

## IT426: ARTIFICIAL INTELLIGENCE SYSTEMS

# **PerfectFit: A GA-Based Solution for Optimal Outfit**

*Phase-2*

Prepared by:

Section: 56617 (Group#4)	
Name	ID
Wiam Baalahatar (Leader)	443200416
Hind Alhijailan	443200971
Wijdan Alhashim	443200530
Sarah Aldbasi	443200520
Khlood Aldoayan	443201002

Supervised by:

L.Nouf Alfear

1<sup>st</sup> Semester 1446

## Table of Contents

Introduction .....	3
Problem description.....	4
Solution approach .....	4
Solution Representation .....	5
Fitness function.....	6
Selection process... ..	7
Genetic operators.....	8
Termination condition .....	9
Graph of GA Performance .....	10
Results Analysis.....	12

## Table of Figure

Figure 1: Genetic Algorithm Workflow .....	3
Figure 2: user input .....	10
Figure 3: user output.....	11
Figure4 : GA Performance Graph .....	11

# 1. Introduction

Choosing the right outfit for an event can often be a difficult task, especially when trying to balance aesthetic preferences with practical considerations like dress code, budget, and comfort. Most individuals face the challenge of selecting clothes that not only meets social expectations but also aligns with their personal style and comfort. This decision-making process becomes even more complex when the wardrobe offers a plethora of choices, each differing in style, color, and suitability for the occasion.

To address this common dilemma, our project, PerfectFit, employs a genetic algorithm (GA) to optimize outfit selection. This approach leverages the principles of natural selection and genetics to systematically explore and evaluate combinations of clothing items from a predefined inventory. The objective is to automate the process of assembling outfits that adhere to specified criteria including dress code, color palette, comfort level, and budget constraints.

By simulating the process of evolution through selection, crossover, and mutation, the genetic algorithm aims to find an optimal combination of garments that maximizes a user-defined fitness function. This function assesses the suitability of potential outfits based on weighted factors such as compliance with the dress code, adherence to the desired color scheme, overall comfort, and cost-effectiveness. The application of a genetic algorithm to outfit selection not only simplifies personal decision-making but also introduces an innovative way to utilize AI in everyday life decisions.

This document outlines how a genetic algorithm is used to select the ideal outfit, starting with defining the problem and the challenges associated with choosing clothing. It describes the solution approach, explains how solutions are represented, and provides a detailed description of the fitness function used to evaluate outfits. The selection process is also discussed, highlighting the binary tournament selection method and other genetic operators such as crossover, mutation, and replacement. Additionally, the termination condition for the algorithm is explained. The report further includes an analysis of the Genetic Algorithm's performance using a fitness graph that shows results over multiple generations and runs, demonstrating its ability to converge on best solutions efficiently. Finally, the document presents the user input process, the output of the best-selected outfit, and the overall results analysis to evaluate the robustness and practicality of the solution.

## 2. Problem description

Selecting the right outfit for various social and professional occasions is a challenge that most people encounter. The process involves multiple considerations, such as adhering to specific dress codes, staying within a budget, choosing appropriate colors, and maintaining personal comfort. The complexity of these decisions increases with the size and variety of one's wardrobe and the diversity of available stores and products.

For example, preparing for a corporate evening event that demands semi-formal attire requires a balance between elegance and comfort for long hours of mingling. Additionally, one may prefer using pieces already owned to stay within budget. This highlights the need for a tool like PerfectFit, which uses a genetic algorithm to automate and optimize outfit selection. By evaluating potential combinations based on event criteria, PerfectFit helps users efficiently select an outfit that matches their style, comfort level, and budget.

## 3. Solution approach

In order to address the common challenges in selecting the perfect outfit, discussed in the problem section above, we present PerfectFit, a system designed to optimize outfit selection using a Genetic Algorithm (GA). The process begins by generating an initial population of solutions, where each solution represents a different outfit. Each outfit consists of items selected from various categories, such as Top, Bottom, Shoes, Neck, and Purse, ensuring a wide range of combinations to evaluate.

The GA iteratively applies selection, crossover, mutation, and replacement operations to evolve the population toward better solutions (see Figure 1). The fitness function evaluates the quality of each outfit based on its alignment with the desired dress code, color palette, comfort level, and budget. This iterative process continues until a specified termination criterion, such as reaching a set number of generations or achieving a satisfactory fitness score, is met.

