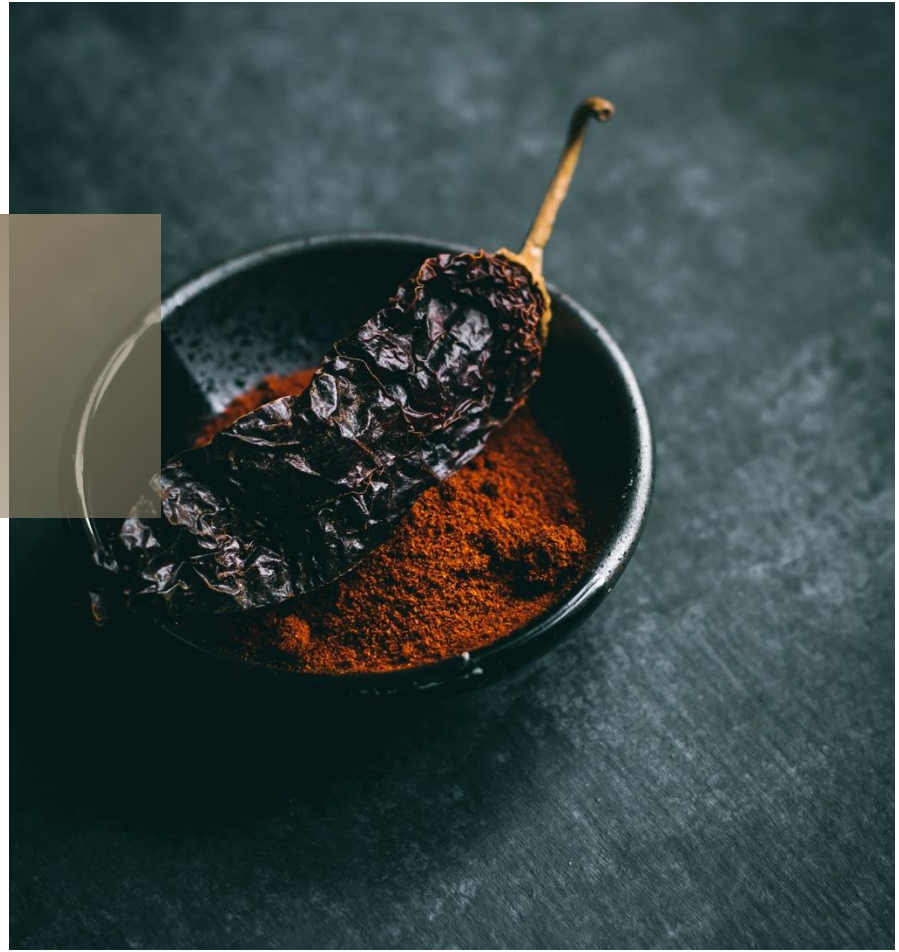
INTRODUCTION TO MULTICRITERIA DECISION ANALYSIS (MCDA)

06 – PAPRIKA

Fellipe Martins

TABLE OF CONTENTS



PAPRIKA

The young one in the class

PAPRIKA

Today’s class notes are based on Dr. Paul Hansen’s (University of Otago, New Zealand) class notes on MCDA and PAPRIKA that he graciously made available for today.

What does PAPRIKA stand for?

PAPRIKA method (short for (Potentially All Pairwise RanKings of all possible Alternatives) was developed in the 2009 by Hansen and Ombler and is one of the most recent multicriteria decision-making methods**.**

What each word means:

• **Potentially** – The method doesn't ask you to rank all possible combinations, but it could, if needed. It finds the *minimum number of comparisons* necessary to figure out your preferences.

• **All Pairwise Rankings** – You're comparing *pairs of alternatives* (two at a time), and saying which one you prefer (or if you're indifferent).

• **of All Possible Alternatives** – It considers *all logically possible combinations of the criteria and their levels* (e.g., all the different ways something can score "high" on one criterion and "low" on another).

Key concepts

- You **don’t have to assign numbers yourself**.

