Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Vícekriteriální analýza variant

2023

Marek Brodacký

Vedoucí práce:
doc. RNDr. Miroslav Kolařík, Ph.D.

Studijní program: Informatika,
Specializace: Obecná informatika

Bibliografické údaje

Autor:	Marek Brodacký
Název práce:	Vícekriteriální analýza variant
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní program:	Informatika, Specializace: Obecná informatika
Vedoucí práce:	doc. RNDr. Miroslav Kolařík, Ph.D.
Počet stran:	47
Přílohy:	elektonická data v úložišti Katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Marek Brodacký
Title:	Multiple Criteria Decision Making
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study program:	Computer Science, Specialization: General Computer Science
Supervisor:	doc. RNDr. Miroslav Kolařík, Ph.D.
Page count:	47
Supplements:	electronic data in the repository of the Department of Computer Science
Thesis language:	Czech

Anotace

Model vícekriteriální analýzy variant se používá pro pomoc při rozhodování komplexních problémů. Text této práce je zaměřen na popis a definici tohoto modelu společně s metodami pro normalizaci, určení vah a získání požadovaného rozhodnutí. Na základě nastudované teorie je vytvořen Python balíček s názvem „mymcdm“, který slouží k práci s modelem vícekriteriální analýzy variant.

Synopsis

The multiple attribute decision making model is used to aid in decision making for complex problems. The text of this paper focuses on the description and definition of this model along with methods for normalization, determining weights, and obtaining a requested decision. Based on the theory studied, a Python package called „mymcdm“ is created to work with the multicriteria variance analysis model.

Klíčová slova: vícekriteriální analýza variant; vícekriteriální rozhodování; normalizační metody; metody vážení; Python balíček

Keywords: multiple attribute decision making; multiple-criteria decision making; normalization methods; weighing methods; Python package

Rád bych podělkoval doc. RNDr. Miroslavu Kolaříkovi, Ph.D. za cenné rady, poznatky a ochotu vést tuto práci.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

U každodenních problémů, kde se lidé rozhodují primárně podle své intuice, obvykle není nutné využívat rozhodovacích nástrojů. Avšak komplexní problémy, které mohou ušetřit peníze, čas nebo zachránit lidské životy, potřebují více než jen intuici.[1] U takto komplexních problémů je nutné vzít v úvahu všechna obvykle protichůdná kritéria a učinit rozhodovací proces více strukturalizovaný a objektivní. Z tohoto důvodu vznikla rozhodovací disciplína zvaná vícekriteriální rozhodování.

Vícekriteriální rozhodování pomáhá při řešení komplexních problémů. Bere v úvahu všechna rozhodovatelem určená kritéria, která slouží pro práci s problémem. Rozhodnutí na základě vícekriteriálních modelů jsou strukturalizované, objektivní a transparentní. Vícekriteriální rozhodování se dělí na dva modely, a to na vícekriteriální analýzu variant a vícekriteriální programování.[2] Tyto modely mají rozdílná využití. Vícekriteriální analýza variant se zabývá výběrem z konečné množiny variant, zatímco vícekriteriální programování je určeno pro návrh a optimalizaci.

Cílem mé práce bylo nastudovat a zpracovat model vícekriteriální analýzy variant a jeho teorie. Na základě nově načerpaných informací, jsem vytvořil nástroj, který umožní zpracovat komplexní problémy s pomocí modelu vícekriteriální analýzy variant. Nástroj obsahuje subjektivní a objektivní metody určení důležitosti kritérií, metody pro normalizaci variant a rozhodovací metody pro určení výsledku analýzy. Nástroj je zpracován ve formě balíčku pro jazyk Python. Ten lze použít pro naprogramování postupu nebo je možno využít poskytnutého konzolového rozhraní, které umožňuje použití rozhodovacích metod vícekriteriální analýzy variant v příkazové řádce.

Práce je rozdělena na teoretickou a praktickou část. Teoretická část začíná kapitolou 2, která je úvodem do vícekriteriální analýzy variant. Tato kapitola obsahuje kroky práce s modelem vícekriteriální analýzy variant a možné cíle rozhodnutí. Následuje kapitola 3 obsahující vysvětlení pojmů variant, kritérií a normalizace. Poslední kapitolou teoretické části je kapitola 4. Ta obsahuje rozhodovací metody modelu vícekriteriální analýzy variant. Praktická část nejprve uvádí použité technologie a nástroje a následně podrobně popisuje vytvořený nástroj. Nakonec v příloze A je ukázka práce s modelem vícekriteriální analýzy variant a v příloze B je ukázka použití balíčku.

