MCEWR

Multi-Criteria Evaluation Model for Water Resources

Method Reference

The mathematics of the ten evaluation methods and the analyses – equations, options, output files, verification

Model developed by Mohamed Rami Mahmoud, 1995-2026

Prof. Mohamed Rami Mahmoud, Ph.D., P.E.

Research Professor Emeritus, National Water Research Center, Cairo, Egypt

rami@m-rami.org · www.m-rami.org · rami.mrami.org

ORCID: 0000-0002-3393-987X · LinkedIn: linkedin.com/in/mohamed-rami-64947570

WAM · PROMETHEE II · CP · ELECTRE II · AHP · EV · RA · TOPSIS · VIKOR · ELECTRE III

September 2026

Python edition, version 2.0. Every formula in this document is the formula of the implementation itself: the five 1995 methods are audited against the dissertation and the original output files, and the 2.0 additions are verified against the primary literature and independent checks. Companion documents: MCEWR_User_Guide.pdf, PORTING_NOTES.md and MCEWR_Modernization_Plan.pdf.

Contents

1. Introduction 1
1.1 Purpose and how to read this document 1
1.2 Notation 1
1.3 The 1995 mode and the text-book mode 2

2. Machinery shared by the five methods 3
2.1 The decision problem and the criteria hierarchy 3
2.2 Standardization of the data 3
2.3 Weights and their normalization 4
2.4 Saaty pair-wise matrices: priorities and consistency 4
2.5 Aggregation from Level to Level 5
2.6 Ranking, ties and domination 5

3. Weighted Average Method (WAM) 6
3.1 Purpose 6
3.2 Equations 6
3.3 Options 6
3.4 Output 6
3.5 1995 mode and text-book mode 7
3.6 Verification 7

4. PROMETHEE II 8
4.1 Purpose 8
4.2 Equations 8
4.3 Options 9
4.4 Output 9
4.5 1995 mode and text-book mode 9
4.6 Verification 9
4.7 PROMETHEE I: the partial ranking (version 2.0) 10

5. Compromise Programming (CP) 11
5.1 Purpose 11
5.2 Equations 11
5.3 Options 11
5.4 Output 12
5.5 1995 mode and text-book mode 12
5.6 Verification 12
5.7 SPOTIS: the same method under a 2020 name 13

6. ELECTRE II 14
6.1 Purpose 14
6.2 Equations 14
6.3 Options 15
6.4 Output 16
6.5 1995 mode and text-book mode 16
6.6 Verification 16

7. Analytic Hierarchy Process (AHP) 17
7.1 Purpose 17
7.2 Equations 17
7.3 Options 17
7.4 Output 18
7.5 1995 mode and text-book mode 18
7.6 Verification 18

8. Expected Value (EV) - provisional 19

8.1 Purpose....19
8.2 Equations....19
8.3 Options....19
8.4 Output....20
8.5 Status and open points....20
8.6 Verification....20

9. Regime Analysis (RA) - provisional....21
9.1 Purpose....21
9.2 Equations....21
9.3 Options....21
9.4 Output....22
9.5 Status and open points....22
9.6 Verification....22

10. Weights beyond the file: equal, Best-Worst, entropy, CRITIC, MEREC - provisional....23
10.1 Purpose....23
10.2 The three schemes....23
10.3 Use....23
10.4 Equal weights and the Best-Worst Method (September 2026)....23
10.5 Status and open points of the objective schemes....24

11. TOPSIS - provisional....25
11.1 Purpose....25
11.2 Equations....25
11.3 Options....25
11.4 Output....25
11.5 Status and open points....25
11.6 Verification....26

12. VIKOR - provisional....27
12.1 Purpose....27
12.2 Equations....27
12.3 Options....27
12.4 Output....27
12.5 Status and open points....27
12.6 Verification....28

13. ELECTRE III - provisional....29
13.1 Purpose....29
13.2 Equations....29
13.3 Options....30
13.4 Output....30
13.5 Status and open points....30
13.6 Verification....30

14. Comparing methods, sensitivity, rank reversal....31
14.1 Purpose....31
14.2 Comparison of methods....31
14.3 Sensitivity of the ranking to the weights....31
14.4 Sensitivity of the decision values, of the method parameters, and Monte Carlo robustness (version 2.1)....31
14.5 Rank-reversal diagnostics....32
14.6 Use....33
14.7 Verification....33

15. Verification summary....34

16. References....35

1. Introduction

1.1 Purpose and how to read this document

This document is the mathematical specification of the five evaluation methods of MCEWR as they are implemented in the Python edition: for every method it gives the purpose, the equations, the options, the content of the output files, the points on which the 1995 program departs from the text-book formulation, and the evidence that the implementation is correct. It is written for three readers: the student who wants to understand what the program computes, the analyst who has to report a method in a publication, and the developer who adds a method and needs the conventions that the existing ones follow.

Every formula printed here is the formula of the audited implementation, not a formula copied from a book. The Python package of version 2.0 reproduces the arithmetic of the audited release 1.1.0 exactly (5,980 paired runs, every number and every output file identical), release 1.1.0 reproduces the nine output files of the 1995 C++ program and the verification examples of the dissertation, and 61 independent checks recompute the methods from text-book formulas outside the program. Chapter 8 summarizes this evidence method by method.

The companion documents are the **User Guide** (how to prepare a file, run the program and read the results), `PORTING_NOTES.md` (the conversion from C++, the defects corrected and the full mathematical review) and the **Modernization Plan**. Section numbers in the form "User Guide 4.6" refer to the guide; "PORTING_NOTES 7.4" to the porting notes.

1.2 Notation

Equations are written in linear notation, on one line, without subscripts or superscripts: a quantity that depends on a criterion and an alternative is written with two arguments, `s(i,j)`; a power is written with `^`; a sum over a group of criteria is written "Σ over i of …". The following names are used throughout:

Symbol	Meaning
j, a, b	alternatives; n = number of alternatives that take part in the evaluation (mode TRUE)
i	a criterion; m = number of criteria (mode TRUE) in the group of siblings being evaluated
v(i,j)	raw value of criterion i for alternative j as read from the file – or, for an aggregated criterion, the value carried up from its sub-criteria (section 2.5)
s(i,j)	the value after standardization (section 2.2); equal to v(i,j) when no standardization is chosen or when the criterion carries Saaty priorities
min(i), max(i)	smallest and largest v(i,·) over the evaluated alternatives
w(i)	weight of criterion i after normalization inside its group of siblings (section 2.3), or its Saaty priority
t(i)	declared type of criterion i: +1 = larger values are better (maximize), −1 = smaller values are better (minimize)
T(i)	the effective type used by Compromise Programming and ELECTRE II at a given Level (section 2.1, 5.2)
U(i), L(i)	the Upper limit value and the Lower limit value of criterion i given in the input file
p(i), q(i)	PROMETHEE preference threshold and indifference threshold of criterion i
p*, p0, p−, q*, q0	ELECTRE II concordance thresholds (p* ≥ p0 ≥ p−) and discordance thresholds (q* ≥ q0) of a group
s (bare letter)	the power of the Compromise Programming metric, 1 or 2 – not to be confused with s(i,j)
Σ over i of …, ^	summation over the criteria of the group (or over the alternatives when written "over b"); power
λmax, CI, CR, RI(n)	principal eigenvalue of a Saaty matrix; consistency index; consistency ratio; Saaty's random index for a matrix of order n

Symbol	Meaning
R(i,j)	the contribution of criterion i to the value of alternative j – the “RankValue” column of the output files, the quantity ranked per criterion
Level, generation	position of a criterion in the hierarchy, section 2.1

Names of options are printed as on the command line (`--textbook`) and of the Python interface (`textbook=True`); the two are equivalent. Output-file names are the fixed names of the 1995 program (`wam.txt`, `sortwam.txt`, …).

1.3 The 1995 mode and the text-book mode

By default the program behaves exactly like the 1995 C++ program, so that every result ever published with MCEWR can be regenerated. The mathematical review found five points at which the 1995 program departs from the text-book formulation of a method. None of them is hidden: each is switched to the text-book formulation by an option, `--textbook` switches all five, and in default mode the program prints a warning whenever a run is affected by one of the points that can change a ranking. The five points are described where they belong – in the subsection “1995 mode and text-book mode” of the method concerned – and are collected here for orientation:

Option	Method	What it changes	Ranking?
`--exact-thresholds`	PROMETHEE II, CP	rescaling of thresholds and limits under standardization (section 4.2)	can change
`--promethee-n-minus-1`	PROMETHEE II	flows divided by n − 1 instead of n (section 4.5)	no
`--cp-root`	CP	the s-th root of the sum, the true L_s metric (section 5.5)	no
`--parents-only`	CP, ELECTRE II	which criteria are treated as “smaller is better” above Level 1 (section 5.5, 6.5)	can change
`--electre-symmetric`	ELECTRE II	the discordance divisor of minimizing criteria (section 6.5)	can change

Recommendation. Use the default mode to reproduce or extend earlier work; use `--textbook` for new studies; state the mode in the publication. The console summary prints the active options (`Text-book options: …`) and the Python interface returns them in `result["options"]`.

2. Machinery shared by the five methods

2.1 The decision problem and the criteria hierarchy

A problem consists of a set of **alternatives** (identifier, name, and a mode flag: TRUE = takes part in the evaluation, FALSE = excluded) and a tree of **criteria**. The criteria of the top group are the main criteria; any criterion may be divided into sub-criteria, which may be divided again, without limit on the depth. A criterion that has no sub-criteria is a **leaf** and carries data: one value v(i,j) per alternative, or Saaty pair-wise comparisons of the alternatives. A criterion that has sub-criteria is **aggregated** (a parent): its value is not read from the file but computed from its sub-criteria (section 2.5). Every criterion carries an identifier, a name, a type t(i), a weight, a mode flag, and the Upper and Lower limit values U(i) and L(i). Every group of siblings carries a flag telling whether its weights are given as Saaty comparisons, and the ELECTRE II thresholds p*, p0, p−, q*, q0.

Two numberings of the depth are used. The **generation** counts from the top: the main criteria are generation 1, their sub-criteria generation 2, and so on. The **Level** counts from the bottom: Level 1 is the deepest generation of the whole tree, and the Level of a criterion is (depth of the tree) − (generation) + 1. The evaluation proceeds Level by Level, from Level 1 upwards, and the output files print the criteria in that order (sub-criteria first, main criteria last). The Level matters for one rule of the 1995 program: Compromise Programming and ELECTRE II invert the declared type of every criterion above Level 1 (sections 5.5 and 6.5). A tree is **balanced** when all its leaves are at Level 1; in an **unbalanced** tree some leaves sit at Level 2 or higher, next to aggregated criteria.

```
generation 1 (Level 3)   1 Effectiveness           2 Efficiency       3 Acceptability
generation 2 (Level 2)   11 Fish passage (leaf)    21 Benefits (leaf) 31 Commercial
(aggregated)
generation 3 (Level 1)                                                311 Traffic  312 Jobs
(leaves)

Leaf 11 and leaf 21 sit at Level 2: this tree is unbalanced.
```

2.2 Standardization of the data

Criteria are measured in different units. Standardization maps the values of every leaf criterion, separately, to a common scale; it is applied once, when the file is read, over the alternatives in mode TRUE. Four choices exist (`--normalize`):

Choice	Formula	Result	Linear map s = a·v + b
1	s(i,j) = (v(i,j) − min(i)) / (max(i) − min(i))	0 for the smallest value of the sample … 1 for the largest	a = 1/(max − min), b = −min/(max − min)
2	s(i,j) = v(i,j) / Σ over j of v(i,j)	shares of the total; they sum to 1	a = 1/Σv, b = 0
3	s(i,j) = v(i,j) / max(i)	0 … 1, 1 for the largest value	a = 1/max, b = 0
0	s(i,j) = v(i,j)	the data as they are	a = 1, b = 0

Choice 1 is eq. 3.5 of the dissertation with the (v − min)/(max − min) map; choices 2 and 3 are eq. 3.6a and 3.6b, the two formulas of Nijkamp et al. (1990) for the Weighted Average Method. Three rules apply to all choices:

- **Minimizing criteria are not inverted here.** Standardization is the same map for both types; each method takes the type into account in its own way (the sign of the contribution in WAM and AHP, the direction of the difference in PROMETHEE and ELECTRE II, the side of the ideal in CP).
- **Saaty-valued criteria are not standardized.** When the values of a criterion are given as pair-wise comparisons, the priorities of section 2.4 are used as they are (they already sum to 1).

- **A criterion whose values are all equal** (max(i) = min(i)) gives the same standardized value to every alternative and cannot influence the ranking; `python mcewr.py check` reports it.