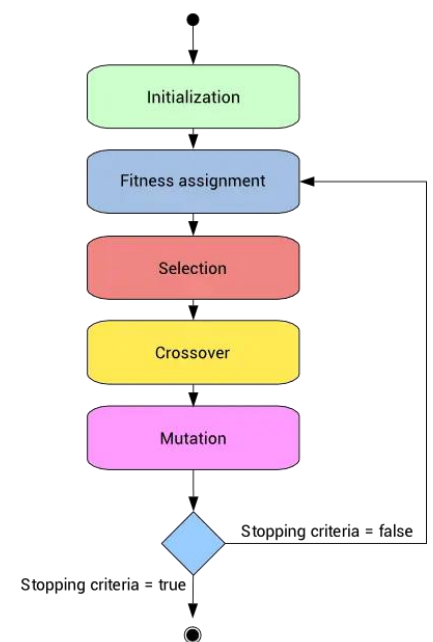


Figure 1: Genetic Algorithm Workflow

## 4. Solution Representation

In this problem, the solution representation is based on a chromosome structure, where each individual (or chromosome) represents a complete outfit. The chromosome is a fixed-length list with each gene corresponding to a category of clothing or accessories (e.g., Top, Bottom, Shoes, Neck, and Purse).

Each gene in the chromosome is an integer that acts as an index, representing a specific item within a category from the predefined search space. For instance, a value of 0 for the “Top” category could represent a “T-shirt,” while a value of 2 in the “Bottom” category could represent “Sports Shorts.”

- **Chromosome Representation:** A list of indices, where each index corresponds to a particular item in the associated category.
- **Gene Representation:** Each gene is an integer that identifies a specific item within a category (e.g., Top, Bottom, Shoes, etc.).

Thus, an individual chromosome might look like: **[0, 4, 4, 1, 3]**

Where:

- 0 refers to the first item in the “Top” category,
- 4 refers to the fifth item in the “Bottom” category,
- and so on.

This encoding allows for the representation of multiple outfit combinations, each of which can be evaluated based on the user’s preferences such as dress code, color palette, comfort level, and budget. The fitness of each chromosome is calculated based on how well it meets these preferences, and the best combinations have a higher probability of being selected to form the next generation of solutions.

## 5. Fitness function

In the PerfectFit system, the fitness function is designed to evaluate how well a given outfit represented by a chromosome satisfies the user's preferences in four key areas: dress code, color palette, comfort level, and budget. Each of these factors is assigned a weight, which reflects its importance in the final fitness score, and the fitness score is a sum of the individual scores for each factor. The overall fitness value is constrained to have a value between 0 and 1, where a higher fitness score represents a better match with the user's preferences.

### Fitness Function Parameters:

- **Dress Code Score:** This measures how well the selected items match the desired dress code. If all items in the outfit match the user's preferred dress code, the score will be 1, otherwise, it is scaled based on the number of matching items.
- **Color Palette Score:** Similar to the dress code score, this assesses how well the selected items match the user's preferred color palette which has a value of either dark or bright.

Both **Dress Code Score** and **Color Palette Score** can be calculated using this formula:  
Number of matching items / 5 (Total number of categories).

- **Comfort Score:** This score evaluates how close the selected items' average comfort level is to the user's desired comfort level. The comfort level ranges from 1 (least comfortable) to 5 (most comfortable).

Comfort Score can be calculated by this formula:

$$1 - (|\text{Average comfort level} - \text{Desired comfort level}|) / 4$$

The division by 4 ensures the score remains in the range [0,1], as the maximum difference between comfort levels is 4.

- **Budget Score:** This score compares the total cost of the selected items with the user's specified budget. If the total cost is within the budget, the score is 1. If the cost total exceeds the budget, the score is penalized proportionally by how much it exceeds the budget by dividing the budget over the total cost.

### Weighting the Fitness Function:

The fitness function is a weighted sum of the four criteria: dress code, color palette, comfort, and budget. In the current implementation, equal weights are assigned to each factor:

$$w1 = w2 = w3 = w4 = 0.25$$

This ensures that all four factors are treated as equally important when calculating the fitness of an outfit. The total fitness score is computed as follows:

$$\text{Fitness} = (w_1 \times \text{Dress Code Score}) + (w_2 \times \text{Color Palette Score}) + (w_3 \times \text{Comfort Score}) + (w_4 \times \text{Budget Score})$$

Since each individual score is between 0 and 1, and the sum of the weights is 1, the final fitness score will also be in the range [0,1] for example, if the fitness score is equal to 0.78, indicating a good match with the user's preferences.

## 6. Selection process

In our project, we utilize a binary tournament selection method as the primary mechanism to select two parent solutions from the population. The binary tournament selection is a well-known and simple selection method in genetic algorithms, which ensures that the fittest individuals in the population have a higher probability of being selected for reproduction while maintaining some level of diversity.

