Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών

Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων

Ανάπτυξη Ολοκληρωμένου Υπολογιστικού Εργαλείου για την Αξιολόγηση Τεχνολογιών Αποθήκευσης Ενέργειας με χρήση Πολυκριτήριας Ανάλυσης

Διπλωματική Εργασία

Θεόδωρος Τσαλίδης

Επιβλέπων: Δημήτριος Ασκούνης

Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2023

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών

Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων

Ανάπτυξη Ολοκληρωμένου Υπολογιστικού Εργαλείου για την Αξιολόγηση Τεχνολογιών Αποθήκευσης Ενέργειας με χρήση Πολυκριτήριας Ανάλυσης

Διπλωματική Εργασία

Θεόδωρος Τσαλίδης

Επιβλέπων: Δημήτριος Ασκούνης

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19^{η} Οκτωβρίου 2023

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
.....	.....	.....
Δημήτριος Ασκούνης	Χρυσόστομος Δούκας	Ιωάννης Ψαρράς
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2023

(Υπογραφή)

……

Θεόδωρος Τσαλίδης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η απόφαση να επιλεγεί η κατάλληλη λύση αποθήκευσης ενέργειας είναι πολύπλοκη λόγω της παρουσίας πολλαπλών αντιφατικών κριτηρίων. Πρέπει να ληφθούν υπόψη κριτήρια όπως η ενεργειακή και η οικονομική αποδοτικότητα, η βιωσιμότητα και η προσαρμοστικότητα, τονίζοντας την ανάγκη για προηγμένη υποστήριξη αποφάσεων. Οι μέθοδοι Πολυκριτήριας Ανάλυσης Αποφάσεων (ΠΑΑ), όπως είναι οι VIKOR και TOPSIS, προσφέρουν μια συστηματική και διαφανή προσέγγιση για την αξιολόγηση διάφορων εναλλακτικών.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη ενός ολοκληρωμένου υπολογιστικού εργαλείου που χρησιμοποιεί τεχνικές ΠΑΑ, επιτρέποντας στους χρήστες να εισάγουν δεδομένα, να προσαρμόσουν κριτήρια και να λάβουν την κατάταξη των εναλλακτικών. Ενσωματώνοντας σύγχρονα πλαίσια ανάπτυξης δικτυακών εφαρμογών, όπως η WebAssembly, και αξιόπιστες τεχνολογίες, όπως η Python, το εργαλείο εξοπλίζει τους αποφασίζοντες στον τομέα της αποθήκευσης ενέργειας, αλλά και αλλού, με τα μέσα για λήψη ενημερωμένων και βιώσιμων αποφάσεων.

Λέξεις Κλειδιά: πολυκριτήρια ανάλυση, λήψη αποφάσεων, ανάλυση αποφάσεων, VIKOR, TOPSIS, υπολογιστικό εργαλείο, ανάπτυξη λογισμικού, αποθήκευση ενέργειας, ανανεώσιμη ενέργεια

Abstract

The decision to choose the appropriate energy storage solution is complex due to the presence of multiple conflicting criteria. Criteria such as energy and economic efficiency, sustainability, and adaptability must be considered, highlighting the need for advanced decision support. Multicriteria Decision Making (MCDM) methods, such as VIKOR and TOPSIS, offer a systematic and transparent approach for evaluating alternatives.

The purpose of this thesis is to develop an integrated computational tool that utilizes MCDM techniques, allowing users to input data, customize criteria, and receive rankings of alternatives. By incorporating modern web application development frameworks, like WebAssembly, and reliable technologies, like Python, the tool equips decision-makers in the field of energy storage, and beyond, with the means to make informed and sustainable decisions.

Keywords: multicriteria analysis, decision making, decision analysis, VIKOR, TOPSIS, computational tool, software development, energy storage, renewable energy

Ευχαριστώ τις τρεις γυναίκες της ζωής μου,

τη μητέρα, την αδερφή, και την κοπέλα μου,

για τη συμπαράσταση που μου έχουν προσφέρει.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Τμημάτων Κώδικα

Κεφάλαιο 1

Εισαγωγή

Σε μια εποχή που χαρακτηρίζεται από διαρκώς αυξανόμενη πολυπλοκότητα και αβεβαιότητα, οι αποφασίζοντες από διάφορους τομείς αντιμετωπίζουν μια αντίστοιχα αυξανόμενη πρόκληση στο να καταλήγουν σε βέλτιστες επιλογές. Είτε πρόκειται για τον τομέα της επιχειρηματικότητας, της μηχανικής, της διαχείρισης του περιβάλλοντος ή της υγείας, η λήψη αποφάσεων που βασίζεται σε πολλαπλά, συχνά αντιφατικά, κριτήρια έχει γίνει απολύτως κρίσιμη. Η ανάγκη να πλοηγηθούμε σε αυτό το πολύπλοκο τοπίο έχει ενθαρρύνει την ανάπτυξη προηγμένων εργαλείων υποστήριξης της λήψης αποφάσεων και μεθοδολογιών, με την Πολυκριτήρια Ανάλυση Αποφάσεων (ΠΑΑ) να προσφέρει δομημένες μεθοδολογίες για την αντιμετώπιση της πολυπλοκότητας που απορρέει από πραγματικά προβλήματα λήψης αποφάσεων.

Οι τεχνικές ΠΑΑ προσφέρουν ένα δομημένο πλαίσιο για την αξιολόγηση και την κατάταξη των εναλλακτικών βάσει πολλαπλών κριτηρίων, μια διαδικασία που θέτει προς αμφισβήτηση τον παραδοσιακό τρόπο μονοκριτηριακής λήψης αποφάσεων. Αυτές οι μεθοδολογίες αναγνωρίζουν ότι οι αποφάσεις στον πραγματικό κόσμο σπάνια λαμβάνονται απομονωμένα, αλλά επηρεάζονται από μια πληθώρα παραγόντων, όπως το κόστος, ο χρόνος, οι περιβαλλοντικές και κοινωνικές συνθήκες καθώς και αντίκτυπος των εναλλακτικών σε αυτές. Ως αποτέλεσμα, η ΠΑΑ επιτρέπει στους αποφασίζοντες να αξιολογούν αυτούς τους παράγοντες ταυτόχρονα, προσφέροντας μια πιο συνολική και ολιστική προσέγγιση.

Εντός του πλαισίου της ΠΑΑ έχει αναπτυχθεί ένα πλήθος από μεθόδους, που επιτρέπει στους αποφασίζοντες να επιλέγουν την πλέον κατάλληλη προσέγγιση για την αντιμετώπιση του πολύπλοκου δικτύου των κριτηρίων, βάσει των πλεονεκτημάτων και τους τομείς εφαρμογής της καθεμιάς. Δύο προεξέχουσες μέθοδοι ΠΑΑ που έχουν αποκτήσει ευρεία αναγνώριση και επιδοκιμασία στην κοινότητα της επιστήμης της λήψης αποφάσεων είναι η μέθοδος VIKOR και η μέθοδος TOPSIS. Τόσο η VIKOR όσο και η TOPSIS αναγνωρίζονται για την αποτελεσματικότητά τους στην αντιμετώπιση προβλημάτων λήψης αποφάσεων που χαρακτηρίζονται από πολλαπλά κριτήρια και στην καθοδήγηση των αποφασιζόντων προς βέλτιστες λύσεις.

Στο πλαίσιο της παρούσας διπλωματικής εργασίας γίνεται αρχικά μια επισκόπηση ορισμένων εκ των μεθόδων ΠΑΑ, συγκεκριμένα των VIKOR και TOPSIS καθώς και κάποιων επεκτάσεων της πρώτης. Στη συνέχεια, παρουσιάζεται το υπολογιστικό εργαλείο που τις ενσωματώνει και ο τρόπος χρήσης του, οι τεχνολογίες (πχ. γλώσσες προγραμματισμού, frameworks) και οι προ-

γραμματιστικές τεχνικές που έχουν χρησιμοποιηθεί για την ανάπτυξή του. Τέλος, γίνεται πιλοτική χρήση του εργαλείου πάνω σε δεδομένα από συγκεκριμένη περίπτωση που αφορά στον τομέα της αποθήκευσης ενέργειας.

Κεφάλαιο 2

Μεθοδολογίες Πολυκριτήριας Ανάλυσης

Στα προβλήματα Πολυκριτήριας Ανάλυσης Αποφάσεων (ΠΑΑ), συνήθως λαμβάνουμε υπόψη ένα πεπερασμένο διακριτό σύνολο εναλλακτικών $A = \{A_1, A_2, \ldots, A_m\}$, καθεμία από τις οποίες αξιολογείται βάσει πολλαπλών κριτηρίων $C = \{C_1, C_2, \ldots, C_n\}$. Ο Πίνακας 2.1 είναι ένας πίνακας απόφασης όπου απεικονίζονται οι διαθέσιμες εναλλακτικές, τα κριτήρια και η επίδοση f_{ij} της κάθε εναλλακτικής i έναντι του αντίστοιχου κριτηρίου j. [2] Επιπλέον ορίζονται τα βάρη των κριτηρίων $W = \{w_1, w_2, \ldots, w_n\}$ για τα οποία ισχύει:

$$\sum_{j=1}^{n} w_j = 1 \quad \text{και} \quad w_j \geq 0 \tag{2.1}$$

Πίνακας 2.1: Πίνακας απόφασης

	C_1	C_2	$\cdots$	C_n
A_1	f_{11}	f_{12}	$\cdots$	f_{1n}
A_2	f_{21}	f_{22}	$\cdots$	f_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
A_m	f_{m1}	f_{m2}	$\cdots$	f_{mn}

Στο κεφάλαιο αυτό γίνεται η θεωρητική παρουσίαση των μεθόδων πολυκριτήριας ανάλυσης αποφάσεων που έχουν υλοποιηθεί στο υπολογιστικό εργαλείο. Αρχικά παρουσιάζεται η μέθοδος VIKOR, στη συνέχεια κάποιες επεκτάσεις αυτής και τέλος η μέθοδος TOPSIS.

2.1. Η κλασική μέθοδος VIKOR

Η μέθοδος VIKOR[1] αναπτύχθηκε για την πολυκριτηριακή βελτιστοποίηση πολύπλοκων συστημάτων [3]. Επικεντρώνεται στην κατάταξη και την επιλογή από ένα σύνολο διαθέσιμων εναλλακτικών παρουσία αντιφατικών κριτηρίων, προτείνοντας μια συμβιβαστική λύση [4]. Η μέθοδος VIKOR συχνά συγκρίνεται με τη μέθοδο TOPSIS (Technique for Order Performance by Similarity to Ideal Solution), αν και κάθε μία χρησιμοποιεί τη δική της συναθροιστική συνάρτηση, καθώς και διαφορετική μέθοδο κανονικοποίησης. Το βέλτιστο σημείο κατά την TOPSIS πρέπει να έχει τη μικρότερη απόσταση από τη θετική ιδεατή λύση και τη μεγαλύτερη απόσταση από την αρνητική ιδεατή λύση. Επομένως, είναι κατάλληλη για έναν επιφυλακτικό αποφασίζοντα που μπορεί να προτιμά μια απόφαση που όχι μόνο προσφέρει το μεγαλύτερο δυνατό κέρδος, αλλά και αποφεύγει το μεγαλύτερο δυνατόν ρίσκο [5]. Το κύριο πλεονέκτημα της μεθόδου VIKOR είναι ότι μπορεί να αντισταθμίσει την μεγιστοποίηση της επίδοσης στο σύνολο των κριτηρίων με την ελαχιστοποίηση του βαθμού δυσαρέσκειας στα μεμονωμένα κριτήρια [4] [6] [7].

Βασιζόμενη στην L^p - μετρική (2.2), η μέθοδος VIKOR κάνει χρήση και των δύο ακραίων περιπτώσεων $p = 1$ και $p = \infty$ σταθμίζοντας σύμφωνα με το βάρος που επιλέγει ο αποφασίζοντας. Η L_i^1 - μετρική (ως S_i, εξ. 2.5) ορίζει την απόσταση από τη μέγιστη χρησιμότητα ως προς όλα τα κριτήρια, ενώ η L_i^∞ - μετρική (ως R_i, εξ. 2.6) το μέγιστο βαθμό δυσαρέσκειας (regret) απέναντι σε κάθε κριτήριο ξεχωριστά.

$$L_i^p = \left(\sum_{j=1}^{n} \left| \frac{f_j^* - f_{ij}}{f_j^* - f_j^-} \right|^p \right)^{\frac{1}{p}} , \quad 1 \le p \le \infty , \quad i = 1, 2, \ldots, m \tag{2.2}$$

Η μέθοδος VIKOR αποτελείται από τα ακόλουθα 5 στάδια:

1. Προσδιορισμός της καλύτερης f_j^* και της χειρότερης f_j^- επίδοσης, αντίστοιχα, για κάθε κριτήριο $j = 1, \ldots, n$:

   - Αν το j-στο κριτήριο εκφράζει όφελος:

   $$f_j^* = \max_i f_{ij} , \quad f_j^- = \min_i f_{ij} \tag{2.3}$$

   - Αν το j-στο κριτήριο εκφράζει κόστος:

   $$f_j^* = \min_i f_{ij} , \quad f_j^- = \max_i f_{ij} \tag{2.4}$$

2. Υπολογισμός των τιμών S_i και R_i, για κάθε εναλλακτική $i = 1, \ldots, m$:

$$S_i = \sum_{j=1}^{n} w_j \left| \frac{f_j^* - f_{ij}}{f_j^* - f_j^-} \right| , \quad i = 1, 2, \ldots, m \tag{2.5}$$

$$R_i = \max_j w_j \left| \frac{f_j^* - f_{ij}}{f_j^* - f_j^-} \right| , \quad i = 1, 2, \ldots, m \tag{2.6}$$

[1] Το όνομα προέρχεται από τα αρχικά γράμματα της σέρβικης φράσης «VIseKriterijumska Optimizacija I Kompromisno Resenje» που σημαίνει «Πολυκριτηριακή Βελτιστοποίηση και Λύση Συμβιβασμού»

3. Υπολογισμός της μετρικής Q_i, για κάθε εναλλακτική $i = 1, \ldots, m$:

$$Q_i = v\frac{S_i - S^*}{S^- - S^*} + (1 - v)\frac{R_i - R^*}{R^- - R^*} \tag{2.7}$$

όπου τα S^*, S^-, R^*, R^- ορίζονται ως:

$$S^* = \min_i S_i \ , \quad S^- = \max_i S_i \tag{2.8}$$

$$R^* = \min_i R_i \ , \quad R^- = \max_i R_i \tag{2.9}$$

Η παράμετρος $v \in [0, 1]$ είναι το βάρος που θέτει ο αποφασίζοντας για την συνολική χρησιμότητα ενώ το $(1 - v)$ είναι το βάρος του βαθμού δυσαρέσκειας. Ένας αποφασίζοντας με ουδέτερη στάση απέναντι στις δύο στρατηγικές συνήθως επιλέγει την τιμή $v = 0.5$

4. Ταξινόμηση των εναλλακτικών βάσει των μετρικών S_i, R_i και Q_i. Τότε, προκύπτουν τρεις λίστες κατάταξης όπου μικρότερη τιμή σημαίνει καλύτερη εναλλακτική.

5. Επιλογή της εναλλακτικής A' που έχει τη μικρότερη τιμή Q ως προτεινόμενη λύση συμβιβασμού, εάν ικανοποιεί τις παρακάτω δύο συνθήκες:

   - Συνθήκη C1 - συγκριτικό πλεονέκτημα:

   $$Q\left(A''\right) - Q\left(A'\right) \geq DQ \tag{2.10}$$

   όπου A'' είναι η αμέσως επόμενη καλύτερη εναλλακτική κατά Q και για πλήθος εναλλακτικών m:

   $$DQ = \frac{1}{m - 1}$$

   - Συνθήκη C2 - αποδεκτή ευστάθεια στη λήψη απόφασης: Η εναλλακτική A' πρέπει επιπλέον να έχει την καλύτερη επίδοση (ελάχιστη τιμή) ως προς S ή/και R

   Εάν κάποια από τις συνθήκες δεν ικανοποιείται, προτείνεται ένα σύνολο λύσεων συμβιβασμού, ανάλογα την περίπτωση:

   - Αν δεν ικανοποιείται μόνο η συνθήκη $C2$, τότε προτείνεται ως λύση συμβιβασμού το σύνολο

   $$\{A', A''\}$$

   - Αν δεν ικανοποιείται η συνθήκη $C1$, τότε προτείνεται ως λύση συμβιβασμού το σύνολο

   $$\{A', A'', \ldots, A^{(M)}\}$$

   όπου $A^{(M)}$ προσδιορίζεται από τη σχέση

   $$Q\left(A^{(M)}\right) - Q\left(A'\right) < DQ$$

   για το μέγιστο M για το οποίο ισχύει η παραπάνω ανισότητα. Οι εναλλακτικές αυτές αποτελούν τις M καλύτερες (πρώτες στην λίστα κατάταξης Q).

2.2. Επεκτεταμένη μέθοδος VIKOR με επιδόσεις αριθμούς-διαστήματα

Στις κλασικές μεθόδους ΠΑΑ, οι βαθμολογίες και τα βάρη των κριτηρίων είναι γνωστά με ακρίβεια, ενώ στον πραγματικό κόσμο, σε ένα ανακριβές και αβέβαιο περιβάλλον, δεν είναι ρεαλιστικό ένας αποφασίζων ή ένας εμπειρογνώμονας να έχει τόσο ακριβή γνώση. Για παράδειγμα, η ανθρώπινη κρίση συμπεριλαμβανομένων των προτιμήσεων είναι συχνά ασαφής και ο αποφασίζων δεν μπορεί να εκτιμήσει την προτίμησή του με ακριβείς αριθμητικές τιμές. Σε αυτές τις περιπτώσεις, ο προσδιορισμός της ακριβούς τιμής των χαρακτηριστικών είναι δύσκολος έως αδύνατος.

Έτσι, για την περιγραφή και την αντιμετώπιση ανακριβών και αβέβαιων στοιχείων που υπάρχουν σε ένα πρόβλημα απόφασης, χρησιμοποιούνται συχνά ασαφείς και στοχαστικές προσεγγίσεις. Στη βιβλιογραφία, στα έργα της ασαφούς λήψης αποφάσεων [8] [9] [10], οι ασαφείς παράμετροι θεωρούνται ότι έχουν γνωστές συναρτήσεις συμμετοχής, και στη στοχαστική λήψη αποφάσεων [11] [12] [13] [14] οι παράμετροι θεωρούνται ότι έχουν γνωστές κατανομές πιθανοτήτων. Ωστόσο, στην πραγματικότητα για έναν αποφασίζων δεν είναι πάντα εύκολο να προσδιορίσει τη συνάρτηση συμμετοχής ή την κατανομή πιθανοτήτων σε ένα ανακριβές περιβάλλον. Τουλάχιστον σε ορισμένες από τις περιπτώσεις, η χρήση αριθμών-διαστημάτων μπορεί να εξυπηρετήσει καλύτερα τον σκοπό. Ένας αριθμός-διάστημα μπορεί να θεωρηθεί ως επέκταση της έννοιας ενός πραγματικού αριθμού για την ενσωμάτωση αυτής της μορφής αβεβαιότητας σε ένα πρόβλημα απόφασης [15].

Οι αριθμοί-διαστήματα είναι πιο κατάλληλοι για την αντιμετώπιση προβλημάτων λήψης αποφάσεων σε ένα ανακριβές και αβέβαιο περιβάλλον, επειδή είναι η απλούστερη μορφή αναπαράστασης της αβεβαιότητας στον πίνακα αποφάσεων. Οι αριθμοί-διαστήματα απαιτούν την ελάχιστη ποσότητα πληροφορίας σχετικά με τις τιμές των χαρακτηριστικών. Ο καθορισμός ενός διαστήματος για μια παράμετρο στον πίνακα αποφάσεων υποδεικνύει ότι η παράμετρος μπορεί να λάβει οποιαδήποτε τιμή εντός του διαστήματος. Σημειώνεται ότι, οι αριθμοί-διαστήματα δεν υποδεικνύουν πόσο πιθανό είναι η τιμή να βρίσκεται στο διάστημα, ούτε ποια τιμή του διαστήματος είναι πιο πιθανό να συμβεί. Ένας αριθμός-διάστημα μπορεί να θεωρηθεί και ως:

1. Μια επέκταση της έννοιας ενός πραγματικού αριθμού και επίσης ως υποσύνολο του πραγματικού άξονα.
2. Ένας εκφυλισμένος επίπεδος ασαφής αριθμός ή ασαφές διάστημα με μηδενικές διασπορές αριστερά και δεξιά.
3. μια α-τομή ενός ασαφούς αριθμού

Στη δημοσίευση [16] επεκτάθηκε η μέθοδος VIKOR για να αντιμετωπίσει προβλήματα Πολυκριτήριας Ανάλυσης Αποφάσεων με τις επιδόσεις της εναλλακτικής A_i έναντι του κριτηρίου C_j να αποτελούν αριθμούς-διαστήματα, δηλαδή $f_{ij} \in [f_{ij}^L, f_{ij}^U]$ για όλα τα i και j, όπως δείχνει ο Πίνακας 2.2.

Η επεκτεταμένη μέθοδος VIKOR αποτελείται από τα ακόλουθα τέσσερα βήματα:

1. Προσδιορισμός της θετικής ιδεατής λύσης A^* και της αρνητικής ιδεατής λύσης A^-:

$$A^* = \{f_1^*, \ldots, f_n^*\} = \left\{ \left(\max_i f_{ij}^U \mid j \in I \right) \text{ ή } \left(\min_i f_{ij}^L \mid j \in J \right) \right\},\ j = 1, \ldots, n \quad (2.11)$$

$$A^- = \{f_1^-, \ldots, f_n^-\} = \left\{ \left(\min_i f_{ij}^L \mid j \in I \right) \text{ ή } \left(\max_i f_{ij}^U \mid j \in J \right) \right\},\ j = 1, \ldots, n \quad (2.12)$$

όπου το σύνολο I περιέχει τα κριτήρια οφέλους και το σύνολο J περιέχει τα κριτήρια κόστους.

2. Υπολογισμός των S_i και R_i σε μορφή διαστημάτων $[S_i^L, S_i^U]$ και $[R_i^L, R_i^U]$ αντίστοιχα:

$$S_i^L = \sum_{j \in I} w_j \left(\frac{f_j^* - f_{ij}^U}{f_j^* - f_j^-} \right) + \sum_{j \in J} w_j \left(\frac{f_{ij}^L - f_j^*}{f_j^- - f_j^*} \right),\ i = 1, \ldots, m \quad (2.13)$$

$$S_i^U = \sum_{j \in I} w_j \left(\frac{f_j^* - f_{ij}^L}{f_j^* - f_j^-} \right) + \sum_{j \in J} w_j \left(\frac{f_{ij}^U - f_j^*}{f_j^- - f_j^*} \right),\ i = 1, \ldots, m \quad (2.14)$$

$$R_i^L = \max \left\{ w_j \left(\frac{f_j^* - f_{ij}^U}{f_j^* - f_j^-} \right) \middle| j \in I,\ w_j \left(\frac{f_{ij}^L - f_j^*}{f_j^- - f_j^*} \right) \middle| j \in J \right\},\ i = 1, \ldots, m \quad (2.15)$$

$$R_i^U = \max \left\{ w_j \left(\frac{f_j^* - f_{ij}^L}{f_j^* - f_j^-} \right) \middle| j \in I,\ w_j \left(\frac{f_{ij}^U - f_j^*}{f_j^- - f_j^*} \right) \middle| j \in J \right\},\ i = 1, \ldots, m \quad (2.16)$$

3. Υπολογισμός του αριθμού-διαστήματος $Q_i = [Q_i^L, Q_i^U]$ για κάθε εναλλακτική i:

$$Q_i^L = v \frac{S_i^L - S^*}{S^- - S^*} + (1 - v) \frac{R_i^L - R^*}{R^- - R^*},\ i = 1, \ldots, m \quad (2.17)$$

$$Q_i^U = v \frac{S_i^U - S^*}{S^- - S^*} + (1 - v) \frac{R_i^U - R^*}{R^- - R^*},\ i = 1, \ldots, m \quad (2.18)$$

Πίνακας 2.2: Πίνακας απόφασης στην επέκταση της VIKOR για επιδόσεις που είναι αριθμοί-διαστήματα

	C_1	C_2	$\cdots$	C_n
A_1	$[f_{11}^L, f_{11}^U]$	$[f_{12}^L, f_{12}^U]$	$\cdots$	$[f_{1n}^L, f_{1n}^U]$
A_2	$[f_{21}^L, f_{21}^U]$	$[f_{22}^L, f_{22}^U]$	$\cdots$	$[f_{2n}^L, f_{2n}^U]$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
A_m	$[f_{m1}^L, f_{m1}^U]$	$[f_{m2}^L, f_{m2}^U]$	$\cdots$	$[f_{mn}^L, f_{mn}^U]$

όπου τα S^*, S^-, R^*, R^- ορίζονται ως:

$$S^* = \min_i S_i^L \ , \quad S^- = \max_i S_i^U \tag{2.19}$$

$$R^* = \min_i R_i^L \ , \quad R^- = \max_i R_i^U \tag{2.20}$$

και εδώ ο παράγοντας v έχει την ίδια έννοια όπως στην εξ. (2.7) της κλασικής μεθόδου VIKOR.

4. Ταξινόμηση των εναλλακτικών βάσει των τιμών Q_i και επιλογή της λύσης συμβιβασμού. Επειδή πλέον τα Q_i είναι διαστήματα αριθμών μπορεί να χρησιμοποιηθεί ο παρακάτω αλγόριθμος για την ταξινόμηση τους. Για δύο διαστήματα αριθμών $[a^L, a^U]$ και $[b^L, b^U]$ που θέλουμε να συγκρίνουμε και να βρούμε τον μικρότερο, υπάρχουν τέσσερις περιπτώσεις:

   i. Εάν τα δύο διαστήματα δεν έχουν καμία τομή, το ελάχιστο είναι αυτό που έχει τις χαμηλότερες τιμές. Δηλαδή, εάν $a^U \leq b^L$, τότε επιλέγουμε το $[a^L, a^U]$ ως ελάχιστο.

   ii. Εάν τα δύο διαστήματα έχουν τα ίδια άκρα, τότε έχουν την ίδια προτεραιότητα.

   iii. Εάν $a^L \leq b^L < b^U \leq a^U$, επιλέγουμε το μικρότερο διάστημα με τον εξής τρόπο:

   - Εφόσον ισχύει
   $$k(b^L - a^L) \geq (1-k)(a^U - b^U) \tag{2.21}$$
   τότε το διάστημα $[a^L, a^U]$ θεωρείται ελάχιστο
   - αλλιώς είναι το $[b^L, b^U]$.

   iv. Εάν $a^L < b^L < a^U < b^U$, επιλέγουμε το μικρότερο διάστημα με τον εξής τρόπο:

   - Εφόσον ισχύει
   $$k(b^L - a^L) \geq (1-k)(b^U - a^U) \tag{2.22}$$
   τότε το διάστημα $[a^L, a^U]$ θεωρείται ελάχιστο
   - αλλιώς είναι το $[b^L, b^U]$.

Στις εξ. (2.21) και εξ. (2.22) ο παράγοντας $k \in [0, 1]$ παραμετροποιεί το βαθμό αισιοδοξίας του αποφασίζοντα. Ο αποφασίζοντας που είναι αισιόδοξος έχει μεγαλύτερη τιμή k από έναν αποφασίζοντα που είναι απαισιόδοξος. Για αποφασίζοντα με ουδέτερο προφίλ ως προς την αισιοδοξία, $k = 0.5$.

2.3. Επεκτεταμένη μέθοδος VIKOR με βάρη με ελλιπή πληροφόρηση

Πέραν όμως των στοιχείων του πίνακα απόφασης, η προτεραιοποίηση των κριτηρίων αποτελούν καθοριστικό παράγοντα για τη λήψη μιας απόφασης. Οι μέθοδοι κατάταξης των κριτηρίων που χρησιμοποιούνται στις τροποποιήσεις της μεθόδου VIKOR είναι πολυποίκιλες, κυμαίνονται από τις ακριβείς μέχρι τις ασαφείς. Οι γνωστές κατά τη βιβλιογραφία είναι: τα ίσα βάρη [17] [18] [4] [16], η μέθοδος άμεσης επιλογής βαρών [19], η μέθοδος εντροπίας [20] [21] [22], η μέθοδος του ιδιοδιανύσματος της AHP (Αναλυτική Ιεραρχική Διαδικασία) [20] και η ασαφής μέθοδος [23] [19] [24] [25]. Τα ίσα βάρη αντιπροσωπεύουν ένα μοναδικό σημείο στο χώρο των δυνητικών βαρών και υποδεικνύουν την ίση σημαντικότητα των κριτηρίων. Η μέθοδος της εντροπίας, βασισμένη στη θεωρία πληροφορίας, αποδίδει μικρό βάρος σε ένα κριτήριο με παρόμοιες επιδόσεις για όλες τις εναλλακτικές, επειδή αυτού του είδους το κριτήριο δε βοηθά στη διαφοροποίηση των εναλλακτικών [26]. Η μέθοδος της εντροπίας είναι μια αντικειμενική προσέγγιση που προτεραιοποιεί τα κριτήρια χρησιμοποιώντας μαθηματικά μοντέλα, αλλά παραμελεί τις υποκειμενικές πληροφορίες κρίσης των αποφασιζόντων [27]. Στις ασαφείς μεθόδους, είναι απαραίτητο να εκτελεστεί η διαδικασία αποασαφοποίησης που μετατρέπει ασαφή σύνολα σε συγκεκριμένες τιμές, που στο μεταξύ μπορεί να προκαλέσει την απώλεια πληροφορίας [28].

Στη μελέτη [2] προτείνεται μια νέα μέθοδος προτεραιοποίησης κριτηρίων, η μέθοδος των ελλιπών βαρών (Incomplete criteria weights), που μπορεί να παρέχει στον αποφασίζοντα περισσότερη ελευθερία επιλογής και προσδιορισμού [29]. Τα ελλιπή βάρη μπορούν να πάρουν μία από τις ακόλουθες μορφές:

- Κατώτατα όρια (LB): $W_{LB} = \{\boldsymbol{w} : w_j \geq a_j > 0,\ j = 1, 2, \ldots, n\}$
- Ασθενείς ανισότητες (WI): $W_{WI} = \{\boldsymbol{w} : w_1 \geq w_2 \geq \ldots \geq w_n \geq 0\}$
- Αναλογικές ανισότητες κλίμακας (RI): $W_{RI} = \{\boldsymbol{w} : w_1 \geq a_1 w_2, \ldots, w_{n-1} \geq a_{n-1} w_n, w_n \geq 0,\ a_j > 0,\ \forall j\}$
- Αυστηρές ανισότητες (SI): $W_{SI} = \{\boldsymbol{w} : w_j - w_{j+1} \geq \varepsilon_j > 0,\ j = 1, 2, \ldots, n-1,\ w_n \geq \varepsilon_n > 0\}$
- Ασθενείς ανισότητες διαφορών (WID): $W_{WID} = \{\boldsymbol{w} : w_1 - w_2 \geq \ldots \geq w_{n-1} - w_n,\ w_n \geq 0\}$.

Η μορφή των ασθενών ανισοτήτων είναι η πιο ευρέως χρησιμοποιούμενη και συναντάται σε μελέτες για την απλή πολυκριτήρια τεχνική αξιολόγησης με βάρη (SMARTS) ([30], [31]). Η αναλογική ανισότητα κλίμακας (ratio scale inequalities) είναι ένας τύπος αξιολόγησης που χρησιμοποιείται συχνά στην AHP όπου μια αριθμητική τιμή, a_{ij}, αντιπροσωπεύει ένα επίπεδο προτίμησης ανάμεσα του i-στοί και του j-στού διαδοχικού κριτηρίου [32]. Στην εργασία [33] υιοθέτησαν τη μορφή των αυστηρών ταξινομικών προτιμήσεων για τη μεγιστοποίηση της διάκρισης μεταξύ των υποψηφίων. Η μορφή των ασθενών ανισοτήτων διαφορών παρουσιάζει δύο επίπεδα της ισχύος της σημασίας των κριτηρίων. Για παράδειγμα, μπορεί να χρησιμοποιηθεί για την περίπτωση όπου το j-στο κριτήριο θεωρείται ισχυρά σημαντικό για το (j + 1)-στο κριτήριο, αλλά το (j + 1)-στο κριτήριο θεωρείται ασθενώς σημαντικό για το (j + 2)-στο κριτήριο [34].

Όταν τα βάρη των κριτηρίων είναι ελλιπώς γνωστά, η προτεινόμενη μέθοδος VIKOR μπορεί να κατατάξει τις εναλλακτικές μέσω μίας από δύο προσεγγίσεις: μέσω γραμμικού προγραμματισμού (LP) ή μέσω της μεθόδου ακραίων σημείων (extreme points).

Γενικά, τα ελλιπή βάρη οδηγούν σε μια περιοχή βαρών, από την οποία μπορούμε να αναγνωρίσουμε πολλά ακραία σημεία, εφόσον αυτή δεν είναι κενή. Στη μελέτη[35], αναπτύχθηκαν τύποι προσδιορισμού των ακραίων σημείων για όλες τις μορφές βαρών. Συγκεκριμένα για την μορφή ασθενών ανισοτήτων τα ακραία σημεία στην περίπτωση n κριτηρίων είναι τα εξής:

$$\boldsymbol{\lambda}_1 = (1, 0, \ldots, 0)^T, \boldsymbol{\lambda}_2 = \left(\frac{1}{2}, \frac{1}{2}, 0, \ldots, 0\right)^T, \ldots, \boldsymbol{\lambda}_n = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)^T \tag{2.23}$$

Αντικαθιστούμε τα ακραία σημεία κατά WI στον πίνακα $\boldsymbol{E} = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_n)$, όπου $\boldsymbol{\lambda}_i$ είναι το διάνυσμα της i-οστής στήλης του οποίου τα στοιχεία είναι το $\frac{1}{i}$ από το πρώτο έως το i-οστό, ενώ τα υπόλοιπα είναι μηδενικά.

$$\boldsymbol{E} = \begin{pmatrix} 1 & \frac{1}{2} & \ldots & \frac{1}{n} \\ 0 & \frac{1}{2} & \ldots & \frac{1}{n} \\ 0 & 0 & \ldots & \frac{1}{n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \frac{1}{n} \end{pmatrix}$$

Για κάθε εναλλακτική, γράφουμε το διάνυσμα $\boldsymbol{d}_i = (d_{i1}, d_{i2}, \ldots, d_{in})$ όπου $d_{ij} = \left|\frac{f_j^* - f_{ij}}{f_j^* - f_j^-}\right|$ και η σειρά εμφάνισης των d_{ij} μέσα στο διάνυσμα $\boldsymbol{d}_i$ εξαρτάται από τη σχετική σπουδαιότητα των κριτηρίων. Αν υποθέσουμε παράδειγμα 4 κριτήρια με σχετική σπουδαιότητα $w_3 \geq w_1 \geq w_4 \geq w_2$ τότε το $\boldsymbol{d}_i$ θα έχει αντίστοιχα τη διάταξη $\boldsymbol{d}_i = (d_{i3}, d_{i1}, d_{i4}, d_{i2})$.

Μετατρέπουμε τις τιμές S_i εξ. (2.5), R_i εξ. (2.6) και Q_i εξ. (2.7) που συναντώνται στην κλασική μέθοδο VIKOR σε αριθμούς-διαστήματα με τις ελάχιστες και μέγιστες τιμές τους να προκύπτουν από τις παρακάτω σχέσεις:

$$S_i^L = \min\{\boldsymbol{d}_i \boldsymbol{E}\}, i = 1, 2, \ldots, m \tag{2.24}$$

$$S_i^U = \max\{\boldsymbol{d}_i \boldsymbol{E}\}, i = 1, 2, \ldots, m \tag{2.25}$$

$$R_i^L = \min_k \Big\{ \max_j \{d_{ij} \lambda_{kj}\} \Big\}, i = 1, 2, \ldots, m \tag{2.26}$$

$$R_i^U = \max_k \Big\{ \max_j \{d_{ij} \lambda_{kj}\} \Big\}, i = 1, 2, \ldots, m \tag{2.27}$$

Ενώ για το $Q_i = [Q_i^L, Q_i^U]$ επαναχρησιμοποιούνται οι εξ. (2.17) και εξ. (2.18)

2.4. Επεκτεταμένη μέθοδος VIKOR με επιδόσεις αριθμούς-διαστήματα και βάρη με ελλιπή πληροφόρηση

Παρακάτω παρουσιάζεται συνοπτικά η επέκταση της μεθόδου VIKOR που συνδυάζει χαρακτηριστικά από τις προηγούμενες δύο επεκτάσεις. Η επέκταση αυτή χρησιμοποιεί τις μετρικές S_i, R_i όπως ορίστηκαν για την περίπτωση με ελλιπή πληροφόρηση στα βάρη εξ. (2.24) εξ. (2.25) εξ. (2.26) εξ. (2.27) προσαρμοσμένες στο γεγονός ότι οι επιδόσεις των εναλλακτικών στα αντίστοιχα κριτήρια είναι πλέον αριθμοί-διαστήματα.

Για κάθε εναλλακτική, ορίζονται πλέον τα διανύσματα $\boldsymbol{d}_i^L = (d_{i1}^L, d_{i2}^L, \ldots, d_{in}^L)$ και $\boldsymbol{d}_i^U = (d_{i1}^U, d_{i2}^U, \ldots, d_{in}^U)$ όπου τα d_{ij} ορίζονται όπως δείχνει ο Πίνακας 2.3 και η σειρά εμφάνισης των d_{ij} μέσα στο διάνυσμα $\boldsymbol{d}_i$ εξαρτάται από τη σχετική σπουδαιότητα των κριτηρίων. Ως υπενθύμιση, το σύνολο I περιέχει τα κριτήρια οφέλους και το σύνολο J περιέχει τα κριτήρια κόστους.

