COMBINED ALGORITHM TO SELECT PROFILES : MCAP+TOPSIS+HONGROIS

By: Abdel YEZZA, Ph.D
Date: Oct. 2025

Summary

🚀 TOPSIS Profile Selection System

A comprehensive system for profile evaluation and ranking, but not only, combining 3 algorithms :

1. The **Profile-Activity Matching** method
2. The **TOPSIS** (Technique for Order Preference by Similarity to Ideal Solution) algorithm
3. Optionally, the **HUNGARIAN algorithm**

with configurable skill-level thresholds and many other options and features as you will discover below.

Author: Abdel YEZZA, Ph.D, 2025

COMBINED ALGORITHM TO SELECT PROFILES: MCAP, TOPSIS et HONGROIS

By : A. YEZZA, Ph.D

Overview

This project combines two powerful concepts and optionally Hungarian algorithm:

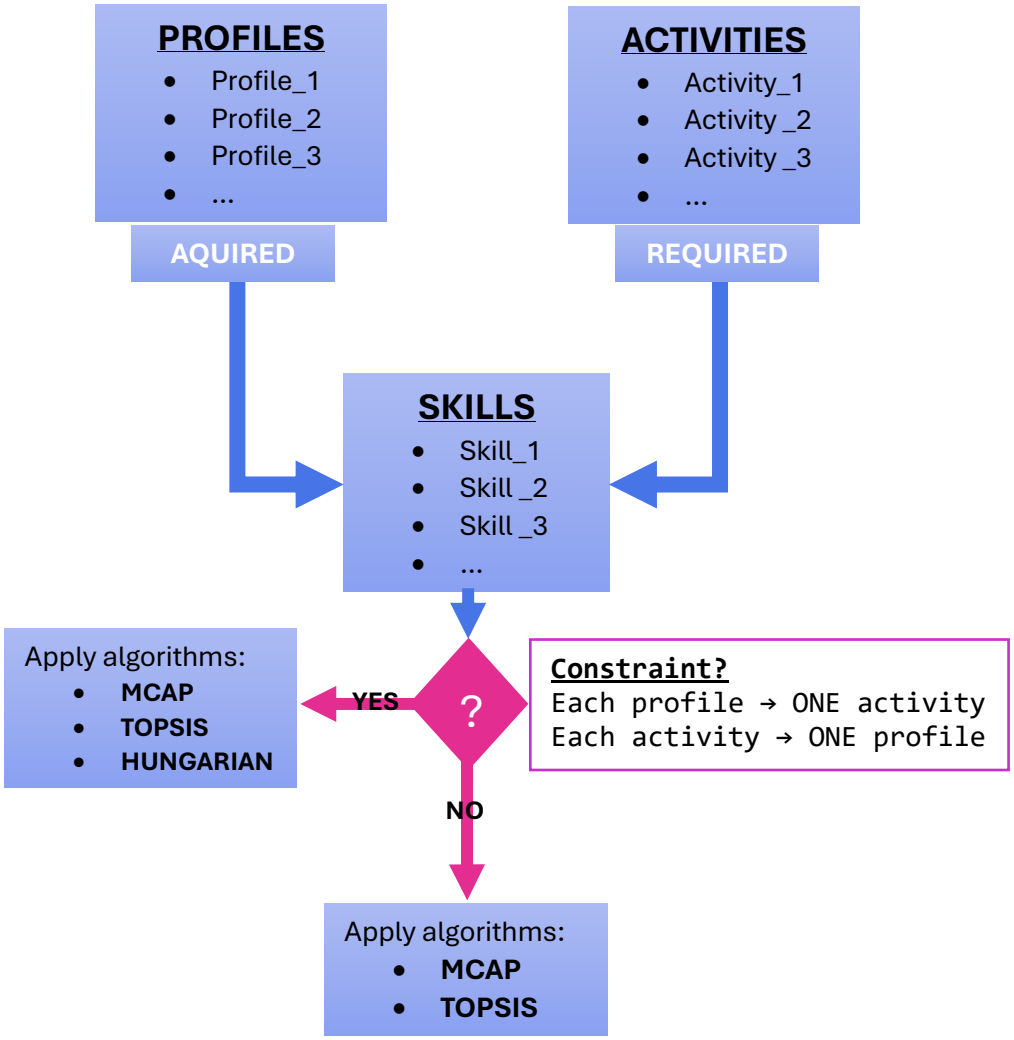
1. **Profile-Activity Matching**: Matching profiles to activities based on competencies - see my article: Profile Assignment System (MCAP) - Abdel YEZZA (Ph.D), 2025
2. **TOPSIS Algorithm** - Multi-criteria decision analysis for ranking - see my article: TOPSIS Algorithm Implementation - Abdel YEZZA (Ph.D), 2025

The system uses a **configurable threshold** to determine how skills are evaluated:

- **Skills >= Threshold:** Treated as **Beneficial** criteria (higher is better)
- **Skills < Threshold**: Treated as **Non-Beneficial** criteria (lower is acceptable)

This approach allows flexible evaluation where some skills are critical (must be maximized) while others are optional (don't penalize lack of proficiency).

