UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digitals

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

“Optimizing Post-Disaster Resource Allocation Using Metaheuristic Techniques”

INFORME DE PRÁCTICAS

Autora:
Nombre: Federica Vinciguerra

Profesor:
Nombre: Antonio Garrido Tejero

VALÈNCIA, 2024

Contents

1 Introduction **3**

2 Problem Description **4**

3 Genetic Algorithm **6**
3.1 Description of the Algorithm Applied to the Problem 6
3.1.1 Objective Functions 6
3.1.2 Genetic Algorithm Workflow 6
3.2 Parameter Variation 9

4 Simulated Annealing **16**
4.1 Description of the Algorithm Applied to the Problem 16
4.1.1 Objective Functions 16
4.1.2 Multi-Objective Simulated Annealing Workflow 16
4.2 Parameter Variation 17

5 Conclusions **20**

1 Introduction

In recent years, metaheuristic optimization techniques have gained significant attention in solving complex multi-objective problems, particularly those with high-dimensional search spaces and conflicting objectives. Two prominent approaches in this domain are the Genetic Algorithm (GA) and Simulated Annealing (SA). Both methods are inspired by natural processes, with GA mimicking the principles of natural evolution and SA modeling the process of annealing in metallurgy. Despite their shared goal of finding optimal solutions in challenging search spaces, they employ fundamentally different strategies in their search processes.

This work applies GA and SA to a resource allocation problem in a post-disaster scenario, where resources must be allocated to different areas based on competing objectives of minimizing travel distance and maximizing demand satisfaction. This type of problem is representative of real-world challenges, where decision-makers must balance multiple conflicting criteria under resource constraints, often in uncertain or time-sensitive environments.

The objective of this work is to compare the performance of GA and SA in terms of Pareto frontiers generated for the distance-demand trade-off in resource allocation.

The following sections provide a detailed description of the algorithms applied to the resource allocation problem, the parameters used for experimentation, and a comparison of the resulting Pareto frontiers. This analysis aims to highlight the strengths and weaknesses of GA and SA in multi-objective optimization and provide insights into their practical application in complex decision-making scenarios.

2 Problem Description

The problem addressed in this project is the optimal allocation of resources to areas affected by a natural disaster, such as a hurricane, earthquake, or flood. After such catastrophic events, multiple regions often experience significant infrastructure damage, and the immediate delivery of essential resources becomes critical for survival and recovery efforts. The challenge lies in effectively managing the limited resources available at multiple distribution centers and delivering them to the affected areas in a way that maximizes the meeting of demands and minimizes the time required for transportation.

Each affected area has specific demand levels for different types of resources. The key resources considered in this project are:

- **Water**
- **Food**
- **Medical Aid:** Medical supplies, such as first-aid kits, medicines, and other health-related materials.

The availability of these resources varies across multiple distribution centers and each center has a limited stock of resources, constrained by prior allocations. The main objective is to allocate resources from these centers to the affected areas in such a way that two primary goals are achieved:

1. **Minimizing Transportation Distance (or Time):** The objective is to reduce the time required to deliver essential resources to affected areas. In the model, the transportation distance is represented as a matrix, where each entry in the matrix specifies the distance between a resource center and a specific area.
2. **Maximizing Demand Satisfaction:** Each area affected by the disaster has specific demands for different types of resources. The goal is to maximize the proportion of this demand that is met, ensuring that areas receive the supplies they require for immediate relief. By efficiently allocating resources based on the demand of each area, the algorithm ensures that the available resources are used optimally to provide the greatest overall benefit.

In disaster response, it is crucial to prioritize certain areas based on the severity of their condition and the urgency of their needs. To address this, each area is assigned a **priority level**, which reflects the urgency of the demand:

- **High-priority areas (Priority Level 3):** These areas are often the most severely impacted by the disaster and require immediate attention.
- **Medium-priority areas (Priority Level 2):** These areas also face significant challenges but are in a relatively more stable condition.

- **Low-priority areas (Priority Level 1):** These areas are less affected but still require resources to aid in recovery efforts. While they are not in immediate danger, they still need assistance to restore normalcy.

The multi-objective nature of the problem involves balancing these objectives while giving priority to the high-priority areas. This balance becomes especially challenging when the total demand for resources exceeds the supply available at the distribution centers. In such cases, a compromise must be made, and resources must be distributed efficiently to maximize overall relief.

Furthermore, the allocation must respect the following constraints:

- The resource limit constraints at each center, ensuring that no center allocates more resources than available.
- The total allocation to an area must not exceed its demand, ensuring that no excess resources are allocated beyond the required amounts.

