AHP Tools for Design-Making for Customized AM Lower-Limb Prosthetics

Isha A. Gujarathi and Albert E. Patterson

Texas A&M University

Overview

The Python code for [1] is attached in this document. This is supported by a tool created in MS Excel [2], where both tools are important supplemental materials for the paper.

References

[1] Gujarathi, I.A., Zahabi, M., Pei, Z.J., & Patterson, A.E. (2024). "Decision-making framework for customized additively-manufactured lower-limb prosthetics". Proceedings of the 2024 Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference, August 11-14, 2024, Austin, Texas.

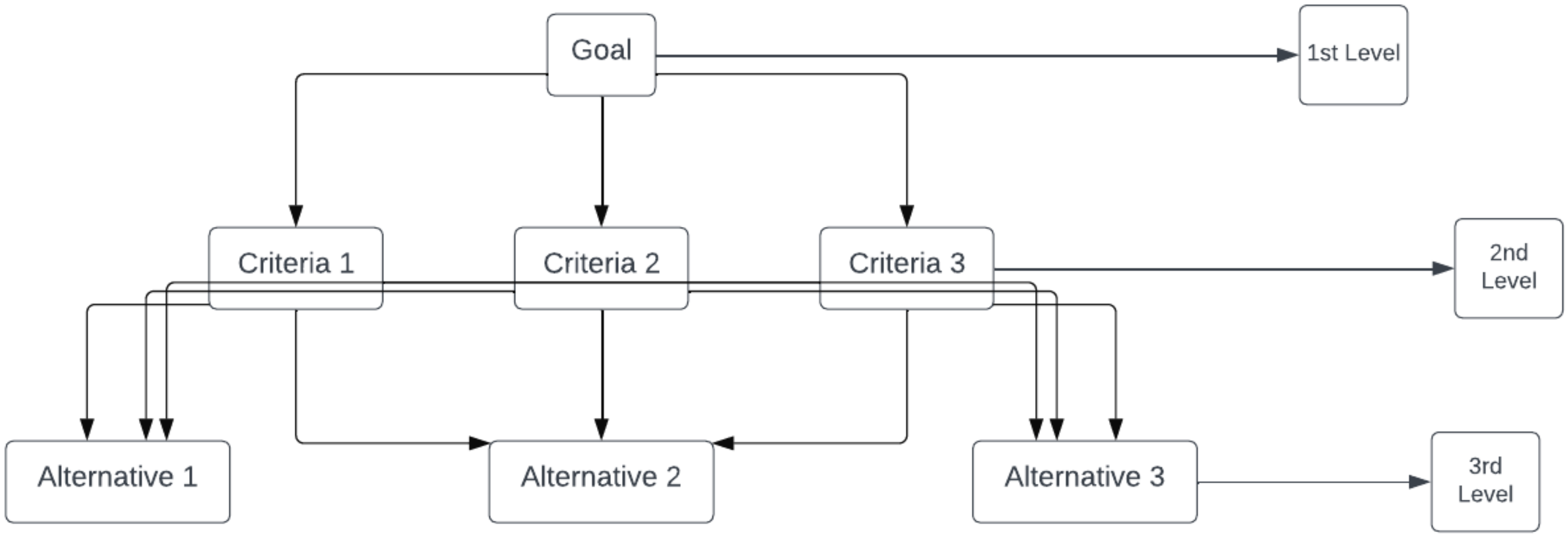
[2] "AHP Charting Tool for Design-Making for Customized AM Lower-Limb Prosthetics". Available electronically from https://hdl.handle.net/1969.1/201424.

In [ ]:
```
!python -m pip install ahpy
```

In [ ]:
```
# Importing all necessary functions
import ahpy
from fractions import Fraction
import tkinter as tk
from tkinter import messagebox
```

In [1]:
```
# The hierarchy for decision making
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open('AHP.png')
img
```

Out[1]:



In [2]:

```
# Comparison Scale
img2 = Image.open('Picture1.png')
img2
```

Out[2]:

Ranking Scale								
9	*7*	*5*	*3*	*1*	*3*	*5*	*7*	*9*
Absolutely More Important	*Very Much More Important*	*Much More Important*	*Somewhat More Important*	*Equal Importance*	*Somewhat More Important*	*Much More Important*	*Very Much More Important*	*Absolutely More Important*