General functional schema



COMBINED ALGORITHM TO SELECT PROFILES: MCAP, TOPSIS et HONGROIS

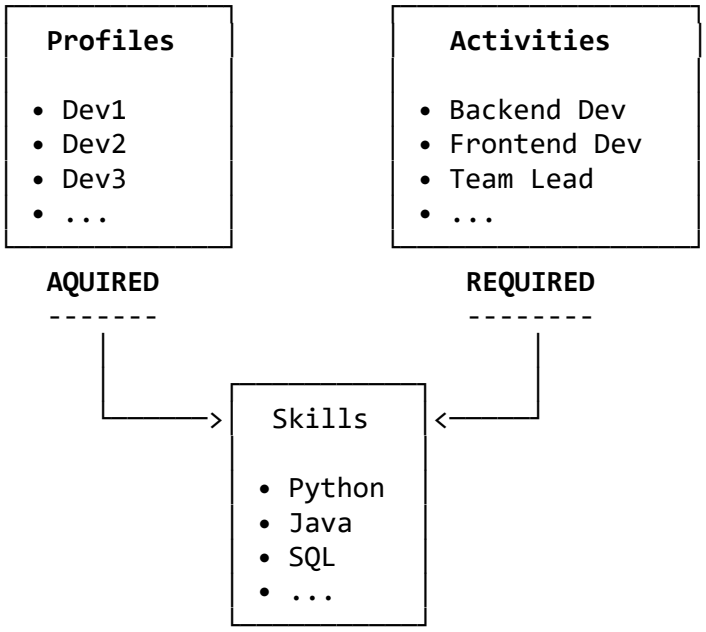
By : A. YEZZA, Ph.D

Let's illustrate this general schema by some scenarios:

Scenario 1: Ranking Profiles for Each Activity

Using: Profile Assignment System (MCAP) + TOPSIS

INPUT:



STEP 1: Profile Assignment System (MCAP)

Skill Transformation Based on Threshold

For each activity requirement:
- If skill level >= Threshold (e.g., 3.0)
  → Mark as BENEFICIAL (higher is better)
- If skill level < Threshold
  → Mark as NON-BENEFICIAL (lower is acceptable)

Example: Backend Dev needs Python=5, Leadership=2
With threshold=3.0:
- Python (5≥3): BENEFICIAL → maximize
- Leadership (2<3): NON-BENEFICIAL → minimize

▼

STEP 2: TOPSIS Algorithm

Multi-Criteria Decision Analysis

1. Normalize profile skills (make comparable)
2. Apply weights (importance of each skill)
3. Find IDEAL profile (best in all skills)
4. Find WORST profile (worst in all skills)
5. Calculate how close each profile is to IDEAL

Result: Proximity Score (0 to 1)

```
  • 1.0 = Perfect match
  • 0.0 = Worst match
```

▼

OUTPUT: Ranked List for Each Activity

```
Backend Development:
1. Dev10 (Score: 0.95) ⭐ Best match
2. Dev7  (Score: 0.87)
3. Dev1  (Score: 0.82)
...

Frontend Development:
1. Dev2  (Score: 0.91) ⭐ Best match
2. Dev5  (Score: 0.85)
...
```

Use Case:

- You have multiple activities and want to know the best candidates for each
- You need to see all qualified profiles ranked by suitability
- Helps in decision-making when you have flexibility in assignments

Scenario 2: Optimal One-to-One Assignment

Using: MCAP + TOPSIS + Hungarian Method

INPUT: Same as Scenario 1

STEPS 1-2: Same as Scenario 1 (MCAP + TOPSIS)

```
Create Compatibility Matrix

          Backend   Frontend   TeamLead
Dev1       0.82       0.91       0.65
Dev2       0.75       0.85       0.70
Dev3       0.90       0.60       0.88

Each cell = TOPSIS proximity score
```

STEP 3: Hungarian Algorithm (Optimal Assignment)

```
Find the BEST overall assignment

Constraint: Each profile → ONE activity
            Each activity → ONE profile

Goal: Maximize total satisfaction

Algorithm finds optimal pairing considering
all possibilities simultaneously
```

OUTPUT: Optimal Assignments

```
Optimal Assignments:

Dev1  →  Frontend Development   (0.91)
Dev2  →  Team Lead              (0.70)
Dev3  →  Backend Development    (0.90)

Total Score: 2.51 (best possible combination)
```

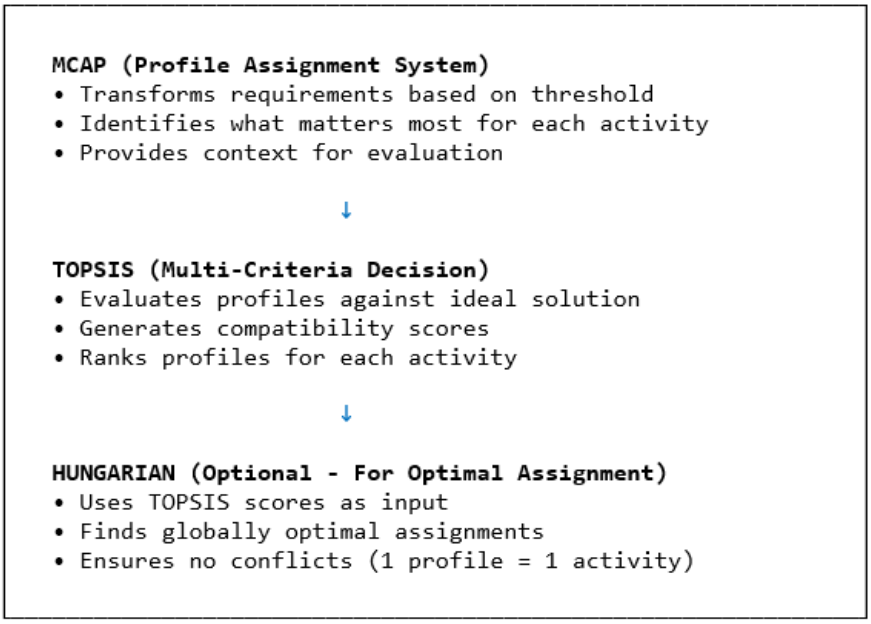
Use Case:

- You need to assign exactly ONE profile to each activity
- No profile can be assigned to multiple activities
- You want the globally optimal solution, not just individual bests
- Example: Assigning team members to project roles