The last column of the table – the slope a and the intercept b of the linear map – is what decides how a threshold or a limit should follow the values. A quantity that is a **value** (an Upper or Lower limit) is mapped with the whole map, a·x + b. A quantity that is a **difference** of two values (a PROMETHEE threshold) is mapped with the slope alone, a·p, because the intercept cancels in a difference. The 1995 program applies the (p − min)/(max − min) map to both; this is the first of the five points of section 1.3 (see section 4.2).

2.3 Weights and their normalization

Weights are local: the weights of a group of sibling criteria are divided by their sum, so that within every group the weights add up to 1,

```
w(i) = weight(i) / Σ over the siblings k (mode TRUE) of weight(k)
```

A criterion in mode FALSE is left out of the sum. With `--raw-weights` the weights are used as they are given, which matters for the printed values of PROMETHEE flows and CP distances (the verification examples of the dissertation use raw weights). When the group header says that the weights are Saaty comparisons, w(i) is the priority vector of section 2.4 instead. Because every group is normalized separately and the groups are aggregated level by level, the effective global weight of a leaf is the product of the local weights along its path to the top – a convention of the AHP that MCEWR applies to all five methods.

2.4 Saaty pair-wise matrices: priorities and consistency

The weights of a group, and the values of the alternatives under a criterion, may be given as pair-wise comparisons on the Saaty scale (1 = equal, 3 = moderately more important, 5 = strongly, 7 = very strongly, 9 = extremely; 2, 4, 6, 8 between; the reciprocal 1/x for the opposite judgement). The file must give every ordered pair: a comparison of k with i that is absent is an error (code 226), and the program does not complete or correct a matrix – a file whose entries violate reciprocity, a(k,i) ≠ 1/a(i,k), is accepted as in 1995 (criterion 5 of `AHP_EG6.MCE` is such a case; the check script notes it). The matrix M of order n is turned into a priority vector by Saaty's **power method**:

- M is raised to the powers p = 1, 2, 3, …; after each power the row sums of M^p are divided by their total. The resulting vector is compared with the one of the previous power; when no component changes by more than 1e−9 the vector is the priority vector (it converges to the principal eigenvector for any positive matrix). The C++ program tested only the last component; the port tests all of them. A cap of 5,000 powers replaces the unbounded loop (error 228), and n > 15 is rejected (error 229) because Saaty's random indices are tabulated up to 15.
- λmax = (1/n) · Σ over i of (M·v)(i) / v(i), the principal eigenvalue estimated from the priority vector v.
- CI = (λmax − n) / (n − 1), the consistency index, and CR = CI / RI(n), the consistency ratio. For n ≤ 2 the ratio is 0 by definition (the C++ program divided by zero).

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RI(n)	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

> **Note.** The output files print the ratio under the label `The Connsistency index for (saaty scall)`, the wording of the 1995 program. The number is Saaty's consistency **ratio** CR; a set of judgements is acceptable when CR ≤ 0.10. The file `AHP_EXM3.MCE` prints 0.0996 for its weights and 0.0158 for its values, both acceptable.

2.5 Aggregation from Level to Level

Each method is a rule that turns one group of sibling criteria into one number per alternative. The evaluation applies that rule to every group of Level 1, then to every group of Level 2, and so on to the top group. The number produced for a group becomes the **value v(i,j) of the parent criterion** for the next Level – it is what the parent's siblings will be compared with – and the number produced for the top group is the **overall ranking value** of the alternative. What is carried up depends on the method:

Method	Value carried to the parent	Direction
WAM	the weighted sum U(j)	larger is better
PROMETHEE II	the net flow φ(j)	larger is better
Compromise Programming	the distance L(j)	smaller is better
ELECTRE II	the median rank	smaller is better
AHP	the weighted sum of priorities U(j)	larger is better

For the three methods of the first kind the parent value has the same sense as the data, so a parent declared "maximize" can be treated like any other criterion at the next Level. For CP and ELECTRE II the parent value is a distance or a rank: smaller is better, whatever the parent's declared type. How the 1995 program handles this, and what the text-book option changes, is the subject of sections 5.5 and 6.5. Independently of the value carried up, every criterion – leaf or aggregated – stores one **ranking value R(i,j)** per alternative, which is the quantity ranked criterion by criterion in the sort file (section 2.6); the detail file prints it in the column `RankValue`.

2.6 Ranking, ties and domination

After the last Level the alternatives are sorted by their ranking value – overall, and for every criterion separately – with a stable sort (straight insertion in the C++ program; Python's stable sort gives the same order): in **descending** order for WAM, PROMETHEE II and AHP, in **ascending** order for CP and ELECTRE II. Alternatives in mode FALSE are listed after the others. Ranks are **dense**: tied alternatives share one rank number and the next distinct value receives the next number (1, 1, 2, …), which is what "Rank 1" means for the four tied alternatives of criterion 1 of the Han River example.

The **domination** of an alternative is the percentage by which it is ahead of the next one in the list,

```
Dom(j) = (value(j) − value(next)) / |value(j)| × 100        (× −1 for CP and ELECTRE II)
```

so that a positive percentage always means "ahead". Two adjacent equal values give 0, the last alternative gets 0, and when value(j) = 0 the percentage is undefined and `NF` (not feasible) is printed. The **domination matrix** of the sort file applies the same formula to every ordered pair of alternatives (row against column). The console summary and `result["overall_ranking"]` give the rank, the identifier, the ranking value and the domination (`None` for NF) of every alternative.

3. Weighted Average Method (WAM)

3.1 Purpose

The Weighted Average Method is the additive value model: every alternative receives the weighted sum of its standardized scores, and the largest sum is best. It is the simplest of the five methods, the easiest to explain to a decision maker, and – with standardization 1 – the method that the Red Bluff study found the most consistent (Mahmoud & Garcia 2000). MCEWR follows Einhorn & McCoach (1977) and Nijkamp, Voogd & Rietveld (1990); it is eq. 3.5 of the dissertation, with the standardizations of eq. 3.6a and 3.6b as alternatives.

3.2 Equations

For a group of criteria the contribution of criterion i to alternative j and the value of the alternative are

```
R(i,j) = w(i) · t(i) · s(i,j)
U(j)   = Σ over i of R(i,j)          (criteria in mode TRUE)
```

U(j) is carried to the parent criterion (section 2.5); the U(j) of the top group is the overall ranking value. A minimizing criterion enters with a negative sign, w(i) · (−1) · s(i,j). The text-book alternative would be the complement, w(i) · (1 − s(i,j)); under standardization 1 the two differ by the constant w(i) for every alternative, so the ranking and the domination percentages are identical and no option is needed (PORTING_NOTES 7.6). With standardization 2 or 3 the same holds because the complement is again a constant minus the value.

3.3 Options

WAM has no method-specific option. The result is shaped by the two common choices: the standardization (section 2.2; 1 is the usual choice, 2 for proportional data, 3 for ratios to the best) and the weight normalization (section 2.3; `--raw-weights` only scales U(j) by the sum of the weights of each group and does not change the ranking).

3.4 Output

`wam.txt` prints, for every Level from the bottom, one table per criterion with a line per alternative:

Column	Content
AltId, AltMode	the alternative and its mode (1 = evaluated, 0 = excluded)
CritId, CritMode	the criterion and its mode
Value	v(i,j) – read from the file, or carried up from the sub-criteria
StValue	s(i,j)
Weight, StWeight	the weight as read, and w(i)
type	t(i)
RankValue	R(i,j) = StValue × StWeight × type (0 for an excluded alternative or criterion)

Each Level ends with the table `AltId ALtMode CritId CritMode value[SumValue]` – the U(j) of every group, which becomes the `Value` of the parent at the next Level – and the file ends with `Alternatives Final Results`, the overall U(j). `sortwam.txt` ranks the alternatives by R(i,j) for every criterion and by U(j) overall, with the domination percentages and matrices of section 2.6.

3.5 1995 mode and text-book mode

None of the five points of section 1.3 concerns WAM; the method computes the same numbers in both modes.

3.6 Verification

- **Dissertation, Appendix 2.1–2.2** (`NIJKAMP.MCE --method wam --normalize 1`): U = 0.6, 0.53333, 0.30667; ranks 1, 2, 3; alternative 1 dominates alternative 2 by 11.1 %. **Appendix 2.3–2.4** (`--normalize 3`): 0.80, 0.66, 0.61.
- **Legacy outputs** `WAM.TXT` (Red Bluff, standardization 2) and `SORTWAM.TXT` (Han River, standardization 1) of the 1995 program: reproduced line by line (differences only in the last printed digit of 32-bit numbers, e.g. 0.60000002 in 1995, 0.6 now).
- **Independent checks**: 5 – on `REDBLUFF.MCE`, `NIJKAMP.MCE`, `WAM.MCE`, `LEVEL3.MCE` and `LEVEL4.MCE` the overall value is recomputed outside the program as Σ over the leaves of (global weight × type × standardized value), the global weight of a leaf being the product of the local weights along its path, and compared with the level-by-level result.
- **Case studies**: Red Bluff (`REDBLUFF.MCE`, Mahmoud & Garcia 2000) and the Egypt siting study (`EGYPT48.MCE`, Table 5.4 of the dissertation) reproduce the published rankings.

4. PROMETHEE II

4.1 Purpose

PROMETHEE (Preference Ranking Organization METHod for Enrichment Evaluations) is an outranking method: instead of scoring each alternative on its own, it compares the alternatives two by two, criterion by criterion, turns each difference into a **preference degree** between 0 and 1 with a preference function chosen by the decision maker, and aggregates the degrees into flows. PROMETHEE I gives a partial ranking from the leaving and entering flows; PROMETHEE II, the version implemented, gives a complete ranking from the net flow, and the largest net flow is best. MCEWR follows Brans, Mareschal & Vincke (1984, 1985) and Brans, Vincke & Mareschal (1986) with all six preference functions of the original articles.

4.2 Equations

For every criterion i of the group and every ordered pair of alternatives (a, b) the difference is taken in the sense of the type, so that a positive d always means “a is better than b on criterion i”:

```
d(i; a,b) = (s(i,a) − s(i,b)) · t(i)
```

The six preference functions

The difference is turned into a preference degree P(i; a,b) in [0, 1] by the function of the criterion, with its thresholds p(i) (preference) and q(i) (indifference). In all six functions P = 0 whenever d ≤ 0.

Type	Name	P(i; a,b) for d > 0	Needs
1	usual criterion	P = 1	–
2	U-shape (quasi-criterion)	P = 0 for d ≤ q; P = 1 for d > q	q
3	V-shape (linear preference)	P = d / p for d ≤ p; P = 1 for d > p	p
4	level criterion	P = 0 for d ≤ q; P = 1/2 for q < d ≤ p; P = 1 for d > p	p, q
5	linear with indifference area	P = 0 for d ≤ q; P = (d − q) / (p − q) for q < d ≤ p; P = 1 for d > p	p, q
6	Gaussian criterion	P = 1 − exp(−d^2 / (2·σ^2)), σ = (p + q) / 2	p or q

Two conventions of the 1995 program are kept. A function whose thresholds make it undefined – type 3 with p = 0, type 5 with p = q, type 6 with p = q = 0 – is replaced by type 1 for that criterion and a warning is printed (messages 87, 88, 89) and returned in `result["warnings"]`. For the Gaussian criterion Brans et al. use a free parameter σ (their s); the 1995 program builds σ from the two thresholds of the file as (p + q)/2, so a file that wants σ = 2 writes, for example, p = 4 and q = 0.

Thresholds under standardization

p(i) and q(i) are **differences** of values in the units of the criterion. When the data are standardized the thresholds must follow the values through the slope of the map (section 2.2): with standardization 1 the correct threshold is p / (max(i) − min(i)). The 1995 program instead applies the whole map, (p − min(i)) / (max(i) − min(i)), as if p were a value. For criterion 2 of `PROM_EX.MCE` (values 58 … 96, p = q = 30) the text-book threshold is 30/38 = 0.789 while the 1995 rule gives (30 − 58)/38 = −0.737, a negative threshold that turns the V-shape function into the usual criterion. The option `--exact-thresholds` (or `--textbook`) applies the slope alone; in default mode a warning names the affected criteria. Nothing changes for type 1, or when the data are not standardized – which is the case of every verification example and of every published MCEWR result.

Flows

```
PM(a,b) = (1/n) · Σ over i of w(i) · P(i; a,b)      the PROMETHEE matrix   (1995: divisor n)
φ+(a)   = Σ over b of PM(a,b)                        leaving flow (strength of a)
φ−(a)   = Σ over b of PM(b,a)                        entering flow (weakness of a)
φ(a)    = φ+(a) − φ−(a)                              net flow: the LARGEST is best
```

φ(a) is carried to the parent criterion; the φ of the top group is the overall ranking value. The ranking value stored per criterion is the weighted leaving flow of that criterion alone, R(i,a) = Σ over b of w(i) · P(i; a,b), which is what `sortprom.txt` ranks criterion by criterion. Brans et al. (1986) divide the flows by n − 1, the number of other alternatives; the 1995 program divides by n. Every flow is then scaled by the constant (n − 1)/n, so the ranking and the domination percentages are unchanged and only the printed values differ.