A detailed explanation of how the binary tournament selection works in our project:

### 1. Random Selection:

To select two parents for reproduction, the algorithm runs the binary tournament selection process twice in each round, two random individuals (solutions) from the population are chosen.

### 2. Fitness Comparison:

Once two individuals are chosen, the algorithm compares their fitness scores. The individual with the higher fitness score is selected as the parent.

### 3. Tie Breaking:

If the two randomly selected individuals have the same fitness score, the algorithm randomly selects one of them to be the parent.

### 4. Selection with Replacement:

After selecting the first parent, the process is repeated to select the second parent. It is possible to select the same individual for both parents, but this rarely happens due to the randomness of the selection.

## 5. Returning Selected Parents:

Once two parents are selected, they are ready for the next steps in the next phase (such as crossover and mutation). In our current phase, the selected parents are simply printed to show their corresponding outfits.

The binary tournament selection is efficient and ensures that solutions (outfits) that are more aligned with the user's preferences are more likely to be selected, while still maintaining some randomness to preserve diversity in the population.

## 7. Genetic operators

The Genetic Algorithm (GA) involves a series of genetic operations designed to evolve a population of solutions over multiple generations, improving their fitness to meet user-defined preferences.

- **Binary Tournament Selection:** As mentioned earlier in the Selection Process section, this operation randomly selects two individuals from the population and compares their fitness values. The better one is chosen as a parent to help create the next generation. This ensures stronger solutions are passed on, while sometimes allowing less optimal solutions to keep diversity.
- **2-Point Crossover:** In this operation, two-parent solutions are chosen, and two random crossover points are determined along their chromosomes. Genetic material between these points is swapped to create new solutions (offspring). This method helps increase diversity by mixing larger sections of genetic information from both parents, leading to new outfit combinations.
- **Mutation:** Mutation introduces genetic variation by randomly altering one or more genes in a solution. For instance, a specific item in an outfit might be replaced with another item from the same category. This operation prevents the algorithm from becoming trapped in local optima by exploring alternative solutions that might lead to better outcomes. Mutation ensures that the population retains its adaptability and continues to explore less conventional but potentially optimal solutions.
- **Replacement:** Replacement incorporates new offspring into the population to form the next generation. This process often involves removing weaker individuals while retaining the strongest ones (a concept known as elitism). By preserving the best solutions and integrating promising new offspring, replacement ensures that the population evolves toward higher fitness over generations. This step balances the exploration of new solutions with the exploitation of well-known solutions.



These genetic operators work in tandem to simulate the process of natural selection, enabling the GA to efficiently search for optimal solutions. By balancing the introduction of new variations and the refinement of existing solutions, the algorithm systematically converges toward outfits that best align with the user's preferences for dress code, color palette, comfort, and budget.

## 8. Termination condition

The termination condition for a Genetic Algorithm (GA) is critical to prevent indefinite execution. For the PerfectFit outfit selection problem, we employed a termination criterion based on the maximum number of generations. This approach, while simple to implement, offers several key advantages:

- **Simplicity and Efficiency:** Setting a maximum generation limit (in our case, 300) provides a straightforward and efficient termination strategy. This simplicity is particularly beneficial during the initial development and testing phases, enabling rapid experimentation and parameter tuning.
- **Guaranteed Completion:** Unlike fitness-based or error-based termination criteria, which might not always be met, a maximum generation limit ensures the algorithm completes execution within a predictable timeframe. This is vital for managing computational resources and adhering to project deadlines.
- **Suitability for Subjective Optimality:** The concept of an "optimal" outfit in PerfectFit is inherently subjective, depending on user preferences and varying interpretations of dress code, color palette, budget, and comfort level. A maximum generation limit avoids the challenge of defining a precise fitness threshold or an error tolerance against an elusive optimal solution.
- **Controlled Computational Cost:** The maximum generation count allows direct control over the algorithm's computational cost. Setting a limit of 300 generations balances efficiency with the need for a reasonable exploration of the solution space.
- **Facilitating Experimentation:** The maximum generation approach greatly simplifies experimentation. Multiple runs with a consistent generation limit allow easy comparison of results across various parameter settings (population size, mutation rate, crossover method, etc.), accelerating the identification of optimal algorithm configurations.

Our choice of 300 generations for termination represents a balance between speed and solution quality. This number allows for a more extensive exploration of the solution space compared to a smaller number of generations, potentially yielding higher-quality outfit suggestions. While still prioritizing efficiency during development and testing, the increased generation count offers a more thorough search for suitable outfits before termination.