Key Differences Between Scenarios

Aspect	Scenario 1 (Ranking)	Scenario 2 (Optimal Assignment)
Output	Ranked list per activity	One-to-one assignments
Flexibility	Multiple candidates per activity	One profile per activity
Optimization	Individual activity level	Global level
Algorithms	MCAP + TOPSIS	MCAP + TOPSIS + Hungarian
When to Use	Exploring felexibiliy options	Final assignments

How the Algorithms Complement Each Other



Features

- **Configurable Threshold System**: (*min_threshold* to *max_threshold* with *threshold* values) - see **config.json file** at **threshold_settings** key
- **Multiple Weight Strategies**: (uniform, requirement-based, custom)
- **Two TOPSIS Proximity Formulas**: (*standard* and *variant*) - the *standard* case uses the TOPSIS standard formulas, whereas *variant* uses a different function already normalized and more disparate than the standard
- **Comprehensive Visualizations**: (*heatmaps*, *bar charts*, *radar* plots and *distance analysis*)
- The **heatmap** graph gives clear and comprehensive ranking and proximity values allowing to choose among the top-ranked profiles.
- **Flexible Input Formats**: (CSV files with profiles and activities)
- **Command-Line Interface**: with simple and extensive options
- **Detailed Results Export**: (rankings, matrices, analysis reports)

🚀 Installation

Prerequisites

- Python 3.8 or higher
- pip package manager

Install Dependencies

```
cd .\topsis_profiles_selection
pip install -r requirements.txt
```

Quick Start

1. Basic Usage (Default Configuration)

```
python main.py
```

This will:

1. Load profiles and activities from data/input/
2. Use threshold = 3.0 (configurable between 0-5) by default or custom values
3. Apply uniform weights to all skills by default
4. Generate rankings for all activities
5. Save results to data/output/

2. Custom Threshold

```
python main.py --threshold 3.5
```

3. Single Activity Processing

```
python main.py --activity "Backend_Development"
```

4. Verbose Mode with Visualizations

```
python main.py -v --viz
```

5. Custom Weight Strategy

```
python main.py --weight-strategy requirement_based
```

6. Custom Input Files

```
python main.py --profiles data/input/profiles.csv --activities data/input/activities.csv
```

Configuration

This option allows to customize all input parameters in one JSON configuration file. You don't have to remember all command-line options.

Edit **config.json** to customize:

```
{
  "project_name": "TOPSIS Profile Selection System",
  "description": "Profile selection using TOPSIS algorithm with configurable skill threshold",
  "version": "1.0.0",
  "author": "Abdel YEZZA (Ph.D)",

  "data": {
    "profiles_file": "data/input/profiles.csv",
    "activities_file": "data/input/activities.csv",
    "output_dir": "data/output"
  },

  "topsis_settings": {
    "proximity_formula_options": ["standard", "variant"],
    "proximity_formula": "variant",
    "proximity_formula_description": {
      "standard": "S*[i] = E-[i] / (E+[i] + E-[i]) - Values between 0 and 1",
      "variant": "S*[i] = E-[i] / E+[i] - Better discrimination, normalized to [0,1]"
    }
  },

  "threshold_settings": {
    "threshold": 3.0,
    "min_threshold": 0.0,
    "max_threshold": 5.0,
    "description": "Skill levels >= threshold are treated as beneficial (maximize), < threshold as non-beneficial (minimize)"
  },

  "weight_settings": {
    "strategy": "uniform",
    "strategy_options": ["uniform", "requirement_based", "custom"],
    "strategy_descriptions": {
      "uniform": "All skills have equal weight",
      "requirement_based": "Weights proportional to required skill levels",
      "custom": "User-defined weights for each skill"
    },
    "custom_weights": null
  },

  "output_settings": {
    "top_n_profiles": 3,
    "save_detailed_results": true,
    "generate_visualizations": true,
    "verbose": false
  },

  "visualization_settings": {
    "chart_types": ["bar", "radar", "heatmap"],
    "figure_format": "png",
    "figure_dpi": 300
  },

  "assignment_settings": {
    "enable_optimal_assignment": false,
    "assignment_method": "auto",
    "assignment_method_options": ["auto", "hungarian", "greedy"],
    "assignment_method_descriptions": {
      "auto": "Automatically select Hungarian (if dimensions match) or Greedy approach",
```

```
      "hungarian": "Use Hungarian Algorithm for optimal 1-to-1 assignment (requires equal number of
profiles and activities)",
      "greedy": "Use Greedy approach for best-effort assignment (works with any dimensions)"
    },
    "generate_assignment_heatmap": true
  }
}
```

The assignment_method: Options are "**auto**", "**hungarian**", or "**greedy**"

- **auto**: Automatically selects **Hungarian** if dimensions match, otherwise uses **Greedy.** This option should be the default assignment method in general.
- **hungarian**: Forces Hungarian Algorithm (requires equal number of profiles and activities). This algorithm is **O(n^3) complexity order.**
- **greedy**: Uses greedy best-effort assignment (works with any dimensions). This is a fast algorithm using local optimisation by iterating over activities to assign all of them. It is fast, **O($n^2 \log n$) complexity order.**

The table below summarizes the 3 options:

Feature	Auto	Greedy algorithm	Hungarian algorithm
Optimality	Depends	Sub-optimal	**Globally optimal**
Dimension Requirement	Any	Any	**Must be equal ($n \times n$)**
Complexity	Depends	O($n^2 \log n$)	O(n^3)
Use Case	Default	Unequal dimensions	Equal dimensions
Total Score	Varies	May be lower	**Maximum possible**

Input Data Format

Make sure that input csv values are in the intervall [min_threshold, max_threshold].