4.3 Options

Option	Effect	Default
preference function type, p, q	in the input file, per leaf criterion (`Pf`, `p`, `q` of the pay-off matrix)	type 1
`--exact-thresholds`	thresholds rescaled as differences under standardization (section 4.2); also the CP limits, section 5.3	1995 rule
`--promethee-n-minus-1`	flows divided by n − 1 as in Brans et al. (1986); values only	divisor n
`--normalize`, `--raw-weights`	sections 2.2 and 2.3; with raw weights the flows are scaled by the sum of the weights of the group	0; normalized

4.4 Output

`promethe.txt` prints, for every group of criteria, the **pay-off matrix** (per criterion: identifier, type, mode, w(i), the function type `Pf` and the thresholds `p`, `q` after rescaling, then the standardized values of the alternatives), the weighted preference degrees w(i) · P(i; a,b) per criterion, the **PROMETHEE matrix** PM(a,b), and the table `THE OVERALL VALUES FOR ALTS` with φ+, φ− and φ. `sortprom.txt` ranks the alternatives by R(i,a) per criterion and by φ overall.

4.5 1995 mode and text-book mode

Point	1995 mode (default)	Text-book mode	Ranking
Thresholds under standardization (`--exact-thresholds`)	p, q mapped as values, (p − min)/(max − min)	p, q mapped as differences, p/(max − min)	can change (types 2–6 with standardized data)
Divisor of the flows (`--promethee-n-minus-1`)	n	n − 1	unchanged; values scaled by n/(n − 1)

Example of the second point: with raw weights and divisor 6 the Brans example gives φ = 1.4645 for alternative 5; with weights normalized to 1 and divisor 5 the text-book value is 0.2929 = 1.4645 / 5 – the same alternative ranks first in both cases.

4.6 Verification

- **Dissertation, Appendix 2.5–2.6** (`PROM_EX.MCE --method promethee --normalize 0 --raw-weights`, the six-alternative example of Brans et al.): φ = 1.4645, 0.0858, −0.1000, −0.2744, −0.4476, −0.7283.

- **Legacy outputs** `PROMETHE.TXT` (Red Bluff, no standardization) and `SORTPROM.TXT` (`EG680_EX.MCE`, standardization 1): reproduced.
- **Independent checks**: 19 – on `PROM_EX.MCE`, `NIJKAMP.MCE` and `EG680_EX.MCE` the 1995 flows are shown to equal the text-book flows (weights normalized to 1, divisor n − 1) times the constant (Σ w)(n − 1)/n with the same order (3 checks); and on `PROM_EX.MCE`, `REDBLUFF.MCE`, `EGYPT48.MCE` and `LEVEL4.MCE`, under each of the four standardizations, the hierarchical flows with exactly rescaled thresholds are recomputed independently and compared with the `--textbook` run (16 checks).

4.7 PROMETHEE I: the partial ranking (version 2.0)

PROMETHEE I (Brans, Vincke & Mareschal 1986) draws a **partial** ranking from the two flows separately instead of from their difference. For an ordered pair (a, b) of the evaluated alternatives of a group:

```
a P b   a outranks b:    φ+(a) > φ+(b) and φ−(a) ≤ φ−(b),   or   φ+(a) ≥ φ+(b) and φ−(a) < φ−(b)
a I b   indifferent:     φ+(a) = φ+(b) and φ−(a) = φ−(b)
a R b   incomparable:    a leaves more than b but also enters more (or the reverse)
```

Two alternatives are incomparable when the leaving and the entering flow disagree about them – a is stronger on the criteria where it beats others, b on the criteria where it is beaten less – and PROMETHEE I refuses to rank them, which is the information PROMETHEE II hides when it computes φ. The option `--promethee-partial` (`promethee_partial=True` in Python) prints, for every group, the matrix of relations with Φ+ and Φ− beside it and the list of incomparable pairs, in `promethe.txt`, and keeps those of the top group in `result["promethee1"]` for the reports (section “PROMETHEE I partial ranking”). The complete ranking, every ranking value and `sortprom.txt` are unchanged – without the option the output is byte-identical to the 1995 layout, which the differential test guarantees – so PROMETHEE I is delivered as a *reading* of the PROMETHEE II computation, not as a separate method with an artificial complete ranking. On the Brans example alternative 5 outranks every other one while six pairs among the rest are incomparable; the relation is verified in `tests/test_phase5.py` against an independent derivation from the flows.

5. Compromise Programming (CP)

5.1 Purpose

Compromise Programming measures how far every alternative is from the **ideal point** – the vector of the best value of every criterion – and ranks the alternatives by that distance: the smallest distance is best, and 0 means that the alternative is ideal on every criterion. The family of distances L_s is indexed by a power s; s = 1 sums the weighted deviations, s = 2 penalizes a large deviation on one criterion more than several small ones, and the two are customarily reported together (Goicoechea et al. 1982). MCEWR follows Zeleny (1973, 1982) and Duckstein & Opricovic (1980); it is eq. 3.20–3.21 of the dissertation.

5.2 Equations

Compromise Programming works with the **effective type** T(i) of the criterion at the current Level. In 1995 mode T(i) = t(i) at Level 1 and T(i) = −t(i) at every Level above 1; with `--parents-only` T(i) = −1 for every aggregated criterion and T(i) = t(i) for every leaf, at any Level (section 5.5). The values are then oriented so that larger is better,

```
x(i,j) = T(i) · s(i,j)
```

The **ideal** and **anti-ideal** of a criterion are the largest and the smallest x(i,·). They are taken either from the input file (`--cp-limits provided`, the default: Max = U(i), Min = L(i) – with the sign change of a minimizing criterion the roles swap, Max = −L(i), Min = −U(i)) or from the data (`--cp-limits recalc`: Max and Min are the extremes of x(i,·) over the evaluated alternatives). Provided limits make the distances independent of which alternatives are present. A value outside [Min, Max] is clamped to the interval. The distance of alternative j to the ideal on criterion i, the contribution, the group value and the value carried up are

```
d(i,j) = (Max(i) − x(i,j)) / (Max(i) − Min(i))        0 at the ideal, 1 at the anti-ideal
         (d = 0 when Max(i) = Min(i))
R(i,j) = (w(i) · d(i,j)) ^ s
L(j)   = Σ over i of R(i,j)                            1995: the SMALLEST is best
```

L(j) is carried to the parent criterion, and the L(j) of the top group is the overall ranking value. Note the order of the operations in R(i,j): the weight multiplies the distance **before** the power, (w · d)^s, which is the definition of Zeleny (1973) and of Duckstein & Opricovic (1980).

The text-book distance is the metric L_s(j) = (Σ over i of (w(i) · d(i,j))^s) ^ (1/s). The 1995 program stores the sum without the root. For s = 1 the two are the same number; for s = 2 the ranking is the same and the 1995 values are the squares of the text-book values. With `--cp-root` the root is taken at every aggregation, so that each parent value and the overall value is an L_s distance.

5.3 Options

Option	Effect	Default
`--cp-power 1|2`	the power s of the metric	1
`--cp-limits provided|recalc`	ideal and anti-ideal from the Upper / Lower limit values of the file, or recomputed from the data. `provided` needs a limit for every criterion, including the aggregated ones in 1995 mode; use `recalc` unless the limits were set on purpose.	provided
`--cp-root`	the s-th root at every aggregation (the L_s metric)	no root
`--exact-thresholds`	under standardization the provided limits are mapped with the same map as the values (section 2.2) instead of the (x − min)/(max − min) rule applied to every standardization	1995 rule

Option	Effect	Default
`--parents-only`	the text-book hierarchy rule of section 5.5	1995 rule

5.4 Output

`compro.txt` prints, for every criterion, one line `UseMaxMinValueProvide = TRUE|FALSE -- Max Value = … -- Min Value = …` giving the ideal and the anti-ideal actually used and where they came from, then the table of section 3.4 in which `StValue` is x(i,j) after clamping and `RankValue` is R(i,j) = (StWeight × distance)^s. The Level tables `value[SumValue]` give L(j) per group and `Alternatives Final Results` the overall L(j). `sortcomp.txt` ranks the alternatives in ascending order of R(i,j) per criterion and of L(j) overall; the domination percentages are sign-corrected so that positive means “ahead” (section 2.6).

5.5 1995 mode and text-book mode

Point	1995 mode (default)	Text-book mode	Ranking
The root (`--cp-root`)	Σ (w·d)^s	(Σ (w·d)^s)^(1/s), at every aggregation	unchanged (s = 2: values are squares)
Which criteria are “smaller is better” above Level 1 (`--parents-only`)	the declared type of EVERY criterion above Level 1 is inverted	an aggregated criterion is always “smaller is better”; a leaf keeps its declared type at any Level; the extremes of aggregated criteria are always recomputed from the data	can change (unbalanced trees; parents declared “minimize”)
Provided limits under standardization (`--exact-thresholds`)	(x − min)/(max − min) for every standardization	the map of the chosen standardization	can change (standardizations 2 and 3 with provided limits)

The second point is the important one. The value carried up from a group is a distance, so at the next Level the parent must be treated as “smaller is better”. The 1995 program achieves this by inverting the declared type of every criterion above Level 1 – which is right for an aggregated criterion declared “maximize”, but reverses the sense of (a) a leaf criterion that sits above Level 1, i.e. in an unbalanced tree, and (b) an aggregated criterion declared “minimize”. It also uses the file limits of aggregated criteria, which have no meaning for a distance. For a balanced tree whose parents are declared “maximize” – `REDBLUFF.MCE` and every verification example – the two rules give identical numbers. The CP column of Table 5.4 of the dissertation (Egypt, `PHD.MCE`, two leaves at Level 2) was produced with the 1995 rule; with `--parents-only` the sea-distance criterion is minimized and the border-distance criterion maximized as declared, and the ranking changes. `python mcewr.py check` reports an unbalanced tree before the run.

5.6 Verification

- **Dissertation, Appendix 2.7–2.8** (`COMP_HAN.MCE --method cp --cp-power 1 --cp-limits provided --normalize 0 --raw-weights`, the Han River example, 9 alternatives and 11 criteria): ranking 8, 9, 4, 5, 7, 3, 6, 2, 1. **Appendix 2.9–2.10** (`--cp-power 2`): 4, 8, 9, 3, 5, 7, 6, 2, 1.
- **Legacy outputs** `COMPRO.TXT` (Han River, s = 1, extremes recomputed, standardization 1) and `SORTCOMP.TXT` (Red Bluff, s = 2, extremes recomputed, no standardization, raw weights): reproduced.
- **Independent checks**: 14 – on `COMP_HAN.MCE`, `EG680_EX.MCE` and `NIJKAMP.MCE`, for s = 1 and 2, the 1995 value equals (L_s)^s with L_s recomputed from the raw data, same order (6 checks); on `REDBLUFF.MCE`, `EGYPT48.MCE`, `LEVEL4.MCE` and `COMP_HAN.MCE`, both powers, the hierarchical L_s with parents as distances is recomputed and compared with the `--textbook` run (8 checks).

5.7 SPOTIS: the same method under a 2020 name

The Stable Preference Ordering Towards Ideal Solution method (Dezert, Tchamova, Han & Tacnet 2020) measures, for every alternative, the weighted sum of its normalized distances to an ideal defined by **bounds given in advance** for every criterion – the property that makes it free of rank reversal, since adding or removing an alternative cannot move the reference points. That is precisely Compromise Programming with the provided Upper/Lower limits and s = 1, as MCEWR has computed it since 1995: the identity holds to eleven decimal places and byte for byte in the output files (`tests/test_phase5.py`). `--method spotis` therefore runs CP with `--cp-power 1 --cp-limits provided` whatever other CP options are given, and the console summary and the reports record "Invoked as SPOTIS". The rank-reversal-free property is thus a property of the 1995 model that the later literature named; Phase 6 will exhibit it alongside the rank-reversal diagnostics of the other methods.

6. ELECTRE II

6.1 Purpose

ELECTRE (ELimination Et Choix Traduisant la REalité) is the outranking method of the French school. ELECTRE I (Benayoun, Roy & Sussman 1966; Roy 1971) selects the alternatives that are preferred on most criteria without being unacceptably bad on any one of them, and gives only a partial ranking; ELECTRE II (Roy & Bertier 1971) adds a second, weaker relation and two **distillations** that produce a complete ranking. For every ordered pair of alternatives the method computes how much of the weight is in favour of the first (**concordance**) and how badly it loses on its worst criterion (**discordance**); five thresholds turn the two numbers into a strong or a weak outranking relation; the relations are then explored forward and backward and the two ranks obtained are averaged. The smallest average is best. MCEWR follows Goicoechea, Hansen & Duckstein (1982); it is eq. 3.22–3.26 and the five steps of section 3.4 of the dissertation.