Πίνακας 2.3: Ορισμός των d_{ij} ανά τύπο κριτηρίου

	$j \in I$	$j \in J$
d_{ij}^L	$\frac{f_j^* - f_{ij}^U}{f_j^* - f_j^-}$	$\frac{f_{ij}^L - f_j^*}{f_j^- - f_j^*}$
d_{ij}^U	$\frac{f_j^* - f_{ij}^L}{f_j^* - f_j^-}$	$\frac{f_{ij}^U - f_j^*}{f_j^- - f_j^*}$

Και οι νέες σχέσεις των S_i, R_i:

$$S_i^L = \min\{\boldsymbol{d}_i^L \boldsymbol{E}\}, i = 1, 2, \ldots, m \tag{2.28}$$

$$S_i^U = \max\{\boldsymbol{d}_i^U \boldsymbol{E}\}, i = 1, 2, \ldots, m \tag{2.29}$$

$$R_i^L = \min_k \Big\{ \max_j \{d_{ij}^L \lambda_{kj}\} \Big\}, i = 1, 2, \ldots, m \tag{2.30}$$

$$R_i^U = \max_k \Big\{ \max_j \{d_{ij}^U \lambda_{kj}\} \Big\}, i = 1, 2, \ldots, m \tag{2.31}$$

Ενώ για το $Q_i = [Q_i^L, Q_i^U]$ επαναχρησιμοποιούνται οι εξ. (2.17) και εξ. (2.18)

2.5. Εναλλακτικές μέθοδοι ταξινόμησης αριθμών-διαστημάτων

Εκτός του Αλγόριθμου στο βήμα 4 που παρουσιάστηκε στην μεθοδολογία της επεκτεταμένης μεθόδου VIKOR με επιδόσεις αριθμούς-διαστήματα, προτείνονται και έχουν υλοποιηθεί στο εργαλείο δύο επιπλέον τρόποι ταξινόμησης των εναλλακτικών βάσει των Q τιμών τους όταν αυτές ορίζονται από αριθμούς διαστήματα.

Ο απλούστερος τρόπος είναι ο υπολογισμός της ενδιάμεσης τιμής $Q_m = \frac{Q^U + Q^L}{2}$ και στη συνέχεια χρήση αυτής για ταξινόμηση. Όπως και με τους προηγούμενους τρόπους επιλογής εναλλακτικής, ως καλύτερη θεωρείται αυτή με την μικρότερη τιμή Q_m.

Στη δημοσίευση [36] παρουσιάζεται μια διαφορετική οπτική για τη σύγκριση αριθμών-διαστημάτων βάσει του βαθμού δυνατότητας (degree of possibility). Ενώ στη μελέτη [37] παρουσιάστηκε μια μέθοδος για την ταξινόμηση αριθμών διαστημάτων λαμβάνοντας υπόψιν την προτίμηση βασισμένη σε μια πιθανοτική μετρική. Η ομοιότητα μεταξύ δύο διαστημάτων μπορεί να εκτιμηθεί από δύο μετρικές που τα χαρακτηρίζουν: την αναλογία της επικάλυψης των δύο διαστημάτων και το βαθμό εγγύτητας των μέσων τους [38]. Παρακάτω παρουσιάζεται η χρήση του βαθμού δυνατότητας για την ταξινόμηση διαστημάτων από τη δημοσίευση [36] προσαρμοσμένη στο πρόβλημα ταξινόμησης των Q τιμών.

Έστω $Q_i = [Q_i^L, Q_i^U]$ και $Q_j = [Q_j^L, Q_j^U]$ και έστω $l_i = Q_i^U - Q_i^L$ και $l_j = Q_j^U - Q_j^L$. Τότε ο βαθμός της δυνατότητας το Q_i να ξεπερνά το Q_j $(Q_i \geq Q_j)$ ορίζεται ως

$$p_{ij} = p(Q_i \geq Q_j) = \max\left\{1 - \max\left\{\frac{Q_j^U - Q_i^L}{l_i + l_j}, 0\right\}, 0\right\} \tag{2.32}$$

Αντίστροφα, ο υπολογισμός για $(Q_j \geq Q_i)$ ορίζεται ως

$$p_{ji} = p(Q_j \geq Q_i) = \max\left\{1 - \max\left\{\frac{Q_i^U - Q_j^L}{l_i + l_j}, 0\right\}, 0\right\} \tag{2.33}$$

Στη συνέχεια, κατασκευάζουμε τον συμπληρωματικό πίνακα $\boldsymbol{P}$ που δείχνει το βαθμό δυνατότητας όλων των διαστημάτων:

$$\boldsymbol{P} = \begin{pmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \dots & p_{nn} \end{pmatrix} \text{ όπου } p_{ij} \geq 0, p_{ij} + p_{ji} = 1, p_{ii} = \frac{1}{2} \tag{2.34}$$

Αθροίζοντας όλα τα στοιχεία σε κάθε γραμμή του πίνακα $\boldsymbol{P}$, καταλήγουμε στον αθροιστικό βαθμό δυνατότητας του κάθε διάστημα να ξεπερνάει τα υπόλοιπα:

$$p_i = \sum_{j=i}^{n} p_{ij}, i = 1 \dots n. \tag{2.35}$$

Στη συνέχεια μπορούμε να ταξινομήσουμε τα διαστήματα $Q_i, i = 1, \ldots, n$ σε φθίνουσα σειρά του αθροιστικού βαθμού δυνατότητας $p_i, i = 1, \ldots, n$ όπου και πάλι η μικρότερη τιμή δηλώνει καλύτερη αναλλακτική.

2.6. Επεκτεταμένη μέθοδος VIKOR με τριγωνικούς ασαφείς αριθμούς

Η επέκτασης της μεθόδου VIKOR με ασάφεια έχει αναπτυχθεί ώστε να καθορίζει τη συμβιβαστική λύση ενός ασαφούς πολυκριτηριακού προβλήματος

$$\underset{i}{mco}\{(\tilde{f}_{ij}(A_i),\ i = 1, \ldots m),\ j = 1, \ldots, n\}$$

όπου m είναι το πλήθος των προς εξέταση εναλλακτικών, $A_i = \{x_1, x_2, \ldots\}$ είναι η i-οστή εναλλακτική που έχει αποκτηθεί (δημιουργηθεί) με συγκεκριμένες τιμές των μεταβλητών συστήματος x.Το f_{ij} είναι η τιμή της j-οστής συνάρτησης κριτηρίου για την εναλλακτική $_i$. Το n είναι ο αριθμός των κριτηρίων. Το mco υποδηλώνει τον τελεστή μιας διαδικασίας πολυκριτηριακής λήψης αποφάσεων για την επιλογή της καλύτερης (συμβιβαστικής) εναλλακτικής από πολυκριτηριακή άποψη. Οι εναλλακτικές μπορούν να παραχθούν και η εφικτότητά τους να ελεγχθεί μέσω μαθηματικών μοντέλων (καθορισμός μεταβλητών x), φυσικών μοντέλων και/ή μέσω πειραμάτων στο υφιστάμενο σύστημα ή άλλα παρόμοια συστήματα. Οι περιορισμοί αντιμετωπίζονται ως στόχοι υψηλής προτεραιότητας, που πρέπει να ικανοποιηθούν στη διαδικασία παραγωγής των εναλλακτικών. Σε αυτή την επέκταση της μεθόδου VIKOR, γίνεται η υπόθεση ότι οι επιδόσεις των εναλλακτικών αποτελούνται από τριγωνικές ασαφείς αριθμητικές τιμές $\tilde{f}_{ij} = (l_{ij}, m_{ij}, r_{ij}),\ i = 1, \ldots, m,\ j = 1, \ldots, n$. [18]

Τα βήματα για την εκτέλεση αυτής της επέκτασης VIKOR είναι τα παρακάτω:

1. Υπολογισμός της βέλτιστης τιμής $\tilde{f}_j^* = (l_j^*, m_j^*, r_j^*)$ και της χειρότερης τιμής $\tilde{f}_j^\circ = (l_j^\circ, m_j^\circ, r_j^\circ)$ για όλα τα κριτήρια $j = 1, \ldots, n$ από τους τύπους:

$$\tilde{f}_j^* = \underset{i}{MAX}\ \tilde{f}_{ij}, \quad \tilde{f}_j^\circ = \underset{i}{MIN}\ \tilde{f}_{ij}, \quad j \in I \tag{2.36}$$

$$\tilde{f}_j^* = \underset{i}{MIN}\ \tilde{f}_{ij}, \quad \tilde{f}_j^\circ = \underset{i}{MAX}\ \tilde{f}_{ij}, \quad j \in J \tag{2.37}$$

   όπου το σύνολο I περιέχει τα κριτήρια οφέλους και το σύνολο J περιέχει τα κριτήρια κόστους.

2. Υπολογισμός των κανονικοποιημένων ασαφών διαφορών $\tilde{d}_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n$ από τους τύπους:

$$\tilde{d}_{ij} = \frac{\tilde{f}_j^* \ominus \tilde{f}_{ij}}{r_j^* - l_j^\circ}, \quad j \in I \tag{2.38}$$

$$\tilde{d}_{ij} = \frac{\tilde{f}_{ij} \ominus \tilde{f}_j^*}{r_j^\circ - l_j^*}, \quad j \in I \tag{2.39}$$

3. Υπολογισμός των $\tilde{S}_i = (S_i^l, S_i^m, S_i^r)$ και $\tilde{R}_i = (R_i^l, R_i^m, R_i^r),\ i = 1, \ldots, m$ από τους τύπους:

$$\tilde{S}_i = \sum_{j=1}^{n} {}_{\oplus}\big(\tilde{w}_j \otimes \tilde{d}_{ij}\big) \tag{2.40}$$

$$\tilde{R}_i = \underset{j}{MAX}\big(\tilde{w}_j \otimes \tilde{d}_{ij}\big) \tag{2.41}$$

όπου το $\tilde{S}$ είναι το ασαφές σταθμισμένο άθροισμα, το $\tilde{S}$ είναι ο ασαφής τελεστής μεγίστου MAX που ορίζεται στον Πίνακα 2.4 και το $\tilde{w}_j$ συμβολίζει τα βάρη των κριτηρίων που εκφράζουν την προτίμηση του αποφασίζοντα ως προς τη σχετική σημαντικότητα τους.

4. Υπολογισμός των ποσοτήτων $\tilde{Q}_i = (Q_i^l, Q_i^m, Q_i^r),\ i = 1, \ldots, m$ από τον τύπο:

$$\tilde{Q}_i = v\frac{\tilde{S}_i \ominus \tilde{S}^*}{S^{\circ r} - S^{*l}} \oplus (1 - v)\frac{\tilde{R}_i \ominus \tilde{R}^*}{R^{\circ r} - R^{*l}} \tag{2.42}$$

όπου

$$\tilde{S}^* = \underset{i}{MIN}\, \tilde{S}_i \tag{2.43}$$

$$S^{\circ r} = \max_i S_i^r \tag{2.44}$$

$$\tilde{R}^* = \underset{i}{MIN}\, \tilde{R}_i \tag{2.45}$$

$$R^{\circ r} = \max_i R_i^r \tag{2.46}$$

και η παράμετρος $v \in [0, 1]$ είναι το βάρος που θέτει ο αποφασίζοντας στη προτίμηση του στη συνολική χρησιμότητα ενώ το (1 − v) είναι το βάρος του βαθμού δυσαρέσκειας. Οι βέλτιστες τιμές των S και R συμβολίζονται από τα $\tilde{S}^*$ και $\tilde{R}^*$, αντίστοιχα.

5. «Κεντρική» ταξινόμηση: Ταξινόμηση των εναλλακτικών ως προς την ενδιάμεση τιμή $Q_i^m,\ i = 1, \ldots m$ σε φθίνουσα σειρά. Η ταξινομημένη λίστα των εναλλακτικών ως προς Q_i^m συμβολίζεται με $\{A\}_{Q^m}$.

6. Ασαφής ταξινόμηση: Η i-οστή θέση ταξινόμησης μέσα στη $\{A\}_{Q^m}$ μια εναλλακτικής $A^{(i)},\ i = 1, \ldots m$ επιβεβαιώνεται εάν

$$\underset{k \in m^i}{MIN}\, \tilde{Q}^{(k)} = \tilde{Q}^{(i)} \tag{2.47}$$

όπου $m^i = \{i, i+1, ...m\}$ και $\tilde{Q}^{(k)}$ είναι η ασαφής ποσότητα Q της εναλλακτικής $A^{(k)}$ που βρίσκεται την k-οστή θέση της κεντρικής ταξινόμησης $\{A\}_{Q^m}$. Η επιβεβαίωση των θέσεων αναπαριστά «ακριβή» ασαφή ταξινόμηση $\{A\}_{\tilde{Q}}$, ωστόσο το σύνολο $\{A\}_{\tilde{Q}}$ μπορεί να μην ορίζει πλήρη ταξινόμηση, αλλά μερική.

7. Αποασαφοποίηση των $\tilde{S}_i, \tilde{R}_i, \tilde{Q}_i,\ i = 1, \ldots m$ με χρήση της σχέσης:

$$Crisp\left(\tilde{N}\right) = \frac{2m + l + r}{4} \tag{2.48}$$

Εδώ χρησιμοποιείται η μέθοδος του «δεύτερου σταθμισμένου μέσου» (2.6.) για την μετατροπή των ασαφών αριθμών σε συγκεκριμένες αριθμητικές τιμές.

8. Ταξινόμηση των εναλλακτικών ως προς τις αριθμητικές τιμές S, R και Q, του προηγούμενου βήματος, σε φθίνουσα σειρά. Το αποτέλεσμα είναι τρεις λίστες ταξινόμησης $\{A\}_S$, $\{A\}_R$, $\{A\}_Q$.

9. Επιλογή της εναλλακτικής $A^{(1)}$, που ταξινομήθηκε ως καλύτερη κατά το μέγεθος Q (στο σύνολο $\{A\}_Q$), ως προτεινόμενη λύση συμβιβασμού, εάν ικανοποιεί τις παρακάτω δύο συνθήκες:

   i. Συνθήκη C1 - συγκριτικό πλεονέκτημα: $Adv \geqslant DQ$ όπου

   $$Adv = \frac{Q(A^{(2)}) - Q(A^{(1)})}{Q(A^{(m)}) - Q(A^{(1)})} \tag{2.49}$$

   είναι ο ρυθμός πλεονεκτήματος της εναλλακτικής $A^{(1)}$ που βρίσκεται στη πρώτη θέση, $A^{(2)}$ είναι η εναλλακτική που βρίσκεται στη δεύτερη θέση της $\{A\}_Q$ ταξινόμησης και το όριο

   $$DQ = \frac{1}{m-1} \tag{2.50}$$

   ii. Συνθήκη C2 - αποδεκτή ευστάθεια στη λήψη απόφασης:
   Η εναλλακτική $Q(A^{(1)})$ πρέπει επιπλέον να έχει και την πρώτη θέση ως προς S (στο $\{A\}_S$) ή/και R (στο $\{A\}_R$).

Εάν κάποια από τις συνθήκες δεν ικανοποιείται, προτείνεται ένα σύνολο λύσεων συμβιβασμού, που αποτελείται από:

i. Τις εναλλακτικές $A^{(1)}$ και $A^{(2)}$ εάν δεν ικανοποιείται μόνο η συνθήκη C2, αλλιώς
ii. Τις εναλλακτικές $A^{(1)}, {}^{(2)}, \ldots, A^{(M)}$ αν δεν ικανοποιείται η συνθήκη C1. Το $A^{(M)}$ καθορίζεται από το μέγιστο M που επαληθεύει τη σχέση

$$Q(A^{(M)}) - Q(A^{(1)}) < DQ$$

Ταξινόμηση ασαφών αριθμών και αποασαφοποίηση

Η Πολυκριτήρια Ανάλυση Αποφάσεων σε ένα ασαφές περιβάλλον απαιτεί τη σύγκριση ασαφών αριθμών. Το πρόβλημα της σύγκρισης ασαφών αριθμών έχει μελετηθεί και φαίνεται να είναι ένα σημαντικό και δύσκολο πρόβλημα. [18] Ένας ασαφής αριθμός χαρακτηρίζεται από το σχήμα, την απόκλιση, το ύψος και τη σχετική θέση του στον άξονα x. Μια καλή μέθοδος κατάταξης θα

έπρεπε να λαμβάνει υπόψη όλους αυτούς τους παράγοντες. Δεδομένου ότι ένας ασαφής αριθμός αντιπροσωπεύει πολλούς δυνητικούς πραγματικούς αριθμούς που έχουν διάφορες συναρτήσεις συμμετοχής, αντιμετωπίζουμε ένα δύσκολο πρόβλημα κατά τη σύγκρισή τους. Έχουν προταθεί πάνω από 20 μέθοδοι κατάταξης ασαφών αριθμών, αλλά καμία από αυτές τις υπάρχουσες μεθόδους δεν είναι τέλεια [9] [39] [40] [41] [42] [43]. Γενικά, χρησιμοποιούνται δύο προσεγγίσεις: (1) σύγκριση ασαφών αριθμών και (2) μετατροπή των ασαφών αριθμών σε ακριβείς αριθμούς (αποασαφοποίηση). Ο αλγόριθμος VIKOR με τριγωνικούς ασαφής αριθμούς στο βήμα 6 χρησιμοποιεί μια διαδικασία κατάταξης με συνεπή αποτελέσματα που παρέχει πλήρη κατάταξη μόνο εάν οι ασαφείς αριθμοί έχουν διαχωρισμένες συναρτήσεις συμμετοχής (χωρίς διασταυρώσεις). Εάν υπάρχει διασταύρωση, αυτή η διαδικασία δεν παρέχει πλήρη ταξινόμηση. Ο ασαφής τελεστής MIN χρησιμοποιείται για την κατάταξη και επιβεβαίωση της κατάταξης. Μια εναλλακτική A_i κατατάσσεται σε καλύτερη θέση από την A_k εάν $\tilde{Q}_i = MIN\left(\tilde{Q}_i, \tilde{Q}_k\right)$ ή αλλιώς $Q_i^l \leqslant Q_k^l,\ Q_i^m < Q_k^m,\ Q_i^r \leqslant Q_k^r$.

Αυτή η επέκταση της VIKOR χρησιμοποιεί μια διαδικασία αποασαφοποίησης στο βήμα 7 για να μετατρέψει τους ασαφείς αριθμούς σε πραγματικούς (crisp) αριθμούς. Στη συνέχεια, στο βήμα 8, η κατάταξη των ασαφών αριθμών πραγματοποιείται μέσω της σύγκρισης των αντίστοιχων πραγματικών αριθμών. Στην εργασία [18] αναπτύσσεται η μέθοδος του k-στού σταθμισμένου μέσου για να χρησιμοποιηθεί ως διαδικασία αποασαφοποίσης. Χρησιμοποιεί τη συνάρτηση μέλους υψωμένη στη δύναμη k ως σταθμισμένο παράγοντα. Η ακριβής τιμή $Crisp(\tilde{N})$ για τον τριγωνικό ασαφή αριθμό $\tilde{N}$ καθορίζεται από τον ακόλουθο τύπο:

$$Crisp(\tilde{N}) = \frac{\int_l^r x\mu^k(x)\,dx}{\int_l^r \mu^k(x)\,dx}$$

Από την ολοκλήρωση του παραπάνω ολοκληρώματος, προκύπτει ο παρακάτω τύπος:

$$Crisp(\tilde{N}) = \frac{km + l + r}{k + 2}$$

ή

$$C = m + \frac{s_r - s_l}{k + 2}$$

και

$$\mu(C) = \begin{cases} \frac{k+1}{k+2} + \frac{s_r}{(k+2)s_l}, & C \leqslant m \\ \frac{k+1}{k+2} + \frac{s_l}{(k+2)s_r}, & C \geqslant m \end{cases}$$

όπου $C = Crisp(\tilde{N})$, ενώ τα $s_l = m - l$ και $s_r = r - m$ είναι το αριστερό και το δεξί στήριγμα αντίστοιχα.

Ο αντίκτυπος της δύναμης k στο αποτέλεσμα της αποασαφοποίησης είναι ο παρακάτω:

$$\text{για } k = 1 \ :\ C = \frac{m + l + r}{3} \quad \text{ή} \quad C = m + \frac{s_r - s_l}{3} \quad \text{και} \quad \mu(C) = \frac{2}{3}$$

Με αύξηση της δύναμης $k(k = 2, 3, \ldots)$ το C κινείται προς το κέντρο m και η συνάρτηση συμμετοχής $\mu(C)$ αυξάνεται. Για παράδειγμα, $k = 4 : C = m + \frac{s_r - s_l}{3}$ και $\lim_{k\to\infty} C(k) = m, \lim_{k\to\infty} \mu(C, k) = 1$.

Η μέθοδος του Κέντρου (Κέντρο βάρους), που παρέχει μια σαφή τιμή βασισμένη στο κέντρο βάρους του ασαφούς συνόλου, θα μπορούσε να θεωρηθεί ως μια ειδική περίπτωση για k = 1. Στο χώρο πολυκριτήριας λήψης αποφάσεων, το $k \geqslant 4$ θα μπορούσε να προτιμηθεί από έναν αποφασίζοντα που αποφεύγει τον κίνδυνο (αυξημένη συνάρτηση μέλους $\mu(C)$), αυτό είναι μια προσέγγιση αποασαφοποίησης προς το κέντρο. Ένας αποφασίζων που αναλαμβάνει ρίσκο μπορεί να έχει διαφορετικές προτιμήσεις [44]. Μια γενική σύσταση θα μπορούσε να είναι η χρήση μίας από τις τιμές $\{2, 3, 4\}$ για τη δύναμη k, και η τιμή της δύναμης k θα πρέπει να είναι ίδια για την αποασαφοποίηση όλων των ασαφών αριθμών σε μια ανάλυση.

Στο βήμα 6 χρησιμοποιείται η μέθοδος του «δεύτερου σταθμισμένου μέσου» $(k = 2)$ ως πρακτικό εργαλείο αποασαφοποίησης.

Τελεστές τριγωνικών ασαφών αριθμών

Για να εκφραστεί μια ασαφής τιμή, όπως "περίπου m" χρησιμοποιείται ο τριγωνικός ασαφής αριθμός $\tilde{N} = (l, m, r)$, συνδεδεμένος με την τριγωνική συνάρτηση συμμετοχής που καθορίζεται ως εξής:

$$\mu_{\tilde{N}}(x) = \begin{cases} \frac{x-l}{m-l}, & l \leqslant x \leqslant m \\ \frac{r-x}{r-m}, & m \leqslant x \leqslant r \\ 0, & x \notin [l, r] \end{cases}$$

Η συνάρτηση μέλους $\mu(x)$ υποδηλώνει το βαθμό αληθείας ότι η ασαφής τιμή είναι ίση με x εντός του πραγματικού διαστήματος $[l, r]$. Ο ασαφής αριθμός $\tilde{N}$ έχει το κέντρο m με$\mu(m) = 1$ και την υποστήριξη $[l, r]$. [18] Η ασαφής μέθοδος VIKOR έχει αναπτυχθεί εφαρμόζοντας μαθηματικές πράξεις σε τριγωνικούς ασαφείς αριθμούς που καθορίζονται στον πινακα: 2.4

Το αποτέλεσμα της πρόσθεσης ή αφαίρεσης τριγωνικών ασαφών αριθμών (TAA) είναι επίσης ένας TAA. Το αποτέλεσμα του ασαφούς πολλαπλασιασμού θεωρείται ως μια προσέγγιση TAA, ειδικά όταν εφαρμόζεται α-τομή [45]. Το αποτέλεσμα των $\underset{i}{MAX}(\tilde{N}_1, \tilde{N}_2)$ ή (MIN) δεν είναι TAA μόνο εάν οι $\tilde{N}_2$ και $\tilde{N}_2$ επικαλύπτονται σε δύο περιπτώσεις: (1) $l_1 < l_2, m_1 \neq m_2, r_1 > r_2$ και (2) $l_1 < l_2, m_1 > m_2, r_1 < r_2$. Σε αυτές τις περιπτώσεις, ο TAA χρησιμοποιείται ως προσέγγιση. Ορισμοί και χαρακτηριστικά των παραπάνω διαδικασιών συζητούνται σε αρκετά επιστημονικά άρθρα [46] [47] [48].

Πρόσθεση ΤΑΑ	$\sum_{i=1}^{n} {}_{\oplus}\tilde{N}_i = \left(\sum_{i=1}^{n} l_i, \sum_{i=1}^{n} m_i, \sum_{i=1}^{n} r_i \right)$
Βαθμωτή πρόσθεση	$\tilde{N} + K = (l + K, m + K, r + K)$
Αφαίρεση ΤΑΑ	$\tilde{N}_1 \ominus \tilde{N}_2 = (l_1 - r_2, m_1 - m_2, r_1 - l_2)$
Βαθμωτή αφαίρεση	$\tilde{N} - K = (l - K, m - K, r - K)$
Βαθμωτός πολλαπλασιασμός	$K \times \tilde{N} = (K \times l, K \times m, K \times r)$, για $K \geqslant 0$
Πολλαπλασιασμός ΤΑΑ	$\tilde{N}_1 \otimes \tilde{N}_2 = (l_1 \times l_2, m_1 \times m_2, r_1 \times r_2)$, για $l_1 \geqslant 0$
Βαθμωτής διαίρεση	$\tilde{N}/K = (l/K, m/K, r/K)$, για $K \geqslant 0$
Τελεστής MAX	$\underset{i}{MAX}\, \tilde{N}_i = (\max_i l_i, \max_i m_i, \max_i r_i)$
Τελεστής MIN	$\underset{i}{MIN}\, \tilde{N}_i = (\min_i l_i, \min_i m_i, \min_i r_i)$

Πίνακας 2.4: Πράξεις τριγωνικών ασαφών αριθμών (ΤΑΑ)

2.7. Η μέθοδος TOPSIS

Η βασική αρχή της μεθόδου TOPSIS [2] είναι ότι η επιλεγμένη εναλλακτική λύση πρέπει να έχει τη μικρότερη απόσταση από την ιδεατή λύση και τη μεγαλύτερη απόσταση από την αρνητική ιδεατή λύση [4] [49] [50]. Η διαδικασία της TOPSIS αποτελείται από τα ακόλουθα βήματα:

1. Υπολογισμός του κανονικοποιημένου πίνακα απόφασης. Οι κανονικοποιημένες τιμές r_{ij} υπολογίζονται ως:

$$r_{ij} = \frac{f_{ij}}{\sqrt{\sum_{k=1}^{m} f_{kj}^2}}, i = 1, \ldots, m, j = 1, \ldots, n \tag{2.51}$$

2. Υπολογισμός του σταθμισμένου κανονικοποιημένου πίνακα απόφασης. Η σταθμισμένη κανονικοποιημένη τιμή v_{ij} υπολογίζεται ως $v_{ij} = w_j r_{ij}, i = 1, \ldots, m, j = 1, \ldots, n$ όπου w_j είναι το βάρος του j-στου κριτηρίου και $\sum_{j=1}^{n} w_j = 1$

3. Υπολογισμός της θετικής ιδεατής και της αρνητικής ιδεατής λύσης:

$$A^* = \{v_1^*, \ldots, v_n^*\} = \{\max_i v_{ij} \mid j \in I \quad \text{ή} \quad \min_i v_{ij} \mid j \in J\} \tag{2.52}$$

$$A^- = \{v_1^-, \ldots, v_n^-\} = \{\min_i v_{ij} \mid j \in I \quad \text{ή} \quad \max_i v_{ij} \mid j \in J\} \tag{2.53}$$

όπου το σύνολο I περιέχει τα κριτήρια οφέλους και το σύνολο J περιέχει τα κριτήρια κόστους.

4. Υπολογισμός της απόστασης από την ιδεατή λύση με χρήστη της Ευκλείδειας απόστασης από αυτή για κάθε εναλλακτική.

[2] Το όνομα προέρχεται από τα αρχικά της φράσης «Technique for Order Preference by Similarity to an Ideal Solution»

$$D_i^* = \sqrt{\sum_{j=1}^{n}(v_{ij} - v_j^*)^2}, i = 1, \ldots, m \tag{2.54}$$

Ομοίως υπολογίζεται και η απόσταση από την αρνητική ιδεατή λύση.

$$D_i^- = \sqrt{\sum_{j=1}^{n}(v_{ij} - v_j^-)^2}, i = 1, \ldots, m \tag{2.55}$$

5. Υπολογισμός της σχετική εγγύτητας προς την ιδεατή λύση για κάθε εναλλακτική με χρήση του τύπου:

$$C_i = \frac{D_i^-}{D_i^* + D_i^-}, i = 1, \ldots, m \tag{2.56}$$

6. Κατάταξη των εναλλακτικών βάση των τιμών C_i, όπου μεγαλύτερη τιμή σημαίνει καλύτερη εναλλακτική.

Κεφάλαιο 3

Υπολογιστικό εργαλείο

3.1. Εκκίνηση

3.1.1 Εγκατάσταση

Το υπολογιστικό εργαλείο είναι διαθέσιμο σε μορφή διαδικτυακής εφαρμογής και ως εκ τούτου δεν απαιτείται κάποια πρόσθετη εγκατάσταση από τον χρήστη. Για την πρόσβασή του σε αυτό, ο χρήστης αρκεί να επισκεφθεί τη διεύθυνση από κάποιο περιηγητή ιστού της επιλογής του. Η εφαρμογή υποστηρίζεται επίσημα και έχει δοκιμαστεί σε πρόσφατες εκδόσεις[1] των Microsoft Edge και Google Chrome, ωστόσο στη πράξη μπορεί να χρησιμοποιηθεί μια πρόσφατη έκδοση κάθε άλλου περιηγητή με βάση το Chromium [2] ή που να υποστηρίζει την τεχνολογία WebAssembly.

3.1.2 Απαιτήσεις συστήματος

Εφόσον το εργαλείο είναι διαδικτυακή εφαρμογή και τρέχει στο περιβάλλον ενός web browser δεν υπάρχει συγκεκριμένος τρόπος για να μετρηθούν και οριστούν συγκεκριμένες απαιτήσεις συστήματος. Όλοι οι υπολογισμοί γίνονται στο backend της εφαρμογής ενώ στο σύστημα του χρήστη γίνεται μόνο η εισαγωγή δεδομένων και η παρουσίαση των αποτελεσμάτων που επιστρέφονται από αυτό. Εκτιμάται ότι απαιτούνται περίπου 200MB[3] μνήμης RAM επιπλέον των απαιτήσεων που έχει ο επιλεγμένος browser για την ομαλή του λειτουργία. Επιπλέον είναι απαραίτητη η πρόσβαση στο διαδίκτυο κατά την φόρτωση των σελίδων και κατά την υποβολή δεδομένων για υπολογισμό.

[1] που έχει λάβει ενημέρωση από τον Οκτώβρη του 2022 και μετά

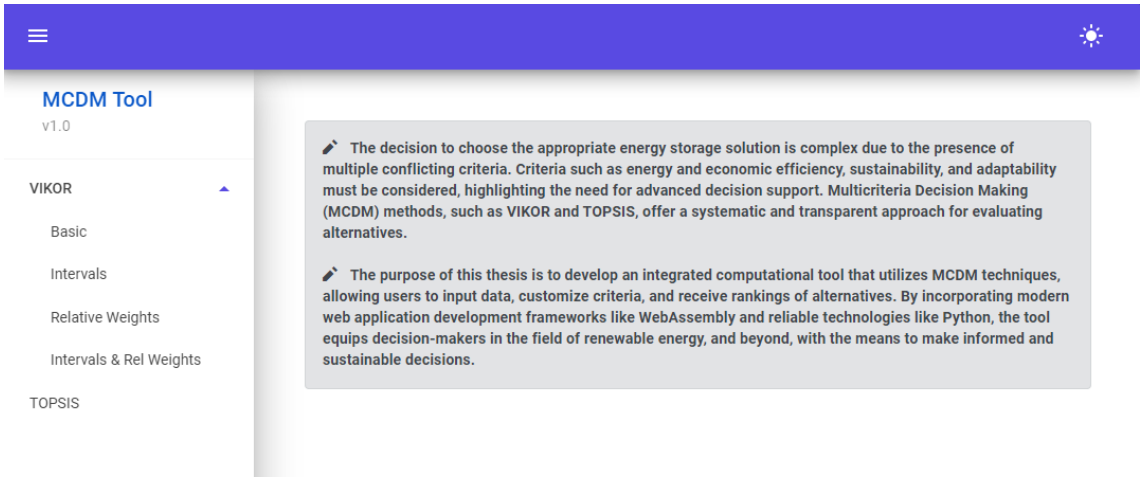
[2] Το Chromium είναι ένα λογισμικό ανοικτού κώδικα που ξεκίνησε και αναπτύχθηκε κυρίως από τη Google και αποτελεί τη βάση πάνω στην οποία αναπτύχθηκαν πολλοί από τους περιηγητές που χρησιμοποιούνται σήμερα.

[3] Ο χώρος αυτός είναι ενδεικτικός και ενδέχεται να αυξηθεί σε επόμενες εκδόσεις της εφαρμογής ή των βιβλιοθηκών που χρησιμοποιήθηκαν στην ανάπτυξή της

3.2. Εγχειρίδιο χρήσης

3.2.1 Αρχική σελίδα

Κατά την είσοδό του στο σύστημα, ο χρήστης, βρίσκεται στην αρχική σελίδα(Εικ. 3.1). Η βασική διάταξη (layout) της γραφικής διεπαφής του χρήστη με την εφαρμογή (graphical user interface) παραμένει σταθερή σε όλες τις προσβάσιμες σελίδες, ενώ το περιεχόμενο της κάθε σελίδας αλλάζει δυναμικά. Για παράδειγμα, στο περιεχόμενο της αρχικής σελίδας καταγράφεται το θέμα της παρούσας διπλωματικής εργασίας.

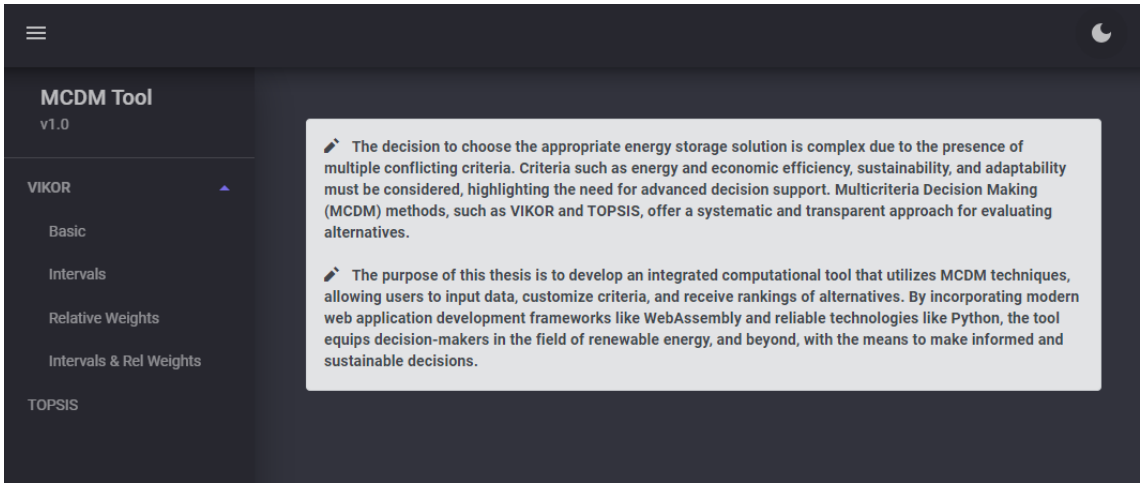


Εικόνα 3.1: Η αρχική σελίδα της εφαρμογής

3.2.2 Περιγραφή βασικής διάταξης

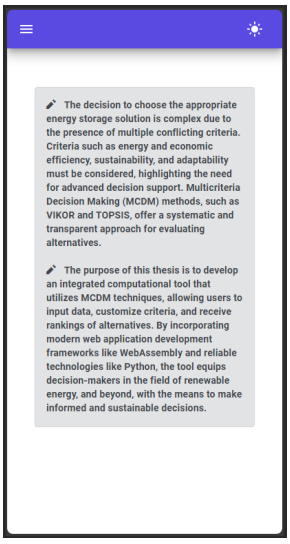
Στην αριστερή πλευρά της σελίδας και σε κάθετη διάταξη βρίσκεται το μενού πλοήγησης. Αυτό αποτελείται από μια επικεφαλίδα που αναγράφει το όνομα της εφαρμογής, τη τρέχουσα έκδοση της και λειτουργεί και ως σύνδεσμος για την αρχική σελίδα. Παρακάτω βρίσκονται σύνδεσμοι για τις υπόλοιπες σελίδες του εργαλείου που φιλοξενούν τα υπολογιστικά εργαλεία. Ειδικά για τις διάφορες μεθόδους ανάλυσης της οικογένειας VIKOR έχει δημιουργηθεί ειδική κατηγορία όπου είναι ομαδοποιημένες. Υποστηρίζεται η επεκτασιμότητα του εργαλείου με ενσωμάτωση περισσότερων μεθόδων ή παραλλαγών τους, οι οποίες μπορούν να ομαδοποιηθούν κατάλληλα βάσει συνάφειας.

Στην κορυφή της σελίδας βρίσκεται η οριζόντια μπάρα εργαλείων. Αριστερά αυτής βρίσκεται το κουμπί που κρύβει ή εμφανίζει το μενού πλοήγησης. Στη δεξιά πλευρά βρίσκεται το κουμπί που ρυθμίζει την λειτουργία σκούρου θέματος. Με τη λειτουργία αυτή όλα τα χρώματα της εφαρμογής αντικαθίστανται με την παλέτα σκούρων χρωμάτων (Εικ. 3.2) ώστε να μειώνεται η συνολική φωτεινότητα της οθόνης, κάνοντας την ανάγνωση πιο ξεκούραστη για τα ματιά, ειδικά σε συνθήκες περιβάλλοντος χαμηλού φωτισμού. Η εφαρμογή, κατά το πρώτο φόρτωμα, διαβάζει και χρησιμοποιεί ως προεπιλογή την τιμή της σχετικής ρύθμισης από το λειτουργικό σύστημα του χρήστη.



Εικόνα 3.2: Η αρχική σελίδα σε λειτουργία σκούρου θέματος

Αξίζει να σημειωθεί ότι τα components που έχουν χρησιμοποιηθεί για την υλοποιήση του εργαλείου ακολουθούν το Responsive UI Design. Αυτό σημαίνει πως η σελίδα προσαρμόζεται ανάλογα με το μέγεθος της οθόνης το χρήστη. Για παράδειγμα, από κινητή συσκευή σε διάταξη πορτρέτου, το μενού πλοήγησης θα είναι αρχικά κρυμμένο για εξοικονόμηση χώρου.