[1] Komplexní problémy se naprosto běžně objevují ve zdravotnictví. Protože na řešení těchto problémů mohou záležet životy, využívají se rozhodovací modely, které napomáhají s rozhodnutím. Více lze najít zde.

[2] V anglickém jazyce se tyto dva modely nazývají „Multiple Attribute Decision Making“ (MADM) a „Multiple Objective Decision Making“ (MODM) a vícekriteriální rozhodování se v anglickém překladu nazývá „Multiple Criteria Decision Making“ (MCDM). Pojmy MCDM a MADM se dosti zaměňují.

2 Vícekriteriální analýza variant

Tato kapitola vychází z poznatků [1], [2], [3], [4] a článku [5].

Vícekriteriální analýza variant, zkráceně VAV, je model vícekriteriálního rozhodování, který se používá jako pomoc při rozhodování komplexních problémů. Pro tento model existuje i samostatná teorie zabývající se způsoby zpracování vstupních dat, určení důležitosti jednotlivých kritérií a metody pro určení výsledku analýzy. Výsledek rozhodovacích metod určuje výsledek celé analýzy (modelu) a měl by sloužit jako podpora při rozhodování problémů.

Model VAV se používá jako nástroj umožňující problém analyzovat, strukturalizovat a najít vlastnosti různých řešení zadaného problému. Model obsahuje prvky, které jsou popsány v následující definici.

Definice 1 (Prvky modelu VAV)
Model vícekriteriální analýzy variant se skládá z následujících prvků:

- **Základní definice problému.** Informace o problému a jeho popis.
- **Cíl modelu VAV.** Označuje formu vráceného výsledku modelu.
- **Kritéria, jejich váhy a typy.** Kritéria jsou hlediska, na základě kterých jsou varianty hodnoceny. Váhy určují důležitost kritérií. Typy kritérií udávají jak s těmito kritérii pracovat.
- **Varianty a její vlastnosti.** Varianty určují konkrétní rozhodovací možnosti. Vlastnosti variant jsou dány hodnotami, které určují hodnocení varianty podle kritérií.

Model VAV je tedy dán požadovaným cílem analýzy a informacemi o problému, variantách a kritériích. Pro dosažení požadovaného cíle se zpracují všechny tyto informace, popřípadě se doplní chybějící, a následně jsou zpracovány pomocí některé rozhodovací metody VAV.[3] Rozhodovací metoda vrátí výsledek, který je zároveň i cílem analýzy. Výsledek rozhodnutí poté slouží jako pomoc při vytváření řešení problému.

S modely VAV pracuje jedna nebo více osob. Ty musí být s problémem obeznámeny, aby s ním dokázaly vhodně pracovat. Od rozhodovatelů je požadováno vytvoření modelu VAV a následná práce s ním. Během práce musí vytvořit strukturu modelu, volit vhodná kritéria vztahující se k problému a zvolit varianty. Dále budou volit vhodné metody pro normalizaci, určení vah nebo pro konečné rozhodnutí modelu. V konečné fázi rozhodovatelé výsledek zkontrolují a na základě něj vytvoří finální řešení.

Definice 2 (Rozhodovatelé)
Osoba či skupina osob pracující s modelem VAV se označují jako rozhodovatelé

[3] Pokud budu v této práci zmiňovat rozhodovací metody, tak mám na mysli ty, které patří k modelu VAV.

nebo experti. Mají za úkol pracovat s problémem a na základě modelu VAV vytvořit odůvodnění pro finální řešení problému.

2.1 Popularita a použití

Vícekriteriální rozhodování a jeho modely byly zpopularizovány publikací, kterou vytvořil S. Zionts s názvem „MCDM: If Not a Roman Numeral, then What?“. Zionts se ve své práci snažil přiblížit vícekriteriální rozhodování širšímu publiku a rok poté v roce 1980 se tak opravdu stalo. Došlo totiž k výraznému pokroku díky vzniku různých skupin a asociací [2].

Vícekriteriální rozhodováním se v posledních letech stalo více populární. To dokazuje výzkum [6], který ukazuje, že od roku 2012 do roku 2021 vzrostl počet studií zabývající se vícekriteriálním rozhodováním. Studie se podle výzkumu nejvíce zaměřují na oblasti strojírenství, energetiku a vědu o životním prostředí.

Vícekriteriální analýza variant se využívá v různých odvětvích a lze ji aplikovat na spousty problémů. Využívá se v odvětvích jako finance, ekonomika, zdravotnictví, vzdělávání a ve spoustě dalších. Konkrétní využití může být například při výběru produktu; rozdělení seznamu studentů pro udělení stipendií; výběr materiálu pro výrobu; nebo hodnocení obnovitelných zdrojů energie.