Profiles CSV (data/input/profiles.csv)

```
Profile,Python,Java,SQL,Communication,Leadership
Dev1,5,3,4,4,2
Dev2,4,5,3,5,4
Dev3,3,2,5,3,3
```

- **First column**: Profile names/IDs
- **Other columns**: Skill levels (0-5 scale or any other scale)

Activities CSV (data/input/activities.csv)

```
Activity,Python,Java,SQL,Communication,Leadership
Backend_Development,5,4,5,3,2
Frontend_Development,3,5,2,4,2
Team_Lead,3,3,3,5,5
```

- **First column**: Activity names
- **Other columns**: Required skill levels (same skills as profiles)
- **Important**: Skill columns must match between profiles and activities

How It Works

1. Skill Transformation

For each activity, required skill levels are analyzed:

Example:

- Backend Development requires Python=5, Leadership=2
- Python (5 >= 3): Beneficial criterion → Higher Python skills preferred
- Leadership (2 < 3): Non-beneficial criterion → Lower Leadership acceptable

2. TOPSIS Algorithm

The system applies TOPSIS in 5 steps:

1. **Normalize** the decision matrix (profiles × skills)
2. **Apply weights** to normalized values
3. **Determine ideal solutions** (best and worst)
4. **Calculate distances** from each profile to ideal solutions
5. **Calculate proximity coefficients** (higher = better match)

3. Ranking

Profiles are ranked by proximity coefficient (0 to 1): - **1.0** = Perfect match with ideal solution - **0.0** = Closest to worst solution

NOTE: if using variant proximity formula, the perfect match is always 1, which facilitates detecting the best choice.

Sample Output

Case: `standard proximity formula`

```
================================================================================
Best Profile for Each Activity
================================================================================
Activity                       Best Profile                   Coefficient
--------------------------------------------------------------------------------
Backend_Development            Profile_10                     0.750578
Frontend_Development           Profile_1                      0.672916
Team_Lead                      Profile_14                     0.659264
Data_Engineer                  Profile_1                      0.703770
DevOps_Engineer                Profile_4                      0.718717
...
================================================================================

================================================================================
RANKING MATRIX - Top 3 Profiles per Activity
================================================================================

            Activity              Rank 1              Rank 2              Rank 3
 Backend_Development Profile_10 (0.7506)  Profile_7 (0.7153)  Profile_1 (0.6702)
Frontend_Development  Profile_1 (0.6729)  Profile_2 (0.6305)  Profile_7 (0.6281)
           Team_Lead Profile_14 (0.6593)  Profile_9 (0.6236) Profile_12 (0.6116)
       Data_Engineer  Profile_1 (0.7038) Profile_10 (0.6652)  Profile_7 (0.6460)
     DevOps_Engineer  Profile_4 (0.7187) Profile_10 (0.6542) Profile_14 (0.6458)
...
```

COMBINED ALGORITHM TO SELECT PROFILES: MCAP, TOPSIS et HONGROIS

By : A. YEZZA, Ph.D

Case: variant proximity formula

```
================================================================================
Best Profile for Each Activity
================================================================================
Activity                       Best Profile                   Coefficient
--------------------------------------------------------------------------------
Backend_Development            Profile_10                     1.000000
Frontend_Development           Profile_1                      1.000000
Team_Lead                      Profile_14                     1.000000
Data_Engineer                  Profile_1                      1.000000
DevOps_Engineer                Profile_4                      1.000000
...
================================================================================

================================================================================
RANKING MATRIX - Top 3 Profiles per Activity
================================================================================

            Activity              Rank 1              Rank 2              Rank 3
 Backend_Development Profile_10 (1.0000)  Profile_7 (0.8348)  Profile_1 (0.6754)
Frontend_Development  Profile_1 (1.0000)  Profile_2 (0.8295)  Profile_7 (0.8207)
           Team_Lead Profile_14 (1.0000)  Profile_9 (0.8564) Profile_12 (0.8140)
       Data_Engineer  Profile_1 (1.0000) Profile_10 (0.8361)  Profile_7 (0.7681)
     DevOps_Engineer  Profile_4 (1.0000) Profile_10 (0.7404) Profile_14 (0.7137)
...
```

4. Command-Line Options

All options are optional. The **config.json** parameters are loaded first and replaced if any parameter is passed on the command-line.

```
usage: main.py [-h] [-c CONFIG] [--profiles PROFILES] [--activities ACTIVITIES]
               [--threshold THRESHOLD] [--min-threshold MIN_THRESHOLD]
               [--max-threshold MAX_THRESHOLD] [--activity ACTIVITY]
               [--weight-strategy {uniform,requirement_based}]
               [--proximity-formula {standard,variant}]
               [-v] [--viz] [-o OUTPUT]
               [--assignment] [--assignment-method {auto,hungarian,greedy}]


Options:
  -h, --help            Show help message
  -c, --config          Configuration file path
  --profiles            Profiles CSV file path
  --activities          Activities CSV file path
  --threshold           Skill level threshold (default: 3.0)
  --min-threshold       Minimum skill level (default: 0.0)
  --max-threshold       Maximum skill level (default: 5.0)
  --activity            Process only specific activity
  --weight-strategy     Weight generation strategy
  --proximity-formula   TOPSIS proximity formula
  -v, --verbose         Enable verbose output
  --viz, --visualize    Generate visualizations
  -o, --output          Output directory
  --assignment          Enable optimal assignment (Hungarian/Greedy)
  --assignment-method   {auto,hungarian,greedy}
                        Assignment method (overrides config)
   Examples:
     # Use default configuration, the most simple
     python main.py

     # Use custom configuration
     python main.py -c my_config.json

     # Override threshold
     python main.py --threshold 3.5

     # Process single activity
     python main.py --activity "Backend_Development"

     # Use custom input files
     python main.py --profiles data/my_profiles.csv --activities data/my_activities.csv

     # Verbose mode with visualizations
     python main.py -v --viz

     # Use different weight strategy
     python main.py --weight-strategy requirement_based

     # Enable optimal assignment with Hungarian/Greedy algorithm
     python main.py --assignment

     # Force Hungarian algorithm (requires activities/profiles CSV equal dimensions)
     python main.py --assignment --assignment-method hungarian

     # Use greedy assignment approach
     python main.py --assignment --assignment-method greedy
```