This adds another layer of complexity, as any excess allocation would result in wasted resources or penalties, reducing the efficiency of the relief distribution.

3 Genetic Algorithm

3.1 Description of the Algorithm Applied to the Problem

The disaster resource allocation problem involves the efficient distribution of resources from a set of distribution centers to multiple areas affected by a disaster. The goal is to allocate limited resources while considering the distance between the centers and the affected areas, the priority levels of the areas, and the resource demands of each area.

3.1.1 Objective Functions

In this multi-objective optimization problem, two competing objectives are considered:

- **Minimization of Distance:** The first objective is to minimize the total distance traveled by the resources allocated from the distribution centers to the affected areas. This distance is weighted by the priority level of each area, where higher-priority areas (those with priority level 3) are given greater importance in the optimization process. The total weighted distance for a solution is calculated as:

$$D = \sum_{\mathrm{c}=1}^{n_c} \sum_{\mathrm{a}=1}^{n_a} \left(\frac{1}{\mathrm{priority_level_a}} \cdot \mathrm{distance_{c,a}} \cdot 1(\mathrm{allocation_{c,a}} > 0) \right) \tag{3.1}$$

  where n_c is the number of centers, n_a is the number of areas, and 1 is an indicator function that ensures the distance is considered only if resources are allocated between a center and an area.

- **Maximization of Demand Met:** The second objective is to maximize the total resource demand met by the allocation plan. This objective can be mathematically represented as follows:

$$\text{Total demand met} = \sum_{\mathrm{c}=1}^{n_c} \sum_{\mathrm{a}=1}^{n_a} \sum_{\mathrm{r}=1}^{n_r} \mathrm{allocation_{c,a,r}} \tag{3.2}$$

  where $\mathrm{allocation_{c,a,r}}$ represents the amount of resource allocated from center c to area a for resource type r.

3.1.2 Genetic Algorithm Workflow

The genetic algorithm applied to this problem follows the classical steps of evolutionary optimization but with adaptations to handle the multi-objective nature of the problem. Below, we describe each step of the algorithm in detail.

Initialization

The algorithm begins by initializing a population of candidate solutions, where each solution (or chromosome) represents a possible allocation plan of resources from distribution centers to affected areas. The initialization process takes into account the priority level of each area, in particular high-priority area are served by multiple centers in the initial allocation. The chromosome structure consists of a 3-dimensional array, where each matrix within the chromosome represents a specific distribution center. Within these matrices:

- Each row corresponds to a specific area.
- Each column represents a specific type of resource.

Example: The following matrices represent allocations from three different centers to four areas, with each entry corresponding to the amount of a specific resource type allocated.

$$\mathbf{C_1} = \begin{bmatrix} 5 & 3 & 2 \\ 0 & 1 & 4 \\ 2 & 0 & 3 \\ 4 & 5 & 1 \end{bmatrix}, \quad \mathbf{C_2} = \begin{bmatrix} 6 & 2 & 3 \\ 1 & 0 & 2 \\ 0 & 5 & 3 \\ 7 & 4 & 1 \end{bmatrix}, \quad \mathbf{C_3} = \begin{bmatrix} 2 & 1 & 5 \\ 4 & 0 & 3 \\ 3 & 2 & 1 \\ 0 & 3 & 2 \end{bmatrix} \tag{3.3}$$

Fitness Evaluation

After the population is initialized, the fitness of each solution is evaluated using the two objectives described earlier and the result is stored as a pair of fitness values: one for distance and one for demand.

Selection

The algorithm uses **tournament selection** to choose parent solutions for crossover. In each tournament, a subset of solutions is randomly selected, and the best solution (based on a weighted combination of the two fitness objectives) is chosen to be a parent. The selection process balances exploitation (favoring solutions with better fitness) and exploration (introducing diversity).

Crossover

Crossover is performed between pairs of selected parents to produce offspring. Two types of crossover are used:

- **Uniform Crossover:** In this type, for each gene (resource allocation decision), a random choice is made between the genes of the two parents.
- **Two-Point Crossover:** A contiguous segment of the allocation plan is swapped between two parents, allowing larger chunks of information to be exchanged.

Mutation

A mutation operator is applied to the offspring to introduce small random changes. The mutation involves reallocating resources randomly, subject to the constraints on area demand and center resource limits. The mutation rate controls how frequently mutations occur.

Elitism

Elitism ensures that the best solutions from one generation are carried over to the next without modification. This prevents the algorithm from losing high-quality solutions as it evolves. A fixed number of elite solutions are preserved in each generation, based on their combined fitness (distance and demand).

Repair Mechanism

To maintain feasibility, a repair mechanism is applied to any solution that violates the resource constraints. The repair function ensures that:

- The total resources allocated from each center do not exceed the resource limits.
- The total resources allocated to each area do not exceed the area's demand.