6.2 Equations

ELECTRE II uses the effective type T(i) of section 5.2 (the same 1995 rule and the same text-book option). For an ordered pair (a, b) the evaluated criteria of the group are split into those on which a is better than b in the sense of T(i), those on which the two are equal, and those on which b is better, and the weights of the three sets are added:

```
W+(a,b) = Σ of w(i) over the criteria where a is better
W=(a,b) = Σ of w(i) over the criteria where the values are equal
W−(a,b) = Σ of w(i) over the criteria where b is better
C(a,b)  = (W+(a,b) + W=(a,b)) / Σ over i of w(i)          the concordance, in [0, 1]
```

Discordance

The discordance measures the largest loss of a against b, criterion by criterion, relative to a scale term; the maximum is taken over the criteria on which b is better than a:

```
D(a,b) = max over i where b beats a of  |s(i,b) − s(i,a)| / θ(i,a)

θ(i,a) = 1                              data standardized with formula 1, group at Level 1
θ(i,a) = max(z(i), s(i,a))              data not standardized, maximizing criterion
θ(i,a) = s(i,a)                         data not standardized, minimizing criterion   (1995 rule)
θ(i,a) = max(z(i), s(i,a))              data not standardized, minimizing criterion   (--electre-
symmetric)
θ(i,a) = max(zmax(i), s(i,a))           every other case, both types
```

Here z(i) is the **scale** of the criterion, the Upper limit value U(i) of the file (or 0 with `--electre-min-discordance-zero`, which makes the loss relative to the value of a itself), and zmax(i) is the largest standardized value of the criterion recomputed from the data – the case of standardizations 2 and 3, and of every group above Level 1 whatever the standardization, where the values are aggregated numbers without a file scale. A scale term equal to 0 stops the method (error 96). The third line is the fifth point of section 1.3: for a minimizing criterion with unstandardized data the C++ program (ELECTRE.CPP, line 459) computes min(z, −s(i,a)), which is −s(i,a) for every z ≥ 0, so the scale is ignored, whereas the recomputed branch of the same function uses the symmetric rule. Whether this was intended is an open question for the author of the model; the port reproduces it by default and offers the symmetric divisor as an option. It only matters for minimizing criteria with unstandardized data and a non-zero Upper limit value – none of the verification examples (`ELEC_EX.MCE` gives Upper limit 0 to its minimizing criteria).

> **Important.** A criterion with weight 0 has no concordance but still enters the discordance, because the discordance does not use the weights. The Red Bluff study noted this as a way in which ELECTRE II can

> mislead a decision maker (Mahmoud & Garcia 2000); `python mcewr.py check` lists active criteria with weight 0 for this reason.

Strong and weak outranking

Five thresholds are given per group in the input file, three levels of concordance p* ≥ p0 ≥ p− and two levels of discordance q* ≥ q0, all in [0, 1]. Alternative a **strongly outranks** b when

```
W+(a,b) ≥ W−(a,b)   and   [ (C(a,b) ≥ p*  and  D(a,b) ≤ q*)   or   (C(a,b) ≥ p0  and  D(a,b) ≤ q0) ]
```

and **weakly outranks** b when

```
W+(a,b) ≥ W−(a,b)   and   C(a,b) ≥ p−   and   D(a,b) ≤ q*
```

Typical sets are (p*, p0, p−; q*, q0) = (0.75, 0.65, 0.55; 0.40, 0.30) or (0.80, 0.75, 0.65; 0.35, 0.25). Lower concordance levels and higher discordance levels create more relations.

Distillations and the ranking value

The **forward distillation** ranks from the top: among the alternatives not yet ranked, every alternative that no other remaining alternative outranks – strongly or weakly – receives the current rank; those alternatives are removed, the rank is increased by one, and the step is repeated until all alternatives are ranked (a pair of alternatives that outrank each other strongly does not block either of them). The **reverse distillation** does the same on the reversed graphs, ranking from the bottom, and its ranks are converted to the forward scale, r' = T + 1 − r(reverse), where T is the number of ranks of that distillation. The ranking value of an alternative is the mean of its two ranks,

```
value(a) = (r(forward, a) + r'(reverse, a)) / 2          the SMALLEST is best
```

– the “median” rank of the output files. It is carried to the parent criterion, and the value of the top group is the overall ranking value. When a step of a distillation ranks nobody the relations contain a cycle and no complete ranking exists for the chosen thresholds; the method stops with error 201 or 206, as the 1995 program did, and the thresholds must be changed. The ranking value stored per criterion is the weighted number of alternatives that beat a on that criterion, R(i,a) = Σ over b of w(i) where b is better than a (smaller is better), which is what `sortele2.txt` ranks criterion by criterion.

6.3 Options

Option	Effect	Default
p*, p0, p−, q*, q0	in the input file, per group of criteria	–
`--electre-min-discordance-zero`	unstandardized data: scale z(i) = 0 instead of the Upper limit value (the loss is relative to the value of a)	Upper limit
`--electre-symmetric`	the symmetric discordance divisor max(z(i), s(i,a)) for minimizing criteria (section 6.5)	1995 rule
`--parents-only`	the text-book hierarchy rule (sections 5.5, 6.5)	1995 rule
`--normalize`, `--raw-weights`	sections 2.2 and 2.3; standardization 1 makes θ = 1 at Level 1 and the discordance a plain difference	0; normalized

6.4 Output

`electreii.txt` (named `ELECTREI.TXT` by the 1995 program) prints, for every group: the pay-off matrix of standardized values; the line `NORMALIZE VALUES = …` and one line `FOR CRITERIA = … <scale>` per criterion with the scale term used; the `Plus`, `Equal` and `Minus` weight matrices W+, W=, W−; the `Concordanace` and `DisConcordanace` matrices C and D; the `Strong` and `Weak Relationship` matrices (1 = the row alternative outranks the column alternative); and the table `AltId CritId RankingVForStrongG RankingVForWeekG Median` with the forward rank, the converted reverse rank and their mean. `sortele2.txt` ranks the alternatives in ascending order of R(i,a) per criterion and of the median rank overall.

6.5 1995 mode and text-book mode

Point	1995 mode (default)	Text-book mode	Ranking
Discordance divisor of minimizing criteria, unstandardized data (`--electre-symmetric`)	θ = s(i,a): the scale is ignored (ELECTRE.CPP line 459)	θ = max(z(i), s(i,a)), as for maximizing criteria	can change (minimizing criteria, no standardization, U(i) ≠ 0)
Which criteria are “smaller is better” above Level 1 (`--parents-only`)	the declared type of EVERY criterion above Level 1 is inverted	an aggregated criterion is always “smaller is better”; a leaf keeps its declared type at any Level	can change (unbalanced trees; parents declared “minimize”)

The second point is the rule of section 5.5 applied to ELECTRE II, where the value carried up is a median rank. The ELECTRE II column of Table 5.4 of the dissertation (Egypt, `PHD.MCE`) was produced with the 1995 rule and changes with `--parents-only`; `REDBLUFF.MCE` and the verification examples give the same numbers under both rules.

6.6 Verification

- **Dissertation, Appendix 2.13–2.14** (`ELEC_EX.MCE --method electre --normalize 0 --raw-weights`, five alternatives): median ranks 3 → 1, 1 → 2, 5 → 2.5, 2 → 3, 4 → 4.
- **Legacy outputs** `ELECTREI.TXT` (Red Bluff, 11 alternatives, unstandardized data with the provided scales – the case of the 1995 discordance rule) and `SORTELE2.TXT` (Han River, 9 alternatives, 11 criteria, standardization 1): reproduced, including the concordance and discordance matrices, both graphs and the distillations.
- **Independent checks**: 8 – the median ranks of `ELEC_EX.MCE` (with and without standardization), `ELEC_HAN.MCE`, `EG680_EX.MCE` and `ELEC2HAN.MCE` are recomputed with an independent implementation of the concordance, the discordance, the two graphs and the two distillations (5 checks); the symmetric divisor on `ELEC_EX.MCE` and `ELEC2HAN.MCE` and the leaves above Level 1 keeping their type on `EGYPT48.MCE` are checked against the `--textbook` runs (3 checks).

7. Analytic Hierarchy Process (AHP)

7.1 Purpose

The Analytic Hierarchy Process of Saaty (1980, 1982) structures a problem as a hierarchy and derives every weight and every score from **pair-wise comparisons** on a 1–9 scale, checked for consistency, instead of asking the decision maker for numbers directly. The score of an alternative is the weighted sum of its priorities, and the largest is best. MCEWR implements the AHP as the model of Saaty with the two treatments of minimizing criteria discussed by Harker (1988) and Goicoechea et al. (1982), and lets a problem mix pair-wise comparisons with measured data: each group of weights, and each criterion, is either Saaty-scaled or numeric, independently of the others.

7.2 Equations

Two kinds of priorities are computed with the power method of section 2.4. The **priority of a criterion** inside its group is the eigenvector of the pair-wise comparison matrix of the siblings when the group header says so, and the normalized weight of section 2.3 otherwise; in both cases it is w(i). The **priority of an alternative under a criterion** is the eigenvector of the pair-wise comparison matrix of the alternatives when the criterion gives Saaty values, and the standardized value s(i,j) of section 2.2 when it gives numbers; in both cases it is written s(i,j) below. Every matrix prints its consistency ratio.

The contribution of criterion i to alternative j depends on how minimizing criteria are treated (`--ahp-min`):

```
method 2 (default):  R(i,j) = w(i) · t(i) · s(i,j)       minimizing criteria: the weight × (−1)
method 1:            R(i,j) = w(i) · s(i,j)              maximizing criteria
                     R(i,j) = w(i) · (1 / s(i,j))        minimizing criteria: the value inverted
U(j) = Σ over i of R(i,j)                                 the LARGEST is best
```

U(j) is carried to the parent criterion and the U(j) of the top group is the overall ranking value – the aggregation of the AHP is the aggregation of section 2.5, and the product of the local priorities along a path is the global priority of a leaf, as in Saaty. Method 2 is the treatment of Goicoechea et al. (1982) and coincides with the WAM rule of section 3.2. Method 1 is Harker's inversion; it cannot be applied to a zero score (error 215), the same limitation as in Harker's formulation. When every criterion is maximizing the two methods give the same numbers.

7.3 Options

Option	Effect	Default
`Saaty scall for Weights: 1`	in the input file, per group: the weights of the group are pair-wise comparisons	0 (numeric weights)
Saaty values per criterion	in the input file: the values of the alternatives are pair-wise comparisons (all ordered pairs must be given)	numeric values
`--ahp-min 1|2`	treatment of minimizing criteria with numeric data (section 7.2)	2
`--normalize`	standardization of the numeric criteria; Saaty priorities are never standardized	0

7.4 Output

`ahp.txt` prints, for every criterion, the Saaty tables that apply – `Saaty Weights for Criteria` (the comparisons of the criterion with its siblings) and `Saaty Values for Criteria` (the comparisons of the alternatives under it), each followed by `The Connsistency index for (saaty scall)`, which is the consistency ratio CR of section 2.4 – and then the table of section 3.4 with `StValue` = the priority or standardized value, `StWeight` = w(i) and `RankValue` = R(i,j). The Level tables and `Alternatives Final Results` give U(j). `sortahp.txt` ranks the alternatives by R(i,j) per criterion and by U(j) overall.

7.5 1995 mode and text-book mode

None of the five points of section 1.3 concerns the AHP. Four details of the implementation were corrected in the port without changing any result of the verification examples, and are listed here because they concern the mathematics: the convergence test of the power method examines all components of the priority vector (the C++ program examined only the last one); a matrix of order 1 or 2 has CR = 0 instead of a division by zero; the power loop is capped at 5,000 powers (error 228) and the order at 15 (error 229); a matrix that is not reciprocal is accepted as in 1995 and reported by the check script.

7.6 Verification

- **Dissertation, Appendix 2.11–2.12** (`AHP_EXM3.MCE --method ahp --ahp-min 2 --normalize 0`, three alternatives and six Saaty-scaled criteria): priorities 0.4019, 0.3445, 0.2536; consistency ratios 0.0996 (weights) and 0.0158 (values).
- **Legacy output** `AHP.TXT` (Red Bluff, standardization 1, minimum method 1, raw weights – the run that exercises the inversion of the values): reproduced.
- **Independent checks**: 8 – for `AHP_EXM3.MCE`, `AHP_EXM2.MCE`, `AHP_EG6.MCE` and `AHP_HARK.MCE` the priority vector of every matrix is compared with the principal eigenvector of a general eigen-solver (numpy; largest difference 3.4e−10) together with the consistency ratio, and the overall value is recomputed as Σ over the leaves of (global priority × priority of the alternative).
- **Case studies**: Red Bluff (`REDBLUFF.MCE`) and the AHP column of Table 5.4 of the dissertation (`EGYPT48.MCE`).