5. Weight Strategies

1. Uniform Weights (Default)

All skills have equal importance as for example:

```
weights = [0.1, 0.1, 0.1, ..., 0.1]
```

2. Requirement-Based Weights

Skills with higher required levels get higher weights:

`weights proportional to required_levels`

3. Custom Weights

Define specific weights in **config.json**:

```
{
  "weight_settings": {
    "strategy": "custom",
    "custom_weights": [0.3, 0.2, 0.2, 0.1, 0.1, 0.1]
  }
}
```

In this example, weights are all normalized since their sum is 1.

6. TOPSIS Formulas

Standard Formula (Default)

$$S^* = \frac{E^-}{(E^+ + E^-)}$$

- Values between 0 and 1
- Easier to interpret
- Smaller differences between alternatives

Variant Formula

$$S^* = \begin{cases} \frac{E^-}{E^+} \; for \; E^+ \neq 0 \\ \frac{E^-}{\max_i (E_i^+)} \; for \; E^+ = 0 \end{cases}$$

Where:

- **E-** = Distance to worst ideal solution
- **E+** = Distance to best ideal solution
- **max(E+)** = Maximum distance to best across all alternatives

Characteristics:

- Better discrimination between alternatives
- Handles edge cases when distance to best is zero
- No normalization required

- Higher values indicate better match
- Ranking results are the same as in the case of the standard formula

7. Usage Examples

Example 1: Find Best Backend Developer

```
python main.py --activity "Backend_Development" -v
```

This will show detailed analysis of which profiles best match the backend development requirements.

Example 2: Adjust Threshold for Different Standards

```
# Strict evaluation (threshold = 4.0)
python main.py --threshold 4.0

# Lenient evaluation (threshold = 2.5)
python main.py --threshold 2.5
```

Higher thresholds mean more skills are treated as “must have” (beneficial).

Example 3: Generate Complete Report with Visualizations

```
python main.py -v --viz --output reports/analysis_2025
```

API Usage (Python)

```
from pathlib import Path
import sys
sys.path.insert(0, 'src')

from core.profile_processor import ProfileProcessor, load_profiles_from_csv, load_activities_from_csv

# Load data
profiles_df = load_profiles_from_csv('data/input/profiles.csv')
activities_df = load_activities_from_csv('data/input/activities.csv')

# Create processor
processor = ProfileProcessor(
    profiles_df=profiles_df,
    activities_df=activities_df,
    threshold=3.0,
    min_threshold=0.0,
    max_threshold=5.0,
    proximity_formula='standard'
)

# Process all activities
results = processor.process_all_activities(
    weight_strategy='uniform',
    verbose=True
)

# Get specific activity results
backend_results = processor.results['Backend_Development']
best_profile = backend_results['best_alternative']
print(f"Best profile for Backend Development: {best_profile}")

# Save results
processor.save_results(Path('data/output'))
```

Real examples

Example 1: standard TOPSIS

Activities.csv

Profile	Python	Java	SQL	Communication	Leadership	Problem_Solving	Teamwork	Project_Management	Data_Analysis	Cloud_Computing
Profile_1	5	3	4	4	2	5	4	2	3	4
Profile_2	4	5	3	5	4	4	5	3	2	3
Profile_3	3	2	5	3	3	4	3	2	5	2
Profile_4	5	4	5	4	4	5	4	4	4	5
Profile_5	2	3	2	5	5	3	5	5	2	1
Profile_6	4	4	4	3	3	4	4	3	4	4
Profile_7	5	5	4	2	2	5	3	1	5	5
Profile_8	3	4	5	4	3	4	4	3	4	3
Profile_9	4	3	3	5	5	3	5	4	3	2
Profile_10	5	5	5	3	2	5	3	2	5	5
Profile_11	2	2	3	4	4	3	4	4	3	2
Profile_12	3	4	4	5	4	4	5	4	3	3
Profile_13	4	3	5	3	3	5	3	3	5	4
Profile_14	5	4	4	4	5	4	4	5	4	4
Profile_15	3	5	3	4	3	3	4	3	4	5

Profiles.csv

Activity	Python	Java	SQL	Communication	Leadership	Problem_Solving	Teamwork	Project_Management	Data_Analysis	Cloud_Computing
Backend_Development	5	4	5	3	2	5	3	2	4	5
Frontend_Development	3	5	2	4	2	4	4	2	2	3
Team_Lead	3	3	3	5	5	4	5	5	3	2
Data_Engineer	5	2	5	3	2	5	3	2	5	4
DevOps_Engineer	4	3	3	3	2	4	3	3	3	5
Project_Manager	2	2	2	5	5	4	5	5	2	2
Full_Stack_Developer	4	4	4	4	3	4	4	3	3	4
Database_Admin	3	2	5	3	2	4	3	2	4	3
Cloud_Architect	4	3	3	3	2	5	3	3	4	5
Tech_Lead	4	4	4	5	5	5	5	5	4	4