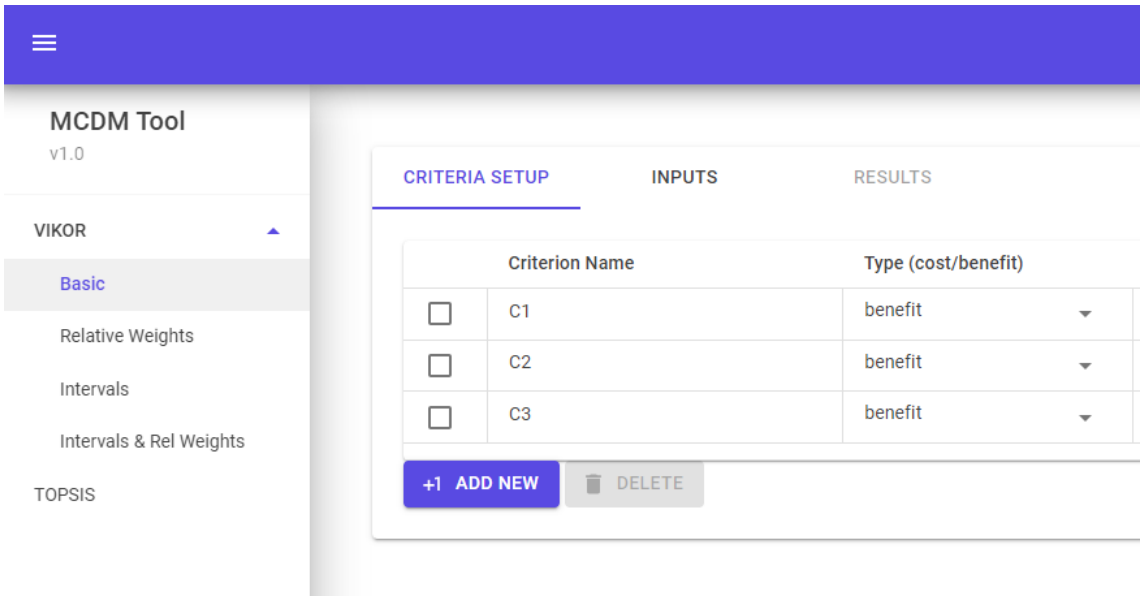


Εικόνα 3.3: Η αρχική σελίδα σε κινητή συσκευή και διάταξη πορτρέτου

Στη συνέχεια θα γίνει παρουσίαση των λειτουργιών των διαφόρων στοιχείων που εμφανίζονται στο γραφικό περιβάλλον της εφαρμογής. Θα γίνει αναφορά τόσο στα στοιχεία που είναι κοινά για όλες τις μεθόδους που έχουν υλοποιηθεί αλλά και στα στοιχεία που έχουν υλοποιηθεί για να ικανοποιηθούν οι ειδικές απαιτήσεις διεπαφής που προκύπτουν από τις ιδιαιτερότητες της κάθε μεθόδου.

3.2.3 Παρουσίαση βασικών στοιχείων του γραφικού περιβάλλοντος

Ένα κοινό χαρακτηριστικό που διέπει όλες τις μεθόδους πολυκριτήριας ανάλυσης αποφάσεων που έχουν ενσωματωθεί στο εργαλείο, είναι η ανάγκη ορισμού των κριτηρίων βάσει των οποίων γίνεται η σύγκριση των εναλλακτικών. Ο αποφασίζων πρέπει να έχει τη δυνατότητα να ορίσει τον αριθμό των κριτηρίων, τα ονόματα τους και το αν ανήκουν στη κατηγορία οφέλους ή κόστους. Αμέσως μόλις γίνει επιλογή της επιθυμητής μεθόδου ΠΑΑ ο χρήστης οδηγείται σε μια σελίδα που κατ' ελάχιστον προσφέρει τις παραπάνω δυνατότητες, καθώς επίσης και δυνατότητες που ορίζονται από την κάθε μέθοδο.

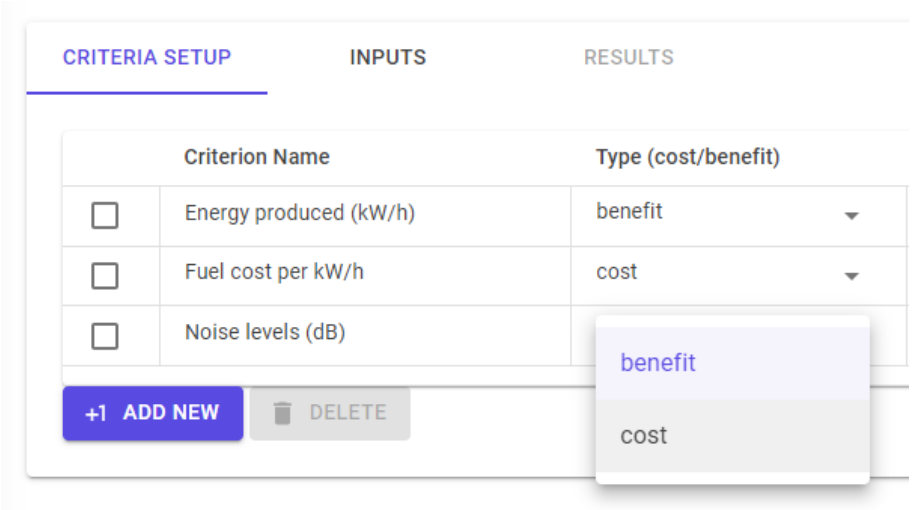


Εικόνα 3.4: Η καρτέλα ορισμού των κριτηρίων επιλογής

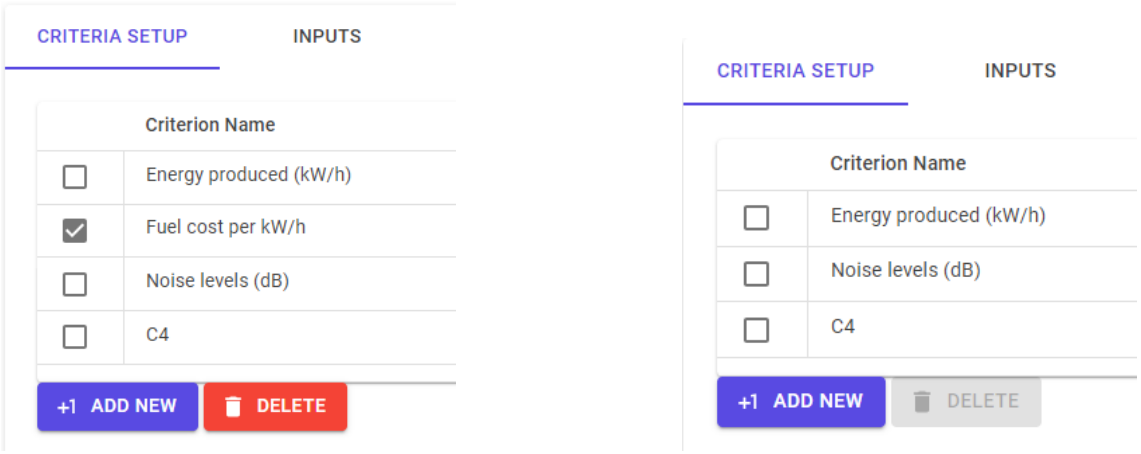
Προκειμένου να περιοριστεί ο χρήστης σε μια συγκεκριμένη σειρά εισαγωγής και τροποποίησης δεδομένων, έχει επιλεγεί η χρήση καρτελών (tabs). Η πρώτη εξ αυτών περιορίζεται στον ορισμό των κριτηρίων επιλογής και των χαρακτηριστικών τους μέσω ενός πίνακα. Κάθε του γραμμή ορίζει ένα νέο κριτήριο, ενώ κάθε του στήλη ένα χαρακτηριστικό που μπορεί να έχει διαφορετική τιμή ανά κριτήριο. Από προεπιλογή, ο πίνακας των κριτηρίων είναι προ-συμπληρωμένος με 3 κριτήρια αρχικών ονομάτων C1, C2, C3.

Η στήλη «Criterion Name» ορίζει σε ένα πεδίο κειμένου για κάθε κριτήριο το όνομά του, το οποίο μπορεί να μεταβληθεί από το χρήστη. Στη στήλη «Type (cost/benefit)» μέσα από μια αναπτυσσόμενη λίστα με δύο επιλογές, «benefit» και «cost», ορίζεται η κατηγορία στην οποία ανήκει το κριτήριο (οφέλους ή κόστους). (Εικ. 3.5)

Ο χρήστης έχει την επιλογή να προσθέσει νέα κριτήρια με το κουμπί «+1 ADD NEW», ενώ το κουμπί «DELETE» παραμένει απενεργοποιημένο ωσότου ένα ή περισσότερα κριτήρια έχουν επιλεχθεί προς διαγραφή από το αντίστοιχο κουτί επιλογής που βρίσκεται στα αριστερά του πίνακα. Η ονομασία των νέων κριτηρίων θα συνεχίσει με το ίδιο μοτίβο (C4, C5, κ.ό.κ.) ενώ οι γραμμές των διαγραμμένων κριτηρίων αφαιρούνται παντελώς από τον πίνακα. (Εικ. 3.6)



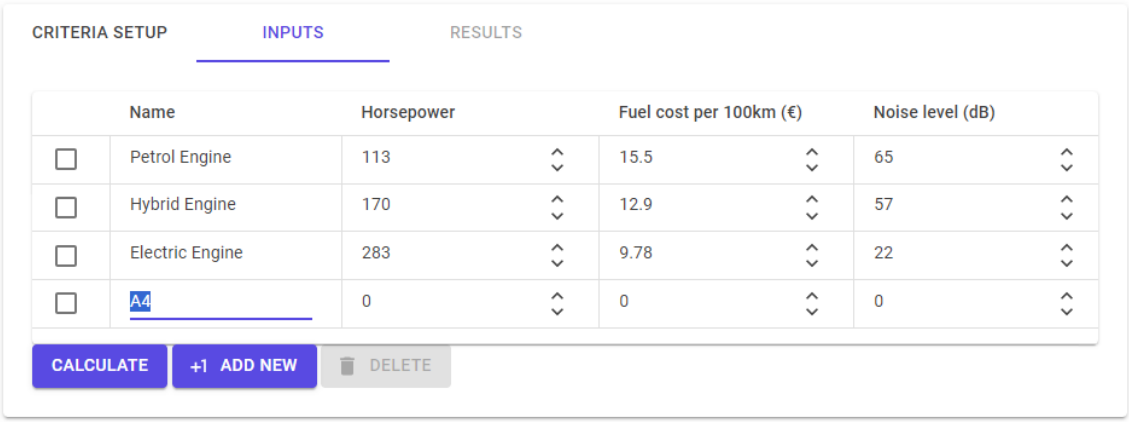
Εικόνα 3.5: Μετονομασία των κριτηρίων και επιλογή κατηγορίας οφέλους ή κόστους



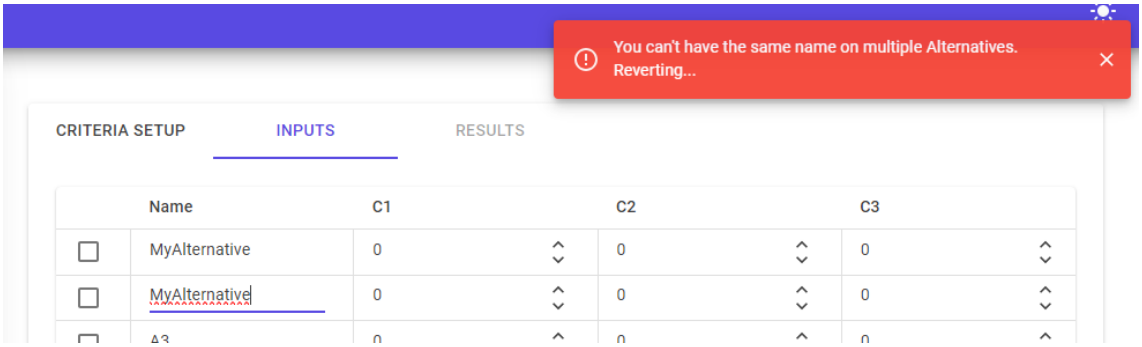
Εικόνα 3.6: Προσθήκη και διαγραφή κριτηρίων

Στην επόμενη καρτέλα «INPUTS», δίδονται στον χρήστη οι δυνατότητες προσθήκης, διαγραφής και μετονομασίας των εναλλακτικών κατ' αντίστοιχο τρόπο. (Εικ. 3.7) Σε αυτό τον πίνακα οι στήλες αποκτούν το όνομα των κριτηρίων, όπως είχαν δηλωθεί στην προηγούμενη καρτέλα. Τα κελιά του πίνακα ορίζουν την επίδοση της εκάστοτε εναλλακτικής στο αντίστοιχο κριτήριο. Οι τιμές που επιτρέπονται στα κελιά της κάθε στήλης, μπορεί να διαφέρουν, ακόμα και σε είδος, ανάλογα με τη μεθοδολογία ΠΑΑ που έχει επιλέξει ο χρήστης.

Ένας από του σκοπούς του περιβάλλοντος διεπαφής με το χρήστη είναι και να περιορίζει τα λάθη που μπορεί να γίνουν από τον τελευταίο, είτε ακούσια είτε ηθελημένα (κακόβουλος χρήστης). Προς αυτή τη κατεύθυνση έχουν υλοποιηθεί κάποιοι βασικοί έλεγχοι στα δεδομένα που εισάγονται στα κελιά των πινάκων. Πιο συγκεκριμένα ως προς τα ονόματα δεν επιτρέπεται η εισαγωγή διπλοεγγραφών ή κενών ονομάτων τόσο στα κριτήρια όσο και στις εναλλακτικές. Αν γίνει τέτοια προσπάθεια ο χρήστης θα λάβει ένα προειδοποιητικό αναδυόμενο μήνυμα σε φόντο έντονου χρώματος και το λάθος θα διορθωθεί με μια προεπιλεγμένη τιμή. (Εικ. 3.8)

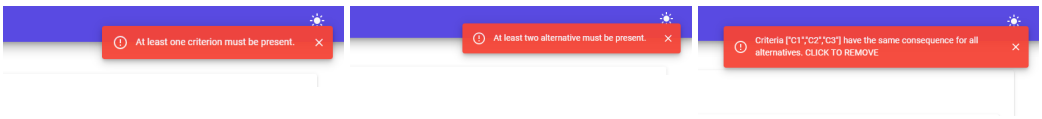


Εικόνα 3.7: Η καρτέλα ορισμού των εναλλακτικών



Εικόνα 3.8: Μήνυμα ελέγχου κατά των διπλοεγγραφών

Μόλις ολοκληρωθεί η εισαγωγή των δεδομένων από τον χρήστη το μόνο που απομένει είναι να υποβάλει τη φόρμα προς υπολογισμό πατώντας το κουμπί «CALCULATE». Τότε γίνονται και οι τελευταίοι έλεγχοι εγκυρότητας των παρεχόμενων δεδομένων. Συγκεκριμένα, η υποβολή παρεμποδίζεται αν ο χρήστης δεν έχει προσθέσει κανένα κριτήριο, αν το πλήθος των εναλλακτικών είναι μικρότερο από δύο ή αν οι επιδόσεις όλων των εναλλακτικών σε κάποιο κριτήριο είναι απόλυτα ίδιες. Ειδικά στην τελευταία περίπτωση ο χρήστης έχει τη δυνατότητα να πατήσει με το ποντίκι του το αναδυόμενο μήνυμα και να αφαιρέσει αυτόματα τα κριτήρια για τα οποία ισχύει αυτή η συνθήκη. Εφόσον, ο τελικός έλεγχος δεν εντοπίζει κάποιο λόγο μη αποδοχής των δεδομένων, αυτά αποστέλλονται για υπολογισμό και η οθόνη μεταβαίνει στη καρτέλα «RESULTS» ανάλυση της οποίας γίνεται στην επόμενη ενότητα.

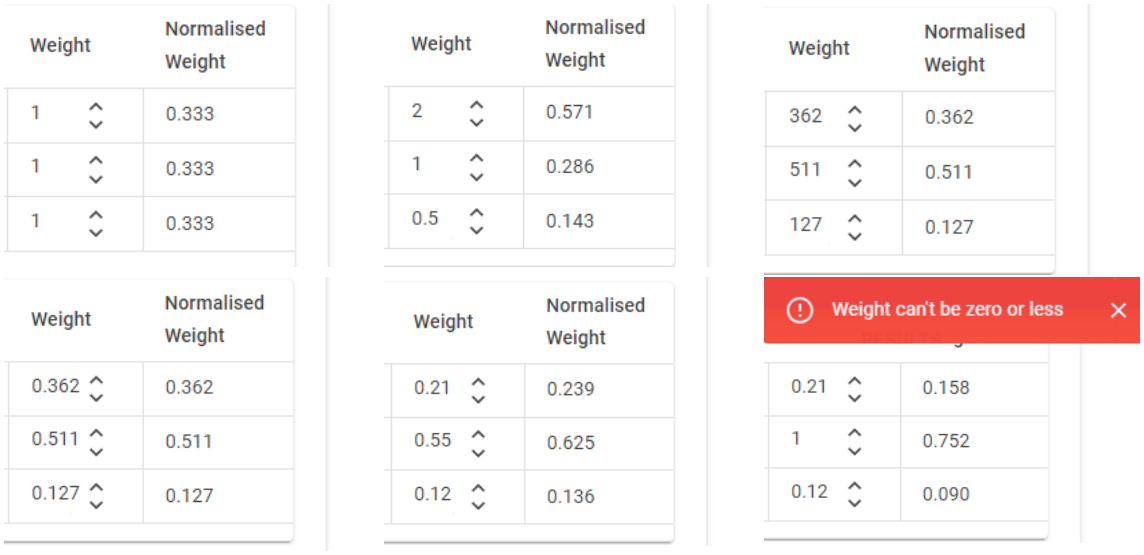


Εικόνα 3.9: Μηνύματα ελέγχου κατά την υποβολή

3.2.4 Παρουσίαση ειδικών στοιχείων του γραφικού περιβάλλοντος

Ενώ ό,τι παρουσιάστηκε στην προηγούμενη ενότητα συναντάται σε όλες τις υλοποιημένες μεθόδους ΠΑΑ, στην τρέχουσα ενότητα θα γίνει παρουσίαση των στοιχείων του γραφικού περιβάλλοντος που υλοποιήθηκαν για να καλύψουν συγκεκριμένες ανάγκες που αποτελούν ιδιαίτερο χαρακτηριστικό της κάθε μεθόδου.

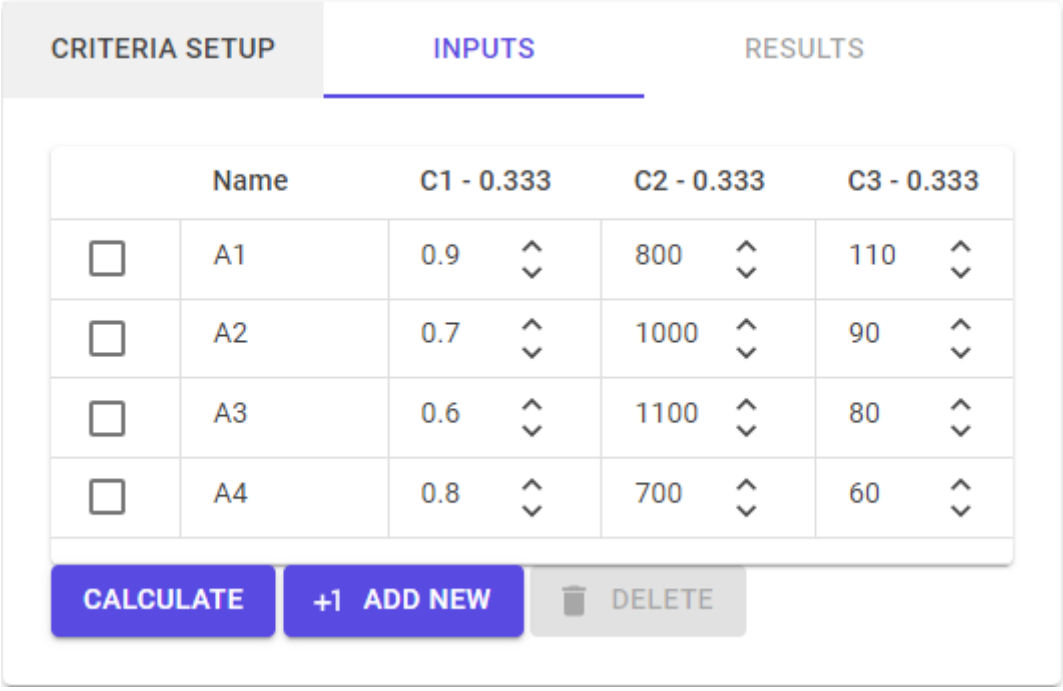
Στη κλασική μέθοδο VIKOR (2.1.), στην επέκτασή της με αριθμούς-διαστήματα (2.2.) αλλά και στη μέθοδο TOPSIS (2.7.) είναι απαραίτητο ο αποφασίζων να θέσει το βάρος του κάθε κριτηρίου. Υπενθυμίζεται από τη θεωρία η σχέση 2.1, δηλαδή ότι το κάθε βάρος είναι ένας δεκαδικός στο διάστημα $(0, 1)$ και το άθροισμα των βαρών όλων των κριτηρίων πρέπει να ισούται με τη μονάδα. Λόγω της δυσκολίας που μπορεί να προκύψει στο να επιτευχθεί αυτό με την εισαγωγή δεκαδικών αριθμών βάρους[4] έχει υλοποιηθεί ένας μηχανισμός που επιτρέπει στον χρήστη να εισάγει το σχετικό βάρος με θετικούς ακέραιους ή δεκαδικούς αριθμούς. Στη συνέχεια τα βάρη που έχουν τοποθετηθεί στην στήλη «Weight» κανονικοποιούνται και το αποτέλεσμα της κανονικοποίησης τυπώνεται στην στήλη «Normalised Weight» ή οποία δεν μπορεί να τροποποιηθεί απευθείας από τον χρήστη. Κάθε νέο κριτήριο που εισάγεται στον πίνακα ξεκινάει με σχετικό βάρος 1. Ένα κριτήριο με σχετικό βάρος 2 έχει και διπλάσιο κανονικοποιημένο βάρος από κάποιο που έχει 1. Με ανάλογο τρόπο, ένα κριτήριο με σχετικό βάρος 0.5 θα έχει το μισό κανονικοποιημένο βάρος από κάποιο που έχει την τιμή 1 στη στήλη «Weight». Στη περίπτωση που ο χρήστης το επιθυμεί, μπορεί να εισάγει απευθείας τους δεκαδικούς αριθμούς βάρους που αθροίζουν στην μονάδα στη στήλη «Weight», και θα είναι ίδιες με τιμές στη στήλη «Normalised Weight», αλλιώς θα γίνει η κανονικοποίηση για να επιτευχθεί η συνθήκη 2.1. Στις εικόνες που ακολουθούν φαίνονται παραδείγματα μετατροπής του σχετικού βάρους σε κανονικοποιημένο βάρος καθώς επίσης και το μήνυμα λάθους και η επαναφορά στην τιμή 1 σε προσπάθεια εισαγωγής μη θετικού σχετικού βάρους.



Εικόνα 3.10: Παραδείγματα κανονικοποίησης σχετικών βαρών και έλεγχος ορθότητας

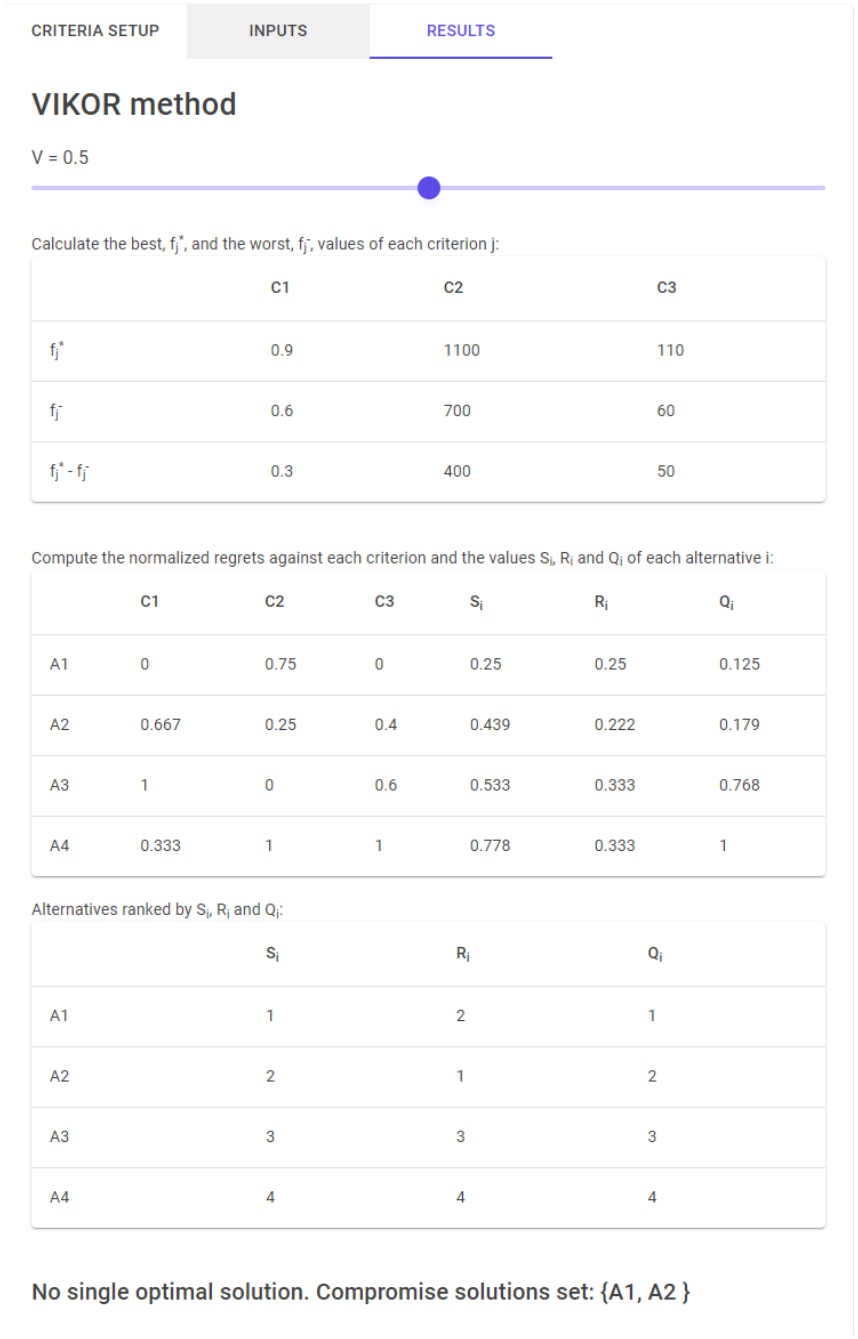
[4] π.χ. στην περίπτωση τριών ισοβαρών κριτηρίων ο χρήστης έχει ένα πεπερασμένο πλήθος χαρακτήρων για να αναπαραστήσει το $\frac{1}{3}$

Στην Εικόνα 3.11 φαίνεται ένα παράδειγμα δεδομένων πάνω στα οποία θα εφαρμοστεί η κλασική μέθοδος VIKOR. Σημειώνεται ότι στην κεφαλίδα κάθε στήλης κριτηρίου αναγράφεται το βάρος του σε δεκαδική κανονικοποιημένη μορφή, αμέσως μετά το όνομά του. Εφόσον γίνει υποβολή των δεδομένων, γίνεται η μετάβαση στην καρτέλα αποτελεσμάτων «RESULTS» που έχει τη μορφή της Εικόνας 3.12. Στη σελίδα αυτή εμφανίζονται τα στάδια επίλυσης της μεθόδου και τα ενδιάμεσα αποτελέσματα εμφανίζονται σε μορφή πινάκων προς ενημέρωση του χρήστη. Η παράμετρος V, v όπως αναφέρθηκε στην αντίστοιχη παράγραφο του Κεφαλαίου 3, είναι το βάρος που θέτει ο αποφασίζοντας στην συνολική χρησιμότητα ενάντια του βαθμού δυσαρέσκειας. Από προεπιλογή η τιμή της είναι 0.5, δηλαδή θεωρείται ουδέτερος ως προς το ρίσκο. Η παράμετρος μπορεί να μεταβληθεί με τη χρήση του οριζόντιου slider[5]. Για κάθε τέτοια μεταβολή πραγματοποιούνται εκ νέοι οι υπολογισμοί και τα νέα αποτελέσματα φαίνονται στην οθόνη. Στο κάτω μέρος της οθόνης αναγράφεται η βέλτιστη εναλλακτική, το σύνολο αποδεκτών εναλλακτικών ή η κατάταξη των εναλλακτικών ανάλογα με την μεθοδολογία που ακολουθήθηκε και τις συνθήκες που ικανοποιούνται.



Εικόνα 3.11: Πίνακας απόφασης παραδείγματος

[5]γραφικό στοιχείο ελέγχου που επιτρέπει την αλλαγή μιας τιμής μέσω της μετακίνησή του στοιχείου σε ένα συγκεκριμένο εύρος θέσεων-τιμών



Εικόνα 3.12: Η καρτέλα αποτελεσμάτων της κλασικής μεθόδου VIKOR για $v = 0.5$

Όπως φαίνεται στην Εικόνα 3.13 για $v = 0.96$ ικανοποιούνται οι συνθήκες ώστε η μέθοδος VIKOR να αναδείξει την εναλλακτική A1 ως βέλτιστη λύση, ενώ για μικρότερες τιμές v, προτείνει ως λύση το σύνολο συμβιβασμού {A1, A2}.

V = 0.96

Compute the normalized regrets against each criterion and the values S_i, R_i and Q_i of each alternative i:

	C1	C2	C3	S_i	R_i	Q_i
A1	0	0.75	0	0.25	0.25	0.01
A2	0.667	0.25	0.4	0.439	0.222	0.344
A3	1	0	0.6	0.533	0.333	0.555
A4	0.333	1	1	0.778	0.333	1

Alternatives ranked by S_i, R_i and Q_i:

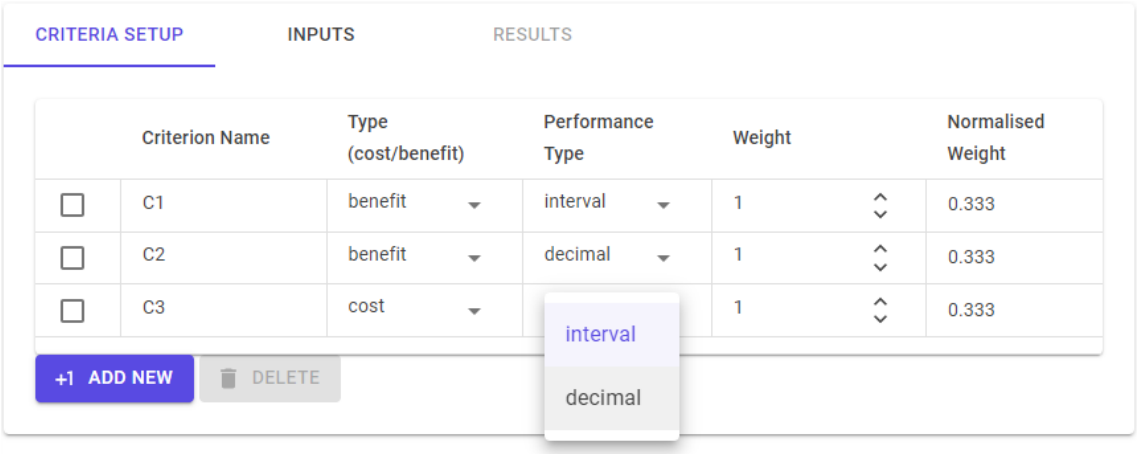
	S_i	R_i	Q_i
A1	1	2	1
A2	2	1	2
A3	3	3	3
A4	4	4	4

Optimal solution is: A1

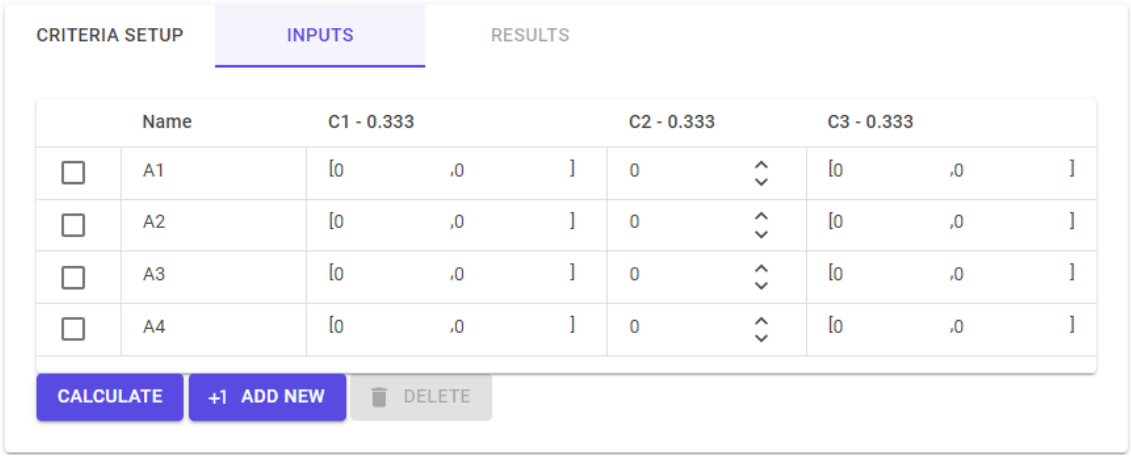
Εικόνα 3.13: Η καρτέλα αποτελεσμάτων για $v = 0.96$

Στο κεφάλαιο που έγινε η παρουσίαση των μεθοδολογιών της οικογένειας VIKOR, είδαμε ορισμένες επεκτάσεις να χρησιμοποιούν αριθμούς-διαστήματα αντί για ακριβείς αριθμούς για τις επιδόσεις των εναλλακτικών στα κριτήρια. Άλλες παραλλαγές μπορεί να χρησιμοποιούν ασαφείς αριθμούς ή χαρακτηρισμούς όπως ”μεγάλη”, ”μεσαία”, ”μικρή”. Σε κάθε περίπτωση, ο χρήστης πρέπει να μπορεί να συνδυάσει κριτήρια όπου οι επιδόσεις τους αποτελούν απλούς αριθμούς, με άλλα που είναι αριθμοί-διαστήματα ή και κάτι άλλο. Για να επιτευχθεί αυτή η πολυμορφική συμπεριφορά στον πίνακα απόφασης, μπορεί ο χρήστης από την καρτέλα των κριτηρίων να ρυθμίσει το είδος των τιμών που θέλει να δέχεται η αντίστοιχη στήλη του κάθε κριτηρίου. Στην Εικόνα 3.14 φαίνεται ο τρόπος με τον οποίο επιτυγχάνεται αυτό. Στο συγκεκριμένο παράδειγμα, ενώ βρισκόμαστε στην επέκταση της μεθόδου VIKOR για αριθμούς-διαστήματα, ο χρήστης έχει επιλέξει το κριτήριο C2 να είναι δεκαδικός αριθμός. Αντίστοιχα, η Εικόνα 3.15 δείχνει το πως αλλάζει ο πίνακας βάσει της προηγούμενης ρύθμισης.

Οι επεκτάσεις της μεθόδου VIKOR που παρουσιάστηκαν, έχουν το κοινό χαρακτηριστικό ότι οι μετρικές Q_i που προκύπτουν είναι αριθμοί-διαστήματα. Ταυτόχρονα παρουσιάστηκαν και

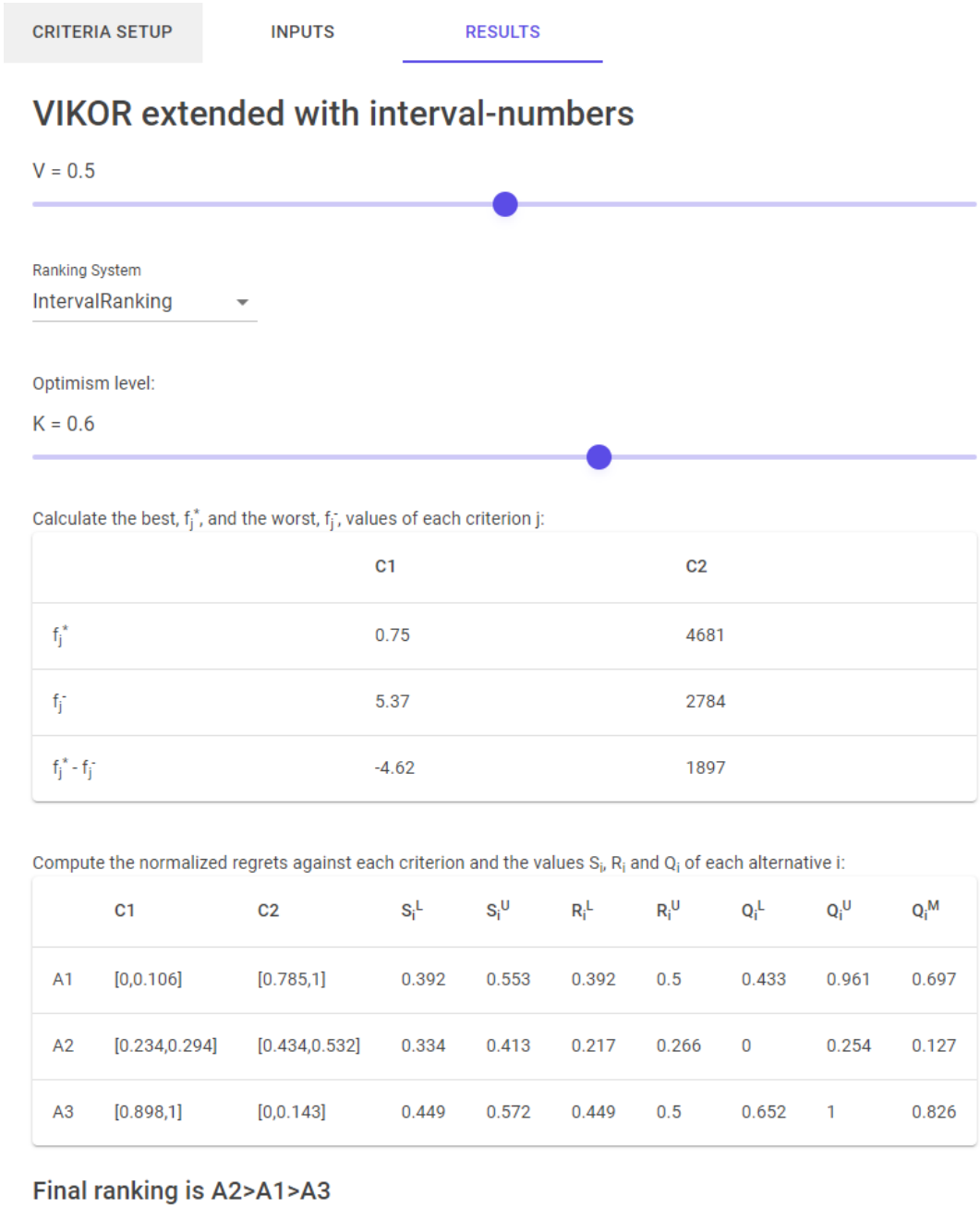


Εικόνα 3.14: Αλλαγή του τύπου επιδόσεων από αριθμό-διάστημα σε δεκαδικό αριθμό



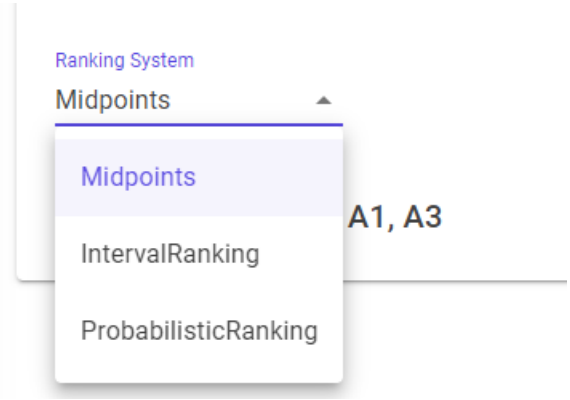
Εικόνα 3.15: Το κριτήριο C2 διαφέρει ως προς τον τύπο των επιδόσεων

κάποιες τεχνικές ταξινόμησης των εναλλακτικών βάσει αυτών των αριθμών-διαστημάτων. Στο υπολογιστικό εργαλείο έχουν υλοποιηθεί τρεις τεχνικές ταξινόμησης από της οποίες μπορεί να επιλέξει ο χρήστης μέσω ενός dropdown που βρίσκεται στο κάτω μέρος της καρτέλας αποτελεσμάτων των τεχνικών αυτών. Η Εικόνες 3.16 και 3.17 δείχνουν πως από τη καρτέλα των υπολογισμών μιας τέτοιας επέκτασης VIKOR, προσφέρεται στον χρήστη η δυνατότητα να επιλέξει την μέθοδο ταξινόμησης που επιθυμεί.