This is done by iteratively reducing excess allocations until the constraints are satisfied.

Example:
1. **Initial Chromosome**:

$$\text{Chromosome} = \begin{bmatrix} 5 & 3 & 4 \\ 6 & 7 & 5 \\ 4 & 2 & 6 \end{bmatrix} \tag{3.4}$$

Here, the rows represent different areas, and the columns represent different resource types.
2. **Resource Limits**:

$$\text{Resource Limits} = \begin{bmatrix} 10 & 15 & 12 \end{bmatrix} \tag{3.5}$$

These limits specify the maximum total allocation for each resource type across all centers.
3. **Area Demands**:

$$\text{Area Demands} = \begin{bmatrix} 3 & 2 & 5 \\ 2 & 4 & 7 \\ 8 & 6 & 10 \end{bmatrix} \tag{3.6}$$

This indicates the maximum amount of each resource type that each area demands.

1. **Step 1**: Check Resource Limits for Each Center
   - **For Resource Type 1** (column 0):
     - Total allocated $= 5 + 6 + 4 = 15$ (exceeds limit of 10).
     - Excess $= 15 - 10 = 5$.
     - Reduce allocations:
       * Area 1: Reduce by 5 (0 remaining)
       * Area 2: Reduce by 0 (6 remaining)
       * Area 3: Reduce by 0 (4 remaining)
   - **For Resource Type 2** (column 1):
     - Total allocated $= 3 + 7 + 2 = 12$ (within limit).

  - No changes made.
- **For Resource Type 3** (column 2):
  - Total allocated $= 4 + 5 + 6 = 15$ (exceeds limit of 12).
  - Excess $= 15 - 12 = 3$.
  - Reduce allocations:
    * Area 1: Reduce by 3 (1 remaining)
    * Area 2: Reduce by 0 (5 remaining)
    * Area 3: Reduce by 0 (6 remaining)

$$\text{Step 1 Repaired Chromosome} = \begin{bmatrix} 0 & 3 & 1 \\ 6 & 7 & 5 \\ 4 & 2 & 6 \end{bmatrix} \tag{3.7}$$

2. **Step 2**: Check Area Demands for Each Area

   Now, the allocations are checked and compared with the area demands matrix, eliminating exceeds. This results in the following repaired chromosome:

$$\text{Repaired Chromosome} = \begin{bmatrix} 0 & 2 & 1 \\ 2 & 4 & 5 \\ 4 & 2 & 6 \end{bmatrix} \tag{3.8}$$

Stopping Condition

The algorithm terminates based on a combination of the following conditions:

- **Convergence:** If there is no significant improvement in the fitness values over a specified number of generations (stagnant generations).
- **Fixed Number of Generations:** The algorithm runs for a fixed number of generations, ensuring a controlled computation time.

3.2 Parameter Variation

In the course of this study, the parameters of the genetic algorithm were varied to examine their effects on solution quality and convergence speed. The following parameters were investigated, with the chosen solutions highlighted in **bold text**:

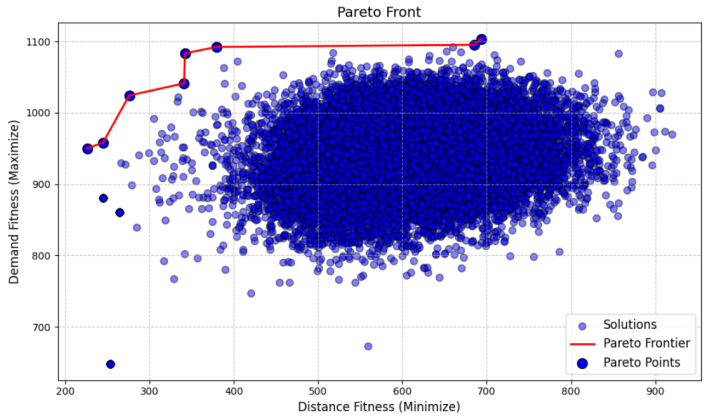
Population Size

A key parameter is the size of the population. Larger populations provide more diversity in the search space, potentially leading to better solutions, but at the cost of increased computational effort. Population sizes ranging from 50 to 300 were tested.