## 9. Graph of GA Performance

This section highlights the Graph performance of the Genetic Algorithm (GA) used in the PerfectFit system to optimize outfit selection. The graph illustrates how the algorithm evolves over generations to improve the population's fitness. By tracking the average fitness score across 20 independent runs, the graph demonstrates the algorithm's ability to refine solutions and consistently achieve high-quality outcomes. Below, we will explain the user inputs, the resulting optimal outfit, and the observed trends in the GA performance graph.

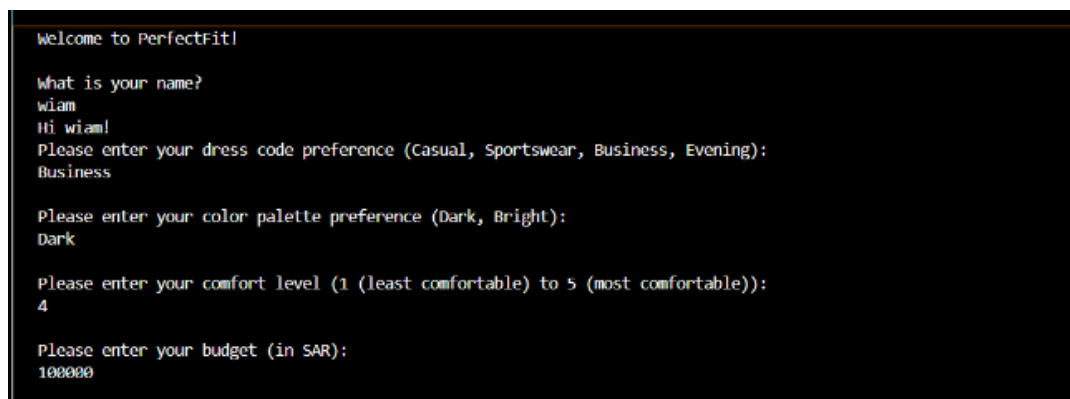
### Input and Output Details:

#### 1. User Input

In Figure 2, we see an example of the user input process in the PerfectFit system. The user is prompted to enter their preferences, including:

- **Dress Code:** Choices like Casual, Sportswear, Business, or Evening.
- **Color Palette:** Either Dark or Bright.
- **Comfort Level:** A scale from 1 (least comfortable) to 5 (most comfortable).
- **Budget:** The maximum amount the user is willing to spend.

The system gathers this input to tailor the outfit selection, ensuring the suggested solutions align with the user's preferences and constraints.



```
Welcome to PerfectFit!

What is your name?
wiam
Hi wiam!
Please enter your dress code preference (Casual, Sportswear, Business, Evening):
Business

Please enter your color palette preference (Dark, Bright):
Dark

Please enter your comfort level (1 (least comfortable) to 5 (most comfortable)):
4

Please enter your budget (in SAR):
100000
```

Figure 2: user input

## 2. Best Output

In Figure 3, the system displays the optimal outfit generated by the GA based on the user's inputs. This output includes:

- **Selected Items:** The output presents a list of clothing items chosen for each category, such as Top, Bottom, Shoes, Neck, and Purse, based on the user's preferences and constraints.
- **Affirmation Message :**"Hope you feel fabulous in your outfit!".
- **Overall Fitness Score:** The output provides a fitness score that indicates how well the selected outfit matches the user's defined criteria. This score is calculated as the average fitness across 20 runs.

This step showcases the effectiveness of the GA in generating practical, user-specific solutions.

```
We Are Working on preparing your optimal outfit...

Your Best Outfit Selection:
Top: Formal Shirt
Bottom: Chinos
Shoes: Loafers
Neck: Silk Scarf
Purse: Leather Briefcase

Hope you feel fabulous in your outfit!

Overall average fitness across 20 runs: 0.90
```

Figure 3: user output

### GA Performance Graph:

Figure 4 represents the performance of the Genetic Algorithm across 300 generations. The graph plots the average fitness score of the population for each generation during 20 independent runs.

