - (0.13298872180451127), Temperature - 407.349
Generation 23: Best latency = 90, Best rate = 2, Fitness std - (0.033045038440924664) Highest fitness for this gen - (0.13298872180451127), Temperature - 391.055
Generation 24: Best latency = 90, Best rate = 2, Fitness std - (0.039208600968179595) Highest fitness for this gen - (0.13298872180451127), Temperature - 375.413
Generation 25: Best latency = 90, Best rate = 2, Fitness std - (0.04535522848809049) Highest fitness for this gen - (0.13298872180451127), Temperature - 360.397
Generation 26: Best latency = 90, Best rate = 2, Fitness std - (0.050192505006030584) Highest fitness for this gen - (0.13298872180451127), Temperature - 345.981
Generation 27: Best latency = 90, Best rate = 2, Fitness std - (0.04973963988862326) Highest fitness for this gen - (0.13298872180451127), Temperature - 332.142
Generation 28: Best latency = 90, Best rate = 2, Fitness std - (0.051806191353932794) Highest fitness for this gen - (0.13298872180451127), Temperature - 318.856
Generation 29: Best latency = 90, Best rate = 2, Fitness std - (0.05161934413732822) Highest fitness for this gen - (0.13298872180451127), Temperature - 306.102
Generation 30: Best latency = 90, Best rate = 2, Fitness std - (0.05245070904780988) Highest fitness for this gen - (0.13298872180451127), Temperature - 293.858
Generation 31: Best latency = 90, Best rate = 2, Fitness std - (0.052760488260663714) Highest fitness for this gen - (0.13298872180451127), Temperature - 282.103
Generation 32: Best latency = 90, Best rate = 2, Fitness std - (0.053008057263426535) Highest fitness for this gen - (0.13298872180451127), Temperature - 270.819
Generation 33: Best latency = 90, Best rate = 2, Fitness std - (0.05300476074017983) Highest fitness for this gen - (0.13298872180451127), Temperature - 259.986
Generation 34: Best latency = 90, Best rate = 2, Fitness std - (0.05205961500759709) Highest fitness for this gen - (0.13298872180451127), Temperature - 249.587
Generation 35: Best latency = 90, Best rate = 2, Fitness std - (0.05177696453479471) Highest fitness for this gen - (0.13298872180451127), Temperature - 239.603
Generation 36: Best latency = 90, Best rate = 2, Fitness std - (0.05319709545622377) Highest fitness for this gen - (0.13298872180451127), Temperature - 230.019
Generation 37: Best latency = 90, Best rate = 2, Fitness std - (0.05205964707056334) Highest fitness for this gen - (0.13298872180451127), Temperature - 220.819
Generation 38: Best latency = 90, Best rate = 2, Fitness std - (0.05287730143744693) Highest fitness for this gen - (0.13298872180451127), Temperature - 211.986
Generation 39: Best latency = 90, Best rate = 2, Fitness std - (0.052366707418505694) Highest fitness for this gen - (0.13298872180451127), Temperature - 203.506
Generation 40: Best latency = 90, Best rate = 2, Fitness std - (0.052210159266780964) Highest fitness for this gen - (0.13298872180451127), Temperature - 195.366
Generation 41: Best latency = 90, Best rate = 2, Fitness std - (0.0517800544252435) Highest fitness for this gen - (0.13298872180451127), Temperature - 187.552
Generation 42: Best latency = 90, Best rate = 2, Fitness std - (0.05304682824066447) Highest fitness for this gen - (0.13298872180451127), Temperature - 180.049
Generation 43: Best latency = 90, Best rate = 2, Fitness std - (0.05273728665162975) Highest fitness for this gen - (0.13298872180451127), Temperature - 172.847
Generation 44: Best latency = 90, Best rate = 2, Fitness std - (0.05199201382071853) Highest fitness for this gen - (0.13298872180451127), Temperature - 165.934
Generation 45: Best latency = 90, Best rate = 2, Fitness std - (0.052596523705188886) Highest fitness for this gen - (0.13298872180451127), Temperature - 159.296
Generation 46: Best latency = 90, Best rate = 2, Fitness std - (0.05343453172952808) Highest fitness for this gen - (0.13298872180451127), Temperature - 152.924
Generation 47: Best latency = 90, Best rate = 2, Fitness std - (0.05194816337470012) Highest fitness for this gen - (0.13298872180451127), Temperature - 146.807
Generation 48: Best latency = 90, Best rate = 2, Fitness std - (0.051865231779812726) Highest fitness for this gen - (0.13298872180451127), Temperature - 140.935
Generation 49: Best latency = 90, Best rate = 2, Fitness std - (0.052451926423515836) Highest fitness for this gen - (0.13298872180451127), Temperature - 135.298
Generation 50: Best latency = 90, Best rate = 2, Fitness std - (0.05193776922440991) Highest fitness for this gen - (0.13298872180451127), Temperature - 129.886
Generation 51: Best latency = 90, Best rate = 2, Fitness std - (0.05225460411096472) Highest fitness for this gen - (0.13298872180451127), Temperature - 124.690
Generation 52: Best latency = 90, Best rate = 2, Fitness std - (0.05131218397454783) Highest fitness for this gen - (0.13298872180451127), Temperature - 119.703
Generation 53: Best latency = 90, Best rate = 2, Fitness std - (0.05111433429154709) Highest fitness for this gen - (0.13298872180451127), Temperature - 114.915
Generation 54: Best latency = 90, Best rate = 2, Fitness std - (0.05149158152554663) Highest fitness for this gen - (0.13298872180451127), Temperature - 110.318
Generation 55: Best latency = 90, Best rate = 2, Fitness std - (0.05249553064996309) Highest fitness for this gen - (0.13298872180451127), Temperature - 105.905
Generation 56: Best latency = 90, Best rate = 2, Fitness std - (0.051540110283176684) Highest fitness for this gen - (0.13298872180451127), Temperature - 101.669
Generation 57: Best latency = 90, Best rate = 2, Fitness std - (0.05060673943349537) Highest fitness for this gen - (0.13298872180451127), Temperature - 97.602
Generation 58: Best latency = 90, Best rate = 2, Fitness std - (0.05116544915468647) Highest fitness for this gen - (0.13298872180451127), Temperature - 93.698
Generation 59: Best latency = 90, Best rate = 2, Fitness std - (0.05074062149987919) Highest fitness for this gen - (0.13298872180451127), Temperature - 89.950
Generation 60: Best latency = 90, Best rate = 2, Fitness std - (0.05054681841223935) Highest fitness for this gen - (0.13298872180451127), Temperature - 86.352