Output:

```
Configuration Summary:
--------------------------------------------------------------------------------
  Profiles File: data/input/profiles.csv
  Activities File: data/input/activities.csv
  Output Directory: data/output

Threshold Settings:
  Threshold: 3.0
  Range: [0.0, 5.0]
  Description: Skill levels >= threshold are treated as beneficial (maximize), < threshold as non-beneficial (minimize)

TOPSIS Settings:
  Proximity Formula: standard

Weight Strategy:
  Strategy: uniform
  Description: All skills have equal weight
--------------------------------------------------------------------------------

Loading data...
  Loaded 15 profiles
  Loaded 10 activities
  Skills: 10
```

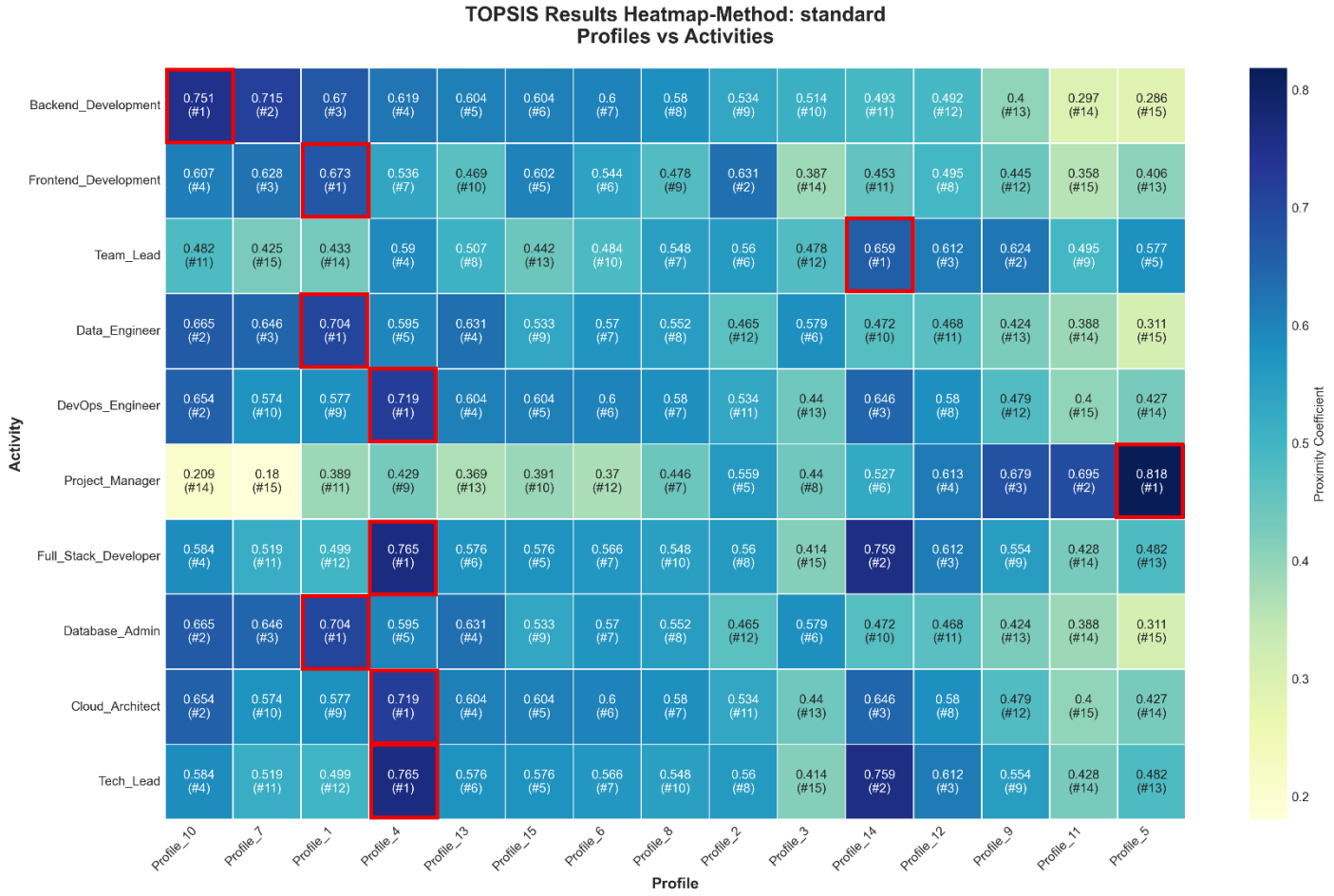
...

COMBINED ALGORITHM TO SELECT PROFILES: MCAP, TOPSIS et HONGROIS

By : A. YEZZA, Ph.D

```
================================================================================
Best Profile for Each Activity
================================================================================
Activity                       Best Profile                   Coefficient
--------------------------------------------------------------------------------
Backend_Development            Profile_10                     0.750578
Frontend_Development           Profile_1                      0.672916
Team_Lead                      Profile_14                     0.659264
Data_Engineer                  Profile_1                      0.703770
DevOps_Engineer                Profile_4                      0.718717
Project_Manager                Profile_5                      0.818082
Full_Stack_Developer           Profile_4                      0.765258
Database_Admin                 Profile_1                      0.703770
Cloud_Architect                Profile_4                      0.718717
Tech_Lead                      Profile_4                      0.765258
================================================================================

================================================================================
PROCESS COMPLETED SUCCESSFULLY
================================================================================
```