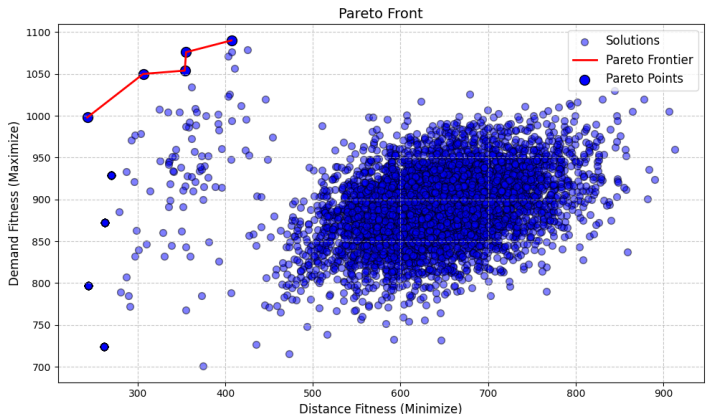
Mutation Rate (Figure 3.1)

The mutation rate controls the extent to which new random changes are introduced to solutions, balancing exploration and exploitation within the solution space.

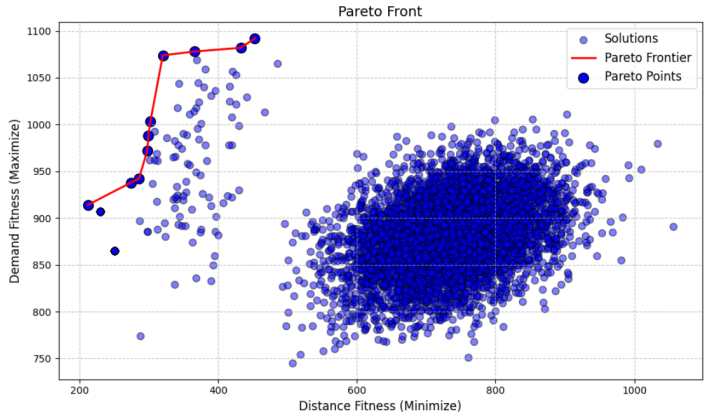
- Mutation Rate = 0.3 (Figure 3.1a): With a low mutation rate of 0.3, the algorithm prioritizes exploitation, focusing on refining solutions near the best-known points. This tendency results in a concentrated clustering of solutions, visible in the figure as a dense grouping close to the Pareto front.
- **Mutation Rate = 0.6** (Figure 3.1b): A mutation rate of 0.6 offers a balanced approach between exploration and exploitation, allowing the algorithm to explore a wider area of the objective space. In the figure, this balance is seen in the broader spread of solutions across the objective space, indicating that the algorithm is discovering more diverse trade-offs between transportation distance and demand satisfaction.
- Mutation Rate = 0.9 (Figure 3.1c): At a high mutation rate of 0.9, the algorithm emphasizes exploration, introducing substantial randomness in the search process. This is reflected in the figure by the wide dispersion of solutions throughout the objective space, as the algorithm avoids converging prematurely to any single region. This reduces the concentration of solutions along the Pareto front, as seen by the more diffuse spread of points.



(a) Mutation Rate = 0.3



(b) Mutation Rate = 0.6



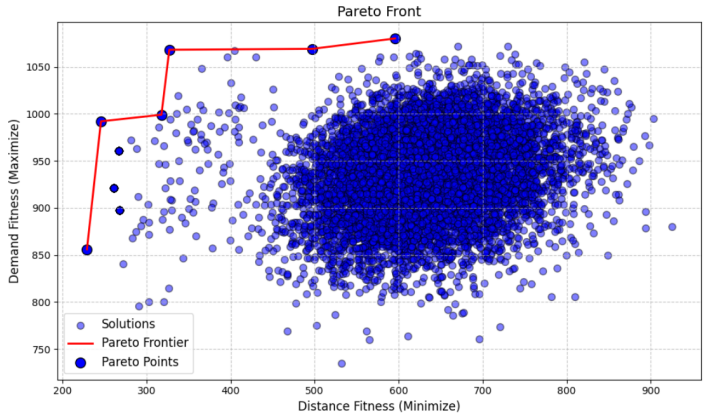
(c) Mutation Rate = 0.9

Figure 3.1: Effect of Mutation Rates on Solution Quality.

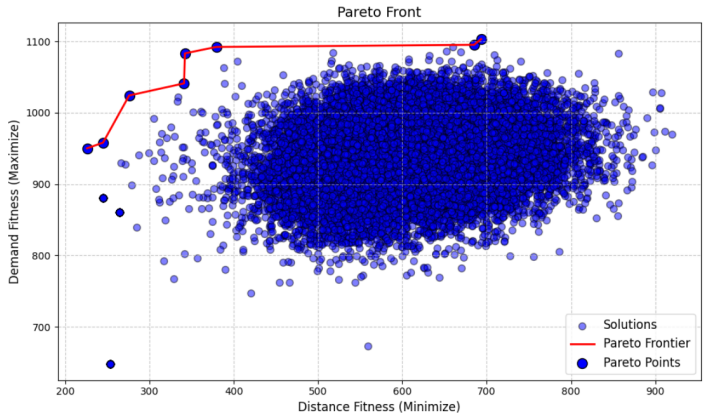
Elitism (Figure 3.2)

Elitism helps retain the best solutions across generations, ensuring high-quality solutions are not lost, but excessive elitism can reduce diversity.