8. Expected Value (EV) - provisional

8.1 Purpose

The Expected Value method of Rietveld (1980; Nijkamp, Rietveld & Voogd 1990) handles **ordinal** information: a ranking of the criteria by importance, and a ranking of the alternatives under each criterion, without numbers attached. The ordinal information is interpreted as unknown quantitative numbers of which only the order is known; every set of numbers consistent with the order is taken to be equally likely, each ordinal statement is replaced by its **expected value** under that assumption, and the expected values are combined with the Weighted Average Method of chapter 3. The largest value is best. It is one of the two methods the 1999 desalination study added to MCEWR; the original 1999 code has not survived, so this is a new implementation from the primary literature.

8.2 Equations

For a group of n sibling criteria the **importance ranking** r(i) is derived from the declared weights: the largest weight gets rank 1, equal weights share the average of their positions (section 2.6). The set of weight vectors consistent with the ranking is the ordered simplex {w : w(1) ≥ w(2) ≥ … ≥ w(n) ≥ 0, Σ w = 1}; under the uniform distribution on it the expected weight of the criterion in position i is the **rank-order-centroid** weight (Barron & Barrett 1996),

```
E[w(i)] = (1/n) · (1/i + 1/(i+1) + ... + 1/n)          e.g. n = 3:  0.6111, 0.2778, 0.1111
tied criteria share the mean of the weights of the positions they occupy
```

Under every criterion the m evaluated alternatives are ranked in the sense of the type (best value = rank 1, ties averaged). The set of score vectors consistent with the ranking is {x : 1 ≥ x(1) ≥ … ≥ x(m) ≥ 0} – m ordered uniform numbers – and the expected score of the alternative in position k is that of the k-th largest of m uniform numbers,

```
E[x(k)] = (m + 1 - k) / (m + 1)          e.g. m = 3:  0.75, 0.50, 0.25;  a tied rank interpolates
linearly

R(i,j) = E[w(i)] · E[x(i,j)]             the contribution of criterion i (RankValue)
U(j)   = Σ over i of R(i,j)              the LARGEST is best
```

U(j) is carried to the parent criterion exactly as in WAM (section 2.5) and the U(j) of the top group is the overall ranking value; the declared type of every criterion is used at every Level, as in the other "larger is better" methods. Two properties follow directly. The expected weights fall steeply with the rank (a convex profile: the gap between the first and the second weight is the largest), while the expected scores fall **linearly** with the rank. And because only the *order* of the values of a criterion enters, the standardization of section 2.2 can never change an EV result in the default mode – a property the tests check.

8.3 Options

Option	Effect	Default
`--ev-weights ordinal|cardinal`	`ordinal`: the rank-order-centroid expected weights from the ranking of the declared weights. `cardinal`: the declared (normalized) weights as they are.	ordinal
`--ev-scores ordinal|cardinal`	`ordinal`: the expected scores (m + 1 – rank)/(m + 1). `cardinal`: the standardized values with the sign of the type, as in WAM.	ordinal

With both options `cardinal` the method is **identical to WAM** – an identity the test suite verifies to ten decimal places on the Red Bluff problem – so EV can be read as WAM with the cardinal information replaced, wholly or in part, by ordinal expectations.

8.4 Output

`expval.txt` prints, per group, the importance ranks and expected weights of the criteria, then one table per criterion with the columns of section 3.4 plus `Rank` (the ordinal rank of the alternative under the criterion), `E[score]` and `E[weight]`, with `RankValue` = E[weight] × E[score]; the Level tables and `Alternatives Final Results` give U(j). `sortev.txt` ranks the alternatives by R(i,j) per criterion and by U(j) overall, with the dominations of section 2.6.

8.5 Status and open points

EV is **provisional**: the registry, the console summary, the reports and `python mcewr.py methods` all say so. What is in place: the closed-form checks above (the sampled ordered simplex reproduces the rank-order-centroid weights and the order-statistic scores), the Nijkamp example of section 3.6 computed by hand inside the test and matched to ten decimal places, an independent recursive implementation matched on the hierarchical examples (`REDBLUFF.MCE`, `LEVEL4.MCE`, `COMP_HAN.MCE`), and the WAM identity. What is still required before the method is promoted to audited: the reproduction of a **published worked example** (Nijkamp, Rietveld & Voogd 1990 print one; the tables of the 1999 desalination paper were found not to contain enough information to reproduce its rankings and are excluded as evidence). One formulation question is open for the author: the 1999 paper applied an additional constraint to the ordinal scores (its equations 4–5, described as giving a concave transformation); that variant is not implemented until its exact form is confirmed.

8.6 Verification

- **Closed forms** (`tests/test_phase4.py`): E[w] of 20,000 sampled ordered-simplex vectors = the ROC formula within 0.006 for n = 3 and 5; the mean k-th largest of m uniform numbers = (m+1−k)/(m+1); exact ROC fractions for n = 4 (25/48, 13/48, 7/48, 3/48); tie conventions.
- **Hand example**: the Nijkamp problem – ranks and expected values written out by hand in the test – gives U = 0.583333, 0.527778, 0.388889 (alternatives 2, 1, 3), matched to ten decimal places.
- **Independent implementation**: a small recursive EV written without the model machinery reproduces the hierarchical runs to nine decimal places.
- **Identities**: cardinal/cardinal ≡ WAM; the result is unchanged under standardizations 0, 1 and 3.

9. Regime Analysis (RA) - provisional

9.1 Purpose

The regime method of Hinloopen, Nijkamp & Rietveld (1983; Nijkamp, Rietveld & Voogd 1990) is the ordinal member of the outranking family: it compares the alternatives two by two, but of each difference it keeps only the **sign**, and of the weights only their **ranking**. Every weight vector consistent with the ranking is taken to be equally likely, and the share of that set on which a pair is won becomes the probability of winning it; the probabilities are aggregated into a complete ranking. The largest score is best. It is the second method the 1999 desalination study added to MCEWR, reimplemented here from the primary literature.

9.2 Equations

For an ordered pair of alternatives (a, b) the **regime vector** over the evaluated criteria of the group is

```
sigma(i) = sign( (s(i,a) − s(i,b)) · t(i) )        +1  a better on i;   0  tie;   −1  b better
```

With **cardinal weights** (`--ra-weights cardinal`) the weighted sum Σ over i of w(i) · sigma(i) decides the pair outright: positive → P(a,b) = 1, negative → 0, zero → 1/2. With **ordinal weights** (the default) the criteria are ranked by their declared weights as in section 8.2 and

```
P(a,b) = Prob over the ordered simplex of  Σ over i of w(i) · sigma(i) > 0
         (a zero sum counts one half;  P(b,a) = 1 − P(a,b))
```

The probability is estimated on `--ra-samples` weight vectors (default 10,000) drawn **uniformly** from the ordered simplex – the gaps between sorted uniform numbers, sorted in decreasing order – with the seed `--ra-seed` (default 1995) **recorded** in the detail file, so that a run is repeatable byte for byte on any machine; pairs with the same regime vector share one estimate. Criteria with equal declared weights are tied in the ranking and receive the same weight, the mean over the positions of the tie. The score of an alternative, carried to the parent criterion and, at the top group, the overall ranking value, is its mean probability of success,

```
p(a) = (1 / (m − 1)) · Σ over b ≠ a of P(a,b)          the LARGEST is best
R(i,a) = (1 / (m − 1)) · Σ over b ≠ a of sigma(i; a,b)   net dominance per criterion, in [−1, 1]
```

R(i,a) is the ranking value stored per criterion (what `sortra.txt` ranks criterion by criterion). Because only signs enter, RA is invariant under any monotone rescaling of the data: the standardization of section 2.2 can never change an RA result – a property the tests check. The declared type of every criterion is used at every Level.

9.3 Options

Option	Effect	Default
`--ra-weights ordinal|cardinal`	probabilities over the ordinal weight set, or the declared weights deciding each pair outright	ordinal
`--ra-samples <N>`	weight vectors drawn from the ordered simplex per group (at least 100)	10000
`--ra-seed <seed>`	seed of the drawings; printed in `regime.txt` so the run can be repeated exactly	1995

9.4 Output

`regime.txt` prints, per group: the declared weights and their importance ranks; the sampling line (count and seed); the pay-off matrix of standardized values; one net-dominance table per criterion (`RankValue` = R(i,a)); the **regime probability matrix** P(row beats column); and the scores p(a). `sortra.txt` ranks the alternatives by R(i,a) per criterion and by p(a) overall.

9.5 Status and open points

RA is **provisional**, marked as such wherever it appears. What is in place: the sampler is validated against the closed forms of chapter 8; the sampled probabilities are matched against an **exact analytic evaluation** – on the ordered simplex the weighted sum equals Σ over k of v(k) · c(k) with v uniform on the standard simplex and c(k) the cumulative mean of the ordered regime vector, whose positive-side probability has the closed simplex-splitting form Σ over the k with c(k) > 0 of c(k)^(n−1) / Π over j ≠ k of (c(k) − c(j)) – as well as cases solvable by hand (n = 2 certainties; the exact value 0.75 for sigma = (+1, −1, −1)); the cardinal mode is matched against a brute-force pairwise count; and a recorded seed reproduces every output byte. What is still required: the reproduction of a **published worked example** (Hinloopen, Nijkamp & Rietveld 1983 print one; the 1999 desalination tables are excluded for the reason given in section 8.5). A planned refinement, once the method is audited, is to compute the probabilities analytically by that same formula instead of sampling whenever the cumulative means are distinct.

9.6 Verification

- **Exact probabilities** (`tests/test_phase4.py`): five regime vectors with distinct cumulative means compared with the simplex-splitting formula within 0.02 at 40,000 samples; the hand cases P(+1,−1) = 1, P(−1,+1) = 0, P(0,0) = 1/2 for n = 2, vertex-certain vectors giving exactly 1 and 0, and the exact 0.75.
- **Hand example**: on the Nijkamp problem every pair is decided with certainty on the ordinal weight set (worked out by hand in the test); the scores are exactly 1, 1/2, 0.
- **Brute force**: cardinal mode reproduces a direct pairwise computation to twelve decimal places.
- **Determinism**: two runs with the same seed give byte-identical `regime.txt` and `sortra.txt`; a different seed moves a genuinely uncertain probability.

10. Weights beyond the file: equal, Best-Worst, entropy, CRITIC, MEREC - provisional

10.1 Purpose

The schemes of this chapter are not evaluation methods: they derive the **weights** of a group of criteria from the data – from how much each criterion differentiates the alternatives – instead of asking the decision maker, and any of the seven methods can then run with them. They are applied per group of siblings, and only to groups whose members are all **leaf criteria with numeric values**; a group containing aggregated or Saaty criteria keeps its given weights and a warning names it (the values of an aggregated criterion do not exist before the evaluation runs). The computed weights of a group sum to 1 and replace its normalized weights, so every method and every report sees them as `StWeight`.

10.2 The three schemes

Each scheme is implemented exactly as defined in its source, with its own data requirements (violations raise error 246):

Scheme	Formula (per group; i = criterion, j = alternative, m alternatives)	Needs
entropy (Hwang & Yoon 1981)	p(i,j) = x(i,j) / Σ over j of x(i,j); e(i) = −(1/ln m) Σ over j of p·ln p (0·ln 0 = 0); w(i) ∝ 1 − e(i). A criterion whose values differ much gets a large weight; a constant criterion gets 0.	raw values ≥ 0 with a positive sum per criterion
CRITIC (Diakoulaki, Mavrotas & Papayannakis 1995)	on values standardized to [0, 1] in the direction of the type: σ(i) = standard deviation; r(i,k) = linear correlation; w(i) ∝ σ(i) · Σ over k of (1 − r(i,k)). Important = varies much AND little correlated with the rest.	some dispersion; any sign of data
MEREC (Keshavarz-Ghorabaee et al. 2021)	n(i,j) = min/x for a maximizing, x/max for a minimizing criterion; S(j) = ln(1 + mean over i of |ln n(i,j)|); S′(j,k) leaves criterion k out; w(k) ∝ Σ over j of |S′(j,k) − S(j)|. Important = its removal changes the overall performances much.	raw values strictly > 0

10.3 Use

```
python mcewr.py weights study.mce                    # the three schemes side by side, per group
python mcewr.py study.mce --method wam --weights merec --normalize 1 -o results
mcewr.entropy_weights(matrix, types)                  # and critic_weights, merec_weights
```

The run records the scheme in `result["run"]["weight_scheme"]`; the report's Summary and Criteria sections show which weights were used.