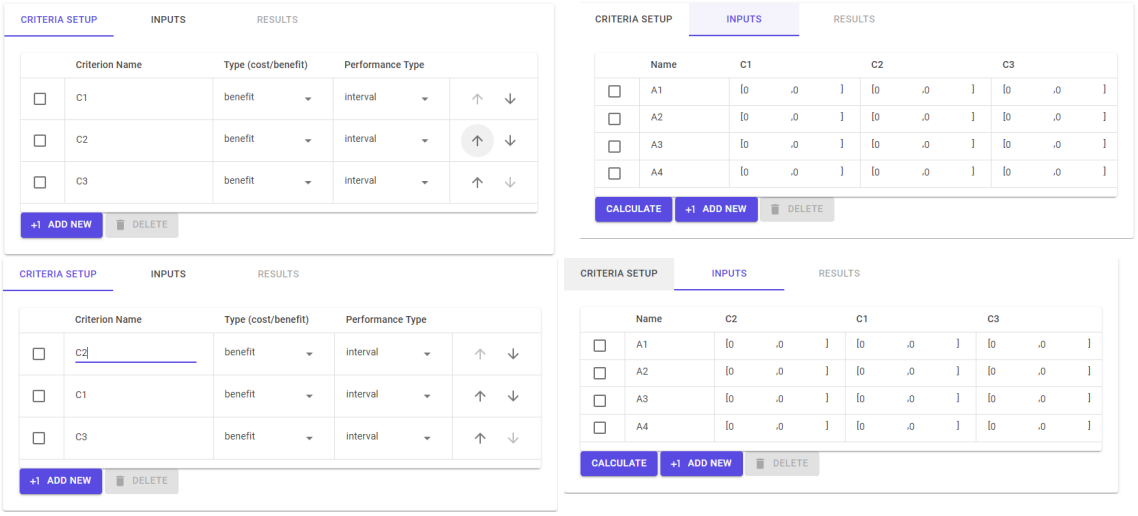
Εικόνα 3.16: Η καρτέλα αποτελεσμάτων της επέκτασης VIKOR με αριθμούς-διαστήματα για $v = 0.5$, $k = 0.6$ στη μέθοδο ταξινόμησης διαστημάτων

Στις επεκτάσεις όπου η σημαντικότητα των κριτηρίων ορίζεται από ανισότητες (ενότητα 2.3.), η σχετική προτίμηση τους ρυθμίζεται από την σειρά τους στον πίνακα κριτηρίων. Ο χρήστης μπορεί φυσικά να εισάγει εξ' αρχής τα κριτήρια στον πίνακα με τη σειρά προτίμησης, από το σημαντικότερο στο λιγότερο σημαντικό. Δίνεται επιπρόσθετα η δυνατότητα να αναδιατάξει τα κριτήρια μετά τον ορισμό τους, ακόμα και αφού συμπληρωθούν οι επιδόσεις των εναλλακτικών.



Εικόνα 3.17: Επιλογή μεθόδου ταξινόμησης των εναλλακτικών βάσει των μετρικών Q_i που είναι αριθμοί-διαστήματα

Αυτό επιτυγχάνεται με τα βοηθητικά κουμπιά σε σχήμα άνω και κάτω βέλους που βρίσκονται στη δεξιότερη στήλη του πίνακα κριτηρίων, για τις σχετικές επεκτάσεις. Κάθε πάτημα ενός βέλους θα μετακινήσει το αντίστοιχο κριτήριο κατά μια γραμμή πάνω ή κάτω στον πίνακα. Η αναδιάταξη στον πίνακα κριτηρίων θα προκαλέσει και αναδιάταξη στις στήλες του πίνακα απόφασης, διατηρώντας όμως τις ως τώρα τοποθετημένες τιμές(Εικ. 3.18).



Εικόνα 3.18: Αναδιάταξη του κριτηρίου C2 με χρήση του άνω βέλους

Κεφάλαιο 4

Ανάπτυξη του υπολογιστικού εργαλείου

4.1. Τεχνολογικό υπόβαθρο

Ένα από τα πρώτα βήματα που γίνονται κατά την έναρξη ενός νέου τεχνολογικού έργου αφορά στον καθορισμό του τεχνολογικού πλαισίου πάνω στο οποίο θα αναπτυχθεί. Πολλά χρήσιμα προγράμματα λειτουργούν, δέχονται εντολές και δεδομένα, μέσω μια γραμμής εντολών. Η εξέλιξη όμως της τεχνολογίας με έγχρωμα μέσα αναπαραγωγής εικόνας υψηλής ευκρίνειας έχουν οδηγήσει στην άνθηση των εφαρμογών με γραφικό περιβάλλον. Μια εφαρμογή με γραφικό περιβάλλον χρήστη μπορεί να έχει διάφορες μορφές, οι πιο διαδεδομένες από τις οποίες είναι: εφαρμογές για κινητές συσκευές, εφαρμογές για υπολογιστές, εφαρμογές διαδικτύου, ακόμα και τεχνολογίες εικονικής ή επαυξημένης πραγματικότητας.

Η επιλογή τεχνολογιών που βασίζονται στο διαδίκτυο προσφέρει αρκετά προτερήματα ενάντια των υπολοίπων. Τα σημαντικότερα είναι: α) η πρόσβαση από οποιαδήποτε υπολογιστική συσκευή μέσω ενός περιηγητή ιστού(web browser), κάτι που υποστηρίζεται από τις περισσότερες σύγχρονες συσκευές ανεξαρτήτως λογισμικού, β) δεν απαιτείται πρότερη εγκατάσταση, γ) εφόσον δεν γίνεται εγκατάσταση, ο χρήστης δεν περιορίζεται σε κάποια έκδοση, αλλά λαμβάνει άμεσα κάθε νεότερη έκδοση τη στιγμή που φορτώνει μια εφαρμογή, δ) μειωμένες απαιτήσεις στο σύστημα του χρήστη αφού συνήθως στις διαδικτυακές εφαρμογές τοπικά, στη συσκευή το χρήστη, σπαταλώνται πόροι μόνο για την παρουσίαση δεδομένων, ενώ οι ακριβοί υπολογισμοί γίνονται σε κάποιον απομακρυσμένο εξυπηρετητή(server). Το βασικό μειονέκτημα της προσέγγισης αυτής είναι ότι μια τέτοια εφαρμογή απαιτεί συνεχή πρόσβαση της συσκευής στο διαδίκτυο προκειμένου να μεταφέρει δεδομένο από και προς τους εξυπηρετητές.

Βάσει της προηγούμενης σύγκρισης, επιλέχθηκε η χρήση τεχνολογιών διαδικτύου για την ανάπτυξη του παρόντος υπολογιστικού εργαλείου. Η αρχιτεκτονική του παραδοτέου συστήματος ακολουθεί κάποια σύγχρονα πρότυπα ανάπτυξης εφαρμογών, όπως είναι ο διαχωρισμός σε microservices. Ένα microservice είναι μια μικρή εφαρμογή που μπορεί να αναπτυχθεί ανεξάρτητα, να κλιμακωθεί ανεξάρτητα και να δοκιμαστεί ανεξάρτητα και έχει μία μόνο ευθύνη. Μια μόνον ευθύνη με την έννοια ότι έχει έναν μόνο λόγο για αλλαγή ή/και έναν μόνο λόγο για αντικατάσταση, αλλά και με την έννοια ότι κάνει ένα και μόνο ένα πράγμα και αυτό μπορεί να γίνει εύκολα κατανοητό [51]. Η ανεξάρτητη ανάπτυξη των microservices επιτρέπει στις ομάδες να επι-

τύχουν καλύτερο χρόνο παράδοσης του συστήματος στην αγορά μέσω ταχύτερων και πιο συνεχών παραδόσεων νέων εκδόσεων [52]. Επιπλέον, επιτρέπει την επιλογή διαφορετικών τεχνολογιών (γλώσσες προγραμματισμού, development frameworks κ.α.) για ανάπτυξή τους. Χρησιμοποιώντας τα οφέλη που προσφέρει η τεχνική των microservices, έχουν χρησιμοποιηθεί δύο διαφορετικές τεχνολογίες για την υλοποίηση του εργαλείου.

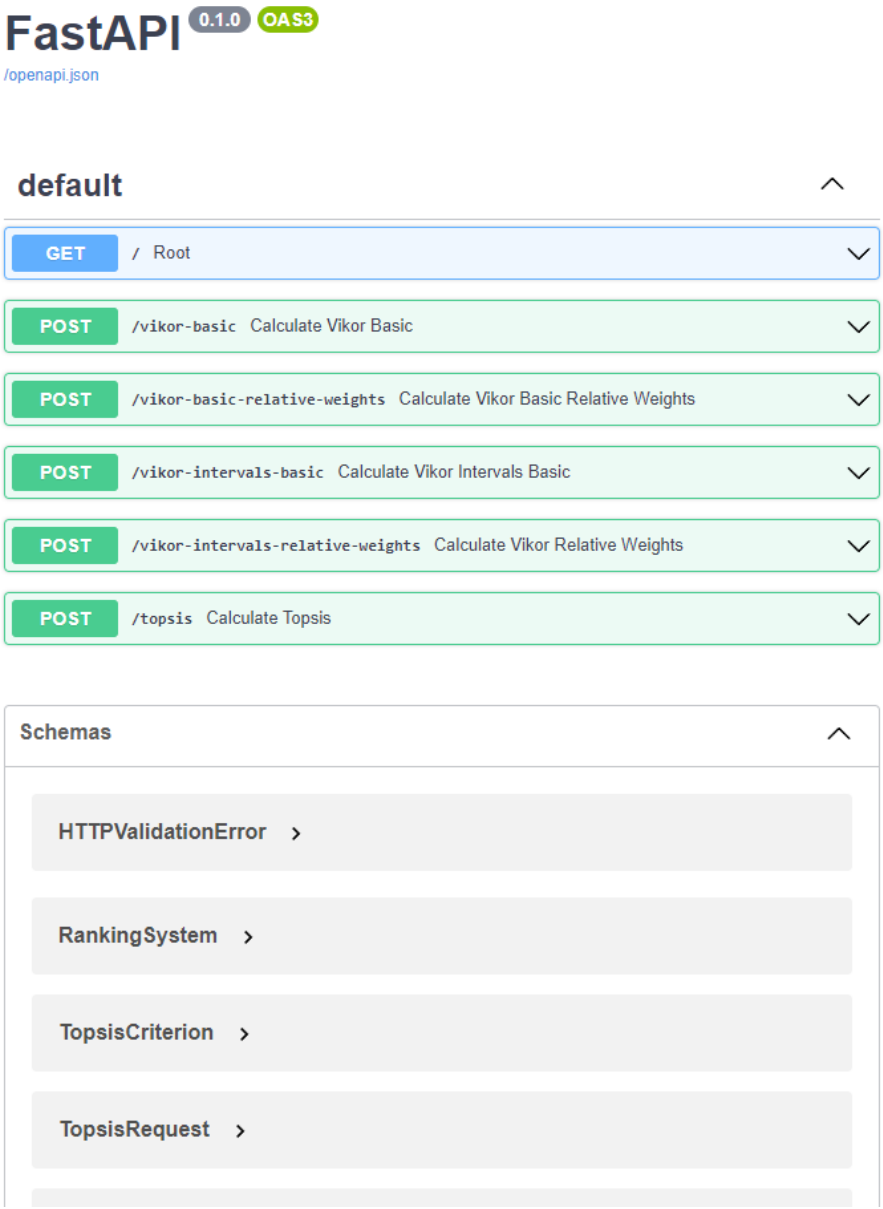
Καθώς αναφερόμαστε σε ένα εργαλείο που σκοπό έχει την χρήση αριθμητικών δεδομένων εισόδου, την επεξεργασία τους και την εξαγωγή κάποιων αποτελεσμάτων, για το backend επιλέχθηκε ως γλώσσα ανάπτυξης η γλώσσα προγραμματισμού Python [1]. Η Python είναι μια ταχέως αναπτυσσόμενη γλώσσα προγραμματισμού [53] που έχει γίνει δημοφιλής για διάφορους λόγους, συμπεριλαμβανομένης της απλής της σύνταξης που μοιάζει με ψευδοκώδικα, της ευέλικτης δόμησης, του αντικειμενοστραφούς σχεδιασμού που υποστηρίζει αλλά κυρίως για την παρουσία ανεπτυγμένων αριθμητικών βιβλιοθηκών στο οικοσύστημά της, που επιτρέπουν την αποτελεσματική αποθήκευση και χειρισμό τεράστιων ποσοτήτων αριθμητικών πληροφοριών κάνοντάς την κατάλληλη για ανάλυση δεδομένων και εφαρμογή στον επιστημονικό τομέα [54].

Καθώς η δυνατότητα επικοινωνίας με άλλα συστήματα ή υποσυστήματα είναι απαραίτητη στις αρχιτεκτονικές που χρησιμοποιούν microservices, επιπλέον του κώδικα που εκτελεί τους απαραίτητους υπολογισμούς για την εξαγωγή αποτελεσμάτων, είναι απαραίτητη και η υλοποίηση του πλαισίου επικοινωνίας. Για τον λόγο αυτό χρησιμοποιήθηκε ένα απλό API [2] που εξυπηρετεί αιτήματα που μεταφέρονται μέσω HTTP πρωτοκόλλου. Για το λόγο αυτό χρησιμοποιήθηκαν δύο βιβλιοθήκες, το FastAPI [3] και το Uvicorn[4]. Το FastAPI είναι ένα πακέτο ανάπτυξης web APIs υψηλής απόδοσης για την γλώσσα Python. Χρησιμοποιείται για τον ορισμό των σημείων εξυπηρέτησης του API, τα μοντέλα αιτημάτων και απόκρισης και άλλες λογικές. Επιπλέον, παρέχει λειτουργίες όπως η αυτόματη δημιουργία τεκμηρίωσης OpenAPI [5] (παλαιότερα γνωστή ως Swagger) (Εικ. 4.1), η επικύρωση και η σειριοποίηση δεδομένων και το dependency injection[6]. Το ίδιο το FastAPI δεν είναι διακομιστής ιστού, αλλά βασίζεται στο ASGI (Asynchronous Server Gateway Interface) για να χειρίζεται αιτήματα HTTP. Για το λόγο αυτό χρησιμοποιείται σε συνδυασμό με έναν διακομιστή ASGI όπως το Uvicorn για την εξυπηρέτηση διαδικτυακών εφαρμογών. Το Uvicorn είναι υπεύθυνο για το χειρισμό των εισερχόμενων αιτημάτων HTTP και τη διαχείριση της ασύγχρονης εκτέλεσης της εφαρμογής.

Το σύστημα διεπαφής χρήστη (FrontEnd)

Για πολλά χρόνια, η γλώσσα JavaScript ήταν η μόνη επιλογή για τη δημιουργία διαδραστικών εφαρμογών που τρέχουν σε web browsers και ειδικά η ανάπτυξη εφαρμογών υψηλών απαιτήσεων σε επεξεργαστική ισχύ (CPU), όπως τα παιχνίδια, είχε ανασταλεί λόγω της μειωμένης ταχύτητας επεξεργασίας που προσφέρει η JavaScript. Για να διορθωθεί αυτό, στο παρελθόν έχουν γίνει αρκετές προσπάθειες από εταιρείες όπως η Microsoft, η Adobe και η Google να πετύχουν επιδόσεις εγγενούς κώδικα σε περιβέλλον web. Ωστόσο, όλες οι προσπάθειες απέτυχαν να γίνουν ευρέως

[1] συγκεκριμένα η έκδοση 3.11 κατά την συγγραφή του παρόντος
[2] API, από το application programming interface (Διεπαφή προγραμματισμού εφαρμογών)
[3] https://fastapi.tiangolo.com/
[4] https://www.uvicorn.org/
[5] https://www.openapis.org/
[6] https://en.wikipedia.org/wiki/Dependency_injection



Εικόνα 4.1: Η αυτόματα παραγόμενη σελίδα τεκμηρίωσης του API που βρίσκεται στην διεύθυνση http://{host}:{port}/docs

αποδεκτές καθώς απαιτούσαν το δέσιμο σε συγκεκριμένη πλατφόρμα, (λειτουργικού ή browser). Τον Μάρτιο του 2017 δημοσιεύτηκε η πρώτη έκδοση της WebAssembly, εν συντομία wasm, ως τυποποιημένη και ανεξάρτητη από πλατφόρμα εναλλακτική. Μέσα σε λίγους μήνες, υλοποιήθηκε και στους τέσσερις μεγαλύτερους περιηγητές ιστού και έκτοτε κέρδισε περισσότερο έδαφος, καθώς η γλώσσα χαμηλού επιπέδου bytecode επιτρέπει σημαντικά ταχύτερη μετάδοση, ανάγνωση και εκτέλεση σε σύγκριση με την JavaScript [55].

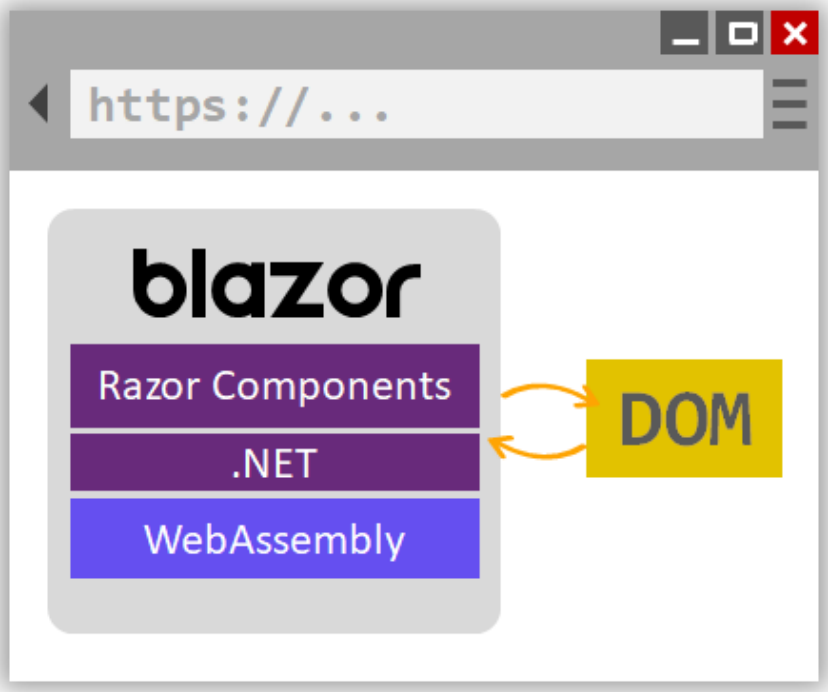
Η νέα αυτή τεχνολογία προκάλεσε την ανάπτυξη πολλών νέων web frameworks που βασίζονται σε αυτή, ένα από αυτά είναι το Blazor WebAssembly[7] που χρησιμοποιήθηκε για την υλοποίηση του συστήματος διεπαφής στο παρόν υπολογιστικό εργαλείο. Το Blazor WebAssembly είναι ένα framework κατάλληλο για κατασκευή διαδραστικών SPA[8] εφαρμογών που τρέχουν στην πλευρά του πελάτη, στον περιηγητή, και είναι γραμμένες σε .NET[9] (ή dotnet)[1]. Το .NET είναι ένα ελεύθερο, ανοιχτού κώδικα, πλαίσιο ανάπτυξης εφαμοργών που αναπτύχθηκε από τη Microsoft και είναι διαθέσιμο στα περισσότερα λειτουργικά συστήματα. Παρέχει ένα σύνολο εργαλείων, βιβλιοθηκών και περιβαλλόντων εκτέλεσης που επιτρέπουν στους προγραμματιστές να δημιουργήσουν ένα ευρύ φάσμα εφαρμογών, συμπεριλαμβανομένων εφαρμογών ιστού, εφαρμογών υπολογιστή, εφαρμογών για φορητές συσκευές, υπηρεσιών που βασίζονται σε τεχνολογίες νέφους (cloud computing) και πολλά άλλα. Το .NET υποστηρίζει πολλές γλώσσες προγραμματισμού, με τις C#, F# και VB.NET να είναι οι πιο συχνά χρησιμοποιούμενες γλώσσες. Οι εφαρμογές γραμμένες σε .NET μεταγλωττίζονται σε μια ενδιάμεση γλώσσα που ονομάζεται Common Intermediate Language (CIL) ή Microsoft Intermediate Language (MSIL). Το Common Language Runtime (CLR) είναι υπεύθυνο για την εκτέλεση αυτής της ενδιάμεσης γλώσσας. Δεδομένου ότι το CLR είναι διαθέσιμο σε πολλές πλατφόρμες, είναι δυνατόν η εφαρμογή να μεταγλωττιστεί μια φορά και να εκτελεστεί σε οποιαδήποτε πλατφόρμα έχει συμβατή υλοποίηση του CLR.

Σύμφωνα με το άρθρο [1] ο κώδικας WebAssembly μπορεί να έχει πρόσβαση στην πλήρη λειτουργικότητα του προγράμματος περιήγησης μέσω JavaScript, που ονομάζεται JavaScript interoperability, εν συντομία JavaScript interop ή JS interop. Ο κώδικας .NET που εκτελείται μέσω WebAssembly στον browser, τρέχει σε sandboxed[10] περιβάλλον αυτού και παρέχει προστασία έναντι κακόβουλων ενεργειών στον υπολογιστή του χρήστη. Τα βήματα για την εκτέλεση μιας εφαρμογής γραμμένης σε blazor είναι:

- Ο κώδικας C# μεταγλωττίζεται στην CIL του .NET και τοποθετείται σε πακέτα
- Τα παραπάνω πακέτα και το CLR κατεβάζονται στον περιηγητή του χρήστη.
- Η WebAssembly του Blazor ξεκινάει το Runtime και εκτελεί σε αυτό τα πακέτα που περιέχουν τον κώδικα της της εφαρμογής.
- Το περιβάλλον εκτέλεσης της Blazor WebAssembly χρησιμοποιεί JavaScript interop προκειμένου να διαχειριστεί της αλλαγές στο DOM[11] (Εικ. 4.2) της σελίδας και να αλληλεπιδράσει με το API του περιηγητή.

Το μέγεθος της μεταγλωττισμένης εφαρμογής, που φτάνει δηλαδή στο μηχάνημα του χρήστη, είναι ένας κρίσιμος παράγοντας απόδοσης για τη χρηστικότητα μιας εφαρμογής. Μια μεγάλη εφαρμογή χρειάζεται σχετικά μεγάλο χρονικό διάστημα για τη λήψη της στον browser, γεγονός που μειώνει την εμπειρία του χρήστη. Η Blazor WebAssembly βελτιστοποιεί το μέγεθος της τελικής εφαρμογής για να μειώσει τους χρόνους λήψης:

[7] https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-7.0#blazor-webassembly
[8] https://en.wikipedia.org/wiki/Single-page_application
[9] https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet
[10] https://en.wikipedia.org/wiki/Sandbox_(computer_security)
[11] https://en.wikipedia.org/wiki/Document_Object_Model



Εικόνα 4.2: Λειτουργία του Blazor μέσα στον browser. Πηγή: https://learn.microsoft.com/ [1]

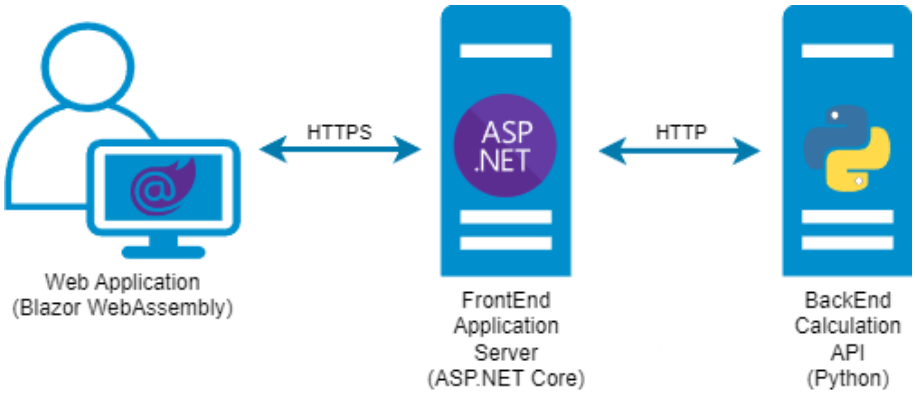
- Ο αχρησιμοποίητος κώδικας αφαιρείται από την εφαρμογή όταν δημοσιεύεται από τον Intermediate Language (IL) Trimmer.
- Τα HTTP μηνύματα συμπιέζονται.
- Το περιβάλλον εκτέλεσης .NET και τα πακέτα κώδικα της εφαρμογής αποθηκεύονται για κάποιο διάστημα στον browser.

Τα απαραίτητα αρχεία για την εκτέλεση της Blazor εφαρμογής (Blazor wasm, το runtime και τα πακέτα επέκτασης .dll με τον κώδικα της εφαρμογής) μπορεί να σερβίρει οποιοσδήποτε διακομιστής ιστού που έχει τη δυνατότητα να σερβίρει στατικά αρχεία. Ωστόσο, είναι βολικό να χρησιμοποιηθεί ένα ASP.NET Core application [12] ως server της εφαρμογής. Μια τέτοια εφαρμογή περιέχει εντός της ίδιας διεργασίας την υλοποίηση ενός HTTP server ο οποίος ακούει τα αιτήματα HTTP και τα προωθεί στην εφαρμογή ως ένα αντικείμενο. Η προεπιλεγμένη υλοποίηση του HTTP server που έρχεται εντός των ASP.NET core εφαρμογών λέγεται Kestrel.

[12] https://learn.microsoft.com/en-us/aspnet/core/fundamentals/servers/?view=aspnetcore-7.0&tabs=windows

4.2. Αρχιτεκτονική παρουσίαση

Στο παρακάτω διάγραμμα (Εικ. 4.3) παρουσιάζεται η αρχιτεκτονική του υπολογιστικού εργαλείου που κατασκευάστηκε. Ο χρήστης αποκτά πρόσβαση στο σύστημα μέσω του περιηγητή του συστήματος του. Επικοινωνεί με τον server της εφαρμογής που στη προκειμένη περίπτωση είναι μια ASP.NET εφαρμογή και εξυπηρετείται από τον Kestrel που τρέχει εντός της διεργασίας αυτής. Μέσω πολλαπλών αιτημάτων HTTP πρωτοκόλλου σερβίρει τα απαραίτητα αρχεία για την εκτέλεση της εφαρμογής του γραφικού περιβάλλοντος στον browser. Όταν ο χρήστης έχει ολοκληρώσει την εισαγωγή των δεδομένων που επιθυμεί για υπολογισμό, και υποβάλει την φόρμα του, το αίτημά του στέλνετε και πάλι μέσω HTTP αιτήματος πίσω στον application server. Ο δεύτερος είναι υπεύθυνος να μεταβιβάσει το αίτημα, είτε ως έχει, είτε με κάποιες μετατροπές αν απαιτούνται, στο microservice που εκτελεί τους υπολογισμούς για τις μεθόδους πολυκριτήριας ανάλυσης.



Εικόνα 4.3: Αρχιτεκτονικό διάγραμμα του υπολογιστικού εργαλείου

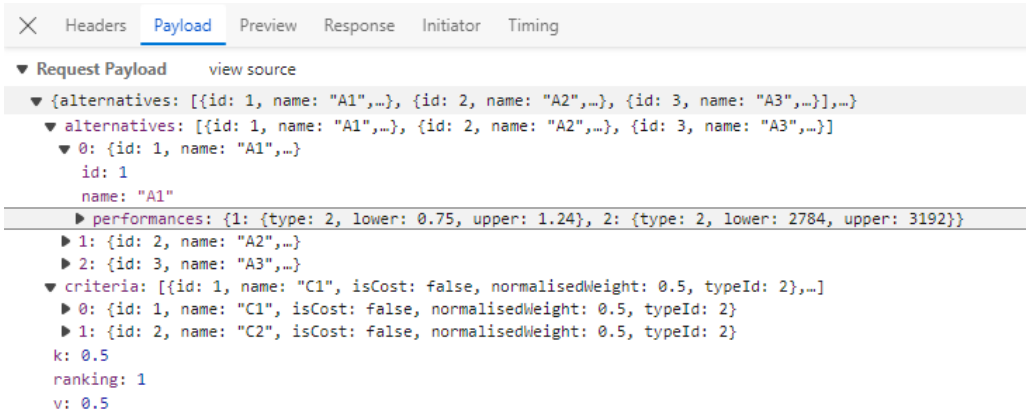
Επιβεβαιώνεται από την παρακάτω εικόνα 4.4 που δείχνει ίχνη από το αρχείο καταγραφής των δύο εφαρμογών στην κονσόλα τους η διαδρομή που ακολουθεί το αίτημα για υπολογισμό που πραγματοποιείται από την Blazor εφαρμογή. Ως «mcdm-frontend» αναφέρεται η application server εφαρμογή γραμμένη σε ASP.NET Core ενώ ως «mdcm-backend» η Python εφαρμογή, το υπολογιστικό API. Συνοπτικά περιγράφεται:

- Λήψη του HTTP POST αιτήματος στην ASP.NET Core εφαρμογή η οποία ακούει στην τοπική πόρτα 8002
- Από εκεί εκκίνηση νέου HTTP POST αιτήματος, προς την τοπική πόρτα 8001 που ακούει η Python εφαρμογή
- Λήψη του δεύτερου HTTP POST αιτήματος από την Python εφαρμογή. Εδώ εννοείται η ολοκλήρωση των απαραίτητων υπολογισμών, και στη συνέχεια δίνεται απάντηση με κωδικό 200 (OK)
- Λήψη της απάντησης από την ASP.NET Core εφαρμογή
- Ολοκλήρωση του πρώτου HTTP POST αιτήματος και επιστροφή της πληροφορίας στον browser του χρήστη.

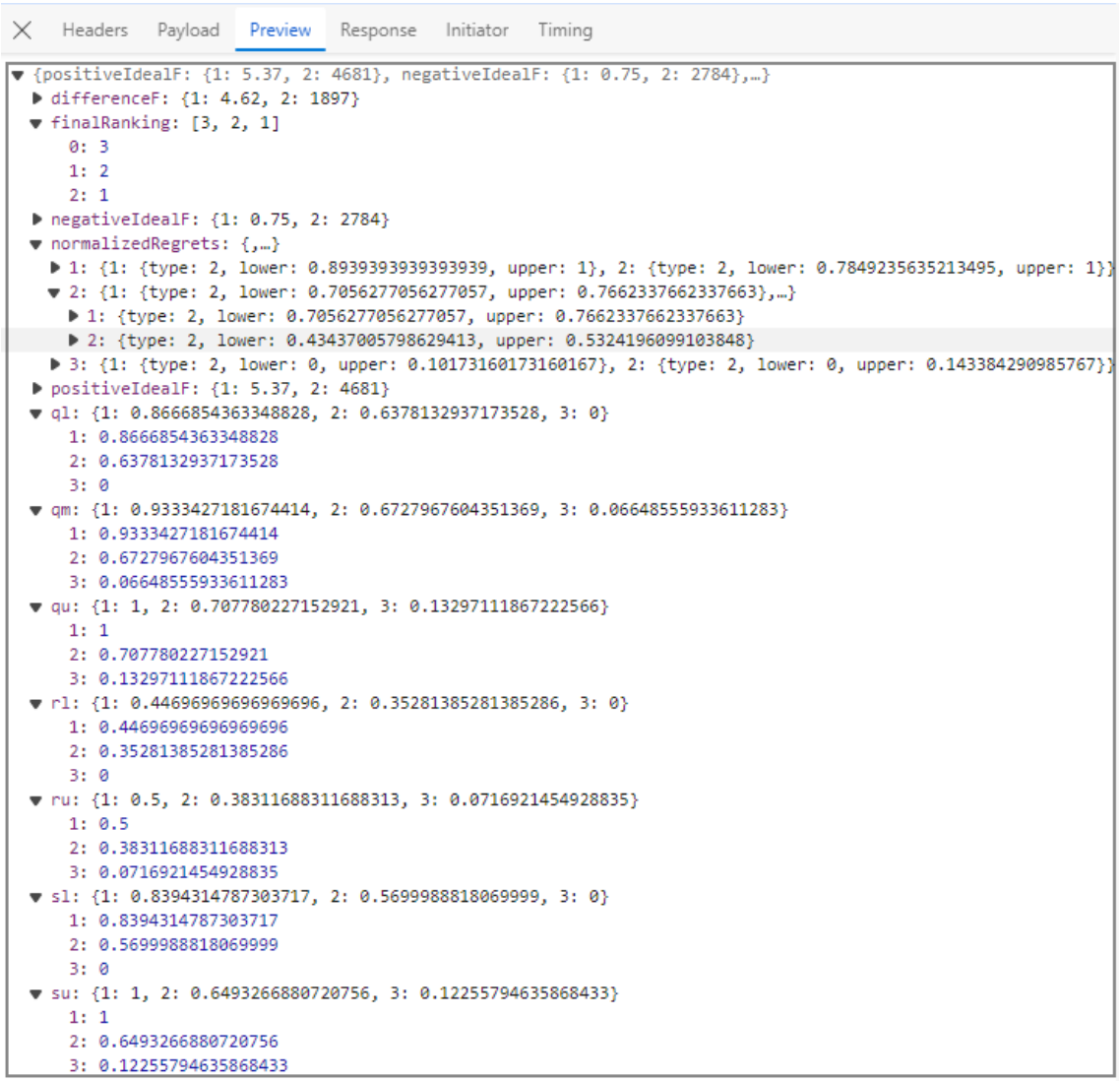
```
mcdm-frontend   | info: Microsoft.Hosting.Lifetime[0]
mcdm-frontend   |       Hosting environment: Production
mcdm-frontend   | info: Microsoft.Hosting.Lifetime[0]
mcdm-frontend   |       Content root path: /app
mcdm-frontend   | info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
mcdm-frontend   |       Request starting HTTP/1.1 POST http://localhost:8002/api/VikorIntervalsBasic application/json;+charset=utf-8
558
mcdm-frontend   | warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
mcdm-frontend   |       Failed to determine the https port for redirect.
mcdm-frontend   | info: Microsoft.AspNetCore.Routing.EndpointMiddleware[0]
mcdm-frontend   |       Executing endpoint 'MCDM.Server.Controllers.VikorIntervalsBasicController.Get (MCDM.Server)'
mcdm-frontend   | info: Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker[102]
mcdm-frontend   |       Route matched with {action = "Get", controller = "VikorIntervalsBasic", page = ""}. Executing controller acti
on with signature System.Threading.Tasks.Task`1[MCDM.Shared.VikorIntervalsBasic.VikorIntervalsBasicResponse] Get(MCDM.Shared.VikorIn
tervalsBasic.VikorIntervalsBasicRequest) on controller MCDM.Server.Controllers.VikorIntervalsBasicController (MCDM.Server).
mcdm-frontend   | info: System.Net.Http.HttpClient.mcdmCalculationApi.LogicalHandler[100]
mcdm-frontend   |       Start processing HTTP request POST http://mcdm-backend:8001/vikor-intervals-basic
mcdm-frontend   | info: System.Net.Http.HttpClient.mcdmCalculationApi.ClientHandler[100]
mcdm-frontend   |       Sending HTTP request POST http://mcdm-backend:8001/vikor-intervals-basic
mcdm-backend    | alternatives=[PolymorphicAlternative(id=1, name='A1', performances={1: IntervalPerformance(type=<PerformanceType.In
terval: 2>, lower=0.75, upper=1.24), 2: IntervalPerformance(type=<PerformanceType.Interval: 2>, lower=2784.0, upper=3192.0)}), Polym
orphicAlternative(id=2, name='A2', performances={1: IntervalPerformance(type=<PerformanceType.Interval: 2>, lower=1.83, upper=2.11),
 2: IntervalPerformance(type=<PerformanceType.Interval: 2>, lower=3671.0, upper=3857.0)}), PolymorphicAlternative(id=3, name='A3', p
erformances={1: IntervalPerformance(type=<PerformanceType.Interval: 2>, lower=4.9, upper=5.37), 2: IntervalPerformance(type=<Perform
anceType.Interval: 2>, lower=4409.0, upper=4681.0)})] criteria=[PolymorphicWeightedCriterion(id=1, name='C1', isCost=False, normalis
edWeight=0.5, typeId=<PerformanceType.Interval: 2>), PolymorphicWeightedCriterion(id=2, name='C2', isCost=False, normalisedWeight=0.
5, typeId=<PerformanceType.Interval: 2>)] v=0.5 k=0.5 ranking=<RankingSystem.Midpoints: 1>
mcdm-backend    | INFO:     172.19.0.3:41202 - "POST /vikor-intervals-basic HTTP/1.1" 200 OK
mcdm-frontend   | info: System.Net.Http.HttpClient.mcdmCalculationApi.ClientHandler[101]
mcdm-frontend   |       Received HTTP response headers after 62.9364ms - 200
mcdm-frontend   | info: System.Net.Http.HttpClient.mcdmCalculationApi.LogicalHandler[101]
mcdm-frontend   |       End processing HTTP request after 73.4958ms - 200
mcdm-frontend   | info: Microsoft.AspNetCore.Mvc.Infrastructure.ObjectResultExecutor[1]
mcdm-frontend   |       Executing ObjectResult, writing value of type 'MCDM.Shared.VikorIntervalsBasic.VikorIntervalsBasicResponse'.
mcdm-frontend   | info: Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker[105]
mcdm-frontend   |       Executed action MCDM.Server.Controllers.VikorIntervalsBasicController.Get (MCDM.Server) in 243.3024ms
mcdm-frontend   | info: Microsoft.AspNetCore.Routing.EndpointMiddleware[1]
mcdm-frontend   |       Executed endpoint 'MCDM.Server.Controllers.VikorIntervalsBasicController.Get (MCDM.Server)'
mcdm-frontend   | info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
mcdm-frontend   |       Request finished HTTP/1.1 POST http://localhost:8002/api/VikorIntervalsBasic application/json;+charset=utf-8
558 - 200 - application/json;+charset=utf-8 357.1832ms
```

Εικόνα 4.4: Συγκεντρωμένα ίχνη από το αρχείο καταγραφής των εφαρμογών

Από την πλευρά του browser μπορούμε να ελέγξουμε το HTTP αίτημα που διενεργείται καθώς και την απάντηση που λαμβάνει. Στην εικόνα 4.5 βλέπουμε τα χαρακτηριστικά των κριτηρίων, όπως το όνομα, το βάρος τους, αν είναι κριτήρια κόστους και τα ονόματα των εναλλακτικών με τις επιδόσεις τους σε κάθε κριτήριο. Στην εικόνα 4.6 φαίνεται το Json φορτίο της απάντησης που λαμβάνει ο περιηγητής. Εκτός της ταξινομημένης λίστας «finalRanking» που περιέχει τα αναγνωριστικά των εναλλακτικών σε αυτό το πρόβλημα απόφασης και τα κατατάσσει βάσει της μεθόδου ΠΑΑ που έχει επιλεχθεί, επιστέφονται και κάποια επιπλέον πεδία για κάθε εναλλακτική που προκύπτουν από ενδιάμεσους υπολογισμούς και μπορούν να εκτυπωθούν στην οθόνη του χρήστη.