- The **system figures out point values** just based on your ordinal preferences (this one is better than that one).

- It **avoids overwhelming the user** by using transitivity to reduce how many comparisons are needed.

Inference

Let's imagine we need to consider **8 criteria** (a, …, h) and each has **4 levels** (1, .. 4, i.e., very low to very high).

4^8 = 65,536 possible alternatives (on a.b.c.d.e.f.g.h criteria)

4.4.4.4.4.4.4.4 a4 + b4 + c4 + d4 + e4 + f4 + g4 + h4	**3.4.4.4.4.4.4.4** a3 + b4 + c4 + d4 + e4 + f4 + g4 + h4	**4.3.4.4.4.4.4.4** a4 + b3 + c4 + d4 + e4 + f4 + g4 + h4	**2.2.2.2.2.2.2.2** a2 + b2 + c2 + d2 + e2 + f2 + g2 + h2
3.3.3.3.3.3.3.3 a3 + b3 + c3 + d3 + e3 + f3 + g3 + h3	**4.3.2.1.4.3.2.1** a4 + b3 + c2 + d1 + e4 + f3 + g2 + h1	**4.1.1.1.1.1.1.1** a4 + b1 + c1 + d1 + e1 + f1 + g1 + h1	**1.1.1.1.1.1.1.1** a1 + b1 + c1 + d1 + e1 + f1 + g1 + h1

Now we need to find the **32 point values / variables** (a4, a3, a2, a1 … h4, h3, h2, h1, where a4>a3>a2>a1, etc.) that produce the decision-maker's preferred ranking of the 65,536 alternatives.

Inference

Do this by having the decision-maker pairwise rank the 65,536 alternatives, but so that she has to make the minimum number of pairwise rankings (i.e. < **2,147,450,880 possible**!)

Of these 2,147,450,880 pairs, **~100 million are intrinsically ranked via 'domination' / monotonicity** …

- e.g. 4.4.4.4.4.4.4.4 > 3.3.3.3.3.3.3.3; 4.3.4.2.2.1.2.4 > 3.3.3.1.2.1.2.3

Still, 2,047,516,416 'undominated' pairs to be ranked (subjective judgements are required)

- e.g. 4.3.2.1.4.3.2.1 vs 1.2.3.4.1.2.3.4; 4.3.2.1.4.3.2.1 vs 1.2.3.4.1.2.3.4

Of these 2,047,516,416 pairs, 1,645,415,856 are **replicas once 'cancellations' performed**:

- 4.3.1.1.1.1.1.1 vs 3.4.1.1.1.1.1.1;
- 4.3.2.2.2.2.2.2 vs 3.4.2.2.2.2.2.2
- 4.3.1.2.3.4.1.2 vs 3.4.1.2.3.4.1.2
- 4.3._._._._._._ vs 3.4._._._._._._

Inference

Do this by having the decision-maker pairwise rank the 65,536 alternatives, but so that she has to make the minimum number of pairwise rankings (i.e. < **2,147,450,880 possible**!)

Of these 2,147,450,880 pairs, ~**100 million are intrinsically ranked via 'domination' / monotonicity** …

- e.g. 4.4.4.4.4.4.4.4 > 3.3.3.3.3.3.3.3; 4.3.4.2.2.1.2.4 > 3.3.3.1.2.1.2.3

Still, 2,047,516,416 'undominated' pairs to be ranked (subjective judgements are required)

- e.g. 4.3.2.1.4.3.2.1 vs 1.2.3.4.1.2.3.4; 4.3.2.1.4.3.2.1 vs 1.2.3.4.1.2.3.4

Of these 2,047,516,416 pairs, 1,645,415,856 are **replicas once 'cancellations' performed**:

- 4.3.1.1.1.1.1.1 vs 3.4.1.1.1.1.1.1;
- 4.3.2.2.2.2.2.2 vs 3.4.2.2.2.2.2.2
- 4.3.1.2.3.4.1.2 vs 3.4.1.2.3.4.1.2
- 4.3._._._._._._ vs 3.4._._._._._._