10.4 Equal weights and the Best-Worst Method (September 2026)

Two judgement-based schemes were added in version 2.1 (`--weights equal`, `--weights bwm`; the Weights tab of the desktop and step 5 of the website). **Equal weights** give every evaluated criterion of a group the weight 1/n; unlike the objective schemes they apply to every group, aggregated criteria included (a Saaty group keeps its priorities, with a warning). The **Best-Worst Method** (Rezaei 2015) derives the weights of a group of n criteria from 2n − 3 pairwise comparisons: the decision maker names the best criterion B and the worst criterion W and gives the Best-to-Others vector a(B,j) and the Others-to-Worst vector a(j,W) on the 1–9 scale (a(B,B) = a(W,W) = 1; a(B,W) is shared by the two vectors and must agree). MCEWR implements the **linear model** of Rezaei (2016), which has a unique optimum:

```
minimize xi  subject to  |w(B) - a(B,j) w(j)| <= xi  and  |w(j) - a(j,W) w(W)| <= xi  for every
j,
                         sum of w(j) = 1,  w(j) >= 0
```

The program is solved by the package's own dense two-phase simplex (`mcewr/lp.py`, Bland's rule so that the degenerate BWM programs cannot cycle; exact fraction arithmetic on request); `mcewr/bwm.py` builds the constraints, validates the judgements (scale 1–9, best ≠ worst, one a(B,W)) and reports, with the weights: the optimum ξ* (the largest violation of a comparison by the weights, 0 for a fully consistent system, in which case w(j) = a(j,W) / Σ a(k,W)); the **consistency ratio** CR = ξ*/CI(n, a(B,W)), where CI is the largest ξ* any system with n criteria and that best-to-worst value can produce (the definition of Rezaei 2015 with the consistency index of the linear model derived by Ratandhara and Kumar 2025, eq. 22); and the **input-based ratio** of Liang, Brunelli and Rezaei (2020), max |a(B,j)·a(j,W) − a(B,W)| / (a(B,W)² − a(B,W)). The judgements are stored in the model (`am["bwm"]`, one entry per group keyed 'root' or the parent id) and travel in the `.mcewr` project file; `gui_logic.prune_bwm` drops the judgements of a group whose best or worst criterion is removed. A run with `--weights bwm` applies the judgements group by group; groups without judgements keep their given weights and a warning names them; a problem without any judgements is refused (error 245).

Verification (`tests/test_phase10.py`): the simplex reproduces the five worked examples of Ratandhara and Kumar (2025) – weights and ξ* to four decimals, consistency ratios to the rounding of their CI table – and their table of consistency indices; consistent systems give the closed form with ξ* = 0; on 150 random comparison systems the simplex agrees to 10^{-10} with an independent implementation of their analytical optimum (`bwm_analytical`, Theorem 2) and with the exact-arithmetic simplex, and the optimal weights satisfy every constraint at the reported ξ*; the validation refuses each malformed judgement; a BWM run with a consistent system equals a run with the closed-form weights entered as given weights; the judgements survive the project round trip and are pruned when their criterion disappears. Status: provisional, like the objective schemes – the formulation follows the sources and every published number available is reproduced; the author's own worked example is still to be signed off. Methods Explained, section 13.8, solves one system by hand.

10.5 Status and open points of the objective schemes

Provisional, like chapters 8 and 9. In place: an independent brute-force recomputation of each formula and its defining properties – weights sum to 1 and are invariant to permuting the alternatives; entropy and MEREC are invariant to rescaling a criterion and CRITIC to any affine change; a constant criterion gets weight 0 under entropy and CRITIC; perfectly correlated criteria carry no CRITIC information (error 246); reversing the sign together with the type is neutral. Still required: the reproduction of the worked example printed in each source article (Diakoulaki et al. 1995; Keshavarz-Ghorabaee et al. 2021), which needs the articles at hand.

11. TOPSIS - provisional

11.1 Purpose

The Technique for Order Preference by Similarity to Ideal Solution (Hwang & Yoon 1981) judges every alternative by two distances at once: how close it lies to the **ideal** – the best value of every criterion – and how far from the **anti-ideal** – the worst value of every criterion. The alternative that is at the same time nearest the first and farthest from the second wins. It is the natural companion of Compromise Programming (chapter 5), which measures the distance to the ideal only, and the two can rank differently – the classic teaching comparison. The largest value is best.

11.2 Equations

For a group of criteria, over the m evaluated alternatives, on the standardized values s(i,j) of section 2.2 (with `--normalize 0` these are the raw data and the method is the text-book TOPSIS):

```
r(i,j) = s(i,j) / sqrt( Σ over j of s(i,j)² )        vector normalization (a zero column keeps r
= 0)
y(i,j) = w(i) · r(i,j)                                weighting
A+(i)  = the best y(i,·) in the sense of the type     the ideal      (largest for max, smallest
for min)
A−(i)  = the worst y(i,·)                             the anti-ideal
D+(j)  = sqrt( Σ over i of (y(i,j) − A+(i))² )        D−(j) alike, to A−
C(j)   = D−(j) / ( D+(j) + D−(j) )                    the relative closeness; the LARGEST is
best
```

C(j) lies in [0, 1]: 1 for an alternative that coincides with the ideal, 0 for one that coincides with the anti-ideal (when every alternative is identical both distances vanish and C = 0.5 for all). C(j) is carried to the parent criterion and the C(j) of the top group is the overall ranking value; the declared type of every criterion is used at every Level, as in the other “larger is better” methods. The ranking value stored per criterion is the weighted normalized value with the sign of the type, R(i,j) = w(i) · r(i,j) · t(i), so `sorttop.txt` ranks each criterion by its own values.

11.3 Options

TOPSIS has no option of its own: the standardization (`--normalize`) and the weight normalization (`--raw-weights`) of section 2 apply, and the vector normalization of step 1 is always performed inside the method. Note that vector normalization is not invariant to shifting the data, so – unlike VIKOR – TOPSIS on standardized values is a different computation from TOPSIS on raw values; the report records which one was run.

11.4 Output

`topsis.txt` prints, per group, the weighted normalized matrix with the A+ and A− rows beneath it, then D+, D− and C for every alternative. `sorttop.txt` ranks the alternatives by R(i,j) per criterion and by C(j) overall, with the dominations of section 2.6.

11.5 Status and open points

Provisional. In place: an independent recursive implementation, written without the model machinery, reproduces the flat and hierarchical runs to ten decimal places; a hand-solved case (column norms 5, closeness 5/6) and the defining properties (C = 1 for the ideal, 0 for the anti-ideal, ties sharing rank 1) are matched exactly. Still required: the reproduction of the worked example printed by Hwang & Yoon (1981).

11.6 Verification

- **Reference implementation** (`tests/test_phase5.py`): `NIJKAMP.MCE`, `COMP_HAN.MCE`, `REDBLUFF.MCE` and `LEVEL4.MCE`, to ten decimal places.
- **Hand cases**: a two-criteria problem with exact fractions; an ideal and an anti-ideal alternative.
- **Round trips**: the Phase 2 suite converts every example to every input format and requires byte-identical `topsis.txt` and `sorttop.txt`.

12. VIKOR - provisional

12.1 Purpose

VIKOR (Opricovic 1998; Opricovic & Tzeng 2004) ranks by a balance between the **group utility** – how good an alternative is over all criteria together – and the **individual regret** – how bad it is on its single worst criterion – and then asks whether the first-ranked alternative is a *compromise solution*: clearly ahead of the second, and stable under a change of strategy. The smallest value is best. Its two ingredients are the L1 and the L∞ distances of Compromise Programming (chapter 5) to the best values of the data.

12.2 Equations

For a group of criteria, over the m evaluated alternatives, on the standardized values s(i,j):

```
f*(i), f−(i)  the best and the worst value of criterion i in the sense of the type
g(i,j) = w(i) · (f*(i) − s(i,j)) / (f*(i) − f−(i))     0 = the best, w(i) = the worst (a constant
criterion gives 0)
S(j)   = Σ over i of g(i,j)                             group utility     (the L1 distance)
R(j)   = max over i of g(i,j)                           individual regret (the L∞ distance)
Q(j)   = v · (S(j) − S*)/(S− − S*)  +  (1 − v) · (R(j) − R*)/(R− − R*)
         S* = min S, S− = max S, R* = min R, R− = max R;  the SMALLEST Q is best
```

v is the weight of the decision strategy: v = 1 ranks by the group utility alone, v = 0 by the individual regret alone, v = 0.5 (the default) by consensus. A term whose denominator vanishes is dropped and the other takes the whole weight; when both vanish every Q is 0. At the **top group** the two acceptance conditions of Opricovic & Tzeng are tested and written into the detail file and the run warnings: C1, *acceptable advantage*, Q(2nd) − Q(1st) ≥ DQ = 1/(m − 1); C2, *acceptable stability*, the first by Q is also first by S or by R. When C1 fails, every alternative within DQ of the first belongs to the set of compromise solutions; when only C2 fails, the first and the second do.

Q(j) is carried to the parent criterion and the Q(j) of the top group is the overall ranking value. The hierarchy follows the text-book rule of the minimize-direction methods (section 5.5, `--parents-only`): a leaf criterion keeps its declared type at any Level, an aggregated criterion is always “smaller is better”, because the value carried up is a regret – VIKOR has no 1995 mode to preserve. The ranking value stored per criterion is g(i,j), what `sortvik.txt` ranks, ascending. Because only normalized differences enter, any linear standardization with a positive slope leaves the result unchanged (formula 1 always; formulas 2 and 3 for positive data) – a property the tests check.

12.3 Options

Option	Effect	Default
`--vikor-v <v>`	the weight of the decision strategy, 0 ≤ v ≤ 1 (error 245 outside that range)	0.5

12.4 Output

`vikor.txt` prints, per group, f* and f− of every criterion, then S, R and Q of every alternative, and at the top group the two conditions with their verdicts and, when they fail, the set of compromise solutions (also issued as a warning). `sortvik.txt` ranks by g(i,j) per criterion and by Q(j) overall.

12.5 Status and open points

Provisional. In place: an independent recursive implementation reproduces flat and hierarchical runs with several values of v to ten decimal places; the Nijkamp example is computed by hand in the test (the g, S, R and Q of every alternative) and matched; v = 1 is shown to order the alternatives exactly by S and v = 0 exactly by R; the

standardization invariance holds to ten decimal places. Still required: the reproduction of the worked example of Opricovic & Tzeng (2004).

12.6 Verification

- **Reference implementation** (`tests/test_phase5.py`): four example files, v = 0.3, 0.5 and 0.7.
- **Hand example**: the Nijkamp problem, every intermediate quantity written out; the per-criterion ranking values matched to twelve decimal places.
- **Properties**: the v = 1 / v = 0 orderings; Q spanning exactly [0, 1]; invariance under standardizations 0, 1 and 3; ascending sort; the conditions printed.
- **Round trips**: byte-identical `vikor.txt` and `sortvik.txt` from every input format.

13. ELECTRE III - provisional

13.1 Purpose

ELECTRE III (Roy 1978; Roy & Bouyssou 1993; Figueira, Mousseau & Roy 2005) is the outranking method built on **pseudo-criteria**: for every criterion an indifference threshold q, a preference threshold p and, optionally, a veto threshold v (q ≤ p ≤ v) turn the crisp "better / worse" of ELECTRE II (chapter 6) into graded statements, and the outranking relation into a **credibility** in [0, 1] that two *distillations* turn into a ranking. The smallest final position is best. The p and q of every criterion are the thresholds already in the input file, read according to the declared PROMETHEE preference function so that the same numbers mean the same thing in both methods – a type-1 usual criterion is a true criterion (q = p = 0), a type-2 quasi-criterion has q only, types 3 and 6 p only, types 4 and 5 both – and the veto is the new per-criterion field `veto` of the project and spreadsheet formats (section 3.7 of the User Guide; 0 = no veto; the 1995 `.MCE` format has no place for it and the writer warns that it is dropped).

13.2 Equations

For a group of criteria, over the evaluated alternatives, with diff(i; a, b) = (s(i,b) − s(i,a)) · t(i) – how much b exceeds a on criterion i in the sense of the type – and the thresholds multiplied by the slope of the standardization map in force, so that they remain differences on the scale of the values the method sees (ELECTRE III has no 1995 mode and never applies the value-map of section 4.5 to its thresholds):

```
c(i; a,b) = 1                              if diff ≤ q(i)          concordance per criterion
          = 0                              if diff ≥ p(i)
          = (p(i) − diff) / (p(i) − q(i))  between
C(a,b)    = Σ w(i) · c(i; a,b) / Σ w(i)                             comprehensive concordance
d(i; a,b) = 0                              if no veto, or diff ≤ p(i)    discordance per
criterion
          = 1                              if diff ≥ v(i)
          = (diff − p(i)) / (v(i) − p(i))  between
σ(a,b)    = C(a,b) · Π over {i : d(i) > C(a,b)} of (1 − d(i)) / (1 − C(a,b))     credibility
```

σ equals C when no criterion discords more than the concordance, and a full veto (d = 1) makes σ = 0 whatever the concordance. With q = p = 0 and no veto, σ is the classic concordance index of chapter 6 (ties count as concordant) – an identity the tests check.