Pro doplnění uvedu hlavní rozdíly mezi modely vícekriteriálního rozhodování, aby bylo jasné jejich rozdílné použití. Model vícekriteriální analýzy variant se používá pro výběr nebo práci s množinou variant, zatímco model vícekriteriálního programování (VP) je spojen s návrhem (optimalizací) podle cílů a omezení. Ostatní rozdíly jsou shrnuty Tabulce 1.

	VAV	**VP**
Kritéria definují	vlastnosti variant	zadané cíle
Varianty	zadány explicitně	nejsou předem určeny
Interakce s rozhodovatelem	nízká	vysoká
Použití	výběr	návrh

Tabulka 1: Rozdíly modelů vícekriteriálního rozhodování

2.2 Práce s modelem VAV

Tato podkapitola vychází z [2] a [5].

Pro získání požadovaného cíle modelu VAV je potřeba dodržet řadu kroků. Dodržením kroků se zajistí objektivnost a správnost výsledného rozhodnutí modelu. V následujících šesti krocích popíšu postup při práci s modelem VAV.

Krok 1: Vytvoření struktury modelu VAV
Základním krokem je vytvoření struktury modelu VAV. Identifikuje se problém společně s jeho vlastnostmi. Vlastnosti problému později poslouží pro volbu roz-

hodovacích metod. Vytvoří se souhrn informací o problému a požadovaný cíl problému.

Krok 2: Vytvoření kritérií
Vytvoří se kritéria, která budou reprezentovat vlastnosti variant. Kritéria budou použita jak u vlastností variant, tak u rozhodovacích metod. Při výběru kritérií je nutné dbát na to, aby byla co nejvíce relevantní vzhledem k zadanému problému. Vybraná kritéria by měla být nezávislá a neměla by se překrývat. Pokud takové vlastnosti nemají, pak se musí vybrat kritéria tak, aby je měly.

Překrytím je myšlena skutečnost, že dvě kritéria označují stejnou věc. Například pokud při výběru automobilu zároveň zvolíme kritéria atraktivity a krásy [5]. Z příkladu je pochopitelné, že by se váhy kritérií započítaly dvakrát.

Kritérium je závislé, pokud jej jiné kritérium přímo ovlivňuje. Například pokud při výběru počítače zvolíme kritéria grafická karta a herní výkon, hned můžeme vidět, že kritérium herní výkon je závislé na kritériu grafická karta.

Krok 3: Zvolení a ohodnocení variant
Nejprve se zvolí varianty, které určují možné výsledky rozhodnutí. Následně se určí hodnoty vlastností variant na základě zvolených kritérií. Tomuto procesu se říká ohodnocení variant. Ohodnocení variant by mělo probíhat co nejvíce objektivně a spravedlivě vzhledem ke každému kritériu.

Krok 4: Zpracování informací
Zpracování získaných dat se dělí na tři kroky:

- Převedení nečíselných hodnot vlastností variant na číselné.
- Sjednocení měřítka vlastností variant pomocí normalizačních metod. Vybrané normalizační metody jsou popsány v kapitole 3.3.
- Volba metod a následné zjištění vah kritérií, pokud váhy nebyly zadány explicitně.[4] V kapitole 3.2 jsou popsány některé subjektivní a objektivní metody určující váhy kritérií.

Krok 5: Použití rozhodovací metody
Na základě zvoleného cíle a informací o problému se vybere rozhodovací metoda VAV. V kapitole 4 popisuji mnou vybrané rozhodovací metody. Rozhodovací metoda vezme všechny údaje zpracované a vytvořené v předešlých krocích a vypočítá výsledek rozhodnutí.

Krok 6: Využití výsledku rozhodnutí
V posledním kroku je rozhodovateli předán výsledek rozhodnutí metody z předchozího kroku. Výsledek následně slouží jako doporučení pro rozhodnutí celého problému nebo v případě potřeby lze výsledek použít ke sdělení a zdůvodnění konečného rozhodnutí zúčastněným stranám [5].

[4] Některé typy rozhodovacích metod pro výpočet cíle nevyužívají váhy kritérií. Více o těchto metodách lze dohledat v knize [1].

2.3 Cíle modelu

Modely vícekriteriální analýzy variant se rozlišují podle jejich cílů. Cíle určují typ výsledku, ke kterému má model dojít. Cíle řešení modelů VAV mohou být následující:

- **Cílem je vybrat nejlepší variantu.**

  U těchto modelů je cílem vybrat nejvhodnější variantu z dané množiny variant a doporučit ji jako řešení. Modely mající takový cíl jsou například výběr rodinného automobilu nebo start-up projektu pro investici.