402,100,560 unique 'undominated' pairs to be ranked by decision-maker

Inference

Do this by having the decision-maker pairwise rank the 65,536 alternatives,
PAPRIKA method achieves this in **~200 pairwise decisions (~95 is sufficient – i.e. accurate - for most real-world applications**), by exploiting, on a massive scale, the property of 'transitivity':

- e.g. Alternative 1 > 2 and 2 > 3 $\Rightarrow$ 1 > 3

2 efficient processes for:

- Generating unique undominated pairs

Model	Alternatives	All possible pairwise rankings	Unique undominated pairs (to be pairwise ranked)	**Pairwise rankings necessary**
8 criteria, **4** levels each	4^8 = 65,536	2,147,450,880	402,100,560	**~95**
10 criteria, **4** levels each	4^{10} = 1,048,576	549,755,289,600	68,646,770,676	**~160**
12 criteria, **5** levels each	5^{12} = 244,140,625	29,802,322,265,625,000	3,674,775,327,316,600	**~900**
20 criteria, **4** levels each	4^{20} = 1,099,511,627,776	604,462,909,806,765,000,000,000	9,502,402,095,174,090,000,000	**~1,200**

Inference

You’re choosing between laptops based on:
- Battery Life (Low/High)
- Weight (Heavy/Light)
- Price (Expensive/Cheap)

There are 2×2×2 = 8 possible laptops.

Instead of comparing all 28 possible pairs, PAPRIKA:
- Chooses a **smart subset** of comparisons to ask you.
- **Infers the rest using logic**.
- **Builds a point-based scoring model behind the scenes**.

Problems?

PAPRIKA – Step 1

We need to define criteria and levels (here is different from other methods).

Criterion	Levels
Battery Life	Low (1), High (2)
Weight	Heavy (1), Light (2)
Price	Expensive (1), Cheap (2)

You can (but shouldn't) create many levels for each criterion (3, 4, …, n). Why?

So, each alternative is a **triple** $(battery, weight, price)$ and each position is either 1 or 2 (low or high performance).

Total combinations: 2 × 2 × 2 = 8 laptops.
Notice these 8 laptops do not exist, we are talking about 8 theoretical laptops (states) that reflect the combination of all criteria and levels.

PAPRIKA – Step 1

→ 6 point values / variables: a2, a1, b2, b1, c2, c1 (a2 > a1, b2 > b1, c2 > c1)
→ 8 (2^3) possible alternatives (on a.b.c criteria):

2.2.2 a2 + b2 + c2	**2.2.1** a2 + b2 + c1	**2.1.2** a2 + b1 + c2	**1.2.2** a1 + b2 + c2
1.1.2 a1 + b1 + c2	**1.2.1** a1 + b2 + c1	**2.1.1** a2 + b1 + c1	**1.1.1** a1 + b1 + c1

Objective: define the 6 points variables such that they produce the preferred ranking of the 8 alternatives, by pairwise ranking the alternatives but so that the decision-maker has to make the smallest number of decisions possible.

PAPRIKA – Step 1

We need to define criteria and levels (here it is different from other methods).

We define a value (score) for each level of each criterion:

Let (higher = better):
- $v_{bat}2$ = value of High battery
- $v_{weight}2$ = value of Light weight
- $v_{price}2$ = value of Cheap price

We **normalize** the lowest levels to 0:
- $v_{bat}1 = v_{weight}1 = v_{price}1 = 0$

So, we only have to find:
- $v_{bat}2 = v_{weight}2 = v_{price}2 = 0$

PAPRIKA – Step 2

Now we compose a table with alternatives / criteria:

ID	Battery	Weight	Price	Vector	Score Formula
A1	High	Light	Cheap	(2,2,2)	$v_{bat}2 + v_{weight}2 + v_{price}2$
A2	High	Light	Expensive	(2,2,1)	$v_{bat}2 + v_{weight}2$
A3	High	Heavy	Cheap	(2,1,2)	$v_{bat}2 + v_{price}2$
A4	Low	Light	Cheap	(1,2,2)	$v_{weight}2 + v_{price}2$
A5	High	Heavy	Expensive	(2,1,1)	$v_{bat}2$
A6	Low	Light	Expensive	(1,2,1)	$v_{weight}2$
A7	Low	Heavy	Cheap	(1,1,2)	$v_{price}2$
A8	Low	Heavy	Expensive	(1,1,1)	0

PAPRIKA – Step 2

versus (vs)	2.2.2 a2 + b2 + c2	2.2.1 a2 + b2 + c1	2.1.2 a2 + b1 + c2	1.2.2 a1 + b2 + c2	1.1.2 a1 + b1 + c2	1.2.1 a1 + b2 + c1	2.1.1 a2 + b1 + c1	1.1.1 a1 + b1 + c1
2.2.2 a2 + b2 + c2		>	>	>	>	>	>	>
2.2.1 a2 + b2 + c1			a2 + b2 + c1 vs a2 + b1 + c2	a2 + b2 + c1 vs a1 + b2 + c2	a2 + b2 + c1 vs a1 + b1 + c2	>	>	>
2.1.2 a2 + b1 + c2				a2 + b1 + c2 vs a1 + b2 + c2	>	a2 + b1 + c2 vs a1 + b2 + c1	>	>
1.2.2 a1 + b2 + c2					>	>	a1 + b2 + c2 vs a2 + b1 + c1	>
1.1.2 a1 + b1 + c2						a1 + b1 + c2 vs a1 + b2 + c1	a1 + b1 + c2 vs a2 + b1 + c1	>
1.2.1 a1 + b2 + c1							a1 + b2 + c1 vs a2 + b1 + c1	>
2.1.1 a2 + b1 + c1								>
1.1.1 a1 + b1 + c1								

PAPRIKA – Step 2

There are only 6 unique ‘undominated pairs’ (or ‘dilemmas’) to be pairwise ranked:

(1)	b2 + c1	*vs*	b1 + c2 ?
(2)	a1 + c2	*vs*	a2 + c1 ?
(3)	a1 + b2	*vs*	a2 + b1 ?
(4)	a2 + b2 + c1	*vs*	a1 + b1 + c2 ?
(5)	a2 + b1 + c2	*vs*	a1 + b2 + c1 ?
(6)	a1 + b2 + c2	*vs*	a2 + b1 + c1 ?

PAPRIKA – Step 3

PAPRIKA is built on top of preferences that actually obtain a number of inequalities.

Here is PAPRIKA's genius:

We **don't need to rank all possible undominated pairs** in PAPRIKA — **just enough to mathematically determine** the value of each criterion level.

Each **undominated pair** you rank gives you **1 linear inequality**.

To solve for 3 variables, you need at least 2 independent inequalities (but ideally more for redundancy and consistency checks).

So technically, **2 well-chosen undominated pair rankings** are **enough** to produce a solution, as long as:

1. They constrain all variables (no variable is left out).
2. 2. The system is feasible (doesn't contradict itself).

PAPRIKA – Step 3

Let's remember what dominance is in MCDA:

A **dominated pair** is one where one alternative is **as good or better on all criteria** and **strictly better on at least one**.

That means we **already know** which one is better — no need to ask.

An **undominated pair** is a pair of alternatives where **each one is better than the other in at least one criterion**.

In other words:
- One option is better on **some criteria**
- The other is better on **others**
- **Neither one is clearly better overall**

That's why you need to **ask the decision-maker** which one they prefer.