Generation 61: Best latency = 90, Best rate = 2, Fitness std - (0.05043843052047323) Highest fitness for this gen - (0.13298872180451127), Temperature - 82.898
Generation 62: Best latency = 90, Best rate = 2, Fitness std - (0.05064188636965299) Highest fitness for this gen - (0.13298872180451127), Temperature - 79.582
Generation 63: Best latency = 90, Best rate = 2, Fitness std - (0.05088923546010772) Highest fitness for this gen - (0.13298872180451127), Temperature - 76.399
Generation 64: Best latency = 90, Best rate = 2, Fitness std - (0.05104841799797269) Highest fitness for this gen - (0.13298872180451127), Temperature - 73.343
Generation 65: Best latency = 90, Best rate = 2, Fitness std - (0.049992652521370064) Highest fitness for this gen - (0.13298872180451127), Temperature - 70.409
Generation 66: Best latency = 90, Best rate = 2, Fitness std - (0.0512300966397027) Highest fitness for this gen - (0.13298872180451127), Temperature - 67.593
Generation 67: Best latency = 90, Best rate = 2, Fitness std - (0.049861662368493695) Highest fitness for this gen - (0.13298872180451127), Temperature - 64.889
Generation 68: Best latency = 90, Best rate = 2, Fitness std - (0.05083474528106121) Highest fitness for this gen - (0.13298872180451127), Temperature - 62.294
Generation 69: Best latency = 90, Best rate = 2, Fitness std - (0.048594749433056826) Highest fitness for this gen - (0.13298872180451127), Temperature - 59.802
Generation 70: Best latency = 90, Best rate = 2, Fitness std - (0.048963028071329816) Highest fitness for this gen - (0.13298872180451127), Temperature - 57.410
Generation 71: Best latency = 90, Best rate = 2, Fitness std - (0.04916667563447142) Highest fitness for this gen - (0.13298872180451127), Temperature - 55.113
Generation 72: Best latency = 90, Best rate = 2, Fitness std - (0.0500570196417995) Highest fitness for this gen - (0.13298872180451127), Temperature - 52.909
Generation 73: Best latency = 90, Best rate = 2, Fitness std - (0.049407137062814764) Highest fitness for this gen - (0.13298872180451127), Temperature - 50.793
Generation 74: Best latency = 90, Best rate = 2, Fitness std - (0.04993131841166459) Highest fitness for this gen - (0.13298872180451127), Temperature - 48.761
Generation 75: Best latency = 90, Best rate = 2, Fitness std - (0.04777222387695239) Highest fitness for this gen - (0.13298872180451127), Temperature - 46.810
Generation 76: Best latency = 90, Best rate = 2, Fitness std - (0.04857170803426191) Highest fitness for this gen - (0.13298872180451127), Temperature - 44.938
Generation 77: Best latency = 90, Best rate = 2, Fitness std - (0.0484860068178733) Highest fitness for this gen - (0.13298872180451127), Temperature - 43.140
Generation 78: Best latency = 90, Best rate = 2, Fitness std - (0.048220392845954606) Highest fitness for this gen - (0.13298872180451127), Temperature - 41.415
Generation 79: Best latency = 90, Best rate = 2, Fitness std - (0.048844186298344576) Highest fitness for this gen - (0.13298872180451127), Temperature - 39.758
Generation 80: Best latency = 90, Best rate = 2, Fitness std - (0.04777842113156466) Highest fitness for this gen - (0.13298872180451127), Temperature - 38.168
Generation 81: Best latency = 90, Best rate = 2, Fitness std - (0.04814178100524812) Highest fitness for this gen - (0.13298872180451127), Temperature - 36.641
Generation 82: Best latency = 90, Best rate = 2, Fitness std - (0.04778832346833624) Highest fitness for this gen - (0.13298872180451127), Temperature - 35.176
Generation 83: Best latency = 90, Best rate = 2, Fitness std - (0.04624520567171312) Highest fitness for this gen - (0.13298872180451127), Temperature - 33.769
Generation 84: Best latency = 90, Best rate = 2, Fitness std - (0.04751559595975231) Highest fitness for this gen - (0.13298872180451127), Temperature - 32.418
Generation 85: Best latency = 90, Best rate = 2, Fitness std - (0.047080218660983685) Highest fitness for this gen - (0.13298872180451127), Temperature - 31.121
Generation 86: Best latency = 90, Best rate = 2, Fitness std - (0.04736580212968274) Highest fitness for this gen - (0.13298872180451127), Temperature - 29.876
Generation 87: Best latency = 90, Best rate = 2, Fitness std - (0.04594007291922416) Highest fitness for this gen - (0.13298872180451127), Temperature - 28.681
Generation 88: Best latency = 90, Best rate = 2, Fitness std - (0.045657288514070145) Highest fitness for this gen - (0.13298872180451127), Temperature - 27.534
Generation 89: Best latency = 90, Best rate = 2, Fitness std - (0.04504720516158289) Highest fitness for this gen - (0.13298872180451127), Temperature - 26.433
Generation 90: Best latency = 90, Best rate = 2, Fitness std - (0.04637807320246597) Highest fitness for this gen - (0.13298872180451127), Temperature - 25.375
Generation 91: Best latency = 90, Best rate = 2, Fitness std - (0.04529820498268739) Highest fitness for this gen - (0.13298872180451127), Temperature - 24.360
Generation 92: Best latency = 90, Best rate = 2, Fitness std - (0.044769248947213264) Highest fitness for this gen - (0.13298872180451127), Temperature - 23.386
Generation 93: Best latency = 90, Best rate = 2, Fitness std - (0.043195296598983135) Highest fitness for this gen - (0.13298872180451127), Temperature - 22.450
Generation 94: Best latency = 90, Best rate = 2, Fitness std - (0.04413306741213247) Highest fitness for this gen - (0.13298872180451127), Temperature - 21.552
Generation 95: Best latency = 90, Best rate = 2, Fitness std - (0.04344393626718812) Highest fitness for this gen - (0.13298872180451127), Temperature - 20.690
Generation 96: Best latency = 90, Best rate = 2, Fitness std - (0.04296346599648169) Highest fitness for this gen - (0.13298872180451127), Temperature - 19.863
Generation 97: Best latency = 90, Best rate = 2, Fitness std - (0.043727274989179686) Highest fitness for this gen - (0.13298872180451127), Temperature - 19.068
Generation 98: Best latency = 90, Best rate = 2, Fitness std - (0.04265525567245777) Highest fitness for this gen - (0.13298872180451127), Temperature - 18.305
Generation 99: Best latency = 90, Best rate = 2, Fitness std - (0.042403525288778755) Highest fitness for this gen - (0.13298872180451127), Temperature - 17.573