Distillations. With the discrimination threshold s(λ) = α + β λ (α = 0.30, β = −0.15 by default) the credibility matrix is read at decreasing levels: starting from λ = max σ, the next level is the largest σ below λ − s(λ), and at a level λ′ the alternative a **λ′-outranks** b when σ(a,b) > λ′ and σ(a,b) − σ(b,a) > s(σ(a,b)). The **qualification** of a is the number of alternatives it λ′-outranks minus the number that λ′-outrank it. The *descending* distillation extracts the alternatives of highest qualification – when several tie, the procedure is repeated inside that subset at lower levels until one remains or no level is left, and the survivors form one distillate – removes them and starts again; the *ascending* distillation does the same with the lowest qualification, from the bottom up. Each distillation is a complete preorder; the position of a is its distillate's place counted from the top, and

```
value(a) = ( position in the descending preorder + position in the ascending preorder ) / 2
           the SMALLEST is best;  the intersection of the two preorders is the final partial
preorder
           (P = ahead in both, I = equal in both, R = incomparable)
```

value(a) is carried to the parent criterion and the value(a) of the top group is the overall ranking value, as the median rank of ELECTRE II is; the final partial preorder of the top group goes into `result["electre3"]` and the reports. The hierarchy follows the text-book rule of the minimize-direction methods: a leaf keeps its declared type at any Level, an aggregated criterion is always "smaller is better" (its value is a position); its p, q and veto are the file's values for that criterion, in position units. The ranking value stored per criterion is that of ELECTRE II – the

weight of the criterion times the number of alternatives strictly better on it – what `sortele3.txt` ranks, ascending.

13.3 Options

Option	Effect	Default
`--electre3-alpha <α>`, `--electre3-beta <β>`	the discrimination threshold s(λ) = α + β λ; α > 0, β ≤ 0 and α + β > 0 are required so that s > 0 on [0, 1] (error 245)	0.30, −0.15
`veto` column of the file	per-criterion veto threshold, a difference of values; 0 = no veto. Error 247 when q > p (types 4, 5) or p > veto	0

13.4 Output

`electre3.txt` prints, per group, the pseudo-criteria as used (q, p, veto after scaling), the comprehensive concordance matrix, the credibility matrix, the two distillations, the positions and the final value of every alternative, and the final partial preorder with its incomparable pairs. `sortele3.txt` ranks by the per-criterion values and by value(a) overall.

13.5 Status and open points

Provisional. In place: an independent implementation of the whole chain – indices, credibility, both distillations with the inner recursion – written in a different style, reproduces the runs on five files under several standardizations; the identity with the classic concordance for true criteria; a full veto giving σ = 0; a one-criterion case and a two-criteria case with and without veto worked out by hand (every distillation step); error paths for inconsistent thresholds; the exact scaling of thresholds under standardization (crisp and graded files give the same ranking under standardizations 0, 1 and 3). Still required: the reproduction of a published worked example (for instance the one of Roy & Bouyssou 1993, or the site-selection example of Figueira, Mousseau & Roy 2005).

13.6 Verification

- **Reference implementation** (`tests/test_phase5.py`): `ELEC_HAN.MCE`, `PROM_EX.MCE` (two standardizations), `NIJKAMP.MCE`, `COMP_HAN.MCE`, positions and values to twelve decimal places.
- **Identities and hand cases**: classic concordance for true criteria; σ = 0 under a veto; the one-criterion case (values 1, 2, 3); the veto case (values 1.5, 1.0, 2.0 without and 2.0, 1.0, 2.5 with the veto); graded index values (c = 0.5 and d = 0.5 at mid-thresholds; σ = C·(1−d)/(1−C)).
- **Formats**: the veto survives the project and spreadsheet round trips; the `.MCE` writer warns; byte-identical `electre3.txt` / `sortele3.txt` from every input format (Phase 2 suite).

14. Comparing methods, sensitivity, rank reversal

14.1 Purpose

A ranking is a decision aid, not a verdict. Three questions belong to every application: do the methods agree on this problem, and where they do not, why; how much would the weights have to change before the decision changes; and does the method rank the alternatives against fixed reference points, or against each other, so that removing or adding an alternative can reorder the rest. Version 2.0 answers them with three analyses – `compare`, `sensitivity` and `reversal` – that run one or more methods many times on modified copies of the same model through the ordinary evaluation path (`evaluate_model()`, the function `run_mcewr()` itself calls after reading the file), so every number they report is a number an ordinary run would produce.

14.2 Comparison of methods

`compare` runs every requested method with the same settings and reports the rank of every alternative under every method (average ranks for ties), the ranking values, the **mean rank** of every alternative over the methods and the **consensus** ordering it defines, the number of first places, and the agreement between every pair of methods:

```
Spearman's rho   = the Pearson correlation of the two rank vectors (average ranks for ties)
Kendall's tau-b  = (P − Q) / sqrt( (n0 − n1) (n0 − n2) )      P concordant, Q discordant pairs,
                   n0 = m(m−1)/2, n1 / n2 = the pairs tied in the first / second ranking
```

A method that cannot be evaluated – ELECTRE II finding a cycle, CP without limits – is recorded and the comparison continues with the others (error 248 only when none can). The 2000 Red Bluff paper is such a comparison of the five 1995 methods.

14.3 Sensitivity of the ranking to the weights

`sensitivity` varies the weights of one group of sibling criteria – the main criteria by default, or the sub-criteria of a named parent – for one method, everything else as in the base run. **One-at-a-time variation**: the share x of one criterion in the total weight W of its group is swept from 0 to 1 in equal steps while the other members keep their proportions,

```
w'(i) = x · W        w'(k) = w(k) · (W − x·W) / (W − w(i))   for k ≠ i
```

and the alternative ranked first is recorded at every step. Where the winner changes between two steps the **critical share** is located by bisection (14 halvings). The result of each criterion is the list of intervals of its share with their winner, and the nearest critical share to the base share – the smallest change of that weight that changes the decision, in the sense of Triantaphyllou & Sánchez (1997) and Insua & French (1991); the criterion with the smallest relative change is the most sensitive. **Random perturbation**: every weight of the group is multiplied by an independent factor 1 + u, u uniform in [−p, +p], the method is re-run, and the **rank acceptability** of an alternative at a position is the share of the perturbations that put it there (the rank acceptability indices of SMAA-2, Lahdelma & Salminen 2001, restricted to a neighbourhood of the given weights); the **winner stability** is the share of perturbations in which the base winner stays first. The seed is recorded and the analysis repeats exactly. Groups whose weights are Saaty priorities are not varied (the judgements would be the thing to vary).

14.4 Sensitivity of the decision values, of the method parameters, and Monte Carlo robustness (version 2.1)

Three analyses added in September 2026 (`mcewr/robustness.py`; sub-commands `values`, `parameter`, `robustness`; the Sensitivity and Robustness tabs of the desktop, steps 9 and 10 of the website). They are loops over the ordinary evaluator (`analysis.evaluate_copy`) and touch no method.

- **Decision-value sensitivity** (`value_sensitivity`): for each cell of the chosen alternatives (the base winner by default, or all, at most 400 cells) and each numeric leaf criterion j, the value x is swept from x − R(j) to x + R(j), R(j) being the range of the criterion over the evaluated alternatives (|x| or 1 for a constant column), with an even number of steps so that x itself is a grid point; a bisection (14 steps) locates every change of winner; the nearest critical value to x is reported with its distance as a share of R(j) and its direction. The cell with the smallest share is the most sensitive value. Saaty-valued criteria are not varied.
- **Parameter sensitivity** (`parameter_sensitivity`): one numeric setting – `normalize_method` (any method) or a numeric option of the method registry (`vikor_v`, `cp_power`, `electre3_alpha`, `electre3_beta`, …) – is given a list of values (at most 200); for each the method is re-run with that value and the winner, the positions (ties included; the order is printed with '=' between tied alternatives), Spearman's rho of the positions against the base run and a same-positions flag are recorded; a value the engine refuses is reported with its message. The values at which the winner differs from the base winner are listed.
- **Monte Carlo robustness** (`data_robustness`): every numeric decision value x is replaced by x·(1 + u), and optionally every free weight w by w·(1 + u), with independent draws of u uniform in [−p, +p] or from the symmetric triangular distribution on [−p, +p] with mode 0 (`random.triangular`); N samples are evaluated and, for every alternative, the share of samples at each position (rank acceptability), the share in which it keeps its base position, and its mean rank are reported, together with the winner stability. The draws use the same expression as `weight_perturbation`, so that a weights-only uniform run with the same seed reproduces its table exactly. Saaty comparison values and Saaty priorities are not perturbed; a sample the engine refuses (for example a negative value under an objective weight scheme) is counted and the first message is reported; if every sample is refused the analysis raises error 245.

14.5 Rank-reversal diagnostics

A method suffers **rank reversal** when the relative order of two alternatives changes because a third, irrelevant one is removed or added (Belton & Gear 1983; Triantaphyllou 2001). It happens when the reference points of the method – the minimum and maximum of a min–max standardization, the ideal and anti-ideal of TOPSIS, the pairwise preferences of PROMETHEE, the normalization of AHP – are computed from the set of alternatives, so that changing the set changes the scale on which the others are measured. `reversal` runs three tests for every requested method: **remove** (each evaluated alternative is left out in turn), **duplicate** (a copy of each alternative is added in turn – Belton & Gear's test) and **dominated** (an alternative worse than every other on every leaf criterion, one per cent of the range beyond the worst, is added). A pair (a, b) counts as reversed when a is strictly ahead of b in the base run and strictly behind it in the trial; ties are not reversals. The two tests that add an alternative need numeric leaf values and are skipped for files with Saaty comparison values.

Compromise Programming with the **provided limits** – the computation the 2020 literature calls SPOTIS (section 5.7) – measures every alternative against reference points fixed in advance and therefore cannot reverse: the diagnostics report 0 for it on every test, on any file. On the Han River data with the min–max standardization the same run shows TOPSIS reversing in 7 of 9 removals and 8 of 9 duplications (51 pairs in all), PROMETHEE II and VIKOR once each, WAM and AHP not at all – the difference between fixed and data-dependent reference points made visible on the user's own problem, which is what Phase 6 of the modernization plan was to exhibit.

14.6 Use

```
python mcewr.py compare study.mce --methods all --normalize 1 -o results --report compare.pdf
python mcewr.py sensitivity study.mce --method wam --normalize 1 --steps 20 --perturb 0.10 --
samples 500
python mcewr.py reversal study.mce --methods cp,topsis,promethee --normalize 1 --report
reversal.docx

cmp  = mcewr.compare_methods("study.mce", "all", normalize_method=1)        # Python
sens = mcewr.weight_sensitivity("study.mce", "wam", group=None, steps=20, normalize_method=1)
pert = mcewr.weight_perturbation("study.mce", "wam", perturb=0.1, samples=500, seed=1995)
rev  = mcewr.rank_reversal("study.mce", "cp,topsis", tests="remove,duplicate,dominated")
mcewr.write_report_model(mcewr.compare_report(cmp), "compare.xlsx")
```

The three sub-commands take every run option of section 2 and of the methods (standardization, weights, `--cp-limits`, `--vikor-v`, …) and `--report` in the five formats of the User Guide, section 5.6; the text of the report is also printed on the console.

14.7 Verification

- **Rank statistics** (`tests/test_phase6.py`): Spearman's rho and Kendall's tau-b against hand-computed values, including ties in one and in both rankings; the inversion count.
- **Evaluation path**: `evaluate_model()` on a loaded model reproduces `run_mcewr()` on the file – ranking, warnings and output files byte for byte; the differential test of chapter 15 is unchanged.
- **Comparison**: every method's values equal its individual run; the agreement matrices are symmetric with 1 on the diagonal; a failing method is recorded.
- **Sensitivity**: on the Nijkamp example under WAM the critical shares are known in closed form – the winner changes when criterion 1 reaches 7/16 of the weight and when criterion 3 falls to 1/3, criterion 2 at 1/7 – and the bisection finds them within $2 \cdot 10^{-4}$; the perturbation is reproducible from its seed and its acceptabilities sum to 1.
- **Rank reversal**: CP with the provided limits shows 0 reversals on every test; TOPSIS shows the reversals above; Saaty files skip the tests that add an alternative.
- **Decision values** (`tests/test_phase11.py`): on the Nijkamp example under WAM the critical value of alternative 1 on criterion 3 is 70 (at 70 the standardized column is 1, 0.5, 0 and the two weighted sums are equal), found within 10^{-3}; every critical value of every cell is confirmed by two direct runs just below and just above it; the base value lies on the grid; the cell cap is enforced.
- **Parameters**: every row equals a direct run with that value; the VIKOR sweep changes the winner at v = 1 on the Nijkamp example and refuses v = 1.5 with the option's message; the standardization sweep gives the winners 2, 1, 1, 1 for the codes 0–3.
- **Robustness**: with the weights only, uniform u and the same seed, the rank acceptability equals `weight_perturbation` exactly; acceptability rows sum to 1; the mean ranks sum to n(n + 1)/2; the triangular distribution and the combined perturbation run; a base run the engine refuses raises, an impossible distribution or an all-zero perturbation is refused; the three reports render in PDF, HTML and text.