  Metoda určující nejlepší variantu je například metoda VIKOR. Lze vybrat i metody, které uspořádají množinu variant a následně vybrat nejlepší variantu z uspořádání.

- **Cílem je sestavení žebříčku variant.**

  Cílem je uspořádání variant od nejlepší po nejhorší variantu. Příkladem modelu může být vytvoření žebříčku obnovitelných zdrojů energie.

  Existuje spousta metod určujících uspořádání. Příklady metod mající za cíl uspořádání lze najít v kapitole 4. Stejně jako u předchozího cíle se může k dosažení výsledku cíle použít alternativní přístup. Opakovaně se použije proces výběru nejlepší varianty, která se následně odebere z množiny uvažovaných variant a zařadí se nakonec do uspořádání.

- **Cílem je rozdělit množinu variant.**

  Zde se rozdělí množina variant na varianty „dobré“ a „špatné“. Například je požadováno odstranit všechny varianty nesplňující pomyslnou hranici. Tento cíl může mít například rozdělení seznamu studentů pro udělení stipendií.

  Pro určení tohoto cíle je možné použít více způsobu. Nejjednodušší je vytvořit atrapu. Atrapa je varianta uměle vytvořená a sloužící jako pomyslný práh. Atrapa se vloží do množiny variant a následně se použijí takové rozhodovací metody, které uspořádají množinu variant. V uspořádání vznikne rozdělení, kde varianty větší než atrapa se označí jako „dobré“ a ostatní jako „špatné“.

3 Teorie

V následujících podkapitolách popisuji základní pojmy teorie vícekriteriální analýzy variant. Nejprve vysvětlím pojem varianty a definuji označení pro ty se speciálními vlastnostmi. Následně uvedu pojem kritéria a rozdělení metod pro určování jejich vah společně s postupem vybraných metod. Nakonec vysvětlím normalizační proces a metody, které se pro normalizaci využívají.

3.1 Varianty

Tato podkapitola vychází z knih [3] a [4].

Varianty udávají konkrétní rozhodovací možnosti, se kterými se při rozhodování pracuje.[5] Všechna rozhodnutí ve vícekriteriální analýze variant probíhají na základě konečné množiny variant. Varianty konkrétního modelu VAV mají stejný počet vlastností, který je dán počtem kritérií modelu.

Definice 3 (Množina variant)
V modelu VAV je množina variant $A = \{A_1, A_2, \ldots, A_m\}$ konečná a udává možnosti, na základě kterých vznikne výsledek rozhodnutí.

Definice 4 (Varianta)
Varianta $A_i = (r_1, r_2, \ldots, r_n)$ je n-tice dána prvky r_j nazývané vlastnosti. Vlastnosti udávají ohodnocení varianty A_i podle kritéria C_j.

Varianty a jejich vlastnosti lze reprezentovat maticí variant $A_{m\times n}$, kde prvek r_{ij} udává ohodnocení i. varianty podle j. kritéria. Matice variant je zobrazena v Tabulce 3.1.

$$\left(\begin{array}{c|cccc} A_1 & r_{11} & r_{12} & \ldots & r_{1n} \\ A_2 & r_{21} & r_{22} & \ldots & r_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_m & r_{m1} & r_{m2} & \ldots & r_{mn} \end{array}\right)$$

Tabulka 2: Matice variant

Vlastnosti variant mohou obsahovat i nečíselné hodnoty, které je nutné převést na číselné. Například u výběru telefonu je typ displeje nečíselný údaj a musí se určit číselné hodnoty, které budou tyto hodnoty reprezentovat. Zároveň se musí zajistit objektivita při určení a správnost převedení nečíselných hodnot na hodnoty číselné.

3.1.1 Označení variant

Varianty lze označovat, pokud mají speciální vlastnosti nebo jejich hodnoty dosahují extrémních hodnot, podle následujících definic. Nechť máme zadaný model **M** vícekriteriální analýzy variant a některé jeho varianty označené D, C, P, I, B.

- **Dominovaná varianta**, označme ji $D = (d_1, d_2, \ldots, d_n)$, je varianta pro kterou existuje varianta $C = (c_1, c_2, \ldots, c_n)$ taková, že $\forall i \in \mathbb{N}$, kde $i \leq n$, platí $d_i \leq c_i$ a zároveň $\exists i$ takové, že $d_i < c_i$.

[5] V anglickém jazyce se pro popis variant používá „alternatives“ (alternativy) v textu budu pro tento pojem používat slovo „varianty“.