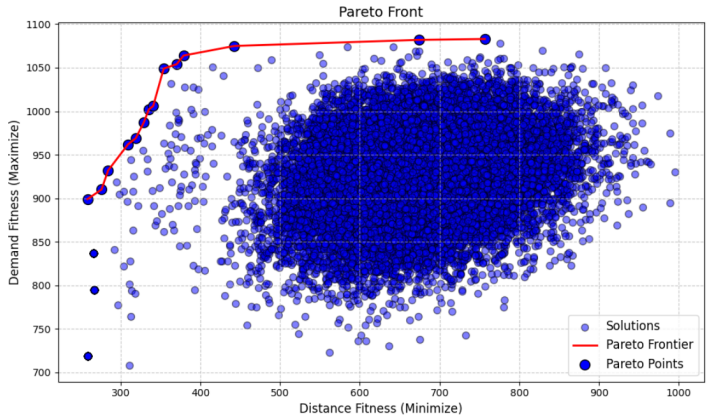
- 1% Elitism (Figure 3.2a): Retaining only 1% of the best solutions maintains diversity but in most of the cases reduces solution quality, resulting in a less prominent Pareto front.
- 5% Elitism (Figure 3.2b): At 5%, high-quality solutions are preserved across generations, resulting in a distinct Pareto front.
- **10% Elitism** (Figure 3.2c): By preserving 10% of the best solutions in each generation, the algorithm maintains a high concentration of solutions along the Pareto front, offering a greater number of points and trade-offs in comparison to lower elitism rates. This helps capture a broader range of optimal solutions, enhancing diversity on the Pareto frontier.



(a) 1% of the population



(b) 5% of the population



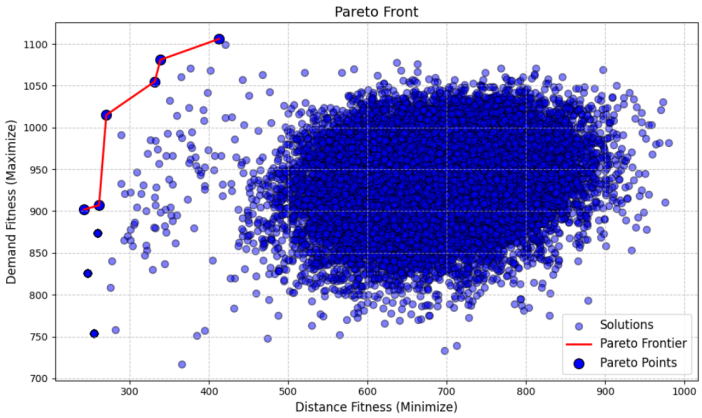
(c) 10% of the population

Figure 3.2: Impact of Elitism on Solution Quality

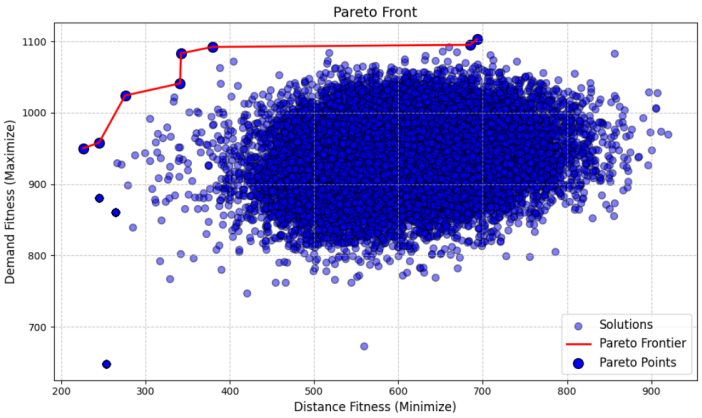
Crossover Type (Figure 3.3)

The crossover method plays a critical role in determining how genetic information is combined in offspring, impacting both solution diversity and convergence speed.