Εικόνα 4.5: Το Φορτίο Json που περιέχεται σε ένα αίτημα του περιηγητή για υπολογισμό της κλασικής μεθόδου VIKOR



Εικόνα 4.6: Το Φορτίο Json που λαμβάνει ως απάντηση στο αίτημα της εικόνας 4.5

4.3. Περιβάλλον ανάπτυξης

Όλες οι τεχνολογίες που χρησιμοποιήθηκαν κατά την υλοποίηση του παρόντος εργαλείου είναι cross-platform, δηλαδή ανεξάρτητες πλατφόρμας, ανεξάρτητες λειτουργικού συστήματος. Στην πράξη αυτό σημαίνει ότι μπορούν να χρησιμοποιηθούν για ανάπτυξη και εγκατάσταση στα πλέον γνωστότερα λειτουργικά συστήματά υπολογιστών. Από επιλογή, η ανάπτυξη έγινε σε προσωπικό υπολογιστή λειτουργικού συστήματος Windows.

Για την ανάπτυξη του FrontEnd που περιλαβάνει το Blazor WebApplication και τον ASP.NET Core application server είναι απαραίτητη η εγκατάσταση του ενός .NET SDK (software development kit), έκδοσης 7.0[13] ή και μεταγενέστερης, ανάλογα με την συμβατότητα που θα ισχύει στο μέλλον. Ως IDE (integrated development environment) χρησιμοποιήθηκε το Visual Studio 2022 Community Edition[14] που είναι δωρεάν για ακαδημαϊκή χρήση, για συγγραφή ανοιχτού κώδικα και για ιδιώτες. Ενώ το .NET SDK διατίθεται πλέον ως πλαίσιο ανάπτυξης σε διάφορα λειτουργικά συστήματα, περιλαμβανομένων και των Linux, το συγκεκριμένο IDE δεν είναι διαθέσιμο σε αυτά. Ως εναλλακτική, για περιβάλλον Linux, προτείνεται η χρήση του Visual Studio Code με επιπλέον εγκατάσταση κάποιου .NET/C# extension ώστε να διευκολύνει στην ανάπτυξη και αποσφαλμάτωση (debugging).

Για την ανάπτυξη του υπολογιστικού BackEnd σε γλώσσα προγραμματισμού Python είναι απαραίτητη η εγκατάσταση κάποιας σύγχρονής έκδοσης της γλώσσας, προτείνεται 3.11 ή μεταγενέστερη. Ως IDE χρησιμοποιήθηκε το Visual Studio Code με επιπλέον εγκατάσταση ενός Python extension που προσθέτει λειτουργίες αυτόματης συμπλήρωσης εντολών, έλεγχος κώδικα, αποσφαλμάτωσης, μορφοποίησης και αναδιάρθρωσης κώδικα κ.ά.

Επιπλέον για την προσομοίωση της εγκατάστασης σε ένα παραγωγικό σύστημα χρειάστηκε εγκατάσταση και χρήση ενός συστήματος Docker[15]. Το Docker είναι μια πλατφόρμα ή εργαλείο που σχεδιάστηκε για να διευκολύνει τη δημιουργία, την ανάπτυξη και την εκτέλεση εφαρμογών μέσα από containers. Τα containers είναι μια μορφή εικονικοποίησης (virtualization) που επιτρέπει την τοποθέτηση του κώδικα μιας εφαρμογής και όλων των εξαρτήσεών της (όπως βιβλιοθήκες, ρυθμίσεις περιβάλλοντος, κλπ.) σε μια αυτόνομη μονάδα. Το Docker παρέχει ένα τυποποιημένο τρόπο δημιουργίας, διαχείρισης και εκτέλεσης αυτών των containers, κάτι που το καθιστά πολύ δημοφιλές στην σύγχρονη ανάπτυξη και παραγωγική εγκατάσταση συστημάτων λογισμικού.

[13] https://dotnet.microsoft.com/en-us/download/dotnet/7.0
[14] https://visualstudio.microsoft.com/free-developer-offers/
[15] https://docs.docker.com/engine/install/ubuntu/

4.4. Εγκατάσταση σε παραγωγικό σύστημα

Προκειμένου να φτάσει μια δικτυακή εφαρμογή στα χέρια των τελικών της χρηστών θα πρέπει η πρώτη να είναι προσβάσιμη από αυτούς από ένα ιδιωτικό δίκτυο ή το διαδίκτυο. Προκειμένου να συμβεί αυτό, πρέπει η εφαρμογή να εγκατασταθεί και να εκτελεστεί σε κάποια υποδομή που είναι προσπελάσιμη από αυτό το δίκτυο. Αυτή η υποδομή πρέπει να πληροί τα χαρακτηριστικά αξιοπιστίας (uptime, σταθερή συνδεσιμότητα) που απαιτεί το εκάστοτε παραγωγικό σύστημα. Συνήθως για το σκοπό αυτό μπορεί να χρησιμοποιηθεί κάποιο από τα παρακάτω συστήματα:

- απευθείας πάνω σε έναν εξειδικευμένο (dedicated) server
- σε κάποιο VirtualMachine (VM) σε κάποιον server που κατανέμει τους πόρους του σε πολλά VMs, on-premise ή σε cloud
- σε κάποιο Platform as a Service (PaaS) στο cloud όπου την ακριβή υλοποίηση της λύσης δεν διαχειρίζεται ο πελάτης και είναι άγνωστη σε αυτόν
- σε πλατφόρμα διαχείρισης containers, σε έναν server με χρήση docker ή κάποια ανάλογη υλοποίηση ή ακόμα και σε ένα σύνολο (cluster) από τέτοιους servers όπου πετυχαίνεται αυτόματος διαμοιρασμός των πόρων και ανάκαμψη από προβλήματα σε μέρος της υποδομής με χρήση των πλεοναζόντων πόρων

Στον κόσμο των microservices, η τελευταία επιλογή είναι και η πιο ελκυστική καθώς οι τεχνολογίες που κάνουν containerization των εφαρμογών μεταξύ άλλων προσφέρουν τα ακόλουθα προτερήματα:

1. Απομόνωση: οι εφαρμογές που τρέχουν σε διαφορετικά containers δεν έχουν πρόσβαση στη μνήμη και στους πόρους άλλων containers. Η εφαρμογή έχει πρόσβαση μόνο σε ότι υπάρχει εντός του ίδιου container. Από τη δική της οπτική, είναι η μόνη που τρέχει σε αυτό το σύστημα. Επίσης οι πόροι στους οποίους έχει πρόσβαση, (CPU, μνήμη, δίσκος) είναι περιορισμένοι και ορισμένοι εκ των προτέρων.
2. Συνοχή: Τα containers πακετάρουν όλα όσα χρειάζεται μια εφαρμογή, συμπεριλαμβανομένων βιβλιοθηκών, εξαρτήσεων και ρυθμίσεων περιβάλλοντος. Αυτό εξασφαλίζει ότι η εφαρμογή λειτουργεί το ίδιο σε διάφορα περιβάλλοντα, από την ανάπτυξη έως την παραγωγή. Οι προγραμματιστές μπορούν να αναπτύξουν και να δοκιμάσουν σε ένα περιβάλλον που χρησιμοποιεί containers και να εξάγουν συμπεράσματα για τη συμπεριφορά της παρόμοια με αυτά που θα έβλεπαν σε περιβάλλον παραγωγής.
3. Φορητότητα: Τα containers είναι εξαιρετικά φορητά. Εφόσον δημιουργηθεί η εικόνα (image) από την οποία παράγεται ένα container σε ένα σύστημα, μπορεί να χρησιμοποιηθεί σε οποιοδήποτε άλλο σύστημα που υποστηρίζει containers και να παράγει το ίδιο αποτέλεσμα, είτε πρόκειται για τον φορητό υπολογιστή ενός προγραμματιστή, έναν περιβάλλον δοκιμών είτε ένα περιβάλλον παραγωγής.
4. Αποτελεσματική αξιοποίηση πόρων: Τα containers μοιράζονται τον πυρήνα του λειτουργικού συστήματος του υπολογιστή, κάτι που τα καθιστά πιο ελαφριά σε σχέση με τις παραδοσιακές εικονικές μηχανές (VMs). Αυτή η αποτελεσματικότητα σημαίνει ότι μπορούν να

εκτελεστούν περισσότερα containers στον ίδιο server, βελτιστοποιώντας τη χρησιμοποίηση των πόρων και μειώνοντας το κόστος της υποδομής.

5. Ευκολία στην κλιμάκωση: Εφόσον απαιτείται από τις ανάγκες φόρτου και χρήσης, μπορούν να δοθούν περισσότεροι πόροι στο container (scale up - κάθετη κλιμάκωση) ή να δημιουργηθούν περισσότερα containers από την ίδια εικόνα και να αναλάβουν το καθένα μέρος της δουλειάς (scale out - οριζόντια κλιμάκωση).

Οι γνωστότερες τεχνολογίες που υλοποιούν τις προηγούμενες λειτουργίες είναι το Docker (μαζί και με τη δυνατότητα διάταξης σε Docker Swarm[16]) και το Kubernetes[17]. Για τους σκοπούς της παρούσας εργασίας όπου δεν υπάρχουν υψηλές απαιτήσεις πόρων ή ανάγκη για οριζόντια κλιμάκωση αρκεί η χρήση του Docker. Ως σύντομο παράδειγμα παρατίθενται τα περιεχόμενα του Dockerfile (Κώδικας 4.1)για το python backend σύστημα που ορίζει πως οι κατασκευάζεται το image που θα παράγει τα containers αυτής της εφαρμογής. Επισημαίνεται πως οι παρακάτω εντολές είναι πολύ γενικές και εξυπηρετούν τους σκοπούς των περισσότερων Python projects.

Κώδικας 4.1: Το περιεχόμενο του αρχείου docker για το Python BackEnd

```
# Use the official Python 3.11 slim image as the base image.
FROM python:3.11-slim-buster

# Create a directory named 'code' inside the container.
RUN mkdir /code

# Set the working directory to '/code' within the container.
WORKDIR /code

# Copy the 'requirements.txt' file from the host into the container.
COPY requirements.txt .

# Install Python dependencies specified in 'requirements.txt'.
RUN pip install -r requirements.txt

# Copy the contents of the './src' directory from the host into the container.
COPY ./src .

# Define the command to run when the container starts.
CMD ["python", "main.py"]
```

Το Docker Compose[18] είναι ένα εργαλείο του docker για τον καθορισμό και την εκτέλεση εφαρμογών αποτελούμενων από πολλαπλά Docker containers. Επιτρέπει τον ορισμό της επιθυμητής τοπολογίας σε ένα αρχείο (συνήθως με το όνομα docker-compose.yml), και με μια εντολή μπορεί να χρησιμοποιηθεί για την εκκίνηση και παραμετροποίηση όλων των container που έχουν καθοριστεί στην τοπολογία του αρχείου. Ως εργαλείο είναι ιδιαίτερα χρήσιμο στη μείωση του χρόνου που διαρκεί ένας κύκλος ανάπτυξης και δοκιμής της εφαρμογής, όταν αυτή αποτελείται από πολλαπλά συνεργαζόμενα containers. Στο επόμενο κομμάτι κώδικα (4.2) φαίνονται τα περιεχόμενα του docker compose αρχείου που χρησιμεύει στην τοπική δοκιμή της εφαρμογής. Με μικρές

[16] https://docs.docker.com/engine/swarm/
[17] https://kubernetes.io/
[18] https://docs.docker.com/compose/

τροποποιήσεις, που αφορούν κυρίως σε δικτυακές ρυθμίσεις, μπορεί να θεωρηθεί κατάλληλο και για την εγκατάσταση σε ένα παραγωγικό σύστημα.

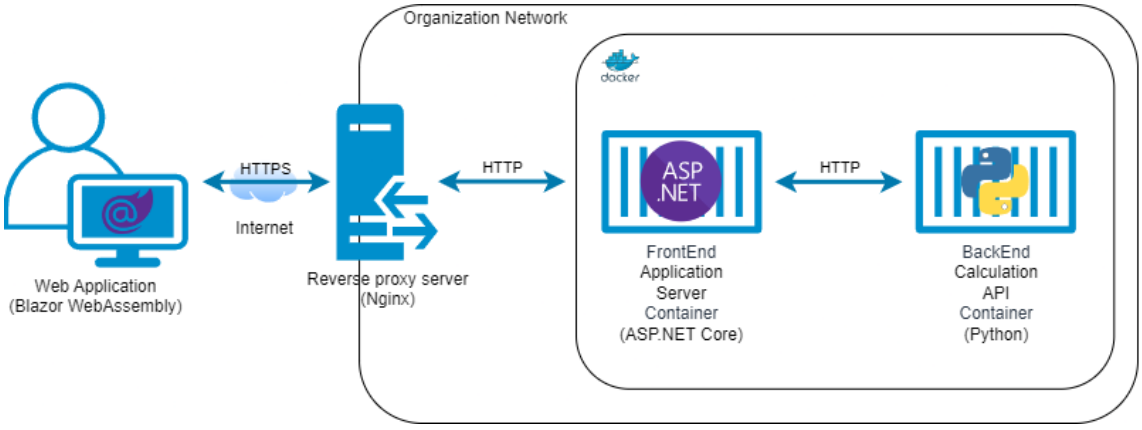
Κώδικας 4.2: Το περιεχόμενο του αρχείου docker-compose.yml

```yaml
version: '3.4'

services:
  backend:
    build: backend
    container_name: mcdm-backend
    image: mcdm/backend
    environment:
      - UVICORN_PORT=8001
    ports:
      - "8001:8001"

  frontend:
    build: frontend
    image: mcdm/frontend
    container_name: mcdm-frontend
    environment:
      - ASPNETCORE_URLS=http://*:8002
      - BACKEND_URL=mcdm-backend:8001
    ports:
      - "8002:8002"
    depends_on:
      - backend
```

Σύμφωνα με τα προηγούμενα, μπορούμε πλέον να συνθέσουμε μια πιο ολοκληρωμένη και ακριβή παρουσίαση του τελικού παραγωγικού συστήματος. Τέτοια είναι η Εικόνα 4.7 στην οποία θεωρείται ότι το σύστημα είναι προσβάσιμο από το διαδίκτυο. Στο διαδίκτυο είναι συχνή τεχνική η χρήση ενός reverse proxy όπως ο Nginx[19] όπου τερματίζει η κρυπτογράφηση SSL και επιτρέπει η υπόλοιπη επικοινωνία, εντός του ιδιωτικού δικτύου του οργανισμού να γίνεται με απλό HTTP πρωτόκολλο.



Εικόνα 4.7: Σχεδιάγραμμα μιας πιθανής τοπολογίας σε παραγωγικές συνθήκες

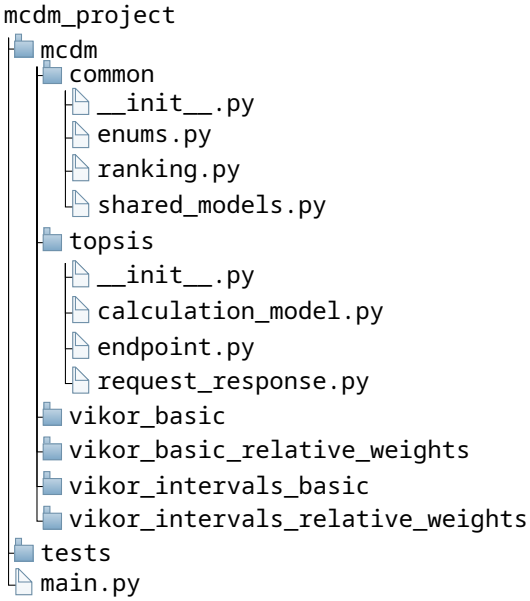
[19] https://www.nginx.com/

4.5. Δομή του κώδικα

Στα Κεφάλαια 4.2. και 4.4. έγινε μια μακροσκοπική παρουσίαση της αρχιτεκτονικής του εργαλείου που έχει υλοποιηθεί. Σε αυτό το κεφάλαιο θα παρουσιαστούν σε μεγαλύτερο βάθος οι δομές των αρχείων του κώδικα καθώς και του ίδιου του κώδικα για τα δύο project, το BackEnd και το FrontEnd. Η παρουσίαση θα γίνει με δεδομένη την υλοποίηση της μεθόδου VIKOR(2.1.), των τριών επεκτάσεων της που παρουσιάστηκαν στα Κεφάλαια 2.2., 2.3., 2.4. και της μεθόδου TOPSIS (2.7.).

4.5.1 Η δομή του Python BackEnd συστήματος

Στη γλώσσα Python ο κώδικας οργανώνεται σε αρχεία επέκτασης *.py που ονομάζονται modules. Τα modules περιέχονται σε φακέλους που με την προσθήκη ενός αρχείου με όνομα «__init__.py» ορίζεται ένα package. Στο Python project του εργαλείου ο κώδικας που αφορά σε μία μέθοδο ΠΑΑ εμπεριέχεται ολόκληρος στο αντίστοιχο package. Στην Εικόνα 4.8 φαίνεται η δομή του project στο σύστημα αρχείων με επεκτεταμένους το φακέλο «common» και τον αντίστοιχο της μεθόδου TOPSIS. Σημειώνεται ότι κάθε package μεθόδου ΠΑΑ έχει ακριβώς τα ίδια modules με των υπολοίπων ώστε να υπάρχει συνέπεια στον τρόπο οργάνωσης.



Εικόνα 4.8: Η δομή του Python BackEnd project

Ο κώδικας του υπολογιστικού μοντέλου της κάθε μεθόδου περιέχεται στο αρχείο με όνομα «calculation_module.py» του αντίστοιχου φακέλου. Στο module με όνομα «request_response.py» ορίζονται τα μοντέλα δεδομένων, ως κλάσεις, για την είσοδο και την έξοδο του υπολογιστικού μοντέλου. Στον Κώδικα 4.3 δίνονται ως παράδειγμα τα μοντέλα δεδομένων εισόδου (ως «TopsisRequest» και εξόδου (ως «TopsisResponse») για την μέθοδο TOPSIS. Το μοντέλο για τα δεδομένα εισόδου απαιτεί την λίστα τον κριτηρίων και την λίστα τον εναλλακτικών. Στη συγκεκριμένη μέθοδο, που

όλα τα κριτήρια έχουν αριθμητικά μεγέθη, έχουν χρησιμοποιηθεί οι κλάσεις «NumericWeightedCriterion» και «NumericAlternative» που ορίζονται στο module «shared_models.py» του πακέτου «common».

Κώδικας 4.3: Το περιεχόμενο του module request_response.py που βρίσκεται εντός του TOPSIS package

```
from pydantic import BaseModel
from typing import Dict, List
from mcdm.common.shared_models import NumericAlternative,
    ↪ NumericWeightedCriterion

class TopsisRequest(BaseModel):
    alternatives: List[NumericAlternative]
    criteria: List[NumericWeightedCriterion]

class TopsisResponse:
    u: Dict[int, Dict[int, float]]
    positiveIdealU: Dict[int, float]
    negativeIdealU: Dict[int, float]
    distanceToPositiveIdeal: Dict[int, float]
    distanceToNegativeIdeal: Dict[int, float]
    closenessToIdeal: Dict[int, float]
    finalRanking: List[int]
```

Στον Κώδικα 4.4 παραθέτονται όλα τα μοντέλα που έχουν σχεδιαστεί για την αναπαράσταση των κριτηρίων και των εναλλακτικών. Οι κλασικές μέθοδοι TOPSIS και VIKOR χρησιμοποιούν πραγματικά αριθμητικά μεγέθη για την εκτίμηση των επιδόσεων των εναλλακτικών στα εξεταζόμενα κριτήρια. Ωστόσο, οι επεκτάσεις των μεθόδων, μπορεί να χρησιμοποιούν κριτήρια που εκτιμώνται από άλλου είδους μεγέθη, όπως οι αριθμοί-διαστήματα ή οι ασαφείς αριθμοί. Για τον λόγο αυτό, έχουν κατασκευαστεί οι πολυμορφικές κλάσεις, όπως η «PolymorphicAlternative», που δέχεται ως επιδόσεις έναν από πολλούς πιθανούς τύπους και διακρίνονται μεταξύ τους μέσω του πεδίου «type» κατά την αποσειριοποίηση (deserialization) ή σε οποιοδήποτε σημείο ετκέλεσης που πρέπει να γίνει διαφορετική διαχείριση για κάθε τύπο δεδομένων.

Κώδικας 4.4: Το περιεχόμενο του module shared_models.py που βρίσκεται εντός του common package

```
from typing import Dict, Literal, Union
from pydantic import BaseModel, Field
from typing_extensions import Annotated
from mcdm.common.enums import PerformanceType

class BaseCriterion(BaseModel):
    id: int
    name: str
    isCost: bool

class NumericRankedCriterion(BaseCriterion):
```

```
    rank: int

class NumericWeightedCriterion(BaseCriterion):
    normalisedWeight: float

class PolymorphicWeightedCriterion(BaseCriterion):
    normalisedWeight: float
    typeId: PerformanceType

class PolymorphicRankedCriterion(BaseCriterion):
    rank: int
    typeId: PerformanceType

class DecimalPerformance(BaseModel):
    type: Literal[PerformanceType.Decimal] = Field(default=
        ↪ PerformanceType.Decimal)
    value: float

class IntervalPerformance(BaseModel):
    type: Literal[PerformanceType.Interval] = Field(default=
        ↪ PerformanceType.Interval)
    lower: float
    upper: float

class NumericAlternative(BaseModel):
    id: int
    name: str
    performances: Dict[int, float]

class PolymorphicAlternative(BaseModel):
    id: int
    name: str
    performances: Dict[int, Annotated[Union[IntervalPerformance,
        ↪ DecimalPerformance] , Field(discriminator='type')]]
```

Τα μοντέλα που υπολογίζουν την κάθε μέθοδο ΠΑΑ, αν και διαφέρουν σε αρκετά σημεία, έχουν παρόμοια μορφή. Είναι κλάσεις που κατά την κατασκευή ενός στηγμιότυπού τους δέχονται ως όρισμα το αντίστοιχο μοντέλο εισόδου δεδομένων. Κάθε μια διαθέτει μια μέθοδο «calculate_result» που εκτελεί τα βήματα υπολογισμού σε σειρά και επιστρέφει το αντικείμενο εξόδου δεδομένων, που περιέχει την τελική κατάταξη των εναλλακτικών, καθώς και όλα τα ενδιάμεσα αποτελέσματα των υπολογισμών που μπορούν να φανούν χρήσιμα στον αποφασίζοντα. Η δομή της μεθόδου θυμίζει τα βήματα που ακολουθούνται για την επίλυση της μεθόδου σύμφωνα με τη θεωρία. Παρατίθεται στον Κώδικα 4.5 η υλοποίηση για τη μέθοδο TOPSIS. Ο πλήρης κώδικας του υπολογιστικού μοντέλου για τη μέθοδο TOPSIS δίνεται στο παράρτημα (A□.1).

Κώδικας 4.5: Η μέθοδος calculate_result το υπολογιστικού μοντέλου TOPSIS

```
    def calculate_result(self) -> TopsisResponse:
        response = TopsisResponse()
        self.calculate_normalised_U_values(response)
        self.calculate_positive_and_negative_ideal_solutions(response)
        self.calculate_distances_from_ideal(response)
        self.calculate_final_ranking(response)
        return response
```

Το τελευταίο module που περιέχεται στο package μιας μεθόδου ΠΑΑ είναι το «endpoint.py» που χρησιμεύει στην ενσωμάτωση της μεθόδου στο API που εξυπηρετεί τα HTTP αιτήματα υπολογισμού. Στο Κώδικα 4.6 φαίνεται ο κώδικας που δημιουργεί το endpoint για την εξυπηρέτηση των αιτημάτων υπολογισμού TOPSIS, και δείχνει τον τρόπο που καλείται το μοντέλο υπολογισμού.

Κώδικας 4.6: Το περιεχόμενο του endpoint.py module της μεθόδου TOPSIS

```
from mcdm.topsis.request_response import TopsisRequest
from mcdm.topsis.calculation_model import TopsisModel
from fastapi import APIRouter, Body

router = APIRouter()

@router.post("/topsis")
async def calculate_topsis(request: TopsisRequest = Body(None)):
    model = TopsisModel(request)
    result = model.calculate_result()
    return result
```

Στο εργαλείο ακολουθήθηκε η αρχιτεκτονική των microservices προκειμένου να υπάρχει η δυνατότητα να χρησιμοποιηθεί το Python BackEnd σύστημα από πολλαπλά συστήματα FrontEnd. Όλα τα δεδομένα που απαιτούνται για την ολοκλήρωση των υπολογισμών μιας μεθόδου ΠΑΑ περιέχονται και αποστέλλονται μαζί με το αίτημα υπολογισμού που φτάνει στο BackEnd. Το BackEnd μπορεί να χαρακτηριστεί λοιπόν ως stateless microservice, αφού δεν υπάρχει κάποια κατάσταση (state) του συστήματος που μπορεί να επηρεάσει τα αποτελέσματα, δηλαδή για την ίδια είσοδο δεδομένων έχει πάντα την ίδια έξοδο αποτελεσμάτων. Επίσης, είδαμε και τον τρόπο καλείται το μοντέλο μιας μεθόδου. Προκειμένου να επιβεβαιωθεί η ορθότητα του κώδικα κατά την ανάπτυξή του και για να αποφευχθούν σφάλματα κατά την οποιαδήποτε προσπάθεια επανασχεδιασμού και βελτιστοποίησης του κώδικα, έχουν καταγραφεί ορισμένα Unit tests[20].

Υπάρχουν πολλά είδη test που μπορούν να επιβεβαιώσουν την ορθότητα ενός συστήματος. Το καταλληλότερο test για τον έλεγχο της λογικής που αποτελεί τον πυρήνα ενός συστήματος υπολογισμών πάνω σε δεδομένα, είναι τα Unit tests. Αυτό ισχύει διότι ελέγχουν ένα μικρό κομμάτι κώδικα (unit) και άρα βρίσκονται πολύ κοντά στο σύστημα που ελέγχουν. Συνήθως εκτελούνται γρήγορα, κάτι που τα κάνει κατάλληλα για συχνή εκτέλεση και άμεσο εντοπισμό σφαλμάτων.

Στον κώδικα 4.7 που αποτελεί απόσπασμα του αρχείου «test_topsis.py» φαίνεται ένα παρά-

[20] https://en.wikipedia.org/wiki/Unit_testing

δειγμα τέτοιου unit test. Το test ακολουθεί μεθοδολογία arrange-act-assert. Στη φάση arrange γίνεται προετοιμασία των δεδομένων, στη φάση act γίνεται χρήση τους από το υπο δοκιμή σύστημα (system under test ή sut) ενώ στη φάση assert ελέγχονται οι ισχυρισμοί του test. Το συγκεκριμένο τεστ χρησιμοποιεί ένα μικρό πίνακα απόφασης, με δύο ισοβαρή κριτήρια, όπου το πρώτο είναι κριτήριο κόστους. Οι ισχυρισμοί που ελέγχει αφορούν τις μετρικές C^* των εναλλακτικών που αναμένεται να είναι 0.762, 0.722 και 0.238 αντίστοιχα, με μέγιστη απόκλιση 0.016. Τέλος, ελέγχεται και η μέθοδος ταξινόμησης των εναλλακτικών βάσει της μετρικής και αναμένεται ότι $A_1 > A_2 > A_3$.

Κώδικας 4.7: Unit test πάνω στη μέθοδο TOPSIS

```
import unittest
from mcdm.common.shared_models import NumericAlternative,
    ↪ NumericWeightedCriterion
from mcdm.topsis.calculation_model import TopsisModel
from mcdm.topsis.request_response import TopsisRequest

class TopsisModelTests(unittest.TestCase):
    def test_smallexample(self):
        #arrange
        inputs = TopsisRequest(criteria=[
            NumericWeightedCriterion(id=1, name="C1", isCost=True,
                ↪ normalisedWeight=1/2),
            NumericWeightedCriterion(id=2, name="C2", isCost=False,
                ↪ normalisedWeight=1/2)
        ],alternatives = [
            NumericAlternative(id=1, name="A1", performances={ 1: 1, 2:
                ↪  3000 }),
            NumericAlternative(id=2, name="A2", performances={ 1: 2, 2:
                ↪  3750 }),
            NumericAlternative(id=3, name="A3", performances={ 1: 5, 2:
                ↪  4500 }),
        ],v = 0.5)

        sut = TopsisModel(inputs)

        #act
        result = sut.calculate_result()

        #assert
        expected_c_values = [0.762, 0.722, 0.238]
        for (i, expected_c) in enumerate(expected_c_values):
            self.assertAlmostEqual(expected_c,
                result.closenessToIdeal[i+1], delta=0.016)
        self.assertListEqual([1,2,3], result.finalRanking)

if __name__ == '__main__':
```

```
    unittest.main()
```

Το σύνολο των test του project μπορεί να εκτελεστεί με την παρακάτω εντολή (Κώδικας 4.8) από μια γραμμή εντολών στο φάκελο «mcdm_project».

Κώδικας 4.8: Εκτέλεση όλως των unit test σε γραμμή εντολών

```
C:\Source\mcdm_tool\backend\mcdm_project>python \-m unittest discover .
.........
----------------------------------------------------------------------
Ran 9 tests in 0.014s

OK
```

Το τελευταίο αρχείο του project, «main.py» είναι το σημείο εκκίνησης του BackEnd microservice. Στο Κεφάλαιο 4.4. εξηγήθηκε πως μπορεί να εγκατσταθεί και να εκτελεστεί η εφαρμογή σε ενα παραγωγικό περιβάλλον. Για τοπική εκτέλεση στο περιβάλλον ανάπτυξης αρκεί η παρακάτω εντολή (Κώδικας 4.9).

Κώδικας 4.9: Εκτέλεση του Python BackEnd σε γραμμή εντολών

```
C:\Source\mcdm_tool\backend\mcdm_project> python.exe main.py
INFO:     Started server process [35676]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8001 (Press CTRL+C to quit)
```

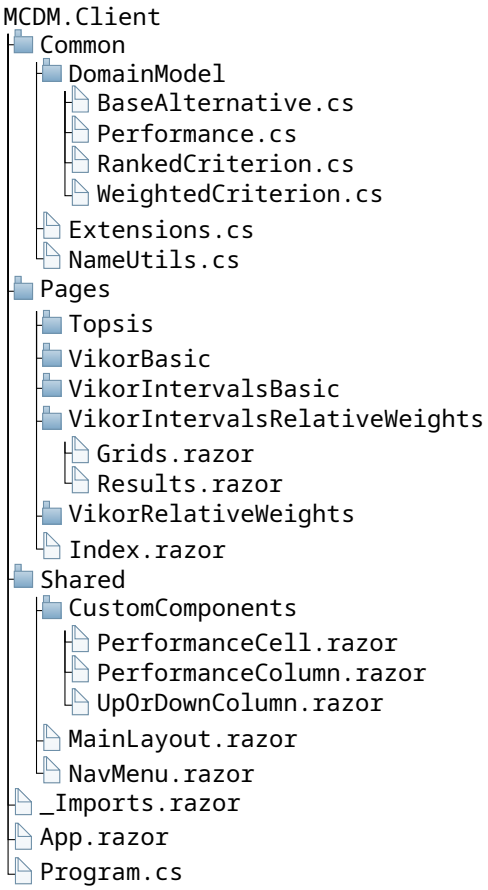
4.5.2 Η δομή του .NET FrontEnd συστήματος

Στην αντικειμενοστραφή γλώσσα προγραμματισμού C#, ο κώδικας οργανώνεται σε κλάσεις, που κάθε μια συνήθως αποτελεί ξεχωριστό αρχείο. Πολλά συναφή αρχεία οργανώνονται σε projects (με επέκταση αρχείου *.csproj) τα οποία μπορεί να χρησιμοποιούν εσωτερικά φακέλους για εσωτερική οργάνωση. Το project είναι η μικρότερη ποσότητα κώδικα που είναι δυνατό να μεταγλωττιστεί στην ενδιάμεση γλώσσα CIL και το αποτέλεσμα της μεταγλώττισης μπορεί να είναι είτε ένα εκτελέσιμο είτε αρχείο είτε μια βιβλιοθήκη (library) που μπορεί να χρησιμοποιηθεί από άλλα εκτελέσιμα. Το FrontEnd του υπολογιστικού εργαλείου αποτελείται από τα τρία παρακάτω projects:

1. MCDM.Client.csproj περιέχει τον κώδικα της web εφαρμογής που κατεβαίνει και εκτελείται στον browser σε περιβάλλον Blazor WebAssembly.
2. MCDM.Shared.csproj περιέχει τον κώδικα που τρέχει στον web application server. Αυτός σερβίρει τα πακεταρισμένα αρχεία της web εφαρμογής και διαχειρίζεται όλα τα HTTP αιτήματα που έρχονται από τον browser.
3. MCDM.Shared.csproj είναι project library που περιέχει τον κώδικα που ειναι απαραίτητα να μοιράζονται τα δύο προηγούμενα projects. Εκεί συνήθως γίνεται ο ορισμός των μοντέλων

δεδομένων που μεταφέρονται σε ένα HTTP αίτημα και είναι απαραίτητο να είναι κοινό στον αποστολέα και στον παραλήπτη.

Στην Εικόνα 4.9 φαίνεται η δομή των φακέλων και των αρχείων του project της web εφαρμογής. Σε αυτό, με εξαίρεση τα μοντέλα δεδομένων και τις βοηθητικές συναρτήσεις που βρίσκονται στον φάκελο «Common», τα αρχεία έχουν επέκταση *.razor. Τα αρχεία Razor στο ASP.NET ορίζουν πρότυπα (templates) παραγωγής κώδικα HTML και CSS, με χρήση γλώσσας C#, παρόμοια με τις δυνατότητες που προσφέρουν τα πιο γνωστά JavaScript frameworks ReactJS, AngularJS, VueJS. Ο κώδικας αυτών συνήθως είναι οργανωμένος σε σελίδες (Pages) και components.



Εικόνα 4.9: Η δομή του MCDM.Client.csproj όπως φαίνεται σε ένα περιβάλλον ανάπτυξης .NET

Προκειμένου να επιτευχθεί ένα ελκυστικό αποτέλεσμα, έγινε κυρίως χρήση των έτοιμων components που παρέχει η ανοιχτού κώδικα βιβλιοθήκη MudBlazor[21]. Εκτός των έτοιμων components απαραίτητη ήταν και η κατασκευή ειδικά προσαρμοσμένων components για να καλύψουν κάποιες ιδιαιτερότητές, όπως για παράδειγμα η εισαγωγή αριθμών-διαστημάτων στους πίνακες απόφασης. Ο Κώδικας 4.10 του αρχείου «PerformanceColumn.razor» ορίζει ένα component στήλης που επεκτείνει το component MudDataGrid της βιβλιοθήκης MudBlazor ώστε να μπορεί να διαχειριστεί τις επιδόσεις των εναλλακτικών που έχουν μοντελοποιηθεί ως λεξικό(dictionary) με κλειδί το

[21] https://mudblazor.com/ και https://github.com/MudBlazor/MudBlazor/

αναγνωριστικό (Id) του κριτηρίου και τιμή την επίδοση της εναλλακτικής σε αυτό.