- **Paretovská varianta** P je varianta, která není dominovaná žádnou variantou.
- **Ideální varianta** I je reálná, či hypotetická varianta, u které všechny vlastnosti nabývají současně nejlepších hodnot oproti ostatním variantám modelu.
- **Bazální varianta** B je opakem ideální varianty.[6] Je to reálná, či hypotetická varianta, u které všechny vlastnosti nabývají současně nejhorších hodnot.

3.2 Kritéria

Tato podkapitola byla vytvořena na základě poznatků z [4], [7], [8], [9].

Kritéria jsou hlediska na základě kterých jsou varianty ohodnoceny. Každé kritérium má přiděleno svou váhu. Čím vyšší je váha daného kritéria, tím vyšší je důležitost vlastností variant, které jsou podle něj ohodnoceny. Konečný výsledek rozhodnutí do značné míry závisí na hodnotách vah kritérií [9].

Definice 5 (Množina kritérií)
V modelu VAV je množina kritérií $C = \{C_1, C_2, \ldots, C_n\}$ konečná. Pojem váhový vektor či vektor vah je vektor $\vec{w} = (w_1, w_2, \ldots, w_m)$, kde w_j značí váhu j. kritéria.

Definice 6 (Kritérium)
Kritérium je určeno svým označením C_j, váhou w_j a typem. Kritérium C_j určuje ohodnocení vlastnosti r_j každé varianty. Typ kritéria může být buď maximalizační nebo minimalizační.[7]

Definice 7 (Váhy)
Váhy určují relativní důležitost kritérií. Váhy jsou hodnoty z uzavřeného intervalu $\langle 0, 1 \rangle$. Musí platit, že $\sum_{i=1}^{m} w_j = 1$.

Kritéria se dělí na dva typy. Maximalizační kritéria mají za cíl maximalizovat hodnoty vlastností variant, zatímco minimalizační kritéria mají cíl opačný a to hodnoty vlastností minimalizovat. Pokud nevezmeme v úvahu typ kritéria, pak je výsledek celé analýzy nesprávný.

Například při výběru telefonu určíme kritéria velikost paměti RAM, rychlost procesoru a cena zařízení. Potom kritéria velikost paměti RAM a rychlost procesoru jsou maximalizačním typem kritérií a kritérium cena zařízení je typickým příkladem minimalizačního typu kritéria.

[6] V anglickém jazyce je bazální varianta nazývána jako „Negative ideal“.

[7] V anglickém jazyce se maximalizačním a minimalizačním kritériím říká „benefitial criteria“ a „cost criteria“.

Definice 8 (Rozhodovací matice)
Rozhodovací matice obsahuje všechny důležité informace, které následně budou použity v rozhodovací metodě pro získání požadovaného cíle modelu. V rozhodovací matici jsou použity prvky: množina kritérií $C = \{C_1, C_2, \ldots, C_n\}$, jejich váhy a typy; matice variant $A_{m \times n}$.

Řádek s označením „Krit.“ (Kritéria) označuje kritéria a pro každé kritérium je na následujícím řádku jeho váha. Maximalizační a minimalizační typy kritérií jsou označeny symboly + a −. Ve spodní části je matice variant $A_{m \times n}$ označena pomocí „Var.“ (Varianty).

Krit.	C_1^+	C_2^-	$\ldots$	$C_n^{\cdots}$
	w_1	w_2	$\ldots$	w_n
Var.				
A_1	r_{11}	r_{12}	$\ldots$	r_{1n}
A_2	r_{21}	r_{22}	$\ldots$	r_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
A_m	r_{m1}	r_{m2}	$\ldots$	r_{mn}

Tabulka 3: Rozhodovací matice

V praxi je obtížné i pro jednoho rozhodovatele dodat číselné relativní váhy různých rozhodovacích kritérií [7]. Přirozeně proto vznikly metody určování vah kritérií. Tyto metody se dělí na tři typy, a to subjektivní, objektivní a integrované metody vážení.

U subjektivních metod se váhy kritérií určují na základě posouzení rozhodovatele nebo skupiny rozhodovatelů. Výhodou subjektivních metod může být využití znalostí a zkušeností rozhodovatelů. Určení může například probíhat na základě seřazení kritérií od nejdůležitějších po nejméně důležité, přiřazením bodů každému kritériu nebo předložením souboru otázek.

Naopak u objektivních metod nejsou zahrnuty posouzení ani názory žádného rozhodovatele a váhy kritérií jsou určeny na základě informací z variant. K objektivnímu určení vah se používají matematické modely nebo i entropie. Výhodou může být menší časová náročnost za cenu vynechání subjektivního názoru rozhodovatele.

Integrované metody vážení jsou kombinací objektivních a subjektivních metod. Do výsledných vah se zahrnují informace vypočtené pomocí objektivních metod i posouzení a subjektivní názor rozhodovatele, kvůli jeho znalostem z příslušné oblasti.