Generation 100: Best latency = 90, Best rate = 2, Fitness std - (0.041495092704667) Highest fitness for this gen - (0.13298872180451127), Temperature - 16.870
Generation 101: Best latency = 90, Best rate = 2, Fitness std - (0.04096812097541189) Highest fitness for this gen - (0.13298872180451127), Temperature - 16.196
Generation 102: Best latency = 90, Best rate = 2, Fitness std - (0.04096668185762407) Highest fitness for this gen - (0.13298872180451127), Temperature - 15.548
Generation 103: Best latency = 90, Best rate = 2, Fitness std - (0.040140089154978395) Highest fitness for this gen - (0.13298872180451127), Temperature - 14.926
Generation 104: Best latency = 90, Best rate = 2, Fitness std - (0.039562095786604516) Highest fitness for this gen - (0.13298872180451127), Temperature - 14.329
Generation 105: Best latency = 90, Best rate = 2, Fitness std - (0.03983637167133207) Highest fitness for this gen - (0.13298872180451127), Temperature - 13.756
Generation 106: Best latency = 90, Best rate = 2, Fitness std - (0.03969421732938935) Highest fitness for this gen - (0.13298872180451127), Temperature - 13.205
Generation 107: Best latency = 90, Best rate = 2, Fitness std - (0.03763334735921467) Highest fitness for this gen - (0.13298872180451127), Temperature - 12.677
Generation 108: Best latency = 90, Best rate = 2, Fitness std - (0.038600368005744276) Highest fitness for this gen - (0.13298872180451127), Temperature - 12.170
Generation 109: Best latency = 90, Best rate = 2, Fitness std - (0.037331548251244163) Highest fitness for this gen - (0.13298872180451127), Temperature - 11.683
Generation 110: Best latency = 90, Best rate = 2, Fitness std - (0.036205154883122086) Highest fitness for this gen - (0.13298872180451127), Temperature - 11.216
Generation 111: Best latency = 90, Best rate = 2, Fitness std - (0.03730842220393324) Highest fitness for this gen - (0.13298872180451127), Temperature - 10.767
Generation 112: Best latency = 90, Best rate = 2, Fitness std - (0.035609949889267606) Highest fitness for this gen - (0.13298872180451127), Temperature - 10.337
Generation 113: Best latency = 90, Best rate = 2, Fitness std - (0.035764377569697) Highest fitness for this gen - (0.13298872180451127), Temperature - 9.923
Generation 114: Best latency = 90, Best rate = 2, Fitness std - (0.03463766938738183) Highest fitness for this gen - (0.13298872180451127), Temperature - 9.526
Generation 115: Best latency = 90, Best rate = 2, Fitness std - (0.03359601617524114) Highest fitness for this gen - (0.13298872180451127), Temperature - 9.145
Generation 116: Best latency = 90, Best rate = 2, Fitness std - (0.03294824327174178) Highest fitness for this gen - (0.13298872180451127), Temperature - 8.779
Generation 117: Best latency = 90, Best rate = 2, Fitness std - (0.03301398040973446) Highest fitness for this gen - (0.13298872180451127), Temperature - 8.428
Generation 118: Best latency = 90, Best rate = 2, Fitness std - (0.033273868917420554) Highest fitness for this gen - (0.13298872180451127), Temperature - 8.091
Generation 119: Best latency = 90, Best rate = 2, Fitness std - (0.03201634027092515) Highest fitness for this gen - (0.13298872180451127), Temperature - 7.767
Generation 120: Best latency = 90, Best rate = 2, Fitness std - (0.03333775386410206) Highest fitness for this gen - (0.13298872180451127), Temperature - 7.457
Generation 121: Best latency = 90, Best rate = 2, Fitness std - (0.03223333825774326) Highest fitness for this gen - (0.13298872180451127), Temperature - 7.158
Generation 122: Best latency = 90, Best rate = 2, Fitness std - (0.03047116360658763) Highest fitness for this gen - (0.13298872180451127), Temperature - 6.872
Generation 123: Best latency = 90, Best rate = 2, Fitness std - (0.0305499933094404) Highest fitness for this gen - (0.13298872180451127), Temperature - 6.597
Generation 124: Best latency = 90, Best rate = 2, Fitness std - (0.030145941315171725) Highest fitness for this gen - (0.13298872180451127), Temperature - 6.333
Generation 125: Best latency = 90, Best rate = 2, Fitness std - (0.030297840198359207) Highest fitness for this gen - (0.13298872180451127), Temperature - 6.080
```

```
Generation 126: Best latency = 90, Best rate = 2, Fitness std - (0.029671875471623414) Highest fitness for this gen - (0.13298872180451127), Temperature - 5.837
Generation 127: Best latency = 90, Best rate = 2, Fitness std - (0.029808525390247722) Highest fitness for this gen - (0.13298872180451127), Temperature - 5.603
Generation 128: Best latency = 90, Best rate = 2, Fitness std - (0.030316508752622647) Highest fitness for this gen - (0.13298872180451127), Temperature - 5.379
Generation 129: Best latency = 90, Best rate = 2, Fitness std - (0.029374837503504062) Highest fitness for this gen - (0.13298872180451127), Temperature - 5.164
Generation 130: Best latency = 90, Best rate = 2, Fitness std - (0.028633525992856217) Highest fitness for this gen - (0.13298872180451127), Temperature - 4.957
Generation 131: Best latency = 90, Best rate = 2, Fitness std - (0.028257316548525223) Highest fitness for this gen - (0.13298872180451127), Temperature - 4.759
Generation 132: Best latency = 90, Best rate = 2, Fitness std - (0.028066975291983570) Highest fitness for this gen - (0.13298872180451127), Temperature - 4.569
Generation 133: Best latency = 90, Best rate = 2, Fitness std - (0.028082119616882034) Highest fitness for this gen - (0.13298872180451127), Temperature - 4.386
Generation 134: Best latency = 90, Best rate = 2, Fitness std - (0.027348077803319323) Highest fitness for this gen - (0.13298872180451127), Temperature - 4.211
Generation 135: Best latency = 90, Best rate = 2, Fitness std - (0.027188599718421066) Highest fitness for this gen - (0.13298872180451127), Temperature - 4.042
Generation 136: Best latency = 90, Best rate = 2, Fitness std - (0.027382604044962466) Highest fitness for this gen - (0.13298872180451127), Temperature - 3.881
Generation 137: Best latency = 90, Best rate = 2, Fitness std - (0.027669902778619980) Highest fitness for this gen - (0.13298872180451127), Temperature - 3.725