Στον Κώδικα 4.11 ορίζεται ένα component κελιού που επεκτείνει το προηγούμενο. Σκοπός του είναι να αλλάζει μορφή ανάλογα με τον τύπο του κριτηρίου. Εκτός της μορφοποίησης, τα components είναι υπεύθυνα και για την διαχείριση των δεδομένων που έχουν δεχτεί ως ορίσματα. Πίσω στο παράδειγμα του «PerformanceCell.razor», όταν έχει μορφή που δέχεται αριθμόδιάστημα και αλλάζει ένα από τα άκρα του, προσαρμόζει αυτόματα και το άλλο άκρο ώστε να ορίζεται σωστό διάστημα ή έστω εκφυλισμένο (μορφής $[a, a]$).

Κώδικας 4.10: Ο κώδικας του component PerformanceColumn.razor

```
@namespace MCDM.Client.Shared.CustomComponents
@using System.Globalization;
@using MCDM.Client.Common.DomainModel;
@using MCDM.Shared;
@using MCDM.Shared.VikorBasic;
@using MCDM.Shared.Enums;
@using MCDM.Client.Common;
@typeparam T where T: BaseAlternative
@inherits MudComponentBase

<TemplateColumn T="T" Title="@Title">
    <EditTemplate>
        <PerformanceCell Type="@Type" Performance="@context.Item.Performances[
            ↪ CriterionId]" Culture="@Culture" />
    </EditTemplate>
    <CellTemplate>
        <PerformanceCell Type="@Type" Performance="@context.Item.Performances[
            ↪ CriterionId]" Culture="@Culture"/>
    </CellTemplate>
</TemplateColumn>

@code{
    [Parameter, EditorRequired] public int CriterionId { get; set; }
    [Parameter] public string? Title { get; set; }
    [Parameter, EditorRequired] public PerformanceType Type { get; set; }
    private CultureInfo? _culture;

    [CascadingParameter] public MudDataGrid<T>? DataGrid { get; set; }
    public CultureInfo? Culture
    {
        get => _culture ?? DataGrid?.Culture;
        set
        {
            _culture = value;
        }
    }
}
```

Κώδικας 4.11: Ο κώδικας του component PerformanceCell.razor

```
@namespace MCDM.Client.Shared.CustomComponents
@using System.Globalization;
@using MCDM.Shared;
@using MCDM.Shared.Enums;
@using MCDM.Client.Common.DomainModel;
```

```
@inherits MudComponentBase

@if (Type == PerformanceType.Interval)
{
    <div style="display: flex;">
        [
        <MudNumericField T="decimal" Value=@IntervalPerformance!.Value.Item1
            ↪ ValueChanged="@(v => IntervalPerformance.Value = (v, Math.Max(v,
            ↪  IntervalPerformance.Value.Item2)))" Margin="@Margin.Dense"
                        Required="true" Variant="@Variant.Text" Culture="
                            ↪ @Culture" Style="margin-top:0" HideSpinButtons=
                            ↪ "true" />,
        <MudNumericField T="decimal" Value=@IntervalPerformance.Value.Item2
            ↪ ValueChanged="@(v => IntervalPerformance.Value = (Math.Min(v,
            ↪ IntervalPerformance.Value.Item1), v))" Margin="@Margin.Dense"
                        Required="true" Variant="@Variant.Text" Culture="
                            ↪ @Culture" Style="margin-top:0" HideSpinButtons=
                            ↪ "true" />]
    </div>
}
else if (Type == PerformanceType.Decimal)
{
    <MudNumericField T="decimal" Value="@DecimalPerformance!.Value"
        ↪ ValueChanged="@(v => DecimalPerformance.Value = v)" Margin="@Margin.
        ↪ Dense" Required="true" Variant="@Variant.Text" Culture="@Culture"
        ↪ Style="margin-top:0" />
}

@code {
    [Parameter, EditorRequired]
    public PerformanceType Type { get; set; }
    [Parameter, EditorRequired]
    public Performance? Performance { get; set; }
    [Parameter, EditorRequired]
    public CultureInfo? Culture { get; set; }


    private IntervalPerformance? IntervalPerformance { get; set; }
    private DecimalPerformance? DecimalPerformance { get; set; }


    protected override Task OnParametersSetAsync()
    {
        @if (Type == PerformanceType.Interval)
        {
            IntervalPerformance = Performance as IntervalPerformance;
        }
        else if (Type == PerformanceType.Decimal)
        {
            DecimalPerformance = Performance as DecimalPerformance;
        }
        return base.OnParametersSetAsync();
    }

    protected override Task OnInitializedAsync()
    {
```

```
        return base.OnInitializedAsync();
    }
}
```

Όπως και στο κώδικα του Python project, υπάρχει η ανάγκη για πολυμορφισμό στα μοντέλα που αναπαριστούν τις επιδόσεις των κριτηρίων. Στον Κώδικα 4.12 δείχνεται πως μπορεί να υλοποιηθεί ένα τέτοιο μοντέλο με χρήση παράγωγων κλάσεων σε γλώσσα C#. Το μοντέλο αυτό συνεργάζεται με τα components «PerformanceColumn» και «PerformanceCell» που αναφέρθηκαν προηγουμένως.

Κώδικας 4.12: Η ιεραρχία κλάσεων που επιτρέπει την πολυμορφική συμπεριφορά των επιδόσεων

```
public class BaseAlternative
{
    public BaseAlternative(int id, string name)
    {
        Id = id;
        Name = name;
    }

    public int Id { get; }
    public string Name { get; set; }
    public Dictionary<int, Performance> Performances { get; set; } = new
        ↪ Dictionary<int, Performance>();
}
public abstract class Performance
{
    public static Performance GetDefault(PerformanceType type) =>
        type switch
        {
            PerformanceType.Decimal => new DecimalPerformance(),
            PerformanceType.Interval => new IntervalPerformance(),
            _ => throw new NotSupportedException(),
        };

    protected PerformanceType _type;
    public PerformanceType Type => _type;

    public abstract object GetValueAsObject();
}

public abstract class Performance<T> : Performance where T : struct //Contraint
    ↪  to struct, else will have to constraint to IEquatable and nullability
{
    public T Value { get; set; } = default;


    public override object GetValueAsObject() => Value;

    public override bool Equals(object? obj)
    {
        if (obj is Performance<T> other)
        {
            return Equals(other);
        }
        return false;
```

```
    }

    public bool Equals(Performance<T> other)
    {
        if (other == null)
            return false;

        return Value.Equals(other.Value);
    }

    public override int GetHashCode()
    {
        return HashCode.Combine(_type, Value.GetHashCode());
    }
}

public class DecimalPerformance : Performance<decimal>
{
    public DecimalPerformance()
    {
        _type = PerformanceType.Decimal;
    }
}

public class IntervalPerformance : Performance<(decimal, decimal)>
{
    public IntervalPerformance()
    {
        _type = PerformanceType.Interval;
    }
}
```

Για να φτάσουν τα δεδομένα που συμπληρώνονται στον πίνακα απόφασης στο Python microservice προκειμένουν να υπολογιστούν τα αποτλέσματα της ΠΑΑ, πρέπει αυτά να τοποθετηθούν σε μοντέλα δεδομένων, όμοια με αυτά που έχουν οριστεί στο HTTP API της Python. Δίδονται στον Κώδικα 4.13 τα μοντέλα δεδομένων εισόδου και εξόδου της μεθόδου TOPSIS, που είναι όμοια σε δομή με αυτά του Κώδικά 4.3.

Κώδικας 4.13: Οι κλάσεις που μοντελοποιούν τα δεδομένα εισόδου και εξόδου για τη μέθοδο TOPSIS

```
public class TopsisRequest
{
    public NumericAlternativeDto[] Alternatives { get; set; }
    public NumericWeightedCriterionDto[] Criteria { get; set; }
}

public class TopsisResponse
{
     public Dictionary<int, Dictionary<int, decimal>> U { get; set; }
     public Dictionary<int, decimal> PositiveIdealU { get; set; }
     public Dictionary<int, decimal> NegativeIdealU { get; set; }
     public Dictionary<int, decimal> DistanceToPositiveIdeal { get; set; }
     public Dictionary<int, decimal> DistanceToNegativeIdeal { get; set; }
     public Dictionary<int, decimal> ClosenessToIdeal { get; set; }
```

```
    public int[] FinalRanking { get; set; }
}
```

Στο φάκελο «Pages» περιέχεται ένας υποφάκελος ανά μέθοδο ΠΑΑ που φιλοξενεί τα σχετικά *.razor αρχεία. Σε κάθε μέθοδο ΠΑΑ αντιστοιχούν ένα αρχείο «Grids.razor» και ένα «Results.razor». Το πρώτο ορίζει το template παραγωγής του πίνακα κριτηρίων και του πίνακα απόφασης, ενώ το δεύτερο είναι υπεύθυνο για την αποστολή του αιτήματος υπολογισμού και για την παρουσίαση του αποτελέσματος και των ενδιάμεσων αποτελεσμάτων της εκτέλεσης της εκάστοτε μεθόδου ΠΑΑ. Ο Κώδικας 4.14 δείχνει πως το component «Results.razor» μετατρέπει τα δεδομένα του πίνακα απόφασης σε μοντέλο δεδομένων για μεταφορά στο δίκτυο και πως εκτελεί ένα τέτοιο αίτημα υπολογισμού. Οι Κώδικες A□.2 και A□.3 του παραρτήματος έχουν τον πλήρη κώδικα των αρχείων.

Κώδικας 4.14: Προετοιμασία και αποστολή HTTP αιτήματος υπολογισμού

```
private async Task Recalculate(double v, double k, RankingSystem r)
{
    IsLoading = true;
    IsFailed = false;
    _v_value = v;
    _k_value = k;
    _ranking = r;
    try
    {
        Response = await (await Http.PostAsJsonAsync("api/
            ↪ VikorIntervalsRelativeWeights", new
            ↪ VikorIntervalsRelativeWeightsRequest
        {
            Alternatives = Alternatives.Select(x =>
            {
                var perf = new Dictionary<int, PerformanceDto>();
                foreach (var p in x.Performances)
                {
                    if (p.Value is IntervalPerformance ip)
                    {
                        perf[p.Key] = new IntervalDto { Lower = ip.Value.Item1,
                            ↪  Upper = ip.Value.Item2 };
                    }
                    else if (p.Value is DecimalPerformance dp)
                    {
                        perf[p.Key] = new DecimalDto { Value = dp.Value };
                    }
                };
                return new PolymorphicAlternativeDto { Id = x.Id, Name = x.Name
                    ↪ , Performances = perf };
            }).ToArray(),
                Criteria = Criteria.Select(x => new
                    ↪ PolymorphicRankedCriterionDto
                {
                    Id = x.Id,
                    Name = x.Name,
                    IsCost = x.IsCost,
                    Rank = x.Rank,
                    TypeId = x.TypeId
                }).ToArray(),
```

```
            V = _v_value,
            K = _k_value,
            Ranking = _ranking
        }, JsonSettings.DefaultJsonOptions)).Content.ReadFromJsonAsync<
            ↪ VikorIntervalsRelativeWeightsResponse>() ?? new
            ↪ VikorIntervalsRelativeWeightsResponse();

    }
    catch (Exception ex)
    {
        IsFailed = true;
        ErrorText = ex.Message;
        Snackbar.Add("Calculation error. For more information see application
            ↪ logs", Severity.Error);
    }
    finally
    {
        IsLoading = false;
    }
}
```

Ένα αίτημα για υπολογισμό που θα πραγματοποιηθεί στο εργαλείο περιήγησης του χρήστη, θα φτάσει πρώτα στο ASP.NET application από όπου απλώς θα προωθηθεί στο επόμενο σύστημα, δηλαδή στο Python microservice. Ο ορισμός του API που είναι υπεύθυνο για την λήψη και προώθηση των αιτημάτων αυτών, για την μέθοδο TOPSIS, φαίνεται στον Κώδικα 4.15. Σε ένα σύστημα με περισσότερη πολυπλοκότητα, όπου απαιτούνται περισσότερα microservices για την διεξαγωγή των υπολογισμών, ο κώδικας θα είχε σαφώς αντίστοιχα πολυπλοκότερη δομή. Η δομή του project «MCDM.Server.csproj» που εξυπηρετεί αυτά τα αιτήματα φαίνεται στην Εικόνα 4.10

Κώδικας 4.15: Το περιεχόμενο του αρχείου TopsisController.cs

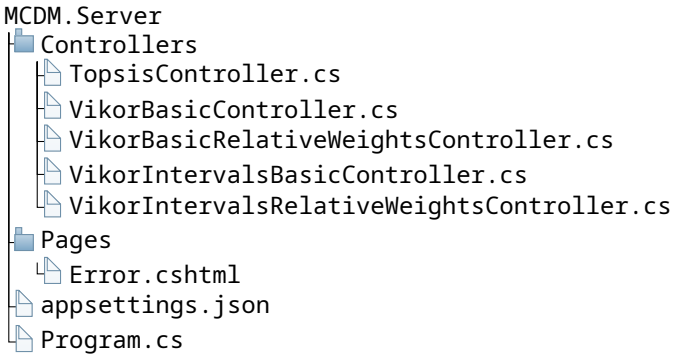
```
using MCDM.Shared;
using MCDM.Shared.Topsis;
using Microsoft.AspNetCore.Mvc;

namespace MCDM.Server.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class TopsisController : ControllerBase
    {
        private readonly IHttpClientFactory _httpClientFactory;
        public TopsisController(IHttpClientFactory httpClientFactory) =>
            _httpClientFactory = httpClientFactory;

        [HttpPost]
        public async Task<TopsisResponse> Get(TopsisRequest request)
        {
            var mcdmCalculationApiClient = _httpClientFactory.CreateClient("
                ↪ mcdmCalculationApi");
            var response = await mcdmCalculationApiClient.PostAsJsonAsync("/
                ↪ topsis", request, JsonSettings.DefaultJsonOptions);
            var result = await response.Content.ReadFromJsonAsync<
                ↪ TopsisResponse>();
            return result!;
```

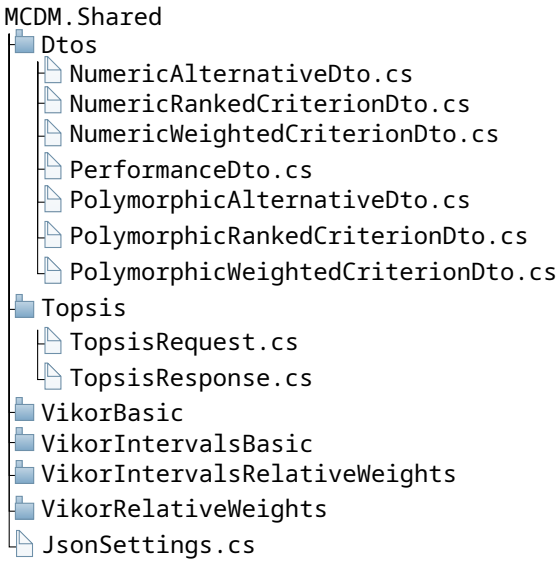
```
        }

    }
}
```



Εικόνα 4.10: Η δομή του MCDM.Server.csproj όπως φαίνεται σε ένα περιβάλλον ανάπτυξης .NET

Για πληρότητα, στην Εικόνα 4.11 δίνεται η δομή του «MCDM.Shared.csproj» όπου βρίσκονται όλα τα μοντέλα που μεταφέρουν δεδομένα στο δίκτυο (Data Transfer Objects - Dtos) ή είναι κοινά ανάμεσα στα δύο εκτελέσιμα C# projects. Όλα τα μοντέλα δεδομένων αυτού του project είναι όμοια σε δομή με τα αντίστοιχα του Python project που εξυπηρετεί τελικά τα αιτήματα υπολογισμού.



Εικόνα 4.11: Η δομή του MCDM.Shared.csproj όπως φαίνεται σε ένα περιβάλλον ανάπτυξης .NET

Κεφάλαιο 5

Πιλοτική Εφαρμογή

Τα παραδοσιακά ορυκτά καύσιμα, όπως ο άνθρακας, το πετρέλαιο και το φυσικό αέριο, είναι οι πιο σημαντικές πηγές ενέργειας. Η χρήση παραδοσιακής ορυκτής ενέργειας έχει προωθήσει την παγκόσμια οικονομική ανάπτυξη, ωστόσο έχει προκαλέσει επίσης σοβαρά περιβαλλοντικά προβλήματα. Για παράδειγμα, η καύση ορυκτών καυσίμων παράγει μεγάλη ποσότητα διοξειδίου του άνθρακα, η οποία είναι η κύρια αιτία του φαινομένου του θερμοκηπίου. Συνεπώς, οι νέες μορφές ενέργειας και οι ανανεώσιμες πηγές ενέργειας με άφθονους πόρους και χαμηλό περιβαλλοντικό αποτύπωμα έχουν σημειώσει σημαντική πρόοδο. Η ανάπτυξη των ανανεώσιμων πηγών ενέργειας είναι θεμελιώδης για τη μείωση των εκπομπών διοξειδίου του άνθρακα και την επίλυση των περιβαλλοντικών προβλημάτων, και αποτελεί σημαντική στρατηγική που αναγνωρίζεται από χώρες σε όλο τον κόσμο για την αντιμετώπιση της ατμοσφαιρικής ρύπανσης και της εξάντλησης των πόρων. Ωστόσο, η ανάπτυξη των ανανεώσιμων πηγών ενέργειας αντιμετωπίζει προκλήσεις και περιορισμούς. Η παραγωγή ενέργειας από ανανεώσιμες πηγές εξαρτάται από τη διαθεσιμότητα φυσικών πόρων, η οποία είναι ασταθής και διακοπτόμενη. Τα χαρακτηριστικά αυτά καθιστούν δύσκολη τη ρύθμιση και τον έλεγχο της παραγωγής ενέργειας, και επηρεάζουν επίσης την ασφαλή και σταθερή λειτουργία του ηλεκτρικού δικτύου. Επομένως, σήμερα, η αποθήκευση ενέργειας είναι ένα βασικό ζήτημα για την ανάπτυξη των ανανεώσιμων πηγών ενέργειας. [56]

Η τεχνολογία αποθήκευσης ενέργειας μεγάλης κλίμακας, η οποία μπορεί να λύσει τα προβλήματα της στοχαστικότητας και της μεταβλητότητας της παραγωγής ενέργειας, είναι ένα αποτελεσματικό μέτρο για την επίλυση αυτών των προβλημάτων που αντιμετωπίζει η τρέχουσα ανάπτυξη των ανανεώσιμων πηγών ενέργειας, με παρόμοιες λύσεις να εφαρμόζονται σε όλη την αλυσίδα ηλεκτρικής ενέργειας, δηλαδή από το στάδιο της παραγωγής ενέργειας έως την διανομή ηλεκτρικής ενέργειας στους τελικούς καταναλωτές. Μπορεί να χρησιμοποιηθεί ως ενέργεια έκτακτης ανάγκης και μπορεί επίσης να χρησιμοποιηθεί για αποφυγή των αιχμών και των κοιλάδων φορτίου για τη μείωση των διακυμάνσεων του δικτύου. Ωστόσο, διαφορετικές τεχνολογίες αποθήκευσης παρουσιάζουν διαφορετικά χαρακτηριστικά από άποψη τεχνολογικών, οικονομικών, κοινωνικών και περιβαλλοντικών πτυχών, και μπορούν να ικανοποιήσουν διαφορετικές απαιτήσεις αποθήκευσης.

Η ανάπτυξη των νέων και ανανεώσιμων πηγών ενέργειας έχει διαφορετικές επιπτώσεις και απαιτήσεις για διαφορετικές γεωγραφικές περιοχές και περιβάλλοντα, και η επιλογή της τεχνολογίας πρέπει να λαμβάνει υπόψη αυτές τις απαιτήσεις σε όλα τα διαφορετικά στάδια ανάπτυξης

αυτών των μεταβάσεων. Ως εκ τούτου, το ερώτημα του πώς να επιλέξετε μια κατάλληλη τεχνολογία αποθήκευσης ενέργειας βάσει της ανάπτυξης των ανανεώσιμων πηγών ενέργειας είναι ένα βασικό ζήτημα στον ενεργειακό σχεδιασμό, και είναι επίσης προϋπόθεση για την επίλυση των προβλημάτων ενεργειακής ασφάλειας.[57]

Συγκεκριμένα, η επιλογή τεχνολογίας αποθήκευσης ενέργειας πρέπει να επιτυγχάνει πολλαπλούς στόχους και να λαμβάνει υπόψη πολλούς παράγοντες, συμπεριλαμβανομένων των οικονομικών, τεχνολογικών, κοινωνικών και περιβαλλοντικών. Χρησιμοποιούνται διαφορετικές προσεγγίσεις για τη βελτιστοποίηση της επιλογής τεχνολογιών αποθήκευσης ενέργειας, με κάποιες από αυτές να χρησιμοποιούν προηγμένες πρακτικές, όπως τεχνικές μηχανικής μάθησης [58] [59] [60] [61] [62], ενώ άλλοι επιστήμονες χρησιμοποιούν μεθόδους πολλαπλών στόχων βελτιστοποίησης για την επιλογή τεχνολογίας [63]. Ωστόσο, διάφορες πτυχές συχνά συγκρούονται μεταξύ τους. Για παράδειγμα, οι τεχνολογίες που εισάγουν χαμηλότερα επίπεδα ρύπανσης είναι συνήθως πιο κοστοβόρες. Αυτό καθιστά δύσκολη για τους υπεύθυνους λήψης αποφάσεων την άμεση επιλογή μιας κατάλληλης τεχνολογίας αποθήκευσης. Με αυτό κατά νου, το πρόβλημα επιλογής τεχνολογίας αποθήκευσης ενέργειας, το οποίο περιλαμβάνει πολυάριθμους ασαφείς παράγοντες, μπορεί να διατυπωθεί ως πρόβλημα ΠΑΑ, με τα πλεονεκτήματα και τα μειονεκτήματα των μεταβλητών απόφασης να απαιτούν συχνά τη χρήση αριθμών-διαστημάτων, καθαρών αριθμών και γλωσσικών όρων που εκφράζονται σε φυσική γλώσσα. Οι μέθοδοι ΠΑΑ χρησιμοποιούνται συνήθως στον τομέα της έρευνας για την ενέργεια, όπως η τοποθέτηση ανανεώσιμων πηγών ενέργειας [64], η επιλογή τοποθεσίας ηλιακών σταθμών παραγωγής ενέργειας [65], η επιλογή φωτοβολταϊκών συστημάτων [66] και η επιλογή κατάλληλων τοποθεσιών για αιολικούς και ηλιακούς σταθμούς [67] [68]. Υπάρχουν πολλά κριτήρια επιλογής για την τεχνολογία αποθήκευσης ενέργειας, όπως η μέθοδος αποθήκευσης, η διάρκεια αποθήκευσης, ο χρόνος απόκρισης κ.λπ.[69] [70] [71]. Η πιο δημοφιλής μέθοδος στις παραπάνω ταξινομήσεις, η οποία έχει αναγνωριστεί από πολλούς επιστήμονες, είναι η μορφή αποθήκευσης [71]. Σύμφωνα με τη μορφή αποθήκευσης, οι τεχνολογίες μπορούν να ταξινομηθούν σε μηχανικές, ηλεκτροχημικές, χημικές, ηλεκτρικές και θερμοχημικές [72] [69].

Η μηχανική αποθήκευση είναι η πιο κοινή τεχνολογία αποθήκευσης και η εγκατεστημένη ισχύς της μηχανικής αποθήκευσης ενέργειας αντιπροσωπεύει περισσότερο από 90% της συνολικής ισχύος [73]. Οι τεχνολογίες που ανήκουν σε αυτή την κατηγορία περιλαμβάνουν κυρίως την υδροηλεκτρική αποθήκευση (PHS - pumped hydro storage), την αποθήκευση ενέργειας πεπιεσμένου αέρα (CAES - compressed air energy storage) και την αποθήκευση ενέργειας σφονδύλου (FES - flywheel energy storage). Η ηλεκτροχημική αποθήκευση περιλαμβάνει συσσωρευτές μολύβδου-οξέος(Lead-acid), ιόντων λιθίου (Li-ion), νικελίου-καδμίου [74] κ.λπ. Ανάμεσά τους, οι Li-ion και οι μολύβδου-οξέος κατέχουν το μεγαλύτερο μερίδιο της αγοράς [75]. Οι χημικές τεχνολογίες αποθήκευσης περιλαμβάνουν το υδρογόνο και την τεχνολογία του συνθετικού φυσικού αερίου [76]. Από τις δύο τεχνολογίες, η τεχνολογία αποθήκευσης υδρογόνου είναι η πιο σημαντική χημική τεχνολογία αποθήκευσης. Η ηλεκτρική αποθήκευση περιλαμβάνει πυκνωτές και υπεραγώγιμη αποθήκευση μαγνητικής ενέργειας (SMES - superconducting magnetic energy storage) [77].

Με βάση τα παραπάνω, χωρίζονται εννέα τεχνολογίες αποθήκευσης σε πέντε κατηγορίες ως εξής:

- υδροηλεκτρική, πεπιεσμένου αέρα και σφονδύλου (μηχανική αποθήκευση)
- συσσωρευτές ιόντων λιθίου και μολύβδου-οξέος (ηλεκτροχημική αποθήκευση)

- Υδρογόνο (χημική αποθήκευση)
- Πυκνωτές και υπεραγώγιμη αποθήκευση μαγνητικής ενέργειας (ηλεκτρική αποθήκευση)
- Θερμοχημική αποθήκευση

Διάφοροι παράγοντες θα πρέπει να λαμβάνονται υπόψη κατά την επιλογή της κατάλληλης τεχνολογίας αποθήκευσης ενέργειας. Αυτοί οι παράγοντες χωρίζονται σε τέσσερις κατηγορίες: τεχνολογικοί παράγοντες [63] [72], οικονομικοί παράγοντες [73], περιβαλλοντικοί παράγοντες και κοινωνικοί παράγοντες. Από τεχνολογική άποψη, χρησιμοποιούνται δείκτες για να εκφράσουν τα χαρακτηριστικά των τεχνολογιών, οι οποίοι περιλαμβάνουν την χωρητικότητα αποθήκευσης, τον χρόνο απόκρισης, τη διάρκεια ζωής, την πυκνότητα ενέργειας και ισχύος, την ασφάλεια, την ενεργειακή απόδοση και την ενεργειακή ένταση. Ανάμεσα σε αυτούς τους δείκτες, ορισμένοι είναι δύσκολο να εκφραστούν με μια συγκεκριμένη τιμή, όπως η χωρητικότητα αποθήκευσης, η οποία υποδηλώνει τη μέγιστη ποσότητα ηλεκτρικής ενέργειας που μπορεί να αποθηκεύσει η τεχνολογία αποθήκευσης ενέργειας. Εξαιρουμένων αυτών των δεικτών, επιλέγουμε ορισμένους αντιπροσωπευτικούς δείκτες, συμπεριλαμβανομένης της ενεργειακής απόδοσης, του χρόνου απόκρισης, της διάρκειας ζωής, της ενεργειακής πυκνότητας και των απωλειών αυτοεκφόρτισης ως τεχνολογικά κριτήρια στο πρόβλημα ΠΑΑ που θέλουμε να επιλύσουμε. Η ενεργειακή απόδοση χρησιμοποιήθηκε για τη μέτρηση του βαθμού αξιοποίησης της ενέργειας. Ο χρόνος απόκρισης μετρά το χρόνο που απαιτείται για να αντιδράσει το σύστημα και να μπορέσει να παρέχει ηλεκτρική ενέργεια. Μπορεί συνήθως να μετρηθεί σε λεπτά ή δευτερόλεπτα. Η διάρκεια ζωής είναι η περίοδος κατά την οποία ο εξοπλισμός τεχνολογίας αποθήκευσης ενέργειας παραμένει λειτουργικός και με αποδεκτή απόδοση. Η ενεργειακή πυκνότητα μετρά την ποσότητα ενέργειας ανά μονάδα μεγέθους της εγκατάστασης. Οι απώλειες αυτοεκφόρτισης υποδηλώνουν την ενέργεια που χάνεται κατά τη διάρκεια μιας περιόδου κατά την οποία το σύστημα παραμένει αδρανές.

Οι οικονομικοί παράγοντες περιλαμβάνουν το κόστος εισροών, το κόστος επένδυσης, το κόστος λειτουργίας, τα οικονομικά οφέλη, τα κόστη κεφαλαίων ισχύος και ενέργειας. Αυτοί οι παράγοντες προέρχονται κυρίως από την προοπτική του οικονομικού κόστους για τη μέτρηση των οικονομικών χαρακτηριστικών της τεχνολογίας. Στην παρούσα ανάλυση θα χρησιμοποιηθούν το κόστος κεφαλαίου ισχύος και το κόστος κεφαλαίου ενέργειας, από τα οικονομικά χαρακτηριστικά.

Οι περιβαλλοντικοί παράγοντες περιλαμβάνουν τις εκπομπές, την παραγόμενη ποσότητα διοξειδίου του άνθρακα, την επιβάρυνση του τοπικού οικοσυστήματος, την προστασία του περιβάλλοντος, την κατανάλωση πόρων και την έκταση γης που απαιτείται. Ωστόσο, ορισμένοι δείκτες είναι δύσκολο να μετρηθούν επειδή η διάρκεια ζωής των τεχνολογιών αποθήκευσης ενέργειας είναι δύσκολο να προσδιοριστεί. Αναφερόμενοι στο [72], εξετάζουμε συνολικά την περιβαλλοντική διάσταση για την αξιολόγηση της τεχνολογίας αποθήκευσης ενέργειας.

Οι κοινωνικοί παράγοντες περιλαμβάνουν τη δημιουργία θέσεων εργασίας, την κοινωνική αποδοχή, τα κρατικά κίνητρα, τον αντίκτυπο στην υγεία και την ασφάλεια. Λαμβάνοντας υπόψη αυτούς τους παράγοντες, επιλέχθηκε η κοινωνική αποδοχή για να εκφράσει τα κοινωνικά χαρακτηριστικά.

Η περίπτωση της επαρχίας Σανσί της Κίνας

Η Κίνα έχει θέσει ως στόχο τη μείωση των εκπομπών διοξειδίου του άνθρακα έως το 2030 και την προσπάθεια επίτευξης της κλιματικής ουδετερότητας έως το 2060. Η χρήση παραδοσιακών ορυκτών καυσίμων έχει προκαλέσει ατμοσφαιρική ρύπανση, και ως εκ τούτου η εφαρμογή των ανανεώσιμων πηγών ενέργειας αποτελεί έναν αποτελεσματικό τρόπο για τη μείωση των εκπομπών διοξειδίου του άνθρακα και την επίλυση των τρεχόντων προβλημάτων. Η παραγωγή ηλεκτρικής ενέργειας από ανανεώσιμες πηγές ενέργειας στην Κίνα αυξάνεται από το 2010. Το 14ο Πενταετές Σχέδιο που διατύπωσε η Κίνα το 2020 επεσήμανε ότι είναι απαραίτητο να προωθηθεί η καθαρή, χαμηλής εκπομπής διοξειδίου άνθρακα, ασφαλής και αποδοτική χρήση της ενέργειας[56].

Η επαρχία Σανσί βρίσκεται στα δυτικά των βουνών Ταϊχάνγκ. Η συνολική έκταση της είναι 156.000 τετραγωνικά χιλιόμετρα και ο μόνιμος πληθυσμός της επαρχίας ήταν 37.292 εκατομμύρια το 2020. Η Σανσί είναι η μεγαλύτερη επαρχία παραγωγής άνθρακα στην Κίνα, και η ετήσια παραγωγή ακατέργαστου άνθρακα της επαρχίας βρίσκεται εδώ και πολύ καιρό στην πρώτη γραμμή της βιομηχανίας άνθρακα της χώρας, αντιπροσωπεύοντας περισσότερο από ένα τέταρτο της συνολικής παραγωγής. Ως παραδοσιακή επαρχία παραγωγής άνθρακα στην Κίνα, η επαρχία Σανσί εφαρμόζει ενεργά το μετασχηματισμό της δομής ενέργειας. Η ετήσια κατανάλωση ηλεκτρικής ενέργειας της Σανσί ήταν $4.98{\times}10^7$ MWh το 2020, και το ποσοστό της παραγωγής ηλεκτρικής ενέργειας από ανανεώσιμες πηγές συνεχίζει να αυξάνεται. Επιπλέον, το 14ο Πενταετές Σχέδιο Ανανεώσιμης Ενέργειας της επαρχίας Σανσί προτείνει ότι κατά την περίοδο του 14ου Πενταετούς Σχεδίου, οι επενδύσεις της Σανσί σε ανανεώσιμες πηγές ενέργειας θα αυξηθούν περαιτέρω, και έως το 2025, η παραγωγή ηλεκτρικής ενέργειας από ανανεώσιμες πηγές θα αντιπροσωπεύει το 40% της συνολικής παραγωγής ηλεκτρικής ενέργειας. Ταυτόχρονα, ορισμένα έργα αποθήκευσης ανανεώσιμης ενέργειας θα υλοποιηθούν με την ανάπτυξη των ανανεώσιμων πηγών ενέργειας. Για ένα έργο αποθήκευσης ανανεώσιμης ενέργειας, είναι απαραίτητη μια προσέγγιση λήψης αποφάσεων για την επιλογή της τεχνολογίας αποθήκευσης ενέργειας[56].

Στην εργασία [56] έχει προταθεί ένα μοντέλο επιλογής που συνδυάζει τεχνικές ΠΑΑ και ασαφούς λογικής το οποίο δοκιμάζεται σε μια προσομοίωση από δεδομένα που συλλέχθηκαν με την συνδρομή δέκα εμπειρογνωμόνων. Αυτοί οι εμπειρογνώμονες προέρχονταν από το Πανεπιστήμιο Μεταλλείων και Τεχνολογίας της Κίνας στο Πεκίνο, το Πανεπιστήμιο Ηλεκτρικής Ενέργειας της Βόρειας Κίνας, την Εταιρεία Διεθνούς Ενέργειας της επαρχίας Σανσί και άλλες σχετικές με την ενέργεια μονάδες και έχουν πλούσια εμπειρία και γνώσεις στην αποθήκευση ενέργειας. Θα χρησιμοποιήσουμε τα δεδομένα-εκτιμήσεις από ορισμένους εξ αυτών και θα εφαρμόσουμε την επέκταση της μεθόδου VIKOR για επιδόσεις που είναι αριθμοί-διαστήματα.

Αρχικά ορίζουμε τις διαστάσεις του προβλήματος, δηλαδή τις πιθανές εναλλακτικές και τα κριτήρια επιλογής αυτών. Οι εναλλακτικές τεχνολογίες αποθήκευσης ενέργειας που συγκρίνονται είναι:

- Υδροηλεκτρική αποθήκευση (PHS - pumped hydro storage). Η μέθοδος αυτή αποθηκεύει ενέργεια με τη μορφή βαρυτικής δυναμικής ενέργειας του νερού, το οποίο αντλείται από μια δεξαμενή χαμηλότερου υψόμετρου σε μια δεξαμενή υψηλότερου υψόμετρου. Όταν υπάρχει περισσευούμενη ηλεκτρική ενέργεια για την οποία δεν υπάρχει ζήτηση, η οποία είναι χαμηλού κόστους, χρησιμοποιείται για τη λειτουργία των αντλιών. Κατά τις περιόδους υψηλής ζήτησης ηλεκτρικής ενέργειας, το αποθηκευμένο νερό απελευθερώνεται για την παρα-

γωγή ηλεκτρικής ενέργειας μέσω υδροηλεκτρικών τουρμπίνων και γεννητριών. Αν και οι απώλειες της διαδικασίας άντλησης καθιστούν το σύστημα συνολικά καθαρό καταναλωτή ενέργειας, εάν η ανώτερη δεξαμενή συλλέγει σημαντική βροχόπτωση ή τροφοδοτείται από ποτάμι, τότε ο σταθμός μπορεί να είναι καθαρός παραγωγός ενέργειας, όπως ένας παραδοσιακός υδροηλεκτρικός σταθμός[78].

- Αποθήκευση ενέργειας πεπιεσμένου αέρα (CAES - compressed air energy storage). Παρόμοια με την προηγούμενη εναλλακτική, με την διαφορά ότι η περισσευούμενη ενέργεια χρησιμοποιείται για την συμπίεση αέρα, ο οποίος μπορεί να απελευθερωθεί αργότερα για την παραγωγή της ζητούμενης ηλεκτρικής ενέργειας. Μια συνεχής πρόκληση στο σχεδιασμό μεγάλης κλίμακας είναι η διαχείριση της θερμικής ενέργειας, καθώς η συμπίεση του αέρα οδηγεί σε μια ανεπιθύμητη αύξηση της θερμοκρασίας, η οποία όχι μόνο μειώνει την αποδοτικότητα λειτουργίας αλλά μπορεί επίσης να οδηγήσει σε ζημιά. Σε σύγκριση με τις παραδοσιακές μπαταρίες, τα συστήματα CAES μπορούν να αποθηκεύσουν ενέργεια για μεγαλύτερες χρονικές περιόδους και απαιτούν λιγότερη συντήρηση[79].

- Αποθήκευση ενέργειας σε σφόνδυλο (FES - flywheel energy storage). Λειτουργεί επιταχύνοντας έναν ρότορα (σφόνδυλο) σε πολύ υψηλή ταχύτητα και διατηρώντας την ενέργεια στο σύστημα ως ενέργεια περιστροφής. Όταν εξάγεται ενέργεια από το σύστημα, η ταχύτητα περιστροφής του σφονδύλου μειώνεται ως συνέπεια της αρχής της διατήρησης της ενέργειας. Αντίστοιχα, η προσθήκη ενέργειας στο σύστημα οδηγεί σε αύξηση της ταχύτητας του σφονδύλου. Τα προηγμένα συστήματα FES έχουν ρότορες κατασκευασμένους από συνθετικά υλικά υψηλής αντοχής με βάση τις ίνες άνθρακα, αναρτημένους σε μαγνητικά έδρανα και περιστρεφόμενους με ταχύτητες από 20.000 έως πάνω από 50.000 σ.α.λ. σε περίβλημα κενού. Τέτοιοι σφόνδυλοι μπορούν να φτάσουν σε ταχύτητα μέσα σε λίγα λεπτά – φτάνοντας την ενεργειακή τους χωρητικότητα πολύ πιο γρήγορα από άλλες μορφές αποθήκευσης[80].