PAPRIKA – Step 3

We ask a few **simple pairwise questions**. Let's say the decision-maker says:

Compare A6 vs A7

- A6 = Light, Expensive = $v_{weight}2$
- A7 = Heavy, Cheap = $v_{price}2$
- Prefers A6 → means: $v_{weight}2 > v_{price}2$

Compare A4 vs A5

- A4 = Low Battery, Light, Cheap = $v_{weight}2 + v_{price}2$
- A5 = High Battery only = $v_{bat}2$
- Prefers A4 → means: $v_{weight}2 + v_{price}2 > v_{bat}2$

PAPRIKA – Step 4

Now we turn to linear programming to solve these inequalities. We know $v_{bat}2, v_{weight}2, v_{price}2 \geq 0$.

With:
1. $v_{weight}2 > v_{price}2$
2. $v_{weight}2 + v_{price}2 > v_{bat}2$

We iteratively try to fit values that satisfy these inequalities:
- $v_{price}2 = 1$
- $v_{weight}2 = 2$ → satisfies (1): 2 > 1
- $v_{bat}2 = 2$ → satisfies (2): 2 + 1 = 3 > 2

So we reach one **valid solution**:
- $v_{bat}2 = 2, v_{weight}2 = 2, v_{price}2 = 1$

PAPRIKA – Step 5

Using these values we can derive the scores and rankings:

ID	Alternative	Score Calculation	Score
A1	(2,2,2)	2 + 2 + 1	5
A2	(2,2,1)	2 + 2 + 0	4
A3	(2,1,2)	2 + 0 + 1	3
A4	(1,2,2)	0 + 2 + 1	3
A5	(2,1,1)	2 + 0 + 0	2
A6	(1,2,1)	0 + 2 + 0	2
A7	(1,1,2)	0 + 0 + 1	1
A8	(1,1,1)	0 + 0 + 0	0

PAPRIKA – Step 6

Now we know scores / rankings. How about weights?

PAPRIKA doesn't ask for weights directly.

Instead, it:

1. **Asks you to compare alternatives** (undominated pairs).
2. 2. **Uses those comparisons** to assign **point values** to **each level of each criterion**.
3. 3. Then uses those **point values** to **derive weights** for the **criteria**.

If:

- v_{ix} is the point value (x) for the best level of criterion i.
- W_i is the weight of criterion I

Then:

$$W_{i=} \frac{v_{ix}}{\sum_{j=1}^{n} v_{jx}}$$

These weights tell you **how much each criterion contributes** to the overall evaluation **when at its best**.

PAPRIKA – Step 6

Assume:
- Each criterion has 2 levels: Level 1 = 0 points, Level 2 = v_{i2}
- For criterion i, its **contribution to the total score** is just v_{i2} if the alternative has Level 2.

Once PAPRIKA solves for all point values:
- $v_{bat}2 = 2$
- $v_{weight}2 = 2$
- $v_{price}2 = 1$

Total points = sum of best levels across all criteria (2 + 2 + 1 = 5):

Weights:
- **Battery**: 2 / 5 = 0.40
- **Weight**: 2 / 5 = 0.40
- **Price**: 1 / 5 = 0.20

These are the **normalized criteria weights**.

PAPRIKA – Step 6

What if criteria have more than 2 levels?

No problem:

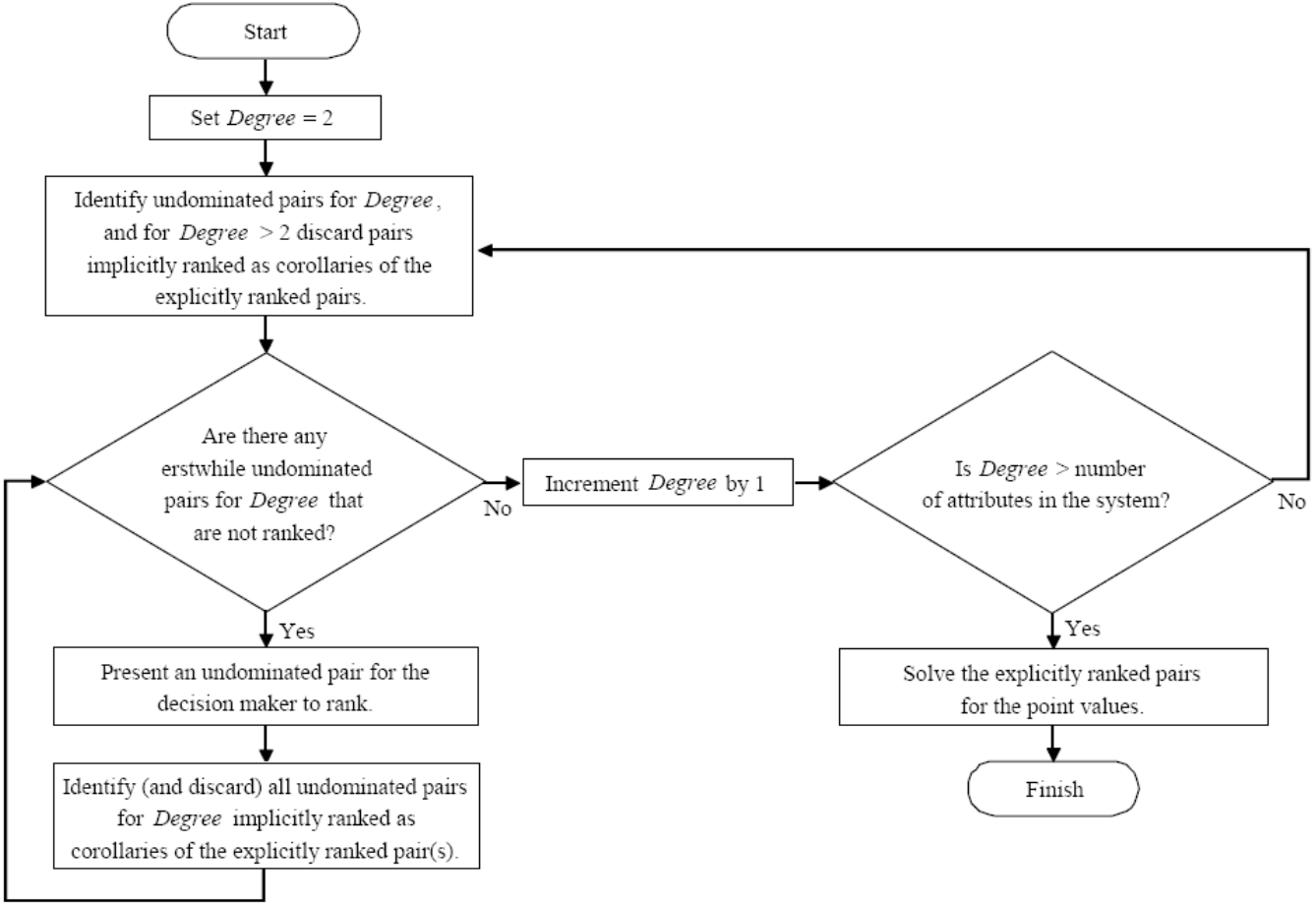
- You sum the **range** of the point values for each criterion:

$$W_i = \frac{v_{i\,\max} - v_{i\,\min}}{\sum_{j=1}^{n}(v_{j\,\max} - v_{j\,\min})}$$