V následujících dvou podkapitolách představím metody určení vah, které jsou použity v mé práci. Vždy uvedu český název metody a v závorce název anglický. Většina objektivních metod využívá matici variant. Knihovna z praktické části žádné integrované metody neobsahuje, proto tuto kategorii vynechám.

3.2.1 Subjektivní metody vážení

Bodovací metoda (The point allocation method) využívá body přiřazené rozhodovateli k určení vah kritérií. Každý z rozhodovatelů rozdělí celkový počet bodů, které rozděluje kritériím. Čím vyšší počet bodů kritérium dostane, tím vyšší je jeho relativní důležitost. Bodové hodnocení lze reprezentovat jako matici $P_{r\times n}$. Každý z řádků označuje rozdělení bodů jednoho z rozhodovatelů. Váhový vektor $\vec{w} = (w_1, w_2, \ldots, w_m)$ se vypočítá pomocí následujícího vzorce:

$$w_j = \frac{\sum_{k=1}^{r} p_{kj}}{m},\ j = 1, 2, \ldots, n\,.$$

Za celkový počet bodů se většinou volí hodnota 100, ale pokud se řádek vhodně normalizuje, pak každý z rozhodovatelů může zvolit vlastní celkový počet bodů. Váhy získané při použití metody přidělování bodů nejsou příliš přesné a metoda se stává obtížnější, jakmile se počet kritérií zvýší na šest nebo více [7].

Metody párových porovnání (The pairwise comparisons methods) se používají pro porovnání důležitosti každého z uspořádaných dvojic kritérií. Párům je přiřazena hodnota z ordinální stupnice. Rozmezí stupnic se může lišit. Metoda párového porovnávání je často kritizována za to, že se jednoduše ptá na relativní důležitost kritérií bez ohledu na měřítka, na kterých jsou kritéria měřena [9]. Pár budu v následujících odstavcích označovat uspořádanou dvojící (x, y), kde x je srovnáno s y.

Nejpoužívanější metodou využívající párového porovnání je Saatyho analytický hierarchický proces (AHP). Metoda využívá Saatyho stupnici, která přiřazuje párům hodnotu z množiny $\{1, 3, 5, 7, 9\}$, kde hodnoty $\{3, 5, 7, 9\}$ označují vyšší intenzitu preference x oproti y a hodnota 1 vyjadřuje stejnou intenzitu důležitost x a y. Pro vyjádření opačné intenzity preference y oproti x se používají hodnoty z množiny $\{1/3, 1/5, 1/7, 1/9\}$, které jsou inverzní k hodnotám $\{3, 5, 7, 9\}$. Významy těchto hodnot lze najít v Tabulce 4. Stupnice se dá rozšířit hodnotami z množiny $\{2, 4, 6, 8\}$ (případně hodnotami inverzními $\{1/2, 1/4, 1/6, 1/8\}$), které představují mezistupně pro případné potřeby kompromisu.

Intenzita důležitosti	**Definice**
1	Stejná důležitost
3	Střední preference
5	Silná preference
7	Velmi silná preference
9	Extrémní preference
2, 4, 6, 8	Mezistupně

Tabulka 4: Základní Saatyho stupnice

Váhový vektor $\vec{w} = (w_1, w_2, \ldots, w_m)$ se vypočítá pomocí následujícího postupu, který je jednou z částí rozhodovací metody AHP a nazývá se metoda

vlastního vektoru. Postup výpočtu vlastního vektoru jsem čerpal z knihy [1] a kroky určení analýzy konzistence jsem čerpal z knihy [2].

1. Nejprve vytvoříme čtvercovou matici párových porovnání $C_{n\times n}$. V ní přiřadíme intenzitu důležitosti pro každý z párů. Matice C obsahuje na hlavní diagonále hodnotu 1, protože prvek je sám sobě rovným.

   Například uvažujeme tři prvky e_1, e_2, e_3. Určíme, že prvek e_1 je silně preferovaný oproti prvku e_2. Poté do matice na pozici c_{12} zapíšeme pět a na pozici c_{21} inverzní hodnotu $1/5$. Pokračujeme dokud neurčíme intenzitu důležitosti pro každý pár.

2. Nakonec vypočítáme hlavní vlastní vektor $\vec{w}$ matice C, který určuje výsledný váhový vektor.[8] Nejprve musíme vypočítat největší vlastní hodnotu λ_{max} matice C vyřešením homogenní lineární rovnice $\det(C - \lambda I) = 0$, kde $I_{n\times n}$ značí jednotkovou matici. Řešení s největší hodnotou λ označíme λ_{max}. Výsledný vektor $\vec{w}$ lze získat vyřešením lineární rovnice $C\vec{w} = \lambda_{max}\vec{w}$, kde z Definice 7 víme, že součet hodnot vektoru $\vec{w}$ je roven 1.