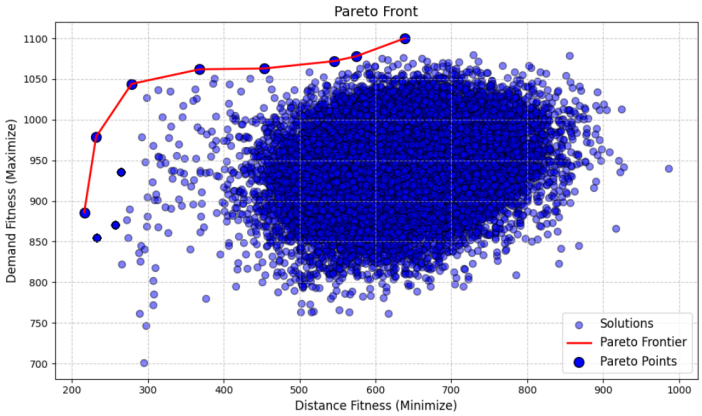
- Uniform Crossover (Figure 3.3a): This method combines genes from both parents at each position with equal probability, resulting in offspring that carry a highly mixed combination of genetic traits. The result, shown in Figure 3.3a, is a diverse set of solutions distributed broadly across the objective space. However, while diversity is enhanced, the Pareto front may not be as well-defined, indicating that convergence to optimal solutions may be slower.
- **Mixed Crossover** (Figure 3.3b): Combining elements of both uniform and two-point crossover methods, mixed crossover balances genetic diversity with faster convergence. As seen in Figure 3.3b, solutions are concentrated closer to optimal trade-offs, leading to a more defined and continuous Pareto front. This method effectively captures high-quality solutions, representing balanced compromises between objectives.
- Two-Point Crossover (Figure 3.3c): This approach swaps larger, contiguous segments of genetic material, leading to offspring that resemble one parent in certain regions. Shown in Figure 3.3c, two-point crossover leads to faster convergence, with solutions forming a concentrated cluster along the Pareto front. While convergence is quicker, the lower diversity may risk trapping the population in local optima.



(a) Uniform crossover



(b) Mixed Crossover



(c) Two-point crossover

Figure 3.3: Effect of Crossover Types on Solution Quality

4 Simulated Annealing

Simulated Annealing (SA) is another optimization technique applied to the resource allocation problem in this project. Like the Genetic Algorithm, SA seeks to allocate resources efficiently from distribution centers to affected areas, balancing two competing objectives: minimizing transportation distance and maximizing the demand met.

4.1 Description of the Algorithm Applied to the Problem

4.1.1 Objective Functions

The Simulated Annealing algorithm uses the same two objective functions used in the GA to guide the optimization process:

- **Minimization of Distance**: The distance calculation formula is given by Equation (3.1).
- **Maximization of Demand Met**: The distance calculation formula is given by Equation (3.2).

4.1.2 Multi-Objective Simulated Annealing Workflow

Initial Solution

A feasible allocation of resources from centers to areas is chosen from the pareto fronter of the previous GA as a starting point.

Neighborhood Generation

Neighborhood generation is a crucial part of the Simulated Annealing process, as it determines how the algorithm explores the solution space. In this problem, a neighborhood is defined as the set of solutions that can be reached by making small modifications to the current solution. The neighborhood generation method used here is based on the following steps:

1. **Random Selection**: A center, area, and resource type are randomly selected from the current solution.
2. **Increment/Decrement and Feasibility Check:** In this phase, a small modification is applied to the selected allocation. This modification involves either increasing or decreasing the amount of resources allocated from the selected center to the selected area by 1 unit.

   The decision on whether to increment or decrement is based on the current allocation status:

- **Increment**: If the current total allocation of the selected resource type in the area is less than the area's demand, and the total resources allocated from the selected center for that resource type is within its limit, the allocation is increased by 1 unit.
- **Decrement**: Conversely, if the current allocation meets or exceeds the area's demand and the allocation for that resource type is greater than zero, the allocation is decreased by 1 unit.

This mechanism ensures that resources are allocated efficiently while adhering to both the area demands and the resource limits constraints.

Cooling Schedule

The algorithm uses an exponential cooling schedule to gradually decrease the temperature parameter, reducing the likelihood of accepting worse solutions over time. The temperature starts at a high value and is gradually reduced by a cooling rate of 0.995 until it reaches a minimum value or a stopping condition is met.

The algorithm iteratively generates new neighboring solutions and accepts them based on their fitness values and a probability determined by the current temperature. It continues to explore the solution space until the temperature is sufficiently low or no improvement is observed after a fixed number of iterations. The best solution found during the process is returned as the final allocation.