Criterion	Level 1	Level 2	Level 3
Battery Life	0	3	6
Weight	0	2	4
Price	0	1	2

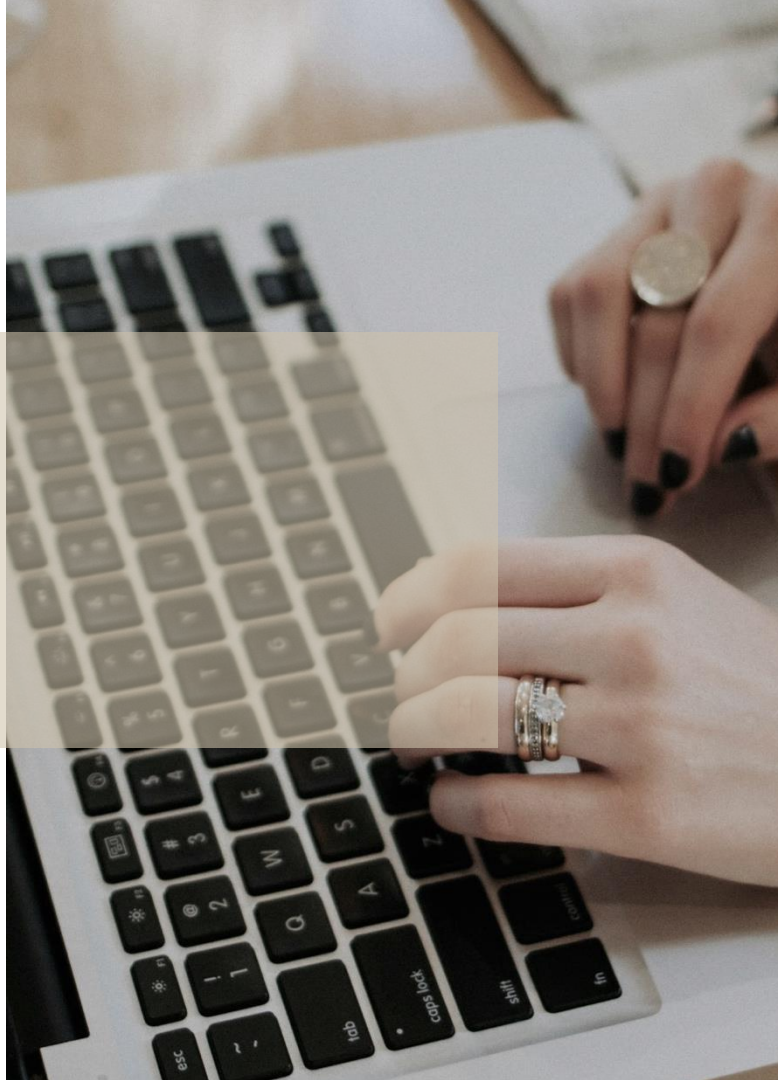
Criterion	v_{max}	v_{min}	Range = v_{max} - v_{min}	Weight
Battery Life	6	0	6	0.50
Weight	4	0	4	0.33
Price	2	0	2	0.17

PAPRIKA


Start
Set *Degree* = 2
Identify undominated pairs for *Degree*, and for *Degree* > 2 discard pairs implicitly ranked as corollaries of the explicitly ranked pairs.
Are there any erstwhile undominated pairs for *Degree* that are not ranked?
No
Increment *Degree* by 1
Is *Degree* > number of attributes in the system?
No
Yes
Present an undominated pair for the decision maker to rank.
Identify (and discard) all undominated pairs for *Degree* implicitly ranked as corollaries of the explicitly ranked pair(s).
Yes
Solve the explicitly ranked pairs for the point values.
Finish

Hands-on approach

- Let’s use 1000Minds (14-day trial)



TIMELINE

Subject to change

04
Fuzzy AHP

06
ELECTRE

03
Analytical Hierarchical Process (AHP)

05
TOPSIS &
Fuzzy TOPSIS

07
PAPRIKA /
MACBETH?

REFERENCES

Today's content was mainly based on

- **Paul Hansen's class notes (University of Otago, New Zealand).**
- **Goodwin, P., & Wright, G. (2014). Decision analysis for management judgment. John Wiley & Sons. Belton, V., & Stewart, T. (2012). Multiple criteria decision analysis: an integrated approach. Springer Science & Business Media.**
- **Greco, S., Figueira, J., & Ehrgott, M. (Eds.). (2016). Multiple criteria decision analysis: state of the art surveys. New York, Springer.**
- **Shih, H. S., & Olson, D. L. (2022). TOPSIS and its extensions: A distance-based MCDM approach (Vol. 447). Springer Nature.**

REFERENCES

IMAGES

- Unsplash

THANKS

Does anyone have any questions?
Contact me at:

fellipe.martins@mackenzie.br
+11 95619 0585 (business hours)
fellipemartins.com.br