Generation 138: Best latency = 90, Best rate = 2, Fitness std - (0.027410193725704034) Highest fitness for this gen - (0.13298872180451127), Temperature - 3.576
Generation 139: Best latency = 90, Best rate = 2, Fitness std - (0.027278672554148464) Highest fitness for this gen - (0.13298872180451127), Temperature - 3.433
Generation 140: Best latency = 90, Best rate = 2, Fitness std - (0.027444159848699273) Highest fitness for this gen - (0.13298872180451127), Temperature - 3.296
Generation 141: Best latency = 90, Best rate = 2, Fitness std - (0.027306201160525755) Highest fitness for this gen - (0.13298872180451127), Temperature - 3.164
Generation 142: Best latency = 90, Best rate = 2, Fitness std - (0.02758425102318263) Highest fitness for this gen - (0.13298872180451127), Temperature - 3.037
Generation 143: Best latency = 90, Best rate = 2, Fitness std - (0.027270543443109814) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.916
Generation 144: Best latency = 90, Best rate = 2, Fitness std - (0.02708431447725690) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.799
Generation 145: Best latency = 90, Best rate = 2, Fitness std - (0.027179572309358125) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.687
Generation 146: Best latency = 90, Best rate = 2, Fitness std - (0.027194130306079953) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.580
Generation 147: Best latency = 90, Best rate = 2, Fitness std - (0.027098036883188615) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.477
Generation 148: Best latency = 90, Best rate = 2, Fitness std - (0.026853079310960325) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.378
Generation 149: Best latency = 90, Best rate = 2, Fitness std - (0.027034644589165290) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.283
Generation 150: Best latency = 90, Best rate = 2, Fitness std - (0.027336954528484260) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.191
Generation 151: Best latency = 90, Best rate = 2, Fitness std - (0.026589257213080645) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.104
Generation 152: Best latency = 90, Best rate = 2, Fitness std - (0.026572068604682860) Highest fitness for this gen - (0.13298872180451127), Temperature - 2.019
Generation 153: Best latency = 90, Best rate = 2, Fitness std - (0.026845484166917360) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.939
Generation 154: Best latency = 90, Best rate = 2, Fitness std - (0.026546877584729510) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.861
Generation 155: Best latency = 90, Best rate = 2, Fitness std - (0.026711960777458243) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.787
Generation 156: Best latency = 90, Best rate = 2, Fitness std - (0.026051133653371982) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.715
Generation 157: Best latency = 90, Best rate = 2, Fitness std - (0.026254851139643265) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.647
Generation 158: Best latency = 90, Best rate = 2, Fitness std - (0.026198496847861855) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.581
Generation 159: Best latency = 90, Best rate = 2, Fitness std - (0.026612184518137787) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.517
Generation 160: Best latency = 90, Best rate = 2, Fitness std - (0.026675220661304790) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.457
Generation 161: Best latency = 90, Best rate = 2, Fitness std - (0.026916253340269160) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.399
Generation 162: Best latency = 90, Best rate = 2, Fitness std - (0.026697808724270848) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.343
Generation 163: Best latency = 90, Best rate = 2, Fitness std - (0.026285914375053187) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.289
Generation 164: Best latency = 90, Best rate = 2, Fitness std - (0.026005581120849930) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.237
Generation 165: Best latency = 90, Best rate = 2, Fitness std - (0.026211044112345268) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.188
Generation 166: Best latency = 90, Best rate = 2, Fitness std - (0.026306466340574715) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.140
Generation 167: Best latency = 90, Best rate = 2, Fitness std - (0.026042346854306737) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.095
Generation 168: Best latency = 90, Best rate = 2, Fitness std - (0.026050491701958950) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.051
Generation 169: Best latency = 90, Best rate = 2, Fitness std - (0.026108886686430390) Highest fitness for this gen - (0.13298872180451127), Temperature - 1.009
Generation 170: Best latency = 90, Best rate = 2, Fitness std - (0.026224004513010837) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.969
Generation 171: Best latency = 90, Best rate = 2, Fitness std - (0.026109512905206418) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.930
Generation 172: Best latency = 90, Best rate = 2, Fitness std - (0.026646249506240147) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.893
Generation 173: Best latency = 90, Best rate = 2, Fitness std - (0.026321672360966258) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.857
Generation 174: Best latency = 90, Best rate = 2, Fitness std - (0.026086516858407540) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.823
Generation 175: Best latency = 90, Best rate = 2, Fitness std - (0.025739312727862330) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.790
Generation 176: Best latency = 90, Best rate = 2, Fitness std - (0.025919520723727607) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.758
Generation 177: Best latency = 90, Best rate = 2, Fitness std - (0.025703873063236893) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.728
Generation 178: Best latency = 90, Best rate = 2, Fitness std - (0.025922815789046810) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.699
Generation 179: Best latency = 90, Best rate = 2, Fitness std - (0.025765083081038948) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.671