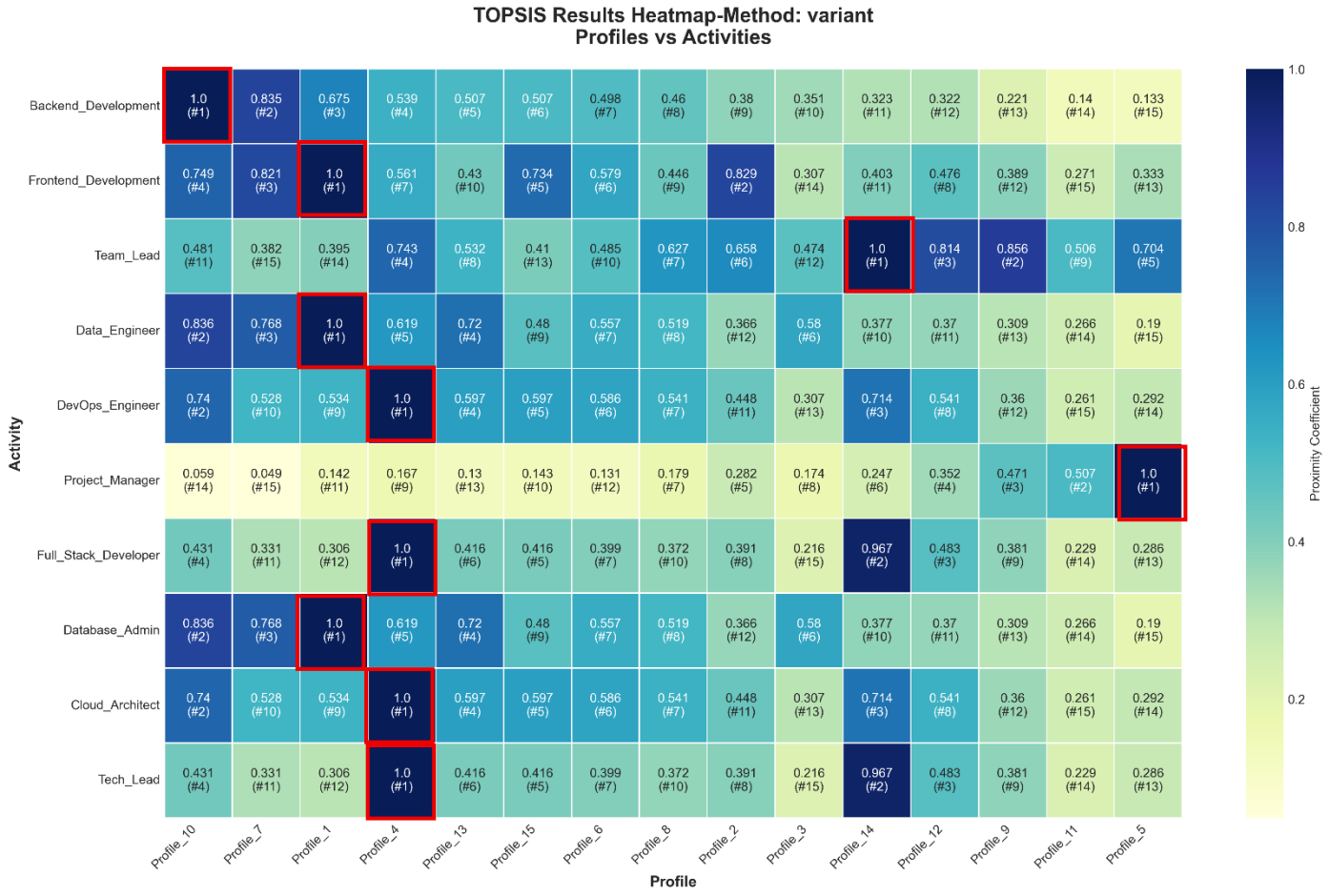


By changing the `proximity_formula='standard'` to `proximity_formula='variant'` we obtain the same **assignment matrix easier to identify optimal assignments by spotting the 1's**:

Example 2: variant TOPSIS

The same as **example 1** by changing the ***proximity_formula*** to ***variant***:

```
  "topsis_settings": {
      "proximity_formula_options": ["standard", "variant"],
      "proximity_formula": "variant",
      …
      }
    },
…
```



In both examples 1, 2 a profile may be assigned to more than one activity even if profiles count is greater than activities count. Nevertheless, This can give recruiters and employers more leeway in choosing from the full range of profiles, not just the best ones!

Example 3: HUNGARIAN assignment constraint (1-to-1 assignement) & proximity_formula = variant

In this case, the number of activities must be the same as the number of profiles (=10):

Activities_2.csv

Profile	Python	Java	SQL	Communication	Leadership	Problem_Solving	Teamwork	Project_Management	Data_Analysis	Cloud_Computing
Profile_1	5	4	5	3	2	5	3	2	4	5
Profile_2	3	5	2	4	2	4	4	2	2	3
Profile_3	3	3	3	5	5	4	5	5	3	2
Profile_4	5	2	5	3	2	5	3	2	5	4
Profile_5	4	3	3	3	2	4	3	3	3	5
Profile_6	2	2	2	5	5	4	5	5	2	2
Profile_7	4	4	4	4	3	4	4	3	3	4
Profile_8	3	2	5	3	2	4	3	2	4	3
Profile_9	4	3	3	3	2	5	3	3	4	5
Profile_10	4	4	4	5	5	5	5	5	4	4

Profiles_2.csv

Activity	Python	Java	SQL	Communication	Leadership	Problem_Solving	Teamwork	Project_Management	Data_Analysis	Cloud_Computing
Backend_Development	5	3	4	4	2	5	4	2	3	4
Frontend_Development	4	5	3	5	4	4	5	3	2	3
Team_Lead	3	2	5	3	3	4	3	2	5	2
Data_Engineer	5	4	5	4	4	5	4	4	4	5
DevOps_Engineer	2	3	2	5	5	3	5	5	2	1
Project_Manager	4	4	4	3	3	4	4	3	4	4
Full_Stack_Developer	5	5	4	2	2	5	3	1	5	5
Database_Admin	3	4	5	4	3	4	4	3	4	3
Cloud_Architect	4	3	3	5	5	3	5	4	3	2
Tech_Lead	5	5	5	3	2	5	3	2	5	5