Po dokončení výpočtu bychom měli provést analýzu konzistence. K určení konzistence potřebujeme vypočítat poměr konzistence (CR). Ten vypočítáme za pomocí vzorců

$$CI = \frac{\lambda_{max} - m}{m - 1},$$

$$CR = \frac{CI}{RI},$$

kde CI označuje index konzistence a RI průměrný náhodný index konzistence. Hodnota RI se zjistí z Tabulky 5. Pokud je poměr konzistence $CR \leq 0{,}10$, pak je rozhodnutí konzistentní. Jinak je rozhodnutí nekonzistentní a matice párových porovnání se musí vytvořit znovu.

Počet prvků	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RI	0,00	0,00	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49	1,51	1,54	1,56	1,57	1,59

Tabulka 5: Saatyho tabulka průměrných náhodných indexů

Metody párových porovnání je možno použít i pro výpočet hodnot variant. Pro každé kritérium vytvoříme matici párových porovnání. Hodnoty matic budou odpovídat porovnání páru variant podle daného kritéria. Použijeme metodu vlastního vektoru pro výpočet výsledných vektorů. Výsledný vektor, určený podle kritéria, je ohodnocením variant podle daného kritéria.

[8] Vlastní vektor se vypočítá následovně. Nechť máme zadanou čtvercovou matici $A_{n\times n}$. Pokud existuje řešení rovnice $A\vec{x} = \lambda\vec{x}$, kde $\lambda \in \mathbb{C}$ a $\vec{x}$ je nenulový sloupcový vektor velikosti $n\times 1$, pak z řešení dostaneme vlastní vektor $\vec{x}$ velikosti n a vlastní číslo λ. Vektor $\vec{x}$ z řešení, které má největší hodnotou vlastního čísla λ (označované λ_{max}) oproti ostatním řešení, nazveme hlavní vlastní vektor.

3.2.2 Objektivní metody vážení

Metoda rovnoměrných vah (Mean Weight) určí všem kritériím stejnou důležitost. Většinou se používá v případě, že rozhodovatel nemá k dispozici žádné informace [7]. Váhový vektor $\vec{w} = (w_1, w_2, \ldots, w_m)$ se vypočítá pomocí následujícího vzorce

$$w_j = \frac{1}{m}, \ j = 1, 2, \ldots, n\,,$$

kde m je počet kritérií.

Entropická metoda (Entropy Method) pro výpočet vah využívá entropii. Entropie je mírou neurčitosti informace formulovanou pomocí teorie pravděpodobnosti [9]. Váhový vektor $\vec{w} = (w_1, w_2, \ldots, w_m)$ se vypočítá pomocí následujícího postupu.

1. Normalizujeme matici variant $A_{m\times n}$ a dostaneme tak matici $N_{m\times n}$.
2. Vypočítáme informační hodnoty entropie e_j pomocí vzorce

$$e_j = -\frac{(\sum_{i=1}^{m} n_{ij} \ln(n_{ij}))}{m}, \ j = 1, 2, \ldots, n\,.$$

3. Nakonec vypočítáme váhový vektor $\vec{w}$ pomocí vzorce

$$w_j = \frac{1 - e_j}{\sum_{i=1}^{n}(1 - e_j)}, \ j = 1, 2, \ldots, n\,.$$

Metoda směrodatné odchylky (Standard Deviation Method) pro výpočet vah se nejprve vypočítá směrodatná odchylka σ_j příslušného sloupce matice A. Váhový vektor $\vec{w} = (w_1, w_2, \ldots, w_m)$ se následně vypočítá pomocí následujícího vzorce

$$w_j = \frac{\sigma_j}{\sum_{i=1}^{n} \sigma_j}, \ j = 1, 2, \ldots, n\,.$$

Metoda rozptylu (Statistical Variance Procedure) váhový vektor $\vec{w} = (w_1, w_2, \ldots, w_m)$ se vypočítá stejně jako Metoda směrodatné odchylky, až na ten rozdíl, že směrodatnou odchylku nahradí rozptylem φ.

CRITIC (Criteria importance through inter-criteria) využívá korelační analýzu ke zjištění kontrastů mezi kritérii a výpočtu jejich vah. Váhový vektor $\vec{w} = (w_1, w_2, \ldots, w_m)$ se vypočítá pomocí následujícího postupu.