Generation 180: Best latency = 90, Best rate = 2, Fitness std - (0.025492665147380190) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.644
Generation 181: Best latency = 90, Best rate = 2, Fitness std - (0.025870677418911520) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.618
Generation 182: Best latency = 90, Best rate = 2, Fitness std - (0.025694229636903160) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.593
Generation 183: Best latency = 90, Best rate = 2, Fitness std - (0.025387989534376910) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.570
Generation 184: Best latency = 90, Best rate = 2, Fitness std - (0.025391952657548112) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.547
Generation 185: Best latency = 90, Best rate = 2, Fitness std - (0.025060107886971005) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.525
Generation 186: Best latency = 90, Best rate = 2, Fitness std - (0.025441820260959710) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.504
Generation 187: Best latency = 90, Best rate = 2, Fitness std - (0.025500477964485947) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.484
Generation 188: Best latency = 90, Best rate = 2, Fitness std - (0.024716838059271946) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.465
Generation 189: Best latency = 90, Best rate = 2, Fitness std - (0.025733579622380678) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.446
Generation 190: Best latency = 90, Best rate = 2, Fitness std - (0.025368317333111180) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.428
Generation 191: Best latency = 90, Best rate = 2, Fitness std - (0.025042186627990598) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.411
Generation 192: Best latency = 90, Best rate = 2, Fitness std - (0.024993377807738650) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.395
Generation 193: Best latency = 90, Best rate = 2, Fitness std - (0.024999753085200412) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.379
Generation 194: Best latency = 90, Best rate = 2, Fitness std - (0.025535966809718200) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.364
Generation 195: Best latency = 90, Best rate = 2, Fitness std - (0.024331545870836087) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.349
Generation 196: Best latency = 90, Best rate = 2, Fitness std - (0.025006116661655937) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.335
Generation 197: Best latency = 90, Best rate = 2, Fitness std - (0.024585508250256010) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.322
Generation 198: Best latency = 90, Best rate = 2, Fitness std - (0.024738204724487940) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.309
Generation 199: Best latency = 90, Best rate = 2, Fitness std - (0.024265162743177400) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.296
Generation 200: Best latency = 90, Best rate = 2, Fitness std - (0.024225310034311290) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.285
Generation 201: Best latency = 90, Best rate = 2, Fitness std - (0.023858357328443436) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.273
Generation 202: Best latency = 90, Best rate = 2, Fitness std - (0.024296086590913300) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.262
Generation 203: Best latency = 90, Best rate = 2, Fitness std - (0.023873405043687096) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.252
Generation 204: Best latency = 90, Best rate = 2, Fitness std - (0.024304690946720987) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.242
Generation 205: Best latency = 90, Best rate = 2, Fitness std - (0.023763336153089204) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.232
Generation 206: Best latency = 90, Best rate = 2, Fitness std - (0.023181232743845040) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.223
Generation 207: Best latency = 90, Best rate = 2, Fitness std - (0.023428634608681530) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.214
Generation 208: Best latency = 90, Best rate = 2, Fitness std - (0.023002151081943912) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.205
Generation 209: Best latency = 90, Best rate = 2, Fitness std - (0.023223236640929807) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.197
Generation 210: Best latency = 90, Best rate = 2, Fitness std - (0.022956923687095076) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.189
Generation 211: Best latency = 90, Best rate = 2, Fitness std - (0.022437504539884204) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.182
Generation 212: Best latency = 90, Best rate = 2, Fitness std - (0.022677533687321064) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.174
Generation 213: Best latency = 90, Best rate = 2, Fitness std - (0.022231107917377035) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.167
Generation 214: Best latency = 90, Best rate = 2, Fitness std - (0.021917110333518654) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.161
Generation 215: Best latency = 90, Best rate = 2, Fitness std - (0.021784173189774873) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.154
Generation 216: Best latency = 90, Best rate = 2, Fitness std - (0.021732448544236590) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.148
Generation 217: Best latency = 90, Best rate = 2, Fitness std - (0.021539302379352170) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.142
Generation 218: Best latency = 90, Best rate = 2, Fitness std - (0.021106336731876396) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.136
Generation 219: Best latency = 90, Best rate = 2, Fitness std - (0.020738080051507330) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.131
Generation 220: Best latency = 90, Best rate = 2, Fitness std - (0.020632350267262827) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.126
Generation 221: Best latency = 90, Best rate = 2, Fitness std - (0.020529547805087686) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.121
Generation 222: Best latency = 90, Best rate = 2, Fitness std - (0.020975360174307494) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.116