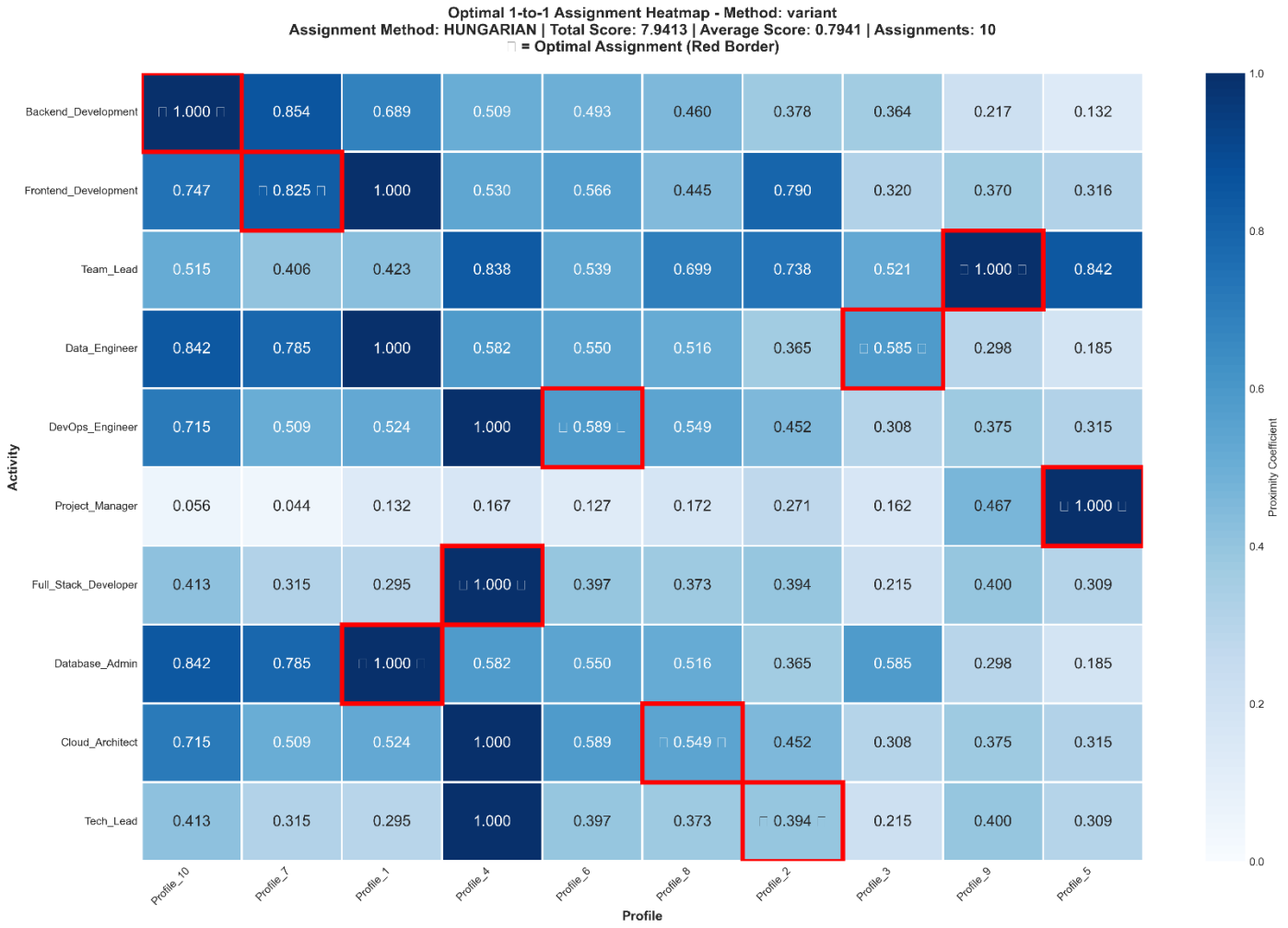
Command-line:

```
python main.py --assignment --assignment-method hungarian
```

We obtain the following assignment represented by a heatmap graph:

COMBINED ALGORITHM TO SELECT PROFILES: MCAP, TOPSIS et HONGROIS

By : A. YEZZA, Ph.D



We note that even a profile has 1 as a proximity factor, he/she is not automatically assigned to the corresponding activity as for example: **Profile_7** is affected to **Frontend_Development** activity with a proximity factor of .825<1.0 even if **profile_1** has 1 as proximity factor for the same activity.

NOTE: The HUNGARIAN assignment matrix is not necessarily the same in the ***standard proximity_formula*** as in the ***variant*** case. Applying **standard** and **variant *proximity_formulas*** gives the following conclusions:

1. **Different Score Ranges:**
    - **Standard**: [0.173, 0.825] - more compressed range
    - **Variant**: [0.044, 1.000] - wider range with better discrimination
2. **Different Total Optimization Scores:**
    - **Standard**: 6.633 (**average**: 0.663)
    - **Variant**: 7.941 (**average**: 0.794)
    - **Difference**: 1.309 points!

References

TOPSIS Algorithm

- Hwang, C.L., & Yoon, K. (1981). Multiple Attribute Decision Making: Methods and Applications. Springer-Verlag.
- Yoon, K.P., & Hwang, C.L. (1995). Multiple Attribute Decision Making: An Introduction. SAGE Publications.

Multi-Criteria Decision Analysis

- Triantaphyllou, E. (2000). Multi-criteria Decision Making Methods: A Comparative Study. Kluwer Academic Publishers.

License

Contributing

For issues, suggestions, or contributions, please contact the author.

Author

Abdel YEZZA, Ph.D

Version: 1.0.0 **Last Updated:** 2025