Stopping Conditions

The algorithm can terminate under different conditions depending on the problem and its parameters. In this case, the stopping conditions are as follows:

- **Minimum Temperature (T_{min})**: The annealing process halts when the temperature reaches a predetermined minimum value ($T_{min} = 1 \times 10^{-5}$). At this point, the probability of accepting worse solutions becomes negligible, and the solution is considered close to optimal.
- **Max Iterations**: The algorithm is set to stop after a fixed number of iterations (10,000 in this case). This serves as a safeguard to ensure the algorithm does not run indefinitely.
- **No Improvement in Solution**: The algorithm also halts if there is no improvement in the fitness values after a certain number of iterations. This condition is checked continuously during the process to detect stagnation in the search.

4.2 Parameter Variation

In the course of this study, the parameters of the simulated annealing algorithm were varied to examine their effects on solution quality. The following parameters were investigated:

Max Iterations

The total number of iterations the algorithm performs. In this case, the maximum number of iterations is set to 10,000.

Initial Temperature (T_0)

This parameter controls the starting level of exploration. A high initial temperature ($T_0 = 1000$) allows the algorithm to explore more freely at the beginning.

Cooling Rate (Ω)

This parameter controls how fast the temperature decreases. In this case, a cooling rate of 0.995 is used, meaning the temperature decreases by 0.5% in each iteration.

Number of Neighbors Generated

During each iteration, the algorithm generates a specified number of neighbor solutions to explore the solution space effectively. For this study, the number of neighbors generated per iteration is set to 50.

Figures 4.1 and 4.2 illustrate the performance of the algorithm with 10 and 50 neighbors generated, respectively. The results demonstrate a direct improvement in solution quality when 50 neighbors are utilized, indicating that a larger neighborhood exploration can lead to better optimization outcomes.

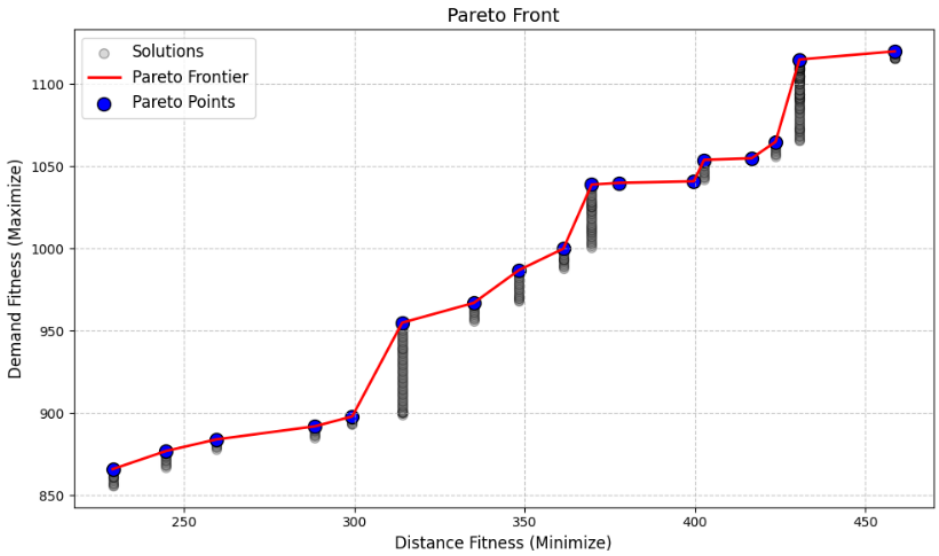


Figure 4.1: Performance with 10 neighbors generated.

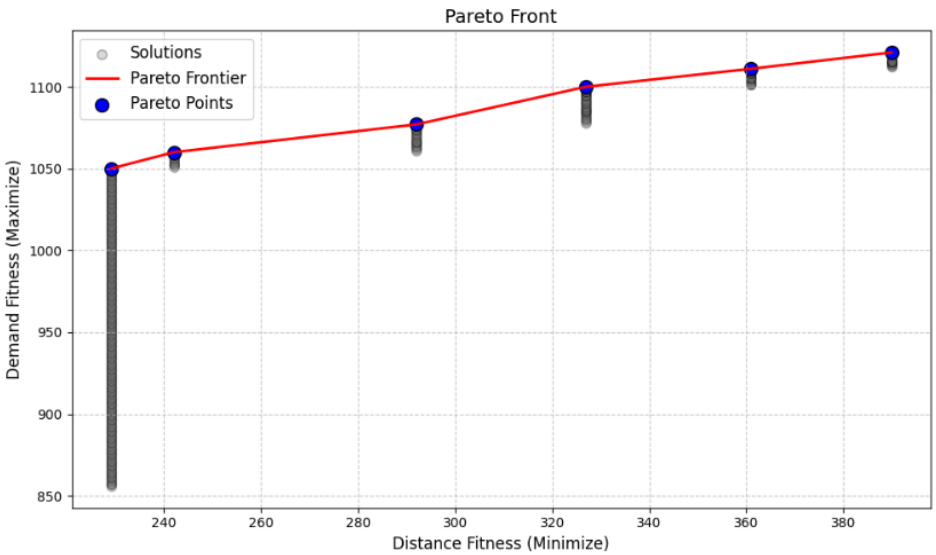


Figure 4.2: Performance with 50 neighbors generated.