Generation 223: Best latency = 90, Best rate = 2, Fitness std - (0.019514489968228240) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.111
Generation 224: Best latency = 90, Best rate = 2, Fitness std - (0.020048000788131144) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.107
Generation 225: Best latency = 90, Best rate = 2, Fitness std - (0.018975576390985290) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.103
Generation 226: Best latency = 90, Best rate = 2, Fitness std - (0.018907553803412120) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.098
Generation 227: Best latency = 90, Best rate = 2, Fitness std - (0.018533511693187878) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.095
Generation 228: Best latency = 90, Best rate = 2, Fitness std - (0.018184131417929760) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.091
Generation 229: Best latency = 90, Best rate = 2, Fitness std - (0.017669082037998110) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.087
Generation 230: Best latency = 90, Best rate = 2, Fitness std - (0.017948436825520983) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.084
Generation 231: Best latency = 90, Best rate = 2, Fitness std - (0.017266390586632420) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.080
Generation 232: Best latency = 90, Best rate = 2, Fitness std - (0.017591677582919880) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.077
Generation 233: Best latency = 90, Best rate = 2, Fitness std - (0.016419158026979973) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.074
Generation 234: Best latency = 90, Best rate = 2, Fitness std - (0.016013581962079432) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.071
Generation 235: Best latency = 90, Best rate = 2, Fitness std - (0.015681686163394094) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.068
Generation 236: Best latency = 90, Best rate = 2, Fitness std - (0.015319259290690900) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.065
Generation 237: Best latency = 90, Best rate = 2, Fitness std - (0.014911511479497127) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.063
Generation 238: Best latency = 90, Best rate = 2, Fitness std - (0.014649649992336306) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.060
Generation 239: Best latency = 90, Best rate = 2, Fitness std - (0.013978439197874245) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.058
Generation 240: Best latency = 90, Best rate = 2, Fitness std - (0.013714265046280720) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.056
Generation 241: Best latency = 90, Best rate = 2, Fitness std - (0.012613555758139480) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.053
Generation 242: Best latency = 90, Best rate = 2, Fitness std - (0.012119000841896812) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.051
Generation 243: Best latency = 90, Best rate = 2, Fitness std - (0.011646630024591029) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.049
Generation 244: Best latency = 90, Best rate = 2, Fitness std - (0.011223962057270085) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.047
Generation 245: Best latency = 90, Best rate = 2, Fitness std - (0.010320927378395479) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.045
Generation 246: Best latency = 90, Best rate = 2, Fitness std - (0.009890506632178612) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.044
Generation 247: Best latency = 90, Best rate = 2, Fitness std - (0.009610858791072733) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.042
Generation 248: Best latency = 90, Best rate = 2, Fitness std - (0.008586758777551332) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.040
Generation 249: Best latency = 90, Best rate = 2, Fitness std - (0.008158942465275478) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.039
Generation 250: Best latency = 90, Best rate = 2, Fitness std - (0.007337661060669798) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.037
Generation 251: Best latency = 90, Best rate = 2, Fitness std - (0.006908751812006219) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.035
```

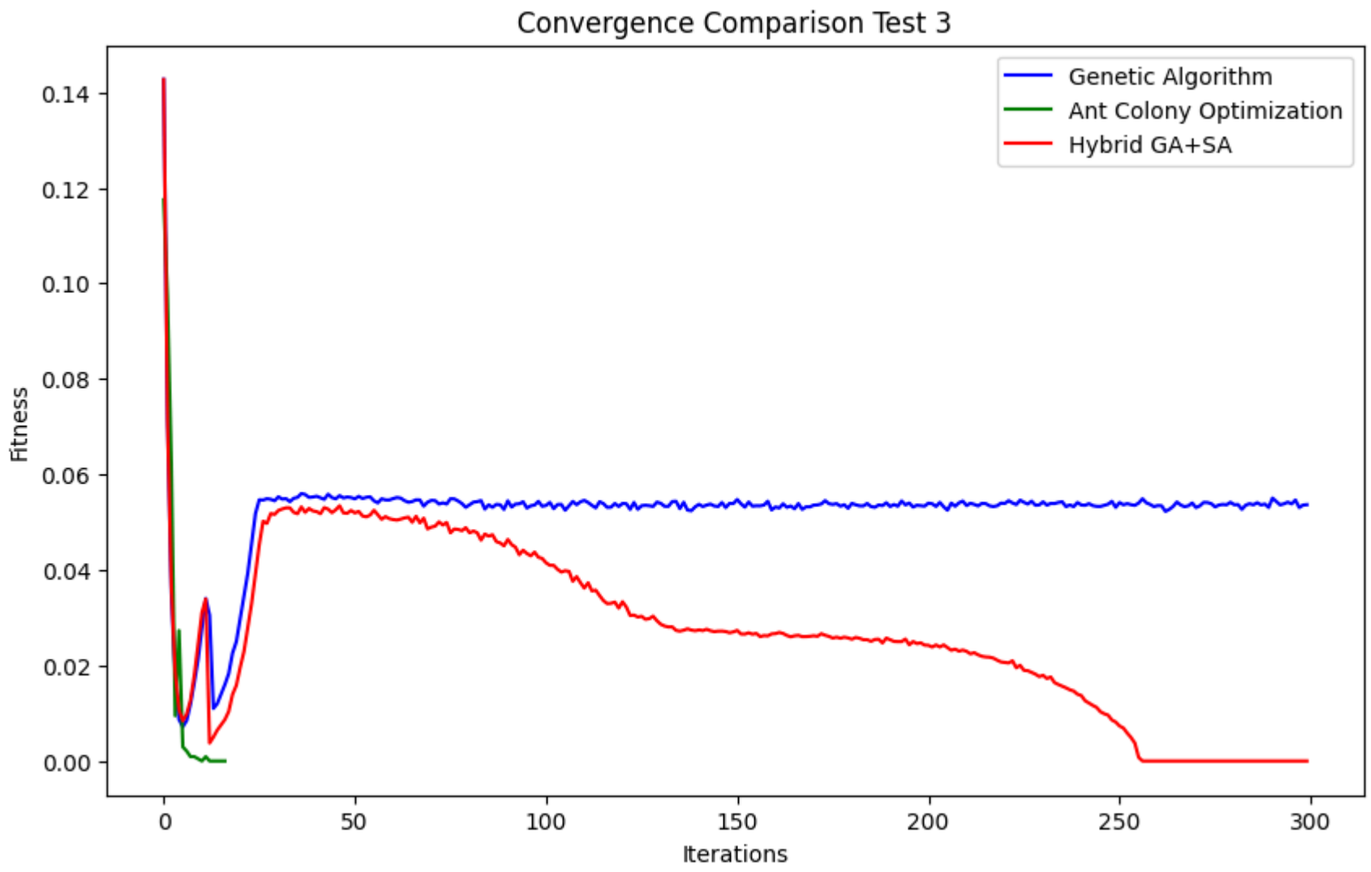
```
Generation 252: Best latency = 90, Best rate = 2, Fitness std - (0.005860556281570312) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.034
Generation 253: Best latency = 90, Best rate = 2, Fitness std - (0.004925439282845719) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.033
Generation 254: Best latency = 90, Best rate = 2, Fitness std - (0.003759383992360401) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.031
Generation 255: Best latency = 90, Best rate = 2, Fitness std - (0.0007528814221894806) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.030
Generation 256: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.029
Generation 257: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.028
Generation 258: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.027
Generation 259: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.026
Generation 260: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.025
Generation 261: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.024
Generation 262: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.023
Generation 263: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.022
Generation 264: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.021
Generation 265: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.020
Generation 266: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.019
Generation 267: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.018
Generation 268: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.018
Generation 269: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.017
Generation 270: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.016
Generation 271: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.016
Generation 272: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.015
Generation 273: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.014