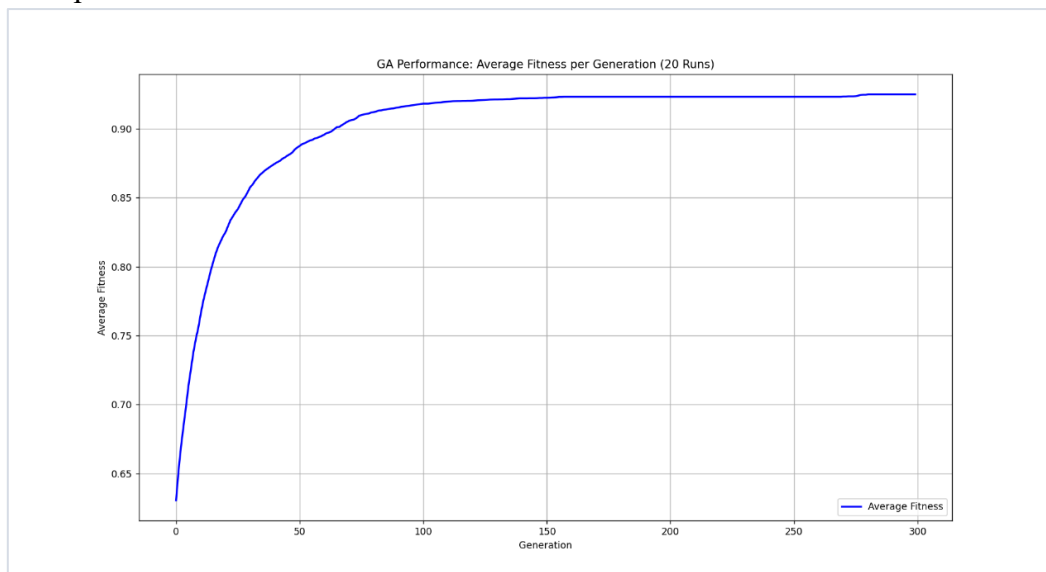


Figure4 : GA Performance Graph

## 10. Results Analysis

The fitness performance graph highlights the efficiency and consistency of the implemented Genetic Algorithm (GA) in identifying the best solutions for the outfit selection problem. Refer to Figure 4 for the graphical representation of the GA performance across generations.

### 1. Initial Phase (Generations 0–50)

- **Sharp Increase in Fitness Scores:** A noticeable sharp increase in average fitness scores is observed during the first 50 generations. This phase represents the exploration stage, where the GA widely searches the solution space to identify promising regions.
- **High Population Diversity:** The diversity in the population is at its peak during this phase. Genetic operators like mutation and crossover introduce new genetic variations, allowing the algorithm to explore a wide array of outfit combinations.
- **Effective Use of Genetic Operators:** The significant improvements in fitness scores indicate that the genetic operators are effectively combining and altering solutions to discover outfits that better match the user's preferences.

### 2. Intermediate Phase (Generations 50–150)

- **Slowing Improvement Rate:** The rate of improvement begins to slow down as the algorithm transitions to the exploitation stage. By this point, the population consists of individuals with higher fitness scores, and the GA focuses on fine-tuning these solutions.
- **Focused Optimization:** The algorithm concentrates on exploiting the most promising areas of the solution space. It refines existing high-quality solutions rather than exploring new ones extensively.
- **Reduced Variability Between Runs:** Variability between different runs becomes less pronounced, indicating that the algorithm consistently identifies similar optimal solutions across multiple executions.

### 3. Convergence Phase (Generations 150–300)

- **Plateauing Fitness Scores:** Fitness scores plateau, demonstrating that the population has largely converged to solutions near the optimal fitness value of 0.90. The improvements per generation become minimal.
- **Stability of Solutions:** The population reaches a stable state where most individuals have similar high fitness scores. This suggests that the GA has effectively optimized the outfit combinations to align with the user's preferences.

- **Minor Fluctuations Due to Stochastic Elements:** Some runs achieve slightly better fitness than others, which may result from the stochastic nature of genetic operations like mutation and selection. These minor fluctuations help prevent premature convergence and maintain genetic diversity.

### Overall Performance

The overall average fitness score of 0.90 across 20 runs demonstrates the effectiveness of the Genetic Algorithm in consistently finding solutions that meet the user's preferences and constraints. This high fitness value reflects the algorithm's ability to balance diverse factors and fine-tune the population's fitness over generations, ultimately converging toward optimal solutions.

### Practical Implications

The consistently high fitness scores across multiple runs emphasize the robustness of the implemented solution. The GA successfully adapts to the user's inputs, consistently providing outfit recommendations that align with the desired criteria.

This performance highlights:

- **Practicality:** The algorithm effectively solves a real-world problem by automating the complex decision-making process of outfit selection.
- **Reliability:** Users can trust the system to provide recommendations that meet their preferences, enhancing user satisfaction.
- **Scalability:** The approach can be extended to larger search spaces with more clothing options and categories, making it a versatile tool for personalized recommendations.

In summary, the Genetic Algorithm effectively optimizes outfit selection, consistently meeting user preferences and demonstrating reliability and practicality for personalized recommendations.