5 Conclusions

The comparison between the Genetic Algorithm (GA) and Simulated Annealing (SA) reveals distinct advantages and limitations in their approach to solving the multi-objective optimization problem. The GA, being a population-based algorithm, evolves multiple candidate solutions simultaneously through operations such as selection, crossover, and mutation. This population diversity enables GA to explore a wider solution space, as shown in the Pareto frontier results, achieving high levels of demand satisfaction even with relatively large distances. This balance of exploration and exploitation makes GA suitable for problems requiring diverse trade-offs.

In contrast, Simulated Annealing (SA) operates with a single candidate solution, making incremental improvements. The initialization of SA significantly affects its performance, as illustrated in Figures 5.1 and 5.2. When SA is initialized with a random solution (Figure 5.1), it favors exploratory moves, achieving solutions with smaller distances but lower demand satisfaction. Conversely, with a well-initialized starting point (Figure 5.2), SA finds solutions that maximize demand but have less variability in distance, focusing on local improvements rather than broad exploration.

One notable advantage of SA is its lower computational complexity compared to GA. This makes SA especially useful for scenarios where computational efficiency is a priority, even though it may not explore the solution space as broadly as GA.

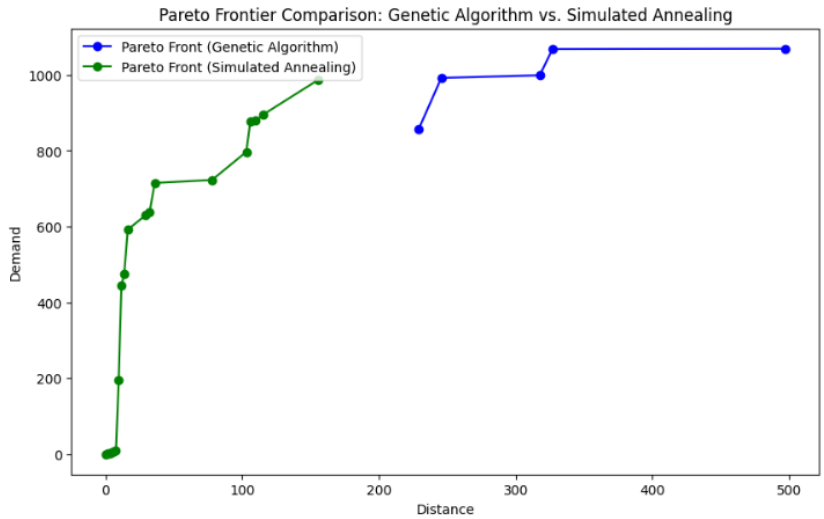


Figure 5.1: Pareto Frontier Comparison: Genetic Algorithm vs. Simulated Annealing with Random Initialization

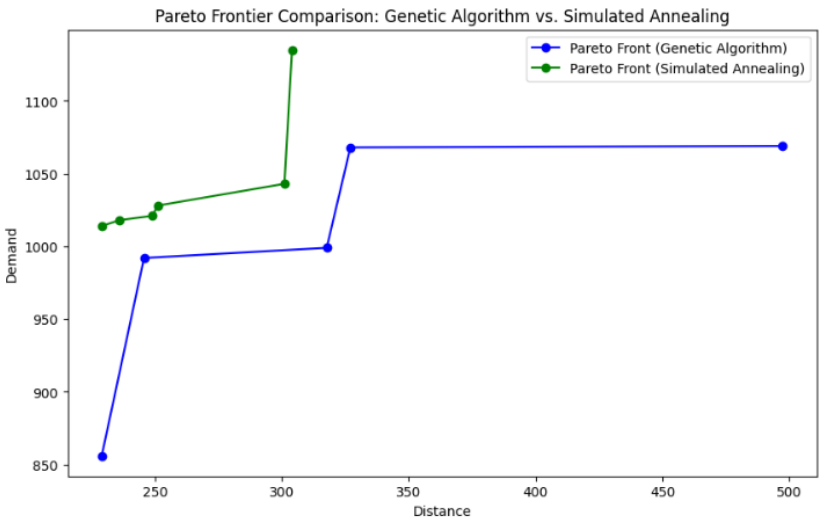


Figure 5.2: Pareto Frontier Comparison: Genetic Algorithm vs. Simulated Annealing with Well-Initialized Starting Point