Generation 274: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.014
Generation 275: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.013
Generation 276: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.013
Generation 277: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.012
Generation 278: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.012
Generation 279: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.011
Generation 280: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.011
Generation 281: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.010
Generation 282: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.010
Generation 283: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.010
Generation 284: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.009
Generation 285: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.009
Generation 286: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.009
Generation 287: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.008
Generation 288: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.008
Generation 289: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.008
Generation 290: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.007
Generation 291: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.007
Generation 292: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.007
Generation 293: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.006
Generation 294: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.006
Generation 295: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.006
Generation 296: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.006
Generation 297: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.005
Generation 298: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.005
Generation 299: Best latency = 90, Best rate = 2, Fitness std - (5.551115123125783e-17) Highest fitness for this gen - (0.13298872180451127), Temperature - 0.005
```




```
Best path [[29, 142, 111, 'BS-2'], [29, 22, 111, 'BS-2']]
Best latency 90
Best transmission rate 2
Execution time 34.93862438201904 seconds
```

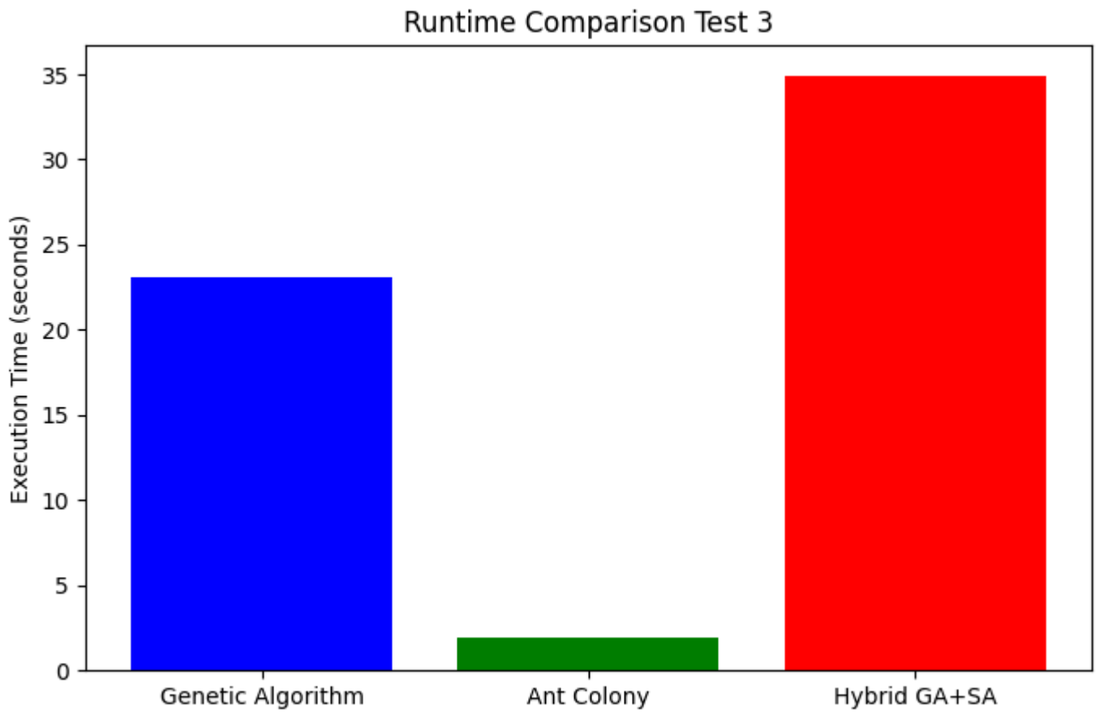
Convergence Curve (Test 3)

In [ ]:
```
plt.figure(figsize=(10, 6))
plt.plot(GA_metrics["Test3"]["Fitness_std"], label='Genetic Algorithm', color='blue')
plt.plot(ACO_metrics["Test3"]["Fitness_std"], label='Ant Colony Optimization', color='green')
plt.plot(HYBRID_metrics["Test3"]["Fitness_std"], label='Hybrid GA+SA', color='red')
plt.xlabel('Iterations')
plt.ylabel('Fitness')
plt.title('Convergence Comparison Test 3')
plt.legend()
plt.show()
```



Runtime Comparison (Test 3)

In [ ]:
```
algorithms = ['Genetic Algorithm', 'Ant Colony', 'Hybrid GA+SA']
runtimes = [
    GA_metrics["Test3"]["Execution_time"],
    ACO_metrics["Test3"]["Execution_time"],
    HYBRID_metrics["Test3"]["Execution_time"]
]

plt.figure(figsize=(8, 5))
plt.bar(algorithms, runtimes, color=['blue', 'green', 'red'])
plt.ylabel('Execution Time (seconds)')
plt.title('Runtime Comparison Test 3')
plt.show()
```



Results Comparison (Test 3)

In [ ]:
```
algorithms = ["Genetic algorithm", "Ant colony algorithm", "Hybrid GA + SA"]
best_paths = [
    GA_metrics["Test3"]["paths"],
    ACO_metrics["Test3"]["paths"],
    HYBRID_metrics["Test3"]["paths"]
]
best_latencies = [
    GA_metrics["Test3"]["latency"],
    ACO_metrics["Test3"]["latency"],
    HYBRID_metrics["Test3"]["latency"]
]
best_transmissions = [
    GA_metrics["Test3"]["transmission"],
    ACO_metrics["Test3"]["transmission"],
    HYBRID_metrics["Test3"]["transmission"]
]
final_score = [
    f"{GA_metrics['Test3']['Fitness'][-1]:.3f}",
    f"{ACO_metrics['Test3']['Fitness'][-1]:.3f}",
    f"{HYBRID_metrics['Test3']['Fitness'][-1]:.3f}"
]
final_fitness_std = [
    f"{GA_metrics['Test3']['Fitness_std'][-1]:.3f}",
    f"{ACO_metrics['Test3']['Fitness_std'][-1]:.3f}",
    f"{HYBRID_metrics['Test3']['Fitness_std'][-1]:.3f}",
]
all_gen_num = [
    GA_metrics["Test3"]["gen"],
    ACO_metrics["Test3"]["gen"],
    HYBRID_metrics["Test3"]["gen"],
]
all_execution_time = [
    f"{GA_metrics['Test3']['Execution_time']:.2f}",
    f"{ACO_metrics['Test3']['Execution_time']:.2f}",
    f"{HYBRID_metrics['Test3']['Execution_time']:.2f}",
]

index = [x for x in range(len(algorithms))]

pd.DataFrame({
    'Algorithms':algorithms,
    'Best path(s)':best_paths,
    'Best latency':best_latencies,
    'Best transmission rate':best_transmissions,
    'Final fitness score':final_score,
    'Final fitness std':final_fitness_std,
    "Number of iteration to converge": all_gen_num,
```

```
    "Execution time (seconds)": all_execution_time
    },index=index)
```

Out[ ]:

	Algorithms	Best path(s)	Best latency	Best transmission rate	Final fitness score	Final fitness std	Number of iteration to converge	Execution time (seconds)
0	Genetic algorithm	[[29, 9, 104, BS-2]]	90	2	0.133	0.054	300	23.05
1	Ant colony algorithm	[[29, 122, 22, 142, 104, 44, 152]]	180	3	0.195	0.000	17	1.91
2	Hybrid GA + SA	[[29, 142, 111, BS-2], [29, 22, 111, BS-2]]	90	2	0.133	0.000	300	34.94

Results Discussion

- The comparison of the three algorithms:
  - Discrete Genetic Algorithm (DGA),
  - Ant Colony Optimization (ACO),
  - Hybrid (GA + SA)
- based on the following performance metrics:
  - transmission rate,
  - latency,
  - fitness scores,
  - convergence speed
  - execution time
- reveals distinct strengths and limitations in solving the multi-objective routing path optimization problem.

DGA discussion

- Across all tests, the DGA displayed a consistent ability to find valid path(s) with balanced latency and transmission rates. For instance, in Test 1 (starting node 14), DGA achieved a latency of 60 ms and a transmission rate of 3 Mbps, indicating its effectiveness in navigating large solution spaces. Furthermore, the algorithm performs calculations rapidly, within seconds, even with a large population set (set to 9000). However, fitness scores of the algorithm continued to improve throughout generations, but the standard deviation of fitness scores showed early convergence, which suggests limited exploration of alternative solutions in later stages.