15. Verification summary

Three independent lines of evidence support every audited method. The **dissertation** (Mahmoud 1995) prints, in Appendix 2, the input and the output of one verification example per method, computed by the 1995 program and checked by hand at the time; the Python edition reproduces every number. The **legacy output files** are nine files written by the original C++ executable – five on the Red Bluff problem, three on the Han River problem and one on `EG680_EX.MCE`, covering every method, both CP powers, both AHP minimum methods, standardized and unstandardized data; the Python edition reproduces them line by line, the only differences being the last printed digit of numbers computed in 32-bit arithmetic. The **independent checks** (`tools/independent_check.py`) recompute the methods from the raw data with text-book formulas and a general eigen-solver, without using the program, and compare the results with the program's output. On top of these, the **differential test** of version 2.0 runs the package and the frozen release 1.1.0 side by side on every example file, every method, every standardization and every option, and the **unit tests** cover the reading of files, the GIS import and export, the error reporting and the text-book switches.

Method	Dissertation (Appendix 2)	Legacy files reproduced	Independent checks	Case studies
WAM	2.1–2.4 (eq. 3.5, 3.6a, 3.6b)	WAM.TXT, SORTWAM.TXT	5	Red Bluff; Egypt Table 5.4
PROMETHEE II	2.5–2.6 (Brans example)	PROMETHE.TXT, SORTPROM.TXT	19 (3 + 16 text-book)	Red Bluff; Egypt Table 5.4
Compromise Programming	2.7–2.10 (Han River, s = 1, 2)	COMPRO.TXT, SORTCOMP.TXT	14 (6 + 8 text-book)	Red Bluff; Egypt Table 5.4
ELECTRE II	2.13–2.14 (five alternatives)	ELECTREI.TXT, SORTELE2.TXT	8 (5 + 3 text-book)	Red Bluff; Egypt Table 5.4
AHP	2.11–2.12 (`AHP_EXM3.MCE`)	AHP.TXT	8	Red Bluff; Egypt Table 5.4
Ranking, ties, domination	every example	all sort files	7	–
Total	**7 runs, 5 files**	**9 files**	**61**	**5,980 paired runs; 52 unit tests**

The analyses of chapter 14 are not methods and carry no status; their evidence is listed in section 14.6 – closed-form critical shares of the Nijkamp example, the zero reversals of CP with provided limits, hand-computed rank statistics – and they are exercised by `tests/test_phase6.py`.

Two limits of this evidence should be stated. First, the text-book options are verified by the independent checks and by internal identities (for instance that the two hierarchy rules coincide on balanced trees), not by legacy files – no 1995 output exists for them, by definition. Second, the rankings printed for two further methods in the 1999 desalination paper, Regime Analysis and Expected Value, could not be reproduced from the tables printed in that paper; when these methods are added to MCEWR 2.0 they will be verified against the primary literature and a published worked example, not against the 1999 tables. The same rule – a published worked example and an independent check before release – applies to every method added later. Chapters 8 to 10 document the additions of version 2.0 – Expected Value, Regime Analysis and the objective weight schemes – which for exactly this reason carry the status PROVISIONAL, as do TOPSIS, VIKOR and ELECTRE III of chapters 11 to 13; the table below records what evidence is in place for them and what is still required.

Provisional (version 2.0)	Evidence in place	Still required
Expected Value (ch. 8)	closed-form ROC and order-statistic checks of the sampler;	a published worked example (Nijkamp,

Provisional (version 2.0)	Evidence in place	Still required
	the Nijkamp example computed by hand (ten decimals); an independent recursive implementation on the hierarchical examples; the identity cardinal/cardinal = WAM; standardization invariance	Rietveld & Voogd 1990); the author's confirmation of the 1999 concave-scores variant
Regime Analysis (ch. 9)	sampled probabilities against the exact simplex-splitting formula and hand-solvable cases; cardinal mode against brute force; byte-for-byte reproducibility with the recorded seed; standardization invariance	a published worked example (Hinloopen, Nijkamp & Rietveld 1983)
Objective weights (ch. 10)	brute-force recomputation of each formula; the defining invariance and degeneracy properties	the worked example of each source article
TOPSIS (ch. 11)	an independent recursive implementation on flat and hierarchical files (ten decimals); hand cases with exact fractions; the ideal / anti-ideal properties	the worked example of Hwang & Yoon (1981)
VIKOR (ch. 12)	an independent recursive implementation for several v; the Nijkamp example by hand; v = 1 / v = 0 orderings; standardization invariance	the worked example of Opricovic & Tzeng (2004)
ELECTRE III (ch. 13)	an independent implementation of indices, credibility and both distillations on five files; the classic-concordance identity; veto = 0 credibility; hand-worked one- and two-criteria cases; exact threshold scaling	a published worked example (Roy & Bouyssou 1993; Figueira, Mousseau & Roy 2005)
PROMETHEE I (§4.7)	the relation derived independently from the flows on the Brans example; default output byte-identical	none – a reading of the audited PROMETHEE II flows
SPOTIS (§5.7)	byte-identical to CP with provided limits and s = 1	none – an identity, not a new computation

16. References

The entries follow the reference lists of the dissertation and of the Red Bluff article, which are the sources the model was built from.

Barron, F.H., Barrett, B.E., 1996. Decision quality using ranked attribute weights. Management Science 42(11), 1515–1523.

Belton, V., Gear, T., 1983. On a short-coming of Saaty's method of analytic hierarchies. Omega 11(3), 228–230.

Benayoun, R., Roy, B., Sussman, B., 1966. ELECTRE: une méthode pour guider le choix en présence de points de vue multiples. Note de Travail 49, SEMA (Metra International), Direction Scientifique, Paris.

Brans, J.P., Mareschal, B., Vincke, Ph., 1984. PROMETHEE: a new family of outranking methods in MCDM. In: Brans, J.P. (Ed.), Operational Research '84. North-Holland, Amsterdam, 477–490.

Brans, J.P., Mareschal, B., Vincke, Ph., 1985. The preference ranking organization method: the PROMETHEE for MCDM. Management Science 31(6), 647–656.

Brans, J.P., Vincke, Ph., Mareschal, B., 1986. How to select and how to rank projects: the PROMETHEE method. European Journal of Operational Research 24, 228–238.

Diakoulaki, D., Mavrotas, G., Papayannakis, L., 1995. Determining objective weights in multiple criteria problems: the CRITIC method. Computers & Operations Research 22(7), 763–770.

Dezert, J., Tchamova, A., Han, D., Tacnet, J.-M., 2020. The SPOTIS rank reversal free method for multi-criteria decision-making support. Proceedings of the 23rd International Conference on Information Fusion (FUSION 2020), IEEE.

Duckstein, L., Opricovic, S., 1980. Multi-objective optimization in river basin development. Water Resources Research 16(1), 14–20.

Einhorn, H.J., McCoach, W., 1977. A simple multi-attribute utility procedure for evaluation. Behavioral Science 22, 270–282.

Figueira, J., Mousseau, V., Roy, B., 2005. ELECTRE methods. In: Figueira, J., Greco, S., Ehrgott, M. (eds), Multiple Criteria Decision Analysis: State of the Art Surveys. Springer, New York, 133–153.

Goicoechea, A., Hansen, D.R., Duckstein, L., 1982. Multi-Objective Decision Analysis with Engineering and Business Applications. Wiley, New York.

Hinloopen, E., Nijkamp, P., 1983. Regime-methode voor ordinale multicriteria-analyses. Kwantitatieve Methoden 4(10), 61–78.

Hinloopen, E., Nijkamp, P., Rietveld, P., 1983. Qualitative discrete multiple criteria choice models in regional planning. Regional Science and Urban Economics 13(1), 77–102.

Hwang, C.L., Yoon, K., 1981. Multiple Attribute Decision Making: Methods and Applications. Springer, Berlin.

Harker, P.T., 1988. The Art and Science of Decision Making: The Analytic Hierarchy Process. Working Paper 88-06-03, Decision Sciences Department, The Wharton School, University of Pennsylvania, Philadelphia.

Insua, D.R., French, S., 1991. A framework for sensitivity analysis in discrete multi-objective decision-making. European Journal of Operational Research 54(2), 176–190.

Kendall, M.G., 1938. A new measure of rank correlation. Biometrika 30(1–2), 81–93.

Keshavarz-Ghorabaee, M., Amiri, M., Zavadskas, E.K., Turskis, Z., Antucheviciene, J., 2021. Determination of objective weights using a new method based on the removal effects of criteria (MEREC). Symmetry 13(4), 525.

Lahdelma, R., Salminen, P., 2001. SMAA-2: stochastic multicriteria acceptability analysis for group decision making. Operations Research 49(3), 444–454.

Mahmoud, M.R., December 1995. Multi-criteria Location and Sizing of Water Resources Projects by GIS and Dynamic Programming. Ph.D. dissertation, Water Resources Planning and Management, Civil Engineering, Colorado State University, Fort Collins, Colorado, USA.

Mahmoud, M.R., April 1999. Integrated Mathematical Models for Evaluating Desalination Technology. Water Science Journal, National Water Research Center, Cairo, Egypt, Issue 25, 21–32.

Mahmoud, M.R., Fahmy, H., Labadie, J.W., March 2002. Multicriteria Siting and Sizing of Desalination Facilities with Geographic Information System. Journal of Water Resources Planning and Management, ASCE, 128(2), 113–120. https://doi.org/10.1061/(ASCE)0733-9496(2002)128:2(113)

Mahmoud, M.R., Garcia, L.A., 1995. MultiCriteria Decision Support System Implementation and Application. First International Conference on Multiple Objective Decision Support Systems for Land, Water, and Environmental Management: Concepts, Approaches, and Applications, Honolulu, Hawaii, July 23–29, 1995.

Mahmoud, M.R., Garcia, L.A., May 2000. Comparison of Different Multi-Criteria Evaluation Methods for the Red Bluff Diversion Dam. Environmental Modelling & Software, Elsevier, 15(5), 471–478. https://doi.org/10.1016/S1364-8152(00)00025-6

Nijkamp, P., Voogd, H., Rietveld, P., 1990. Multicriteria Evaluation in Physical Planning. North-Holland, Amsterdam.

Opricovic, S., 1998. Multicriteria Optimization of Civil Engineering Systems. Faculty of Civil Engineering, Belgrade.

Opricovic, S., Tzeng, G.-H., 2004. Compromise solution by MCDM methods: a comparative analysis of VIKOR and TOPSIS. European Journal of Operational Research 156(2), 445–455.

Rietveld, P., 1980. Multiple Objective Decision Methods and Regional Planning. North-Holland, Amsterdam.

Roy, B., 1978. ELECTRE III: un algorithme de classements fondé sur une représentation floue des préférences en présence de critères multiples. Cahiers du CERO 20(1), 3–24.

Roy, B., Bouyssou, D., 1993. Aide multicritère à la décision: méthodes et cas. Economica, Paris.

Roy, B., 1971. Problems and methods with multiple objective functions. Mathematical Programming 1(2), 239–266.

Roy, B., Bertier, P., 1971. La méthode ELECTRE II: une méthode de classement en présence de critères multiples. Note de Travail 142, Direction Scientifique, Groupe Metra, Paris (published in: Ross, M. (Ed.), OR ’72, North-Holland, 1973).

Saaty, T.L., 1980. The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation. McGraw-Hill, New York.

Saaty, T.L., 1982. Decision Making for Leaders. Lifetime Learning Publications, Belmont, CA.

Spearman, C., 1904. The proof and measurement of association between two things. American Journal of Psychology 15(1), 72–101.

Triantaphyllou, E., 2001. Two new cases of rank reversals when the AHP and some of its additive variants are used that do not occur with the multiplicative AHP. Journal of Multi-Criteria Decision Analysis 10(1), 11–25.

Triantaphyllou, E., Sánchez, A., 1997. A sensitivity analysis approach for some deterministic multi-criteria decision-making methods. Decision Sciences 28(1), 151–194.

Zeleny, M., 1973. Compromise programming. In: Cochrane, J.L., Zeleny, M. (Eds.), Multiple Criteria Decision Making. University of South Carolina Press, Columbia, SC.

Zeleny, M., 1982. Multiple Criteria Decision Making. McGraw-Hill, New York.