- Συσσωρευτές μολύβδου-οξέος(lead-acid batteries). Εφευρέθηκαν το 1859 και είναι ο παλαιότερος τύπος επαναφορτιζόμενης μπαταρίας και χρησιμοποιούν έναν ηλεκτρολύτη σε υγρή μορφή. Η τεχνολογία των μπαταριών μολύβδου-οξέος είναι απλή και το κόστος κατασκευής τους είναι χαμηλό. Ωστόσο, οι μπαταρίες αυτές φορτίζουν αργά, δεν μπορούν να αποφορτιστούν πλήρως και έχουν περιορισμένο αριθμό κύκλων φόρτισης/εκφόρτισης, λόγω της χαμηλής αναλογίας ενέργειας προς βάρος και της χαμηλής αναλογίας ενέργειας προς όγκο τους[81]. Ο μολύβδος και το θειικό οξύ που χρησιμοποιούνται είναι επίσης ιδιαίτερα τοξικά και δημιουργούν περιβαλλοντικούς κινδύνους. Η χημεία των μπαταριών μολύβδου-οξέος μπορεί να τροποποιηθεί για εφαρμογές αποθήκευσης σε δικτύου πέρα από τις εφαρμογές σταθεροποίησης, μέσω τροποποίησης των δομών των ηλεκτροδίων. Στις εφαρμογές αποθήκευσης ενέργειας, πολλές μπαταρίες μολύβδου-οξέος βαθιάς εκφόρτισης, οι οποίες παρέχουν σταθερό ρεύμα για μεγάλο χρονικό διάστημα, συνδέονται μεταξύ τους για να σχηματίσουν μια τράπεζα μπαταριών[82].

- Συσσωρευτές ιόντων λιθίου (Li-ion batteries). Είναι ο κύριος τύπος μπαταριών που χρησιμοποιείται σε φορητές ηλεκτρονικές συσκευές και ηλεκτρικά οχήματα. Χρησιμοποιούνται επίσης σε μεγάλο βαθμό για αποθήκευση ενέργειας σε επίπεδο δικτύου και σε στρατιωτικές και αεροδιαστημικές εφαρμογές. Σε σύγκριση με άλλες τεχνολογίες επαναφορτιζόμενων μπαταριών, οι μπαταρίες ιόντων λιθίου έχουν υψηλή ενεργειακή πυκνότητα, χαμηλή αυτοεκφόρτιση και υψηλή διάρκεια ζωής[83].

- Αποθήκευση ενέργειας υδρογόνου(hydrogen). Είναι μια μορφή αποθήκευσης χημικής ενέργειας, στην οποία η ηλεκτρική ενέργεια μετατρέπεται σε υδρογόνο. Αυτή η ενέργεια μπορεί στη συνέχεια να απελευθερωθεί ξανά χρησιμοποιώντας το αέριο ως καύσιμο σε κινητήρα εσωτερικής καύσης ή σε κυψέλη καυσίμου. Το υδρογόνο μπορεί να παραχθεί από ηλεκτρική ενέργεια με την ηλεκτρόλυση του νερού, μια απλή διαδικασία που μπορεί να πραγματοποιηθεί με σχετικά υψηλή απόδοση, εφόσον είναι διαθέσιμη φθηνή ενέργεια. Το υδρογόνο πρέπει στη συνέχεια να αποθηκευτεί, ενδεχομένως σε υπόγειες σπηλιές για αποθήκευση ενέργειας μεγάλης κλίμακας, αν και για αποθήκευση μικρότερης κλίμακας μπορούν να χρησιμοποιηθούν ατσάλινες δεξαμενές. Το υδρογόνο μπορεί να χρησιμοποιηθεί ως καύσιμο για κινητήρες εμβόλων, αεροστρόβιλους ή κυψέλες καυσίμου υδρογόνου, με τις τελευταίες να προσφέρουν την καλύτερη απόδοση[84].

- Υπερπυκνωτές (Supercapacitors). Είναι πολύ μεγάλης επιφάνειας ενεργοποιημένοι πυκνωτές που χρησιμοποιούν ένα λεπτό στρώμα ηλεκτρολύτη ως διηλεκτρικό υλικό για να διαχωρίσουν τα φορτία. Ο υπερπυκνωτής μοιάζει με έναν κανονικό πυκνωτή, με τη μόνη διαφορά ότι προσφέρει πολύ υψηλή χωρητικότητα σε μικρή συσκευασία. Η αποθήκευση ενέργειας γίνεται με στατικό φορτίο και όχι μέσω μιας ηλεκτροχημικής διαδικασίας όπως στις μπαταρίες[85].

- Υπεραγώγιμη αποθήκευση μαγνητικής ενέργειας (SMES - superconducting magnetic energy storage). Τα συστήματα αυτά αποθηκεύουν ενέργεια σε ένα μαγνητικό πεδίο. Αυτό το μαγνητικό πεδίο δημιουργείται από ένα συνεχές ρεύμα που διέρχεται από έναν υπεραγώγιμο πηνίο. Σε ένα κανονικό σύρμα, καθώς το ηλεκτρικό ρεύμα διέρχεται από το σύρμα, μέρος της ενέργειας χάνεται ως θερμότητα λόγω της ηλεκτρικής αντίστασης. Ωστόσο, σε ένα σύστημα SMES, το σύρμα είναι κατασκευασμένο από υπεραγώγιμο υλικό το οποίο έχει ψυχθεί κρυογενικά κάτω από την κρίσιμη θερμοκρασία του. Ως αποτέλεσμα, το ηλεκτρικό ρεύμα μπορεί να διέρχεται από το σύρμα με σχεδόν μηδενική αντίσταση, επιτρέποντας την αποθήκευση ενέργειας σε ένα σύστημα SMES για μεγαλύτερο χρονικό διάστημα. Τα κοινά υπεραγώγιμα υλικά περιλαμβάνουν τον υδράργυρο, το βανάδιο και το νιόβιο-τιτάνιο. Η ενέργεια που αποθηκεύεται σε ένα σύστημα SMES εκφορτίζεται με τη σύνδεση ενός μετατροπέα εναλλασσόμενου ρεύματος στον αγώγιμο πηνίο. Τα συστήματα SMES είναι μια εξαιρετικά αποδοτική τεχνολογία αποθήκευσης, αλλά έχουν πολύ χαμηλές ενεργειακές πυκνότητες και είναι ακόμα μακριά από το να είναι οικονομικά βιώσιμα[86].

- Θερμική αποθήκευση (TES). Τα συστήματα αποθήκευσης θερμικής ενέργειας αποθηκεύουν θερμική ενέργεια για μελλοντική χρήση. Υπάρχουν διάφοροι τύποι συστημάτων αποθήκευσης θερμικής ενέργειας [87]:

  - Αποθήκευση αισθητής θερμότητας: Σε αυτή τη μέθοδο, η θερμότητα αποθηκεύεται με την αύξηση της θερμοκρασίας ενός υλικού, όπως το νερό, οι πέτρες ή τα λιωμένα άλατα. Όταν χρειάζεται ενέργεια, η αποθηκευμένη θερμότητα απελευθερώνεται επιτρέποντας στο υλικό να ψυχρανθεί και να μεταφέρει τη θερμότητά του σε ένα ρευστό μέσο, όπως ο αέρας ή το νερό, το οποίο στη συνέχεια χρησιμοποιείται για την παραγωγή ηλεκτρικής ενέργειας ή την παροχή θέρμανσης.
  - Αποθήκευση λανθάνουσας θερμότητας: Τα συστήματα αποθήκευσης λανθάνουσας θερμότητας αποθηκεύουν ενέργεια αλλάζοντας τη φάση ενός υλικού, όπως από στερεά σε υγρή ή από υγρή σε αέρια. Τα υλικά αλλαγής φάσης (PCMs - phase change

materials) χρησιμοποιούνται συνήθως για το σκοπό αυτό. Όταν το υλικό στερεοποιείται ή εξατμίζεται, απελευθερώνει ή απορροφά θερμική ενέργεια.

– Θερμοχημική αποθήκευση (TCES): Η θερμοχημική αποθήκευση ενέργειας περιλαμβάνει χημικές αντιδράσεις που αποθηκεύουν και απελευθερώνουν θερμική ενέργεια. Εξειδικευμένα υλικά, γνωστά ως απορροφητικά ή αντιδραστήρια, υφίστανται αναστρέψιμες χημικές αντιδράσεις όταν εκτίθενται σε υψηλές ή χαμηλές θερμοκρασίες. Οι αντιδράσεις αυτές απελευθερώνουν ή απορροφούν θερμότητα όταν το υλικό εκτίθεται στις κατάλληλες συνθήκες. Τα συστήματα TCES έχουν το πλεονέκτημα της υψηλής ενεργειακής πυκνότητας και της δυνατότητας αποθήκευσης θερμότητας για μεγάλα χρονικά διαστήματα χωρίς σημαντικές απώλειες.

Τα τελικά κριτήρια πάνω στα οποία θα εφαρμοστεί η μέθοδος πολυκριτήριας ανάλυσης φαίνονται στον πίνακα 5.1 με τα χαρακτηριστικά τους:

Όνομα κριτηρίου	Μονάδα κριτηρίου	Όφελος/Κόστος	Έυρος τιμών
Ενεργειακή απόδοση	%	Όφελος	$(0, 100)$
Χρόνος απόκρισης	γλωσσικό	Κόστος	Πολύ Μικρός Μικρός Μεσαίος Μεγάλος Πολύ Μεγάλος
Διάρκεια ζωής	Χρόνια	Όφελος	$(0, \infty)$
Ενεργειακή πυκνότητα	Wh/kg	Όφελος	$(0, \infty)$
Απώλειες αυτοεκφόρτισης	%/μέρα	Κόστος	$(0, 100)$
Κόστος κεφαλαίου ενέργειας	$/kW	Κόστος	$(0, \infty)$
Κόστος κεφαλαίου ισχύος	$/kWh	Κόστος	$(0, \infty)$
Κοινωνική αποδοχή	γλωσσικό	Όφελος	Πολύ Μικρή Μικρή Μεσαία Μεγάλη Πολύ Μεγάλη

Πίνακας 5.1: Κριτήρια αξιολόγησης μέσων αποθήκευσης ενέργειας

Θεωρούμε τις επιδόσεις των μέσων αποθήκευσης ενέργειας ενός εμπειρογνώμονα στα επιλεγμένα κριτήρια που δίνονται από την εργασία [56] που δίνονται στον Πίνακα 5.2. Οι εκτιμήσεις στα περισσότερα κριτήρια αποτελούν αριθμούς-διαστήματα κάτι που καθιστά την επέκταση της μεθόδου VIKOR με επιδόσεις αριθμούς-διαστήματα άμεσα συμβατή μέθοδο ανάλυσης. Για ευκολία, στον πίνακα 5.3 δίνεται ένας τρόπος μετατροπής των γλωσσικών τιμών των κριτηρίων σε αριθμητικές τιμές.

	Απόδοση ενέργειας	Χρόνος απόκρισης	Διάρκεια ζωής	Ενεργειακή πυκνότητα	Απώλειες αυτοεκφόρτισης	Κόστος κεφαλαίου ενέργειας	Κόστος κεφαλαίου ισχύος	Κοινωνική αποδοχή
PHS	(75, 80)	Μεσαίος	(30, 60)	(0.5, 1.5)	(0.0001, 0.0001)	(600, 2000)	(5, 100)	Πολύ μεγάλη
CAES	(41, 75)	Μεγάλος	(20, 40)	(30, 60)	(0.0001, 0.0001)	(400, 800)	(50, 150)	Πολύ μεγάλη
FES	(88, 90)	Μεσαίος	(15, 20)	(5, 130)	(20, 100)	(250, 350)	(1000, 5000)	Πολύ μεγάλη
Lead–acid	(75, 80)	Πολύ μικρός	(3, 12)	(30, 50)	(0.1, 0.3)	(300, 600)	(150, 500)	Μεσαία
Li-ion	(65, 75)	Πολύ μικρός	(5, 15)	(75, 250)	(0.1, 0.3)	(1200, 4000)	(600, 2500)	Μεγάλη
Hydrogen	(35, 40)	Μεσαίος	(5, 15)	(800, 1000)	(0.5, 2)	(500, 10,000)	(2, 15)	Μεσαία
Super-capacitors	(85, 98)	Μικρός	(10, 20)	(0.1, 15)	(20, 40)	(100, 300)	(300, 2000)	Μεσαία
SMES	(90, 95)	Μικρός	(20, 30)	(0.5, 5)	(10, 15)	(200, 300)	(1000, 10,000)	Μεσαία
Thermal (TES)	(14, 18)	Μεγάλος	(5, 15)	(30, 60)	(0.05, 1)	(100, 400)	(3, 130)	Μεσαία

Πίνακας 5.2: Πίνακας απόφασης κατασκευασμένος από τις εκτιμήσεις ενός εμπειρογνώμονα

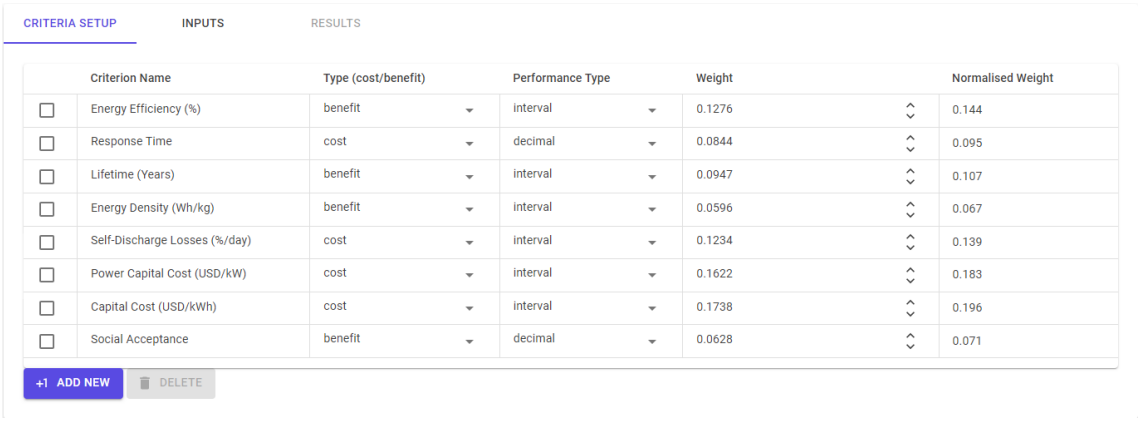
Γλωσσική τιμή	Αριθμητική τιμή
Πολύ καλός/μεγάλος	5
Καλός/Μεγάλος	4
Ουδέτερος/Μέτριος/Μεσαίος	3
Κακός/Μικρός	2
Πολύ κακός/μικρός	1

Πίνακας 5.3: Κανόνας μετατροπής γλωσσικών τιμών σε αριθμητικές τιμές

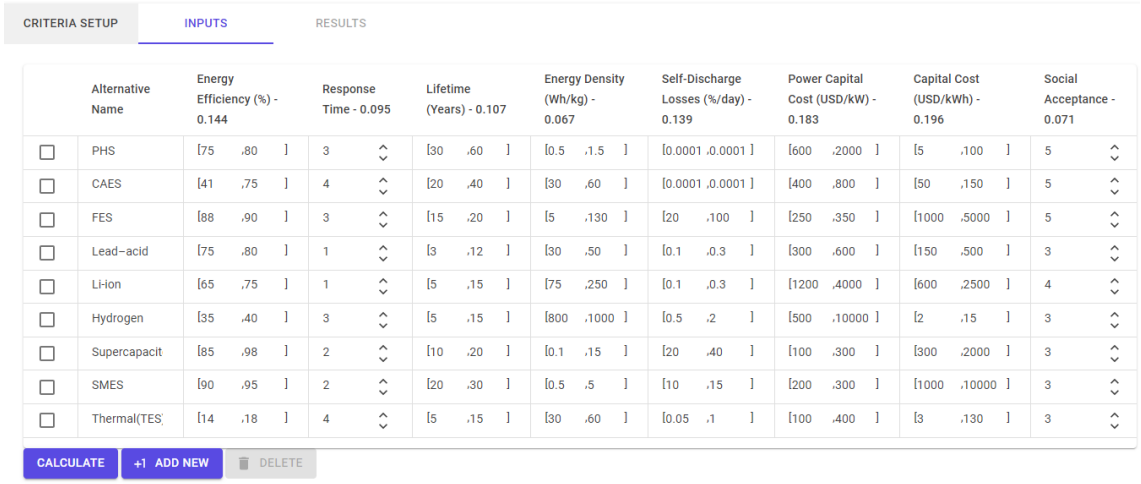
Στο εργαλείο ρυθμίζουμε τα κριτήρια και τις ιδιότητές τους όπως στον πίνακα 5.1. Επίσης, θέτουμε τα βάρη του εμπειρογνώμονα για τα συγκεκριμένα κριτήρια και αφήνουμε το εργαλείο να κάνει αυτόματα την κανονικοποίηση (Εικ. 5.1). Αφού γίνει μετατροπή και των γλωσσικών κριτηρίων, συμπληρώνουμε και τον πίνακα απόφασης στο εργαλείο (Εικ. 5.2). Υποθέτουμε έναν ουδέτερο ως προς το ρίσκο αποφασίζοντα ($v = 0.5$) και επιλέγουμε τη πιθανοτική μέθοδο ταξινόμησης των αριθμών διαστημάτων που παρουσιάστηκε στην Ενότητα 2.5.. Το αποτέλεσμα των

υπολογισμών φαίνεται φαίνεται στην Εικόνα 5.3. Η σειρά κατάταξης των εναλλακτικών σύμφωνα με την παραμετροποίηση της επιλεγμένης επέκτασης VIKOR από την καλύτερη προς την χειρότερη είναι:

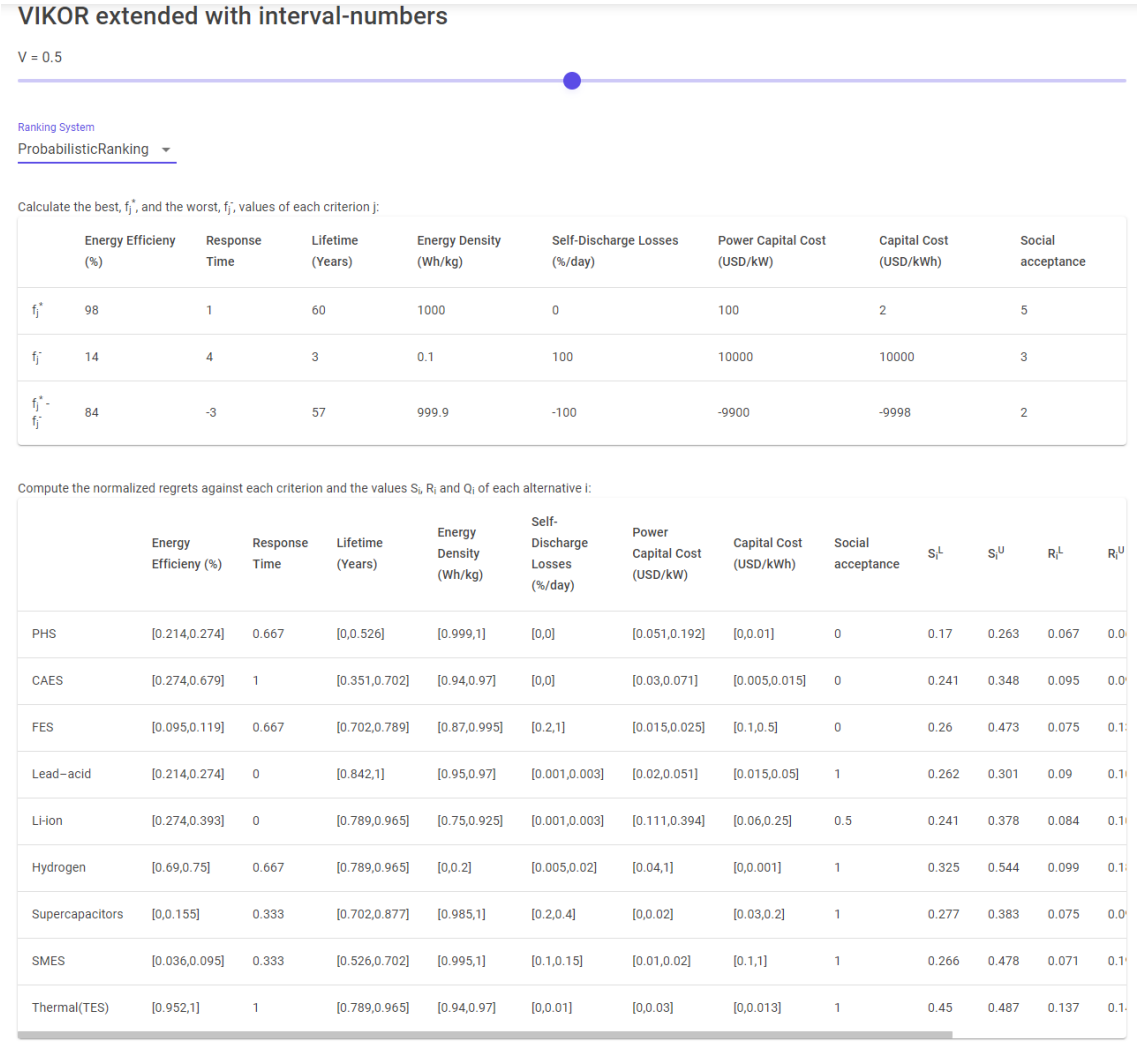
1. υδροηλεκτρική αποθηκευση - PHS
2. συσσωρευτές μολύβδου-οξέος - Lead–acid
3. αποθήκευση πεπιεσμένου αέρα - CAES
4. υπερπυκνωτές - Supercapacitors
5. συσσωρευτές ιόντων λιθίου - Li-ion
6. αποθήκευση σε σφόνδυλο - FES
7. υπεραγώγιμη αποθήκευση μαγνητικής ενέργειας - SMES
8. αποθήκευση υδρογόνου - Hydrogen
9. θερμική αποθήκευση - Thermal(TES)



Εικόνα 5.1: Τα κριτήρια περασμένα στο υπολογιστικό εργαλείο



Εικόνα 5.2: Συμπληρωμένος ο πίνακας απόφασης στο υπολογιστικό εργαλείο



VIKOR extended with interval-numbers

V = 0.5

Ranking System

ProbabilisticRanking

Calculate the best, f_j^*, and the worst, f_j^-, values of each criterion j:

	Energy Efficieny (%)	Response Time	Lifetime (Years)	Energy Density (Wh/kg)	Self-Discharge Losses (%/day)	Power Capital Cost (USD/kW)	Capital Cost (USD/kWh)	Social acceptance
f_j^*	98	1	60	1000	0	100	2	5
f_j^-	14	4	3	0.1	100	10000	10000	3
$f_j^* - f_j^-$	84	-3	57	999.9	-100	-9900	-9998	2

Compute the normalized regrets against each criterion and the values S_i, R_i and Q_i of each alternative i:

	Energy Efficieny (%)	Response Time	Lifetime (Years)	Energy Density (Wh/kg)	Self-Discharge Losses (%/day)	Power Capital Cost (USD/kW)	Capital Cost (USD/kWh)	Social acceptance	S_i^L	S_i^U	R_i^L	R_i^U
PHS	[0.214,0.274]	0.667	[0,0.526]	[0.999,1]	[0,0]	[0.051,0.192]	[0,0.01]	0	0.17	0.263	0.067	0.0
CAES	[0.274,0.679]	1	[0.351,0.702]	[0.94,0.97]	[0,0]	[0.03,0.071]	[0.005,0.015]	0	0.241	0.348	0.095	0.0
FES	[0.095,0.119]	0.667	[0.702,0.789]	[0.87,0.995]	[0.2,1]	[0.015,0.025]	[0.1,0.5]	0	0.26	0.473	0.075	0.1
Lead–acid	[0.214,0.274]	0	[0.842,1]	[0.95,0.97]	[0.001,0.003]	[0.02,0.051]	[0.015,0.05]	1	0.262	0.301	0.09	0.1
Li-ion	[0.274,0.393]	0	[0.789,0.965]	[0.75,0.925]	[0.001,0.003]	[0.111,0.394]	[0.06,0.25]	0.5	0.241	0.378	0.084	0.1
Hydrogen	[0.69,0.75]	0.667	[0.789,0.965]	[0,0.2]	[0.005,0.02]	[0.04,1]	[0,0.001]	1	0.325	0.544	0.099	0.1
Supercapacitors	[0,0.155]	0.333	[0.702,0.877]	[0.985,1]	[0.2,0.4]	[0,0.02]	[0.03,0.2]	1	0.277	0.383	0.075	0.0
SMES	[0.036,0.095]	0.333	[0.526,0.702]	[0.995,1]	[0.1,0.15]	[0.01,0.02]	[0.1,1]	1	0.266	0.478	0.071	0.1
Thermal(TES)	[0.952,1]	1	[0.789,0.965]	[0.94,0.97]	[0,0.01]	[0,0.03]	[0,0.013]	1	0.45	0.487	0.137	0.1

Final ranking is PHS>Lead–acid>CAES>Supercapacitors>Li-ion>FES>SMES>Hydrogen>Thermal(TES)

Εικόνα 5.3: Τα αποτελέσματα της εκτέλεσης της μεθόδου VIKOR

Κεφάλαιο 6

Συμπεράσματα και επεκτάσεις

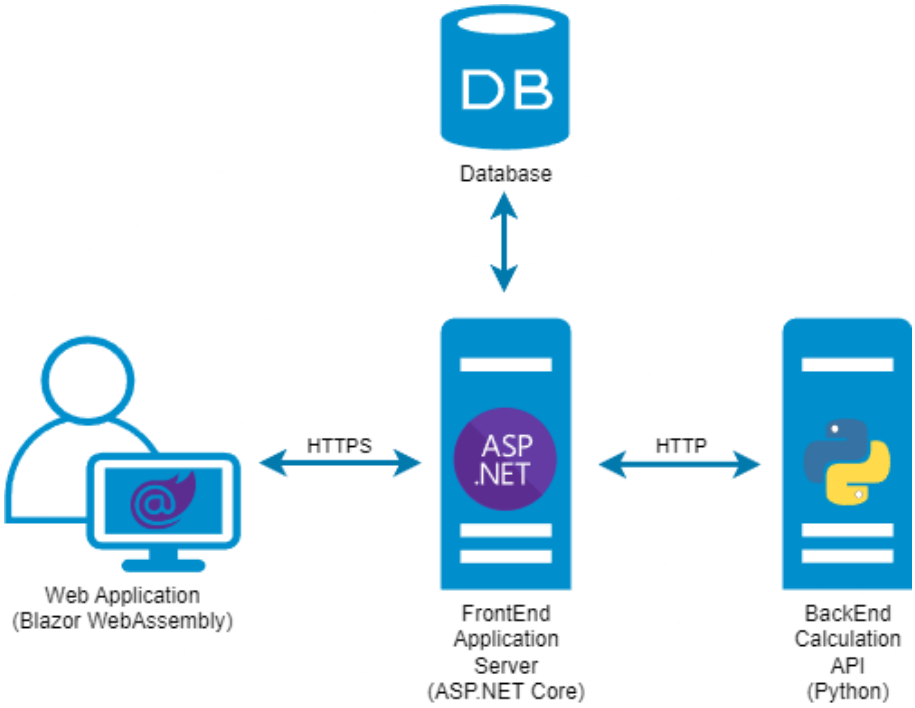
Η τρέχουσα έκδοση του υπολογιστικού εργαλείου μπορεί να ικανοποιεί τις απαιτήσεις αυτής της διπλωματικής εργασίας, όμως οι περισσότερες χρήσιμες εφαρμογές λογισμικού σπανίως παραμένουν αναλλοίωτες. Οι μέθοδοι ΠΑΑ που έχουν ήδη υλοποιηθεί είναι αρκετά ευέλικτες και μπορούν να εφαρμοστούν σε ένα ευρύ φάσμα προβλημάτων, ωστόσο αποτελούν μικρό κλάσμα του συνόλου των μεθοδολογιών που έχουν μελετηθεί στο πλαίσιο της πολυκριτήριας ανάλυσης αποφάσεων.

Χρησιμοποιώντας ως βάση το παρόν, δύναται με σχετικά μικρό κόπο η ενσωμάτωση περισσότερων μεθόδων. Χάρη στη καλά διαχωρισμένη δομή που έχει ακολουθηθεί τόσο στο επίπεδο του αρχιτεκτονικού σχεδιασμού όσο και στο επίπεδο της οργάνωσης του κώδικα, εκτιμάται ότι η σχέση του χρόνου προς τον αριθμό νέων μεθόδων ΠΑΑ προς ενσωμάτωση παραμένει γραμμική. Ακόμα καλύτερα, με κάθε νέο επαναχρησιμοποιήσιμο UI component που κατασκευάζεται, διευκολύνεται περαιτέρω η επιπλέον ανάπτυξη της σελίδας διεπαφής με τον χρήστη. Στη συνέχεια του κεφαλαίου προτείνονται κάποιες ιδέες για την μελλοντική ανάπτυξη του εργαλείου, για την προσθήκη νέων δυνατοτήτων ή βελτίωση των υπαρχόντων.

Σε ένα παραγωγικό υπολογιστικό σύστημα η πρόσβαση των χρηστών πρέπει, για λόγους ασφαλείας, να γίνεται ελεγχόμενα. Για το σκοπό αυτό ορίζονται η ταυτοποίηση (authentication) και η εξουσιοδότηση (authorization) ενός χρήστη. Η ταυτοποίηση αφορά την επαλήθευση της ταυτότητας ενός χρήστη. Σε αυτό το στάδιο, ο χρήστης παρέχει πιστοποιητικά, όπως όνομα χρήστη και κωδικό πρόσβασης, προκειμένου να αποδείξει ότι είναι όντως αυτός που ισχυρίζεται ότι είναι. Η εξουσιοδότηση αφορά στη χορήγηση πρόσβασης και δικαιωμάτων σε έναν χρήστη μετά την ταυτοποίηση. Αφού ο χρήστης ταυτοποιηθεί, η εξουσιοδότηση καθορίζει τι μπορεί να κάνει μέσα στο σύστημα και σε ποιους πόρους μπορεί να έχει πρόσβαση.

Ο ορισμός των χρηστών του συστήματος και η αποθήκευση των προσωπικών τους κωδικών συνήθως γίνεται σε μια βάση δεδομένων. Στην προτεινόμενη αρχιτεκτονική η ταυτοποίηση γίνεται στο server της FrontEnd εφαρμογής ο οποίος πλέον αποκτάει επικοινωνία με μια βάση δεδομένων (Εικόνα 6.1). Κατά την πρώτη επίσκεψη στην εφαρμογή, ο χρήστης θα καλείται να συμπληρώσει τα πιστοποιητικά του σε μια φόρμα, προκειμένου να λάβει ένα token περιορισμένης χρονικής εγκυρότητας. Με την αποστολή αυτού μέσα σε κάθε επόμενο αίτημα, ο server θα αποφασίζει αν

πρέπει να το εξυπηρετήσει ή να το αρνηθεί με κάποιο κωδικό σφάλματος[1].



Εικόνα 6.1: Προσθήκη βάσης δεδομένων στην αρχιτεκτονική του υπολογιστικού εργαλείου

Η ύπαρξη ταυτοποίησης είναι προαπαιτούμενη για την υποστήριξη της έννοιας της συνεδρίας χρήστη στην εφαρμογή. Έχοντας υλοποιημένα τα προηγούμενα, μπορεί πλέον με την είσοδο του χρήστη στο σύστημα να ορίζεται μια συνεδρία, η οποία αποθηκεύεται ακόμα και στην ίδια βάση δεδομένων. Περιοδικά ή με κάθε αλλαγή που πραγματοποιείται στη σελίδα από το χρήστη, η κατάσταση κωδικοποιείται και αποστέλλεται για αποθήκευση. Σε επόμενη ταυτοποιημένη είσοδο του ίδιου χρήστη, η εφαρμογή μπορεί να ξεκινήσει από την κατάσταση που βρισκόταν στην τελευταία συνεδρία του.

Στο Κεφάλαιο 4.5.1 αναφέρθηκε η ανάγκη για την συγγραφή κάποιων Unit Tests στο επίπεδο της υπολογιστικής λογικής. Σε μεγάλα συστήματα που λαμβάνουν συνεχείς ενημερώσεις, ελλοχεύει ο κίνδυνος εισαγωγής σφαλμάτων σε προϋπάρχουσες λειτουργίες του γραφικού περιβάλλοντος ή της επικοινωνίας μεταξύ των microservices. Για το σκοπό αυτό προτείνεται η δημιουργία τριών νέων ειδών test. Τα integration test που επιβεβαιώνουν την σωστή επικοινωνία ανάμεσα στα διάφορα συστήματα, για παράδειγμα την επιτυχή επικοινωνία με τη βάση δεδομένων ή αν απαντάται κάποιο αίτημα HTTP προς ένα υποσύστημα. Τα End-To-End (E2E) test που προσομοιώνουν μέσω frameworks αυτοματισμού την περιήγηση του χρήστη στο γραφικό περιβάλλον, και επιβεβαιώνουν ισχυρισμούς πάνω στο τι θα έπρεπε να υπάρχει σε αυτό, ανάλογα με το κάθε καταγεγραμμένο σενάριο. Τα stress test, τα οποία ελέγχουν αν ο βαθμός απόκρισης του συστήματος παραμένει στα αποδεκτά όρια κάτω από συνθήκες αυξημένου φόρτου, σε αριθμό αιτημάτων ή και σε ποσότητα πληροφορίας.

[1] στο πρωτόκολλο HTTP ο κωδικός σφάλματος είναι 401 Unauthorized

Καθώς ολοκληρώνουμε αυτήν τη διπλωματική εργασία, έχουμε εξερευνήσει τις ουσιώδεις πτυχές της δημιουργίας ενός ολοκληρωμένου υπολογιστικού εργαλείου για την υποστήριξη της λήψης αποφάσεων. Έχουμε συζητήσει δυνητικές βελτιώσεις στους τομείς της ασφάλειας και των test, αναγνωρίζοντας την καίρια σημασία της διασφάλισης της ακεραιότητας των δεδομένων και της διασφάλισης της αξιοπιστίας της εφαρμογής. Είναι απαραίτητο να θυμόμαστε ότι η τεχνολογία, παρόμοια με τις ανανεώσιμες πηγές ενέργειας που σκοπεύουμε να εκμεταλλευθούμε, πρέπει να εξελίσσεται και να προσαρμόζεται για να ανταποκριθεί στις προκλήσεις του μέλλοντος.

Κεφάλαιο 7

Βιβλιογραφία

[1] Blazor webassembly. https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-7.0#blazor-webassembly. Accessed: 27/09/2023.

[2] Jong Hyen Kim and Byeong Seok Ahn. Extended vikor method using incomplete criteria weights. *Expert Systems with Applications*, 126:124–132, 2019.

[3] Serafim Opricovic. Multicriteria optimization of civil engineering systems. *Faculty of civil engineering, Belgrade*, 2(1):5–21, 1998.

[4] Serafim Opricovic and Gwo-Hshiung Tzeng. Compromise solution by mcdm methods: A comparative analysis of vikor and topsis. *European Journal of Operational Research*, 156(2):445–455, 2004.

[5] Nian Zhang and Guiwu Wei. Extension of vikor method for decision making problem based on hesitant fuzzy set. *Applied Mathematical Modelling*, 37(7):4938–4947, 2013.

[6] Madjid Tavana, Reza Kiani Mavi, Francisco J. Santos-Arteaga, and Elahe Rasti Doust. An extended vikor method using stochastic data and subjective judgments. *Computers & Industrial Engineering*, 97:240–247, 2016.

[7] Shu-Ping Wan, Qiang-Ying Wang, and Jiu-Ying Dong. The extended vikor method for multi-attribute group decision making with triangular intuitionistic fuzzy numbers. *Knowledge-Based Systems*, 52:65–77, 2013.

[8] Hsiao-Fan Wang. Fuzzy multicriteria decision making—an overview. *Journal of Intelligent & Fuzzy Systems*, 9(1-2):61–83, 2000.

[9] Shu-Jen Chen and Ching-Lai Hwang. Fuzzy multiple attribute decision making methods. pages 289–486, 1992.

[10] Rita Almeida Ribeiro. Fuzzy multiple attribute decision making: a review and new preference elicitation techniques. volume 78, pages 155–181. Elsevier, 1996.

[11] Jati K Sengupta. *Optimal decisions under uncertainty: methods, models, and management*. Springer Science & Business Media, 2012.

[12] Peter Kall, Stein W Wallace, and Peter Kall. *Stochastic programming*, volume 5. Springer, 1994.

[13] Steven Vajda. *Probabilistic programming*. Academic Press, 2014.

[14] Xinwang Liu. On the methods of decision making under uncertainty with probability information. *International journal of intelligent systems*, 19(12):1217–1238, 2004.

[15] Ramon E Moore. *Methods and applications of interval analysis*. SIAM, 1979.

[16] Mohammad Kazem Sayadi, Majeed Heydari, and Kamran Shahanaghi. Extension of vikor method for decision making problem with interval numbers. *Applied Mathematical Modelling*, 33(5):2257–2262, 2009.

[17] Chia-Ling Chang. A modified vikor method for multiple criteria analysis. *Environmental Monitoring and Assessment*, 168(1):339–344, Sep 2010.

[18] Serafim Opricovic. Fuzzy vikor with an application to water resources planning. *Expert Systems with Applications*, 38(10):12983–12990, 2011.

[19] Tolga Kaya and Cengiz Kahraman. Fuzzy multiple criteria forestry decision making based on an integrated vikor and ahp approach. *Expert Systems with Applications*, 38(6):7326–7333, 2011.

[20] Abbas Aghajani Bazzazi, Morteza Osanloo, and Behrooz Karimi. Deriving preference order of open pit mines equipment through madm methods: Application of modified vikor method. *Expert Systems with Applications*, 38(3):2550–2556, 2011.

[21] Prasenjit Chatterjee and Shankar Chakraborty. A comparative analysis of vikor method and its variants. *Decision Science Letters*, 5:469–486, 09 2016.

[22] Liu P. & Wu X. A competency evaluation method of human resources managers based on multi-granularity linguistic variables and vikor method. *Technological and Economic Development of Economy*, 18(4):696–710, 2012.

[23] Tolga Kaya and Cengiz Kahraman. Multicriteria renewable energy planning using an integrated fuzzy vikor & ahp methodology: The case of istanbul. *Energy*, 35(6):2517–2527, 2010. 7th International Conference on Sustainable Energy Technologies.

[24] Hu-Chen Liu, Ling-Xiang Mao, Zhi-Ying Zhang, and Ping Li. Induced aggregation operators in the vikor method and its application in material selection. *Applied Mathematical Modelling*, 37(9):6325–6338, 2013.

[25] G. Nilay Yücenur and Nihan Çetin Demirel. Group decision making process for insurance company selection problem with extended vikor method under fuzzy environment. *Expert Systems with Applications*, 39(3):3702–3707, 2012.