ACO discussion

- The ACO algorithm outperforms DGA and Hybrid in finding the optimal path. Throughout the tests, the ACO algorithms demonstrated its ability to identify higher-quality paths. The algorithm's tendency to find longer paths negatively impacts the latency. The algorithm's exploration-driven path construction mechanism might be the cause of this problem. ACO exhibited complete convergence, the best solutions output, and fastest execution time throughout the tests compared to the other algorithms, which emphasizes its robustness in network configuration scenarios.

Hybrid discussion

- The Hybrid algorithm demonstrated a strong balance between the two objectives, achieving the lowest latency of all tests. In Test 2, it recorded a latency of 90 ms while maintaining a transmission rate of 2 Mbps. This performance highlights the hybrid algorithm's strength over the Genetic algorithm where the search space is much more explored which can be observed with the standard deviation in test 2 compared to the other algorithms. However, this algorithm suffers from longer convergence and execution time.

Overall results discussion

- Overall, ACO excelled in maximizing transmission rates, making it suitable for scenarios prioritizing data throughput. The hybrid algorithm, on the other hand, provided superior latency minimization, making it ideal for applications requiring a fast internet connection. These findings underscore the importance of algorithm selection based on the specific requirements and constraints of the routing path optimization problem.

Conclusion

- Discrete Genetic Algorithm (DGA), Ant Colony Optimisation (ACO), and a hybrid Genetic Algorithm-Simulated Annealing (GA+SA) were the three algorithms used and compared with the aim to successfully address the optimisation of routing paths in a wireless sensor network with the dual objectives of maximising transmission rates and minimising latency. Also, the advantages and disadvantages of each method were addresses.
- The results indicate that the Ant Colony Optimisation (ACO) approach was highly effective in identifying optimal paths because to its dynamic exploration of the search space. The hybrid methodology, integrating Genetic Algorithm (GA) and Simulated Annealing (SA), achieved an optimal equilibrium between accuracy and efficiency by examining a broad spectrum of options and honing the most promising solutions. Conversely, the Discrete Genetic Algorithm (DGA) functioned adequately as a fundamental approach for pathfinding; nevertheless, it proved less effective in scenarios requiring greater flexibility and adaptability.
- The research employed the Weighted Sum Method (WSM) to simultaneously address multiple objectives, such as optimising transmission rate and minimising latency. This approach facilitated the attainment of an optimal equilibrium between the two objectives. To ensure the efficacy of the solutions across various network types, modifications were implemented to the values and scales utilised in the calculations (normalisation and parameter tuning). This rendered the solutions versatile and responsive to diverse requirements.

Future Work

- The project offers various opportunities to refine and improve the implemented algorithms for this optimization problem:
  - **Algorithmic Improvements:**
    - `Hyperparameter Optimization` : A systematic study of hyperparameter optimisation could be undertaken to determine the parameter configurations that yield optimal performance. This entails examining the mutation rate, pheromone evaporation rate, and cooling schedule variables. This work improves the reliability and robustness of the employed algorithms by calibrating the parameters to each particular problem context.
    - `Hybridization Strategies` : The investigation of additional hybridisation techniques, such as the integration of the Firefly algorithm or the Cuckoo search algorithm, can enhance the balance between exploration and exploitation, ultimately improving the quality of the solution
    - `Scalability Evaluation` : The algorithms' scalability must be evaluated on larger networks that incorporate additional nodes and base stations to guarantee efficiency and effectiveness across various network routing scenarios.
  - **Comparison of Optimization Methods:**
    - `Methodological Alternatives` : Different approaches might have been employed or evaluated throughout the design of the algorithm. For example, during the selection process, methods like weighted pairing, random pairing, or tournament selection may offer further understanding of how different selection strategies influence overall performance. In a comparable manner, the crossover process can be examined through methods such as single-point crossover or uniform crossover to assess their impact on solution diversity and convergence speed.
    - `Cost function investigation` : Examine the application of the Pareto front method as a substitute for the Weighted Sum approach. The Weighted Sum streamlines the optimisation process by consolidating it into a singular objective, whereas the Pareto front presents a collection of trade-off solutions. Analysing these methods can uncover which one excels in achieving an optimal balance between latency and transmission rate.

References

- Alnajjar, A. B., Kadim, A. M., Jaber, R. A., Hasan, N. A., Ahmed, E. Q., Sahib Mahdi Altaei, M. and Khalaf, A. L., 2022. Wireless Sensor Network Optimization Using Genetic Algorithm. Journal of Robotics and Control (JRC) [online], 3 (6), 827–835. Available from: https://journal.umy.ac.id/index.php/jrc/article/view/16526 [Accessed 3 Dec 2024].
- Amit Singh and Dr. Devendra Singh, 2023. Genetic Algorithm-Based Secure Routing Protocol for Wireless Sensor Networks. International Research Journal on Advanced Engineering Hub (IRJAEH), 1 (01), 46–52.
- Bean, J. C. and Hadj-Alouane, A. Ben., 1993. A dual genetic algorithm for bounded integer programs James C. Bean, Atidel Ben Hadj-Alouane. [online]. Available from: http://deepblue.lib.umich.edu/handle/2027.42/3480 [Accessed 22 Dec 2024].
- Blum, C. and López-Ibáñez, M., 2007. Ant colony optimization. Scholarpedia, 2 (3), 1461.
- Deb, K., 2011. Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction. Multi-objective Evolutionary Optimisation for Product Design and Manufacturing [online], 3–34. Available from: https://link.springer.com/chapter/10.1007/978-0-85729-652-8_1 [Accessed 18 Dec 2024].

- Dorigo, M., 1998. AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research [online]. Available from: https://www.academia.edu/760920/AntNet_Distributed_stigmergetic_control_for_communications_networks [Accessed 22 Dec 2024].
- Ghawy, M. Z., Amran, G. A., Alsalman, H., Ghaleb, E., Khan, J., Al-Bakhrani, A. A., Alziadi, A. M., Ali, A. and Ullah, S. S., 2022. An Effective Wireless Sensor Network Routing Protocol Based on Particle Swarm Optimization Algorithm. Wireless Communications and Mobile Computing [online], 2022 (1), 8455065. Available from: https://onlinelibrary.wiley.com/doi/full/10.1155/2022/8455065 [Accessed 3 Dec 2024].
- Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P., 1983. Optimization by Simulated Annealing. Science [online], 220 (4598), 671–680. Available from: https://www.science.org/doi/10.1126/science.220.4598.671 [Accessed 4 Dec 2024].
- Shirazi, A., 2017. Analysis of a hybrid genetic simulated annealing strategy applied in multi-objective optimization of orbital maneuvers. IEEE Aerospace and Electronic Systems Magazine, 32 (1), 6–22.