In [ ]:
```python
from fractions import Fraction
import ahpy

# Enter the number of factors
num_factors = int(input("Enter the number of factors: "))

# Initialize an empty dictionary for the comparison matrix
comparison_matrix = {}

# Function to display consistency message
def display_consistency_message(consistency_ratio):
    if consistency_ratio <= 0.1:
        print("The matrix is consistent.")
    else:
        print("The matrix is inconsistent. Please adjust the comparison matrix.")

# Loop to collect user input for comparisons
# Enter the factor name and also enter the comparison value (Ensure that the value is an integer)
# Add the comparison values to the matrix and print the comparison matrix
for i in range(num_factors):
    for j in range(i + 1, num_factors):
        Factor1 = input(f"Enter the first factor {i + 1}/{num_factors}: ")
        Factor2 = input(f"Enter the second factor {j + 1}/{num_factors}: ")

        # Get user input for comparison value as a string
        comparison_str = input(f"Enter the comparison value for ({Factor1}, {Factor2}): ")

        try:
            # Attempt to convert the string to a fraction
            comparison_value = Fraction(comparison_str)
        except ValueError:
            print("Invalid input. Please enter a valid fraction.")
            continue

        # Add the comparison to the dictionary
        comparison_matrix[(Factor1, Factor2)] = comparison_value
        # Since it's a symmetric matrix, we can also add the reverse comparison
        comparison_matrix[(Factor2, Factor1)] = 1 / comparison_value

# Display the comparison matrix
print("\nComparison Matrix:")
factors = list(set(factor for pair in comparison_matrix.keys() for factor in pair))
```

```
# Print header
print(" " * 6 + " ".join(f"{factor:^10}" for factor in factors))

for i in range(num_factors):
    print(f"{factors[i]:<6}", end=" ")
    for j in range(num_factors):
        if i <= j:
            comparison_value = float(comparison_matrix.get((factors[i], factors[j]), 1))
            print(f"{comparison_value:.2f}", end="\t")
        else:
            comparison_value = float(comparison_matrix.get((factors[j], factors[i]), 1))
            print(f"{1 / comparison_value:.2f}", end="\t")
    print()

# Create an ahpy comparison object
factors_comparisons = {pair: comparison_matrix.get(pair, 1) for pair in [(f1, f2) for f1 in factors
                        for f2 in factors]}
factors_comparison_object = ahpy.Compare(name="Criteria", comparisons=factors_comparisons, precision=
                                         random_index='saaty')

# Perform the AHP analysis to get the weights of the factors and the consistency ratio
target_weights = factors_comparison_object.target_weights
consistency_ratio = factors_comparison_object.consistency_ratio

# Print the results
print("\nAHP Analysis:")
print("Target Weights:", target_weights)
print("Consistency Ratio:", consistency_ratio)

# Display consistency message
display_consistency_message(consistency_ratio)
```

In [ ]:
```
from fractions import Fraction
import ahpy

# Function to display consistency message
def display_consistency_message(consistency_ratio):
    if consistency_ratio <= 0.1:
        print("The matrix is consistent.")
    else:
        print("The matrix is inconsistent. Please adjust the comparison matrix.")

# Enter the number of factors
num_alternatives = int(input("Enter the number of alternatives: "))

# Initialize an empty dictionary for the comparison matrix
comparison_matrix = {}

# Loop to collect user input for comparisons
for i in range(num_alternatives):
    for j in range(i + 1, num_alternatives):
        Alternative1 = input(f"Enter the first alternative {i + 1}/{num_alternatives}: ")
        Alternative2 = input(f"Enter the second alternative {j + 1}/{num_alternatives}: ")

        # Get user input for comparison value as a string
        comparison_str = input(f"Enter the comparison value for ({Alternative1}, {Alternative2}): ")

        try:
            # Attempt to convert the string to a fraction
            comparison_value = Fraction(comparison_str)
        except ValueError:
            print("Invalid input. Please enter a valid fraction.")
            continue

        # Add the comparison to the dictionary
        comparison_matrix[(Alternative1, Alternative2)] = comparison_value
        # Since it's a symmetric matrix, we can also add the reverse comparison
        comparison_matrix[(Alternative2, Alternative1)] = 1 / comparison_value

# Display the comparison matrix
print("\nComparison Matrix:")
alternative = list(set(factor for pair in comparison_matrix.keys() for factor in pair))

# Print header
print(" " * 6 + " ".join(f"{alternative:^10}" for alternative in alternative))
```

```
for i in range(num_alternatives):
    print(f"{alternative[i]:<6}", end=" ")
    for j in range(num_alternatives):
        if i <= j:
            comparison_value = float(comparison_matrix.get((alternative[i], alternative[j]), 1))
            print(f"{comparison_value:.2f}", end="\t")
        else:
            comparison_value = float(comparison_matrix.get((alternative[j], alternative[i]), 1))
            print(f"{1 / comparison_value:.2f}", end="\t")
    print()

# Create an ahpy comparison object
alt_comparisons = {pair: comparison_matrix.get(pair, 1) for pair in [(f1, f2) for f1 in alternative
                                                                      for f2 in alternative]}
alt_comparison_object = ahpy.Compare(name="Criteria", comparisons=alt_comparisons, precision=3,
                                     random_index='saaty')

# Perform the AHP analysis
target_weights = alt_comparison_object.target_weights
consistency_ratio = alt_comparison_object.consistency_ratio

# Print the results
print("\nAHP Analysis:")
print("Target Weights:", target_weights)
print("Consistency Ratio:", consistency_ratio)

# Display consistency message
display_consistency_message(consistency_ratio)
```