[26] Xiaozhan Xu. A note on the subjective and objective integrated approach to determine attribute weights. *European Journal of Operational Research*, 156(2):530–532, 2004.

[27] Jian Ma, Zhi-Ping Fan, and Li-Hua Huang. A subjective and objective integrated approach to determine attribute weights. *European Journal of Operational Research*, 112(2):397–404, 1999.

[28] Yahya Dorfeshan, S Meysam Mousavi, and Behnam Vahdani. A multi-criteria analysis model under an interval type-2 fuzzy environment with an application to production project decision problems. *Journal of Quality Engineering and Production Optimization*, 3(1):43–66, 2018.

[29] Byeong Seok Ahn. Multiattribute decision aid with extended ismaut. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 36(3):507 – 520, 2006.

[30] Ward Edwards. How to use multiattribute utility measurement for social decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(5):326–340, 1977.

[31] Ward Edwards and F.Hutton Barron. Smarts and smarter: Improved simple methods for multiattribute utility measurement. *Organizational Behavior and Human Decision Processes*, 60(3):306–325, 1994.

[32] Thomas L. Saaty. What is the analytic hierarchy process? In Gautam Mitra, Harvey J. Greenberg, Freerk A. Lootsma, Marcel J. Rijkaert, and Hans J. Zimmermann, editors, *Mathematical Models for Decision Support*, pages 109–121, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[33] Wade D. Cook and Moshe Kress. A multiple criteria decision model with ordinal preference data. *European Journal of Operational Research*, 54(2):191–198, 1991.

[34] Behnam Malakooti. Ranking and screening multiple criteria alternatives with partial information and use of ordinal and cardinal strength of preferences. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, 30(3):355 – 368, 2000. Cited by: 88.

[35] Byeong Seok Ahn. Extreme point-based multi-attribute decision analysis with incomplete information. *European Journal of Operational Research*, 240(3):748–755, 2015.

[36] ZS Xu and Qing-Li Da. The uncertain owa operator. *International Journal of Intelligent Systems*, 17(6):569–575, 2002.

[37] Byeong Seok Ahn. The uncertain owa aggregation with weighting functions having a constant level of orness. *International Journal of Intelligent Systems*, 21(5):469–483, 2006.

[38] Sang Hyun Choi, Sungmin Kang, and Young Jun Jeon. Personalized recommendation system based on product specification values. *Expert Systems with Applications*, 31(3):607–616, 2006.

[39] Marcin Detyniecki and Ronald R Yager. Ranking fuzzy numbers using α-weighted valuations. *International Journal of uncertainty, Fuzziness and knowledge-Based systems*, 8(05):573–591, 2000.

[40] Xiangbai Gu and Qunxiong Zhu. Fuzzy multi-attribute decision-making method based on eigenvector of fuzzy attribute evaluation space. *Decision support systems*, 41(2):400–410, 2006.

[41] Young-Jou Lai, Ching-Lai Hwang, Young-Jou Lai, and Ching-Lai Hwang. *Fuzzy multiple objective decision making*. Springer, 1994.

[42] ES Lee and R-J Li. Comparison of fuzzy numbers based on the probability measure of fuzzy events. *Computers & Mathematics with Applications*, 15(10):887–896, 1988.

[43] Ronald R Yager. A procedure for ordering fuzzy subsets of the unit interval. *Information sciences*, 24(2):143–161, 1981.

[44] L Yu Po. *Forming winning strategies: An integrated theory of habitual domains*. Springer Science & Business Media, 2012.

[45] Ronald E Giachetti and Robert E Young. Analysis of the error in the standard approximation used for multiplication of triangular and trapezoidal fuzzy numbers and the development of a new approximation. *Fuzzy sets and systems*, 91(1):1–13, 1997.

[46] Chih-Hui Chiu and Wen-June Wang. A simple computation of min and max operations for fuzzy numbers. *Fuzzy Sets and Systems*, 126(2):273–276, 2002.

[47] Ronald E Giachetti and Robert E Young. A parametric representation of fuzzy numbers and their arithmetic operators. *Fuzzy sets and systems*, 91(2):185–202, 1997.

[48] George J Klir and Bo Yuan. Fuzzy sets and fuzzy logic: theory and applications. *Possibility Theory versus Probab. Theory*, 32(2):207–208, 1996.

[49] Shu-Jen Chen and Ching-Lai Hwang. *Fuzzy Multiple Attribute Decision Making Methods*, pages 289–486. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[50] Ching-Lai Hwang and Kwangsun Yoon. *Methods for Multiple Attribute Decision Making*, pages 58–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 1981.

[51] Johannes Thönes. Microservices. *IEEE Software*, 32(1):116–116, 2015.

[52] Xabier Larrucea, Izaskun Santamaria, Ricardo Colomo-Palacios, and Christof Ebert. Microservices. *IEEE Software*, 35(3):96–100, 2018.

[53] KR Srinath. Python–the fastest growing programming language. *International Research Journal of Engineering and Technology*, 4(12):354–357, 2017.

[54] Abhinav Nagpal and Goldie Gabrani. Python for data analytics, scientific and technical applications. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 140–145, 2019.

[55] Marius Musch, Christian Wressnegger, Martin Johns, and Konrad Rieck. New kid on the web: A study on the prevalence of webassembly in the wild. In Roberto Perdisci, Clémentine Maurice, Giorgio Giacinto, and Magnus Almgren, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 23–42, Cham, 2019. Springer International Publishing.

[56] Xiaotong Qie, Rui Zhang, Yanyong Hu, Xialing Sun, and Xue Chen. A multi-criteria decision-making approach for energy storage technology selection based on demand. *Energies*, 14(20):6592, 2021.

[57] Abdelrahman Azzuni and Christian Breyer. Energy security and energy storage technologies. *Energy Procedia*, 155:237–258, 2018. 12th International Renewable Energy Storage Conference, IRES 2018, 13-15 March 2018, Düsseldorf, Germany.

[58] C Lawrence Zitnick, Lowik Chanussot, Abhishek Das, Siddharth Goyal, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Thibaut Lavril, Aini Palizhati, Morgane Riviere, et al. An introduction to electrocatalyst design using machine learning for renewable energy storage. *arXiv preprint arXiv:2010.09435*, 2020.

[59] Van-Hai Bui, Akhtar Hussain, and Hak-Man Kim. Double deep q-learning-based distributed operation of battery energy storage system considering uncertainties. *IEEE Transactions on Smart Grid*, 11(1):457–469, 2019.

[60] Tianhan Gao and Wei Lu. Machine learning toward advanced energy storage devices and systems. *Iscience*, 24(1), 2021.

[61] Denis Sidorov, Fang Liu, and Yonghui Sun. Machine learning for energy systems. *Energies*, 13(18), 2020.

[62] Bin Xu, Denise Rizzo, and Simona Onori. Machine learning based optimal energy storage devices selection assistance for vehicle propulsion systems. Technical report, SAE Technical Paper, 2020.

[63] Yong Liu and Jun-liang Du. A multi criteria decision support framework for renewable energy storage technology selection. *Journal of Cleaner Production*, 277:122183, 2020.

[64] Abhishek Kumar, Bikash Sah, Arvind R Singh, Yan Deng, Xiangning He, Praveen Kumar, and RC Bansal. A review of multi criteria decision making (mcdm) towards sustainable renewable energy development. *Renewable and Sustainable Energy Reviews*, 69:596–609, 2017.

[65] Chia-Nan Wang, Van Thanh Nguyen, Hoang Tuyet Nhi Thai, and Duy Hung Duong. Multi-criteria decision making (mcdm) approaches for solar power plant location selection in viet nam. *Energies*, 11(6):1504, 2018.

[66] CC Strategy. Optimal decision-making in photovoltaic system selection in saudi arabia. *World Health*, 18:1–14, 2011.

[67] Ljubomir Gigović, Dragan Pamučar, Darko Lukić, and Slobodanka Marković. Gis-fuzzy dematel mcda model for the evaluation of the sites for ecotourism development: A case study of "dunavski ključ" region, serbia. *Land use policy*, 58:348–365, 2016.

[68] Marina Giamalaki and Theocharis Tsoutsos. Sustainable siting of solar power installations in mediterranean using a gis/ahp approach. *Renewable Energy*, 141:64–75, 2019.

[69] Md Mustafizur Rahman, Abayomi Olufemi Oni, Eskinder Gemechu, and Amit Kumar. Assessment of energy storage technologies: A review. *Energy Conversion and Management*, 223:113295, 2020.

[70] Haisheng Chen, Thang Ngoc Cong, Wei Yang, Chunqing Tan, Yongliang Li, and Yulong Ding. Progress in electrical energy storage system: A critical review. *Progress in natural science*, 19(3):291–312, 2009.

[71] Haoran Zhao, Qiuwei Wu, Shuju Hu, Honghua Xu, and Claus Nygaard Rasmussen. Review of energy storage system for wind power integration support. *Applied energy*, 137:545–553, 2015.

[72] Chonghui Zhang, Chao Chen, Dalia Streimikiene, and Tomas Balezentis. Intuitionistic fuzzy multimoora approach for multi-criteria assessment of the energy storage technologies. *Applied Soft Computing*, 79:410–423, 2019.

[73] Jingzheng Ren. Sustainability prioritization of energy storage technologies for promoting the development of renewable energy: A novel intuitionistic fuzzy combinative distance-based assessment approach. *Renewable energy*, 121:666–676, 2018.

[74] Helder Lopes Ferreira, Raquel Garde, Gianluca Fulli, Wil Kling, and Joao Pecas Lopes. Characterisation of electrical energy storage technologies. *Energy*, 53:288–298, 2013.

[75] Doe global energy storage database. 2018.

[76] Jannik Haas, Felix Cebulla, Wolfgang Nowak, Claudia Rahmann, and Rodrigo Palma-Behnke. A multi-service approach for planning the optimal mix of energy storage technologies in a fully-renewable power supply. *Energy Conversion and Management*, 178:355–368, 2018.

[77] Mathew Aneke and Meihong Wang. Energy storage technologies and real life applications–a state of the art review. *Applied Energy*, 179:350–377, 2016.

[78] Luai Al-Hadhrami. Pumped hydro energy storage system: A technological review. *Renewable and Sustainable Energy Reviews*, 44, 04 2015.

[79] Henrik Lund and Georges Salgi. The role of compressed air energy storage (caes) in future sustainable energy systems. *Energy Conversion and Management*, 50(5):1172–1179, 2009.

[80] Davide Castelvecchi. Spinning into control: High-tech reincarnations of an ancient way of storing energy. *Science News*, 171(20):312–313, 2007.

[81] Carl D Parker. Lead–acid battery energy-storage systems for electricity supply networks. *Journal of Power Sources*, 100(1-2):18–28, 2001.

[82] Andreas Poullikkas. A comparative overview of large-scale battery systems for electricity storage. *Renewable and Sustainable Energy Reviews*, 27:778–788, 2013.

[83] Holger C. Hesse, Michael Schimpe, Daniel Kucevic, and Andreas Jossen. Lithium-ion battery storage for the grid—a review of stationary battery storage system design tailored for applications in modern power grids. *Energies*, 10(12), 2017.

[84] Paul Breeze. Chapter 8 - hydrogen energy storage. In Paul Breeze, editor, *Power System Energy Storage Technologies*, pages 69–77. Academic Press, 2018.

[85] Ioannis Hadjipaschalis, Andreas Poullikkas, and Venizelos Efthimiou. Overview of current and future energy storage technologies for electric power applications. *Renewable and Sustainable Energy Reviews*, 13(6):1513–1522, 2009.

[86] Samuel C. Johnson, F. Todd Davidson, Joshua D. Rhodes, Justin L. Coleman, Shannon M. Bragg-Sitton, Eric J. Dufek, and Michael E. Webber. Chapter five - selecting favorable energy storage technologies for nuclear power. In Hitesh Bindra and Shripad Revankar, editors, *Storage and Hybridization of Nuclear Energy*, pages 119–175. Academic Press, 2019.

[87] Hasila Jarimi, Devrim Aydin, Zhang Yanan, Gorkem Ozankaya, Xiangjie Chen, and Saffa Riffat. Review on the recent progress of thermochemical materials and processes for solar thermal energy storage and industrial waste heat recovery. *International Journal of Low-Carbon Technologies*, 14(1):44–69, 12 2018.

Παράρτημα Α

Συμπληρωματικός κώδικας

Κώδικας Α.1: Το πλήρες περιεχόμενο του module calculation_model.py για την μέθοδο TOPSIS

```
from mcdm.topsis.request_response import TopsisRequest, TopsisResponse
import math


class TopsisModel:
    def __init__(self, request: TopsisRequest):
        self.Request = request

    def calculate_normalised_U_values(self, response: TopsisResponse):
        denominators = {crit.id: math.sqrt(
            sum(alternative.performances[crit.id] ** 2 for alternative
                ↪ in self.Request.alternatives)) for crit in self.
                ↪ Request.criteria}
        response.u = {}
        for alternative in self.Request.alternatives:
            response.u[alternative.id] = {}
            for crit in self.Request.criteria:
                response.u[alternative.id][crit.id] = crit.
                    ↪ normalisedWeight * \
                    alternative.performances[crit.id] / denominators[
                        ↪ crit.id]

    def calculate_positive_and_negative_ideal_solutions(self, response:
        ↪  TopsisResponse):
        response.positiveIdealU = {crit.id: (min(response.u[alt.id][
            ↪ crit.id] for alt in self.Request.alternatives) if crit.
            ↪ isCost else max(
            response.u[alt.id][crit.id] for alt in self.Request.
                ↪ alternatives)) for crit in self.Request.criteria}
```

```
        response.negativeIdealU = {crit.id: (max(response.u[alt.id][
            ↪ crit.id] for alt in self.Request.alternatives) if crit.
            ↪ isCost else min(
             response.u[alt.id][crit.id] for alt in self.Request.
                ↪ alternatives)) for crit in self.Request.criteria}

    def calculate_distances_from_ideal(self, response: TopsisResponse):
        response.distanceToPositiveIdeal={}
        response.distanceToNegativeIdeal={}
        response.closenessToIdeal={}
        for alternative in self.Request.alternatives:
            u_values = response.u[alternative.id]
            distanceToPositiveIdeal = math.sqrt(sum(
                (u_values[crit.id] - response.positiveIdealU[crit.id])
                    ↪ **2 for crit in self.Request.criteria))
            response.distanceToPositiveIdeal[alternative.id] =
                ↪ distanceToPositiveIdeal
            distanceToNegativeIdeal = math.sqrt(sum(
                (u_values[crit.id] - response.negativeIdealU[crit.id])
                    ↪ **2 for crit in self.Request.criteria))
            response.distanceToNegativeIdeal[alternative.id] =
                ↪ distanceToNegativeIdeal
            response.closenessToIdeal[alternative.id] =
                ↪ distanceToNegativeIdeal / (distanceToPositiveIdeal +
                ↪  distanceToNegativeIdeal)

    def calculate_final_ranking(self, response: TopsisResponse):
        response.finalRanking = [alt.id for alt in sorted(self.Request.
            ↪ alternatives, key=lambda x: response.closenessToIdeal[x.
            ↪ id], reverse=True)]

    def calculate_result(self) -> TopsisResponse:
        response = TopsisResponse()
        self.calculate_normalised_U_values(response)
        self.calculate_positive_and_negative_ideal_solutions(response)
        self.calculate_distances_from_ideal(response)
        self.calculate_final_ranking(response)
        return response
```

Κώδικας A□.2: Ο κώδικας του αρχείου Grids.razor για την μέθοδο VIKOR με αριθμούς διαστήματα και ελλιπή πληροφόρηση στα βάρη των κριτηρίων

```
@page "/vikor-intervals-relative-weights"

<PageTitle>VIKOR</PageTitle>

@using MCDM.Client.Common.DomainModel;
@using MCDM.Client.Shared.CustomComponents
```

```
@using MCDM.Shared;
@using MCDM.Shared.VikorBasic;
@using MCDM.Shared.VikorRelativeWeights;
@using MCDM.Client.Shared;
@using MudBlazor
@using System.Dynamic;
@using MCDM.Shared.Enums;
@using MCDM.Client.Common;
@inject ISnackbar Snackbar

<MudTabs Elevation="2" Rounded="true" ApplyEffectsToContainer="true" PanelClass
    ↪ ="pa-6" ActivePanelIndex="@_activeTabIndex" ActivePanelIndexChanged="@(x
    ↪  => _activeTabIndex = x)">
    <MudTabPanel Text="Criteria Setup">
        <MudDataGrid T="RankedCriterion" Items="Criteria" ReadOnly="false"
            ↪ EditMode=DataGridEditMode.Cell MultiSelection="true"
            ↪ CommittedItemChanges="@CommitCriteriaChanges"
                    Bordered="true" Dense="true" SelectedItems="
                        ↪ @SelectedCriteria" SelectedItemsChanged="
                        ↪ @SetCriteriaDeletedButton" SortMode="SortMode.None"
                        ↪  SelectOnRowClick="false">
            <Columns>
                <SelectColumn ShowInFooter="false" ShowInHeader="false" />
                <PropertyColumn Title="Criterion Name" Property="x => x!.Name"
                    ↪ />
                <PropertyColumn Title="Type (cost/benefit)" Property="x => x!.
                    ↪ IsCost">
                    <EditTemplate>
                        <MudSelect @bind-Value="context.Item!.IsCost" Required
                            ↪ RequiredError="You must select a Type!!!" Margin
                            ↪ ="@Margin.Dense">
                            <MudSelectItem Value="false">benefit</MudSelectItem
                                ↪ >
                            <MudSelectItem Value="true">cost</MudSelectItem>
                        </MudSelect>
                    </EditTemplate>
                </PropertyColumn>
                <PropertyColumn Title="Performance Type" Property="x => x!.
                    ↪ TypeId">
                    <EditTemplate>
                        <MudSelect T="PerformanceType" Value="context.Item!.
                            ↪ TypeId" Required RequiredError="You must select
                            ↪ a Type!!!" Margin="@Margin.Dense" ValueChanged="
                            ↪ @(x => ChangeCriterionType(context.Item, x))">
                            <MudSelectItem Value="@(PerformanceType.Interval)">
                                ↪ interval</MudSelectItem>
                            <MudSelectItem Value="@(PerformanceType.Decimal)">
                                ↪ decimal</MudSelectItem>
                        </MudSelect>
                    </EditTemplate>
                </PropertyColumn>
                <UpOrDownColumn MoveUp="MoveUp" MoveDown="MoveDown" />
            </Columns>
        </MudDataGrid>
        <MudButton Variant="Variant.Filled" StartIcon="@Icons.Material.Filled.
            ↪ PlusOne" Color="Color.Primary" OnClick="@AddCriterion">Add New</
```

```
            ↪ MudButton>
        <MudButton Variant="Variant.Filled" StartIcon="@Icons.Material.Filled.
            ↪ Delete" Color="Color.Error" Disabled="
            ↪ _isCriteriaDeleteButtonDisabled" OnClick="
            ↪ @DeleteSelectedCriteria">Delete</MudButton>
    </MudTabPanel>
    <MudTabPanel Text="Inputs">
        <MudDataGrid T="BaseAlternative" Items="@Alternatives" ReadOnly="false"
            ↪  EditMode=DataGridEditMode.Cell MultiSelection="true"
            ↪ CommittedItemChanges="@CommittedAlternativeChanges"
                    Bordered="true" Dense="true" SelectedItems="
                        ↪ @SelectedAlternatives" SelectedItemsChanged="
                        ↪ @SetAlternativesDeletedButton" SortMode="SortMode.
                        ↪ None" SelectOnRowClick="false">
            <Columns>
                <SelectColumn ShowInFooter="false" ShowInHeader="false" />
                <PropertyColumn Title="Alternative Name" Property="x => x!.Name
                    ↪ " />
                @foreach (var item in Criteria)
                {
                    <PerformanceColumn CriterionId="@item.Id" Title="@item.Name
                        ↪ " Type="@item.TypeId" />
                }
            </Columns>
        </MudDataGrid>
        <MudButton Variant="Variant.Filled" Color="Color.Primary" OnClick="
            ↪ @Calculate">Calculate</MudButton>
        <MudButton Variant="Variant.Filled" StartIcon="@Icons.Material.Filled.
            ↪ PlusOne" Color="Color.Primary" OnClick="@AddAlternative">Add New
            ↪ </MudButton>
        <MudButton Variant="Variant.Filled" StartIcon="@Icons.Material.Filled.
            ↪ Delete" Color="Color.Error" Disabled="
            ↪ _isAlternativesDeleteButtonDisabled" OnClick="
            ↪ @DeleteSelectedAlternatives">Delete</MudButton>
    </MudTabPanel>
    <MudTabPanel Text="Results" Disabled="@(_activeTabIndex != 2)">
        <Results Alternatives="@Alternatives" Criteria="@Criteria" />
    </MudTabPanel>
</MudTabs>


@code {
    private bool _isCriteriaDeleteButtonDisabled { get; set; } = true;
    private bool _isAlternativesDeleteButtonDisabled { get; set; } = true;

    private void SetCriteriaDeletedButton()
    {
        _isCriteriaDeleteButtonDisabled = !SelectedCriteria.Any();
    }

    private void SetAlternativesDeletedButton()
    {
        _isAlternativesDeleteButtonDisabled = !SelectedAlternatives.Any();
    }

    public HashSet<RankedCriterion> SelectedCriteria = new HashSet<
```

```
    ↪ RankedCriterion>();
public HashSet<BaseAlternative> SelectedAlternatives = new HashSet<
    ↪ BaseAlternative>();



private List<RankedCriterion> Criteria = new List<RankedCriterion>();

private List<BaseAlternative> Alternatives = new List<BaseAlternative>();

private int _activeTabIndex;

protected override Task OnInitializedAsync()
{
    AddCriterion();
    AddCriterion();
    AddCriterion();
    AddAlternative();
    AddAlternative();
    AddAlternative();
    AddAlternative();
    return base.OnInitializedAsync();
}

void AddCriterion()
{
    var currentId = Criteria.Select(x => x.Id).DefaultIfEmpty().Max();
    var nextId = currentId + 1;
    var newCriterion = new RankedCriterion { Id = nextId, Name = NameUtils.
        ↪ GenerateCriteriaName(nextId, Criteria.Select(x => x.Name)),
        ↪ TypeId = PerformanceType.Interval };

    foreach (var alterinative in Alternatives)
    {
        alterinative.Performances.Add(newCriterion.Id, Performance.
            ↪ GetDefault(newCriterion.TypeId));
    }

    Criteria.Add(newCriterion);
    AssignFirstOrLast();
}

void AddAlternative()
{
    var currentId = Alternatives.Select(x => (int)x.Id).DefaultIfEmpty(0).
        ↪ Max();
    var nextId = currentId + 1;
    var newAlternative = new BaseAlternative(nextId, NameUtils.
        ↪ GenerateAlternativeName(nextId, Alternatives.Select(x => x.Name)
        ↪ ));

    foreach (var criterion in Criteria)
    {
        newAlternative.Performances.Add(criterion.Id, Performance.
            ↪ GetDefault(criterion.TypeId));
    }
```

```
    Alternatives.Add(newAlternative);
}

void CommittedAlternativeChanges(BaseAlternative item)
{
    item.Name = item.Name.Trim();
    var newName = item.Name;
    if (string.IsNullOrWhiteSpace(newName))
    {
        item.Name = NameUtils.GenerateAlternativeName((int)item.Id,
            ↪ Alternatives.Select(x => x.Name));
        Snackbar.Add("Forbidden name. Reverting...", Severity.Error);
        return;
    }

    var groupped = Alternatives.GroupBy(x => x.Name);
    if (groupped.Any(x => x.Count() > 1))
    {

        item.Name = NameUtils.GenerateAlternativeName((int)item.Id,
            ↪ Alternatives.Select(x => x.Name));
        Snackbar.Add("You can't have the same name on multiple Alternatives
            ↪ . Reverting...", Severity.Error);
    }
}

void ChangeCriterionType(RankedCriterion item, PerformanceType newType)
{
    var previousType = item.TypeId;
    item.TypeId = newType;

    if (previousType != newType)
    {
        foreach (var alternative in Alternatives)
        {
            alternative.Performances.Remove(item.Id, out var _);
            alternative.Performances.Add(item.Id, Performance.GetDefault(
                ↪ item.TypeId));

        }
    }
}

void CommitCriteriaChanges(RankedCriterion item)
{
    var groupped = Criteria.GroupBy(x => x.Name);
    if (groupped.Any(x => x.Count() > 1))
    {

        item.Name = NameUtils.GenerateCriteriaName(item.Id, Criteria.Select
            ↪ (x => x.Name));
        Snackbar.Add("You can't have the same name on multiple criteria.
            ↪ Reverting...", Severity.Error);
        AssignFirstOrLast();
        return;
```

```
    }

    AssignFirstOrLast();
}

void MoveUp(RankedCriterion criterion)
{
    var temp = Criteria.ToList();

    var toMove = Criteria.Single(x => x.Id == criterion.Id);

    var index = Criteria.IndexOf(toMove);

    Criteria.Clear();

    Criteria.AddRange(temp.Take(Math.Max(0, index - 1)));

    Criteria.Add(toMove);

    Criteria.AddRange(temp.Skip(Math.Max(0, index - 1)).Where(x => x.Id !=
        ↪ toMove.Id));

    AssignFirstOrLast();
}

void MoveDown(RankedCriterion criterion)
{
    var temp = Criteria.ToList();

    var toMove = Criteria.Single(x => x.Id == criterion.Id);

    var index = Criteria.IndexOf(toMove);

    Criteria.Clear();

    Criteria.AddRange(temp.Take(Math.Min(temp.Count, index + 2)).Where(x =>
        ↪  x.Id != toMove.Id));

    Criteria.Add(toMove);

    Criteria.AddRange(temp.Skip(Math.Min(temp.Count, index + 2)));

    AssignFirstOrLast();
}

void AssignFirstOrLast()
{
    for (var i = 0; i < Criteria.Count; i++)
    {
        var rank = i + 1;
        var criterion = Criteria[i];
        criterion.Rank = rank;
        criterion.IsFirst = rank == 1;
        criterion.IsLast = rank == Criteria.Count;
    }
}
```

```
void DeleteSelectedCriteria()
{
    foreach (var criterion in SelectedCriteria)
    {
        Criteria.Remove(criterion);
        foreach (var alternative in Alternatives)
        {
            alternative.Performances.Remove(criterion.Id, out var
                ↪ previous_Value);
        }
    }

    SelectedCriteria.Clear();
    SetCriteriaDeletedButton();
    AssignFirstOrLast();
}

void DeleteSelectedAlternatives()
{
    foreach (var alternative in SelectedAlternatives)
    {
        Alternatives.Remove(alternative);
    }

    SelectedAlternatives.Clear();
    SetAlternativesDeletedButton();
}

void Calculate()
{
    var validationResult = ValuesAreValid();
    if (!validationResult.Item1)
    {
        Snackbar.Add(validationResult.Item2, Severity.Error,
            ↪ validationResult.Item3);
        return;
    }
    _activeTabIndex = 2;
}

(bool, string, Action<SnackbarOptions>?) ValuesAreValid()
{
    if (!Criteria.Any())
    {
        return (false, "At least one criterion must be present.", null);
    }

    if (Alternatives.Count() < 2)
    {
        return (false, "At least two alternative must be present.", null);
    }

    var uselessCriteria = new List<RankedCriterion>();
    foreach (var crit in Criteria)
    {
```

```
            var fValuesForThisCriterion = Alternatives.Select(x => x.
                ↪ Performances[crit.Id]).ToArray();
            var first = fValuesForThisCriterion[0];
            if (fValuesForThisCriterion.Skip(1).All(x => x.Equals(first)))
            {
                uselessCriteria.Add(crit);
            }
        }
        if (uselessCriteria.Any())
        {
            var singular = uselessCriteria.Count() == 1;
            return (false, $"Criteri"+ (singular ? "on" : "a") + " [" + string.
                ↪ Join(',', uselessCriteria.Select(x => $"\"{x.Name}\""))+ "]
                ↪ "+ (singular ? "has" : "have")+ " the same consequence for
                ↪ all alternatives. CLICK TO REMOVE", sbo => sbo.Onclick = sb
                ↪ =>
                InvokeAsync(() =>
                {
                    _activeTabIndex = 0;
                    StateHasChanged();
                    SelectedCriteria.Clear();
                    foreach (var ulc in uselessCriteria)
                    {
                        SelectedCriteria.Add(ulc);
                    }
                    DeleteSelectedCriteria();
                    StateHasChanged();
                })
            );
        }

        return (true, "", null);
    }
}
```

Κώδικας Α.3: Ο κώδικας του αρχείου Results.razor για την μέθοδο VIKOR με αριθμούς διαστήματα και ελλιπή πληροφόρηση στα βάρη των κριτηρίων

```
@using MCDM.Client.Common.DomainModel;
@using MCDM.Shared.Dtos;
@using MCDM.Shared.VikorRelativeWeights;
@using MCDM.Shared.VikorBasic;
@using MCDM.Shared;
@using MudBlazor;
@using System.Runtime.CompilerServices;
@using MCDM.Shared.VikorIntervalsRelativeWeights;
@using MCDM.Shared.Enums;
@using MCDM.Client.Common;
@inject HttpClient Http
@inject ISnackbar Snackbar

<h3>VIKOR extended with incomplete weights (weak inequalities) and interval
    ↪ numbers</h3>
<MudSlider T="double" Size="Size.Medium" Immediate="false" ValueLabel="true"
    ↪ Min="0" Max="1" Step="0.01" Value="@_v_value" ValueChanged="@(x =>
    ↪ Recalculate(x, _k_value, _ranking))">V = @_v_value.ToString()</MudSlider
```

```
    ↪ >
<br>
<br>
<div Style="width: 180px">
    <MudSelect T="RankingSystem" Label="Ranking System" AnchorOrigin="Origin.
        ↪ BottomCenter" Value="@_ranking" ValueChanged="@(r => Recalculate(
        ↪ _v_value, _k_value, r))">
        @foreach (RankingSystem system in Enum.GetValues(typeof(RankingSystem))
            ↪ )
        {
            <MudSelectItem Value="@system">@system</MudSelectItem>
        }
    </MudSelect>
</div>
<br>
<br>

@if (_ranking == RankingSystem.IntervalRanking)
{
    <div>Optimism level:</div>
    <MudSlider T="double" Size="Size.Medium" Immediate="false" ValueLabel="true
        ↪ " Min="0" Max="1" Step="0.01" Value="@_k_value" ValueChanged="@(x =>
        ↪  Recalculate(_v_value, x, _ranking))">K = @_k_value.ToString()</
        ↪ MudSlider>
    <br>
    <br>
}

@if (IsLoading)
{
    <MudProgressLinear Color="Color.Primary" Indeterminate="true" Class="my-7"
        ↪ />
}
else if (IsFailed)
{
    <MudText Color=@Color.Error Typo="@Typo.h6">@ErrorText</MudText>

}
else
{
    <div>
       Calculate the best, f<sub>j</sub><sup>*</sup>, and the worst, f<sub>j</
           ↪ sub><sup>-</sup>, values of each criterion j:
        <MudTable Items="@IdealPerformanceRows">
            <HeaderContent>
                <MudTh></MudTh>
                @foreach (var crit in Criteria)
                {
                    <MudTh>@crit.Name</MudTh>
                }
            </HeaderContent>
           <RowTemplate>
                <MudTd DataLabel="@context.datalabel">@((MarkupString)(context.
                    ↪ html))</MudTd>
                @foreach (var crit in Criteria)
                {
```

```
            <MudTd DataLabel="@(crit.Name)">@((MarkupString)(context.
                ↪ selector(Response, crit.Id).To3DigitDecimal()))</
                ↪ MudTd>
        }
    </RowTemplate>
</MudTable>
<br>
<br>
Compute the normalized regrets against each criterion and the values S<
    ↪ sub>i</sub>, R<sub>i</sub> and Q<sub>i</sub> of each alternative
    ↪  i:
<MudTable Items="@Alternatives">
    <HeaderContent>
        <MudTh></MudTh>
        @foreach (var item in Criteria)
        {
            <MudTh>@item.Name</MudTh>
        }
        <MudTh>S<sub>i</sub><sup>L</sup></MudTh>
        <MudTh>S<sub>i</sub><sup>U</sup></MudTh>
        <MudTh>R<sub>i</sub><sup>L</sup></MudTh>
        <MudTh>R<sub>i</sub><sup>U</sup></MudTh>
        <MudTh>Q<sub>i</sub><sup>L</sup></MudTh>
        <MudTh>Q<sub>i</sub><sup>U</sup></MudTh>
        <MudTh>Q<sub>i</sub><sup>M</sup></MudTh>
    </HeaderContent>
    <RowTemplate>
        <MudTd DataLabel="Alternative">@((MarkupString)(@context.Name))
            ↪ </MudTd>
        @foreach (var crit in Criteria)
        {
            if (crit.TypeId == PerformanceType.Interval)
            {
                var nr = (IntervalDto)Response.NormalizedRegrets[
                    ↪ context.Id][crit.Id];
                <MudTd DataLabel="@($"NR({crit.Name})")">@((
                    ↪ MarkupString)("[" + nr.Lower.To3DigitDecimal() +
                    ↪  "," + nr.Upper.To3DigitDecimal() + "]"))</MudTd
                    ↪ >
            }
            else
            {
                var nr = (DecimalDto)Response.NormalizedRegrets[context
                    ↪ .Id][crit.Id];
                <MudTd DataLabel="@($"NR({crit.Name})")">@((
                    ↪ MarkupString)(nr.Value.To3DigitDecimal()))</
                    ↪ MudTd>
            }
        }
        <MudTd DataLabel="@("S Lower Bound")">@((MarkupString)(Response
            ↪ .Sl[context.Id].To3DigitDecimal()))</MudTd>
        <MudTd DataLabel="@("S Upper Bound")">@((MarkupString)(Response
            ↪ .Su[context.Id].To3DigitDecimal()))</MudTd>
        <MudTd DataLabel="@("R Lower Bound")">@((MarkupString)(Response
            ↪ .Rl[context.Id].To3DigitDecimal()))</MudTd>
        <MudTd DataLabel="@("R Upper Bound")">@((MarkupString)(Response
```

```
                    ↪ .Ru[context.Id].To3DigitDecimal()))</MudTd>
                <MudTd DataLabel="@("Q Lower Bound")">@((MarkupString)(Response
                    ↪ .Ql[context.Id].To3DigitDecimal()))</MudTd>
                <MudTd DataLabel="@("Q Upper Bound")">@((MarkupString)(Response
                    ↪ .Qu[context.Id].To3DigitDecimal()))</MudTd>
                <MudTd DataLabel="@("Q Mean Value ")">@((MarkupString)(Response
                    ↪ .Qm[context.Id].To3DigitDecimal()))</MudTd>
            </RowTemplate>
        </MudTable>
        <br>
        <MudText Typo="@Typo.h6">@($"Final ranking is {string.Join('>',
            ↪ Response.FinalRanking.Select(x => Alternatives.First(a => a.Id
            ↪ == x).Name))}")</MudText>
    </div>
}



@code {
    [Parameter, EditorRequired] public List<BaseAlternative> Alternatives { get
        ↪ ; set; } = new List<BaseAlternative>();
    [Parameter, EditorRequired] public List<RankedCriterion> Criteria { get;
        ↪ set; } = new List<RankedCriterion>();

    private bool IsLoading { get; set; } = true;
    private bool IsFailed { get; set; } = false;
    private VikorIntervalsRelativeWeightsResponse Response { get; set; } = new
        ↪ VikorIntervalsRelativeWeightsResponse();


    private double _v_value = 0.5;
    private double _k_value = 0.5;

    private RankingSystem _ranking = RankingSystem.Midpoints;

    private string ErrorText { get; set; } = string.Empty;

    protected override async Task OnInitializedAsync()
    {
        await Recalculate(_v_value, _k_value, _ranking);
    }

    private async Task Recalculate(double v, double k, RankingSystem r)
    {
        IsLoading = true;
        IsFailed = false;
        _v_value = v;
        _k_value = k;
        _ranking = r;
        try
        {

            Response = await (await Http.PostAsJsonAsync("api/
                ↪ VikorIntervalsRelativeWeights", new
                ↪ VikorIntervalsRelativeWeightsRequest
```

```
        {
            Alternatives = Alternatives.Select(x =>
            {
                var perf = new Dictionary<int, PerformanceDto>();
                foreach (var p in x.Performances)
                {
                    if (p.Value is IntervalPerformance ip)
                    {
                        perf[p.Key] = new IntervalDto { Lower = ip.Value.
                            ↪ Item1, Upper = ip.Value.Item2 };
                    }
                    else if (p.Value is DecimalPerformance dp)
                    {
                        perf[p.Key] = new DecimalDto { Value = dp.Value };
                    }
                };
                return new PolymorphicAlternativeDto { Id = x.Id, Name = x.
                    ↪ Name, Performances = perf };
            }).ToArray(),
                Criteria = Criteria.Select(x => new
                    ↪ PolymorphicRankedCriterionDto
                {
                    Id = x.Id,
                    Name = x.Name,
                    IsCost = x.IsCost,
                    Rank = x.Rank,
                    TypeId = x.TypeId
                }).ToArray(),
            V = _v_value,
            K = _k_value,
            Ranking = _ranking
        }, JsonSettings.DefaultJsonOptions)).Content.ReadFromJsonAsync<
            ↪ VikorIntervalsRelativeWeightsResponse>() ?? new
            ↪ VikorIntervalsRelativeWeightsResponse();

    }
    catch (Exception ex)
    {
        IsFailed = true;
        ErrorText = ex.Message;
        Snackbar.Add("Calculation error. For more information see
            ↪ application logs", Severity.Error);
    }
    finally
    {
        IsLoading = false;
    }
}

private static readonly (Func<VikorIntervalsRelativeWeightsResponse, int,
    ↪ decimal> selector, string html, string datalabel)[]
    ↪ IdealPerformanceRows =
new (Func<VikorIntervalsRelativeWeightsResponse, int, decimal> selector,
    ↪ string html, string datalabel)[] {
    ((x, i) => x.PositiveIdealF[i], "f<sub>j</sub><sup>*</sup>", "
        ↪ IdealPerformance"),
```

```
        ((x, i) => x.NegativeIdealF[i], "f<sub>j</sub><sup>-</sup>", "
            ↪ NegativeIdealPerformance"),
        ((x, i) => x.DifferenceF[i], "f<sub>j</sub><sup>*</sup> - f<sub>j</sub
            ↪ ><sup>-</sup>", "Difference") };

}
```