1. Normalizujeme matici variant $A_{m\times n}$ pomocí normalizační metody Max-min z kapitoly 3.3. Dostaneme tak matici $N_{m\times n}$.

2. Vypočítáme matici $P_{n\times n}$ korelačních koeficientů ρ pomocí vzorce

$$\rho_{jk} = \frac{\sum_{i=1}^{m}(n_{ij} - n_j)(n_{ik} - n_k)}{\sqrt{\sum_{i=1}^{m}(n_{ij} - n_j)^2 \sum_{i=1}^{m}(n_{ij} - n_j)^2}},$$

pro $j, k = 1, \ldots, n$.

3. Vypočítáme vektor $\vec{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ objemu informací pomocí vzorce

$$\beta_j = \sigma_j \sum_{k=1}^{n}(1 - \rho_{jk}),\ j = 1, 2, \ldots, n\,,$$

kde σ_j je směrodatná odchylka příslušného sloupce matice $N_{m\times n}$.

4. Nakonec vypočítáme váhový vektor $\vec{w}$ pomocí vzorce

$$w_j = \frac{\beta_j}{\sum_{k=1}^{n}\beta_k},\ j = 1, 2, \ldots, n\,.$$

3.3 Normalizace variant

Tato podkapitola vychází z poznatků z článků [10], [11] a [12].

Normalizace je pojem, který se vyskytuje v mnoha odvětvích matematiky, statistiky a informatiky. Ve vícekriteriální analýze variant je normalizace transformační proces, jehož cílem je získat číselné a srovnatelné vstupní údaje pomocí společného měřítka [10]. Například při výběru telefonu určíme kritéria velikost paměti RAM, kvalita fotoaparátu, rychlost procesoru. Určená kritéria mají různá měřítka a jednotky. Pro zachování správnosti výpočtů je potřeba sjednotit měřítka na jedno společné.

Výzkumníci bohužel často podceňují význam správného výběru metody normalizace dat [11], přestože má výběr velký dopad na výsledek celé analýzy. Několik studií o vlivu normalizačních metod na pořadí variant v problémech VAV ukázalo, že některé metody jsou pro konkrétní rozhodování vhodnější než jiné [12]. Jahan a Edwards ve svém průzkumu [13] identifikovali 31 normalizačních metod a zkoumali jejich nedostatky. Uvedli, že drobné úpravy metod mohou mít velký vliv na výsledek rozhodnutí.

Ve své práci jsem implementoval nejčastěji používané normalizační metody a ty, které se nejvíce hodí k rozhodovacím metodám z praktické části. Například pro rozhodovací metodu TOPSIS jsem zvolil metodu vektorové normalizace. Ta byla ve studii [14] prokázána jako nejkonzistentnější pro metodu TOPSIS ve srovnání s ostatními běžně používanými metodami. Vybrané normalizační metody jsou popsány v Tabulce 6.

Normalizační metody pro výpočet využívají matici variant a typ kritéria. Pro maximalizační a minimalizační typy kritérií mají normalizační metody různé vzorce. Výsledkem výpočtu jsou normalizované hodnoty n_{ij} vlastností variant,

které náleží do uzavřeného intervalu $\langle 0, 1\rangle$. Zároveň je typ kritérií sjednocen na maximalizační typ.

Normalizační metoda	**Maximalizační kritérium**	**Minimalizační kritérium**
Max-min [12]	$n_{ij} = \frac{r_{ij} - \min r_j}{\max r_j - \min r_j}$	$n_{ij} = \frac{\max r_j - r_{ij}}{\max r_j - \min r_j}$
Max [12]	$n_{ij} = \frac{r_{ij}}{\max r_j}$	$n_{ij} = 1 - \frac{r_{ij}}{\max r_j}$
Vektorová normalizace [12]	$n_{ij} = \frac{r_{ij}}{\sqrt{\sum_{i=1}^{m} r_{ij}^2}}$	$n_{ij} = 1 - \frac{r_{ij}}{\sqrt{\sum_{i=1}^{m} r_{ij}^2}}$
Součtová [12]	$n_{ij} = \frac{r_{ij}}{\sum_{i=1}^{m} r_{ij}}$	$n_{ij} = \frac{1/r_{ij}}{\sum_{i=1}^{m} 1/r_{ij}}$
Logaritmická normalizace [12]	$n_{ij} = \frac{\ln r_{ij}}{\ln(\prod_{i=1}^{m} r_{ij})}$	$n_{ij} = \frac{1 - \frac{\ln r_{ij}}{1 - \ln(\prod_{i=1}^{m} r_{ij})}}{m-1}$
Lineární metoda [11]	$n_{ij} = \frac{r_{ij}}{\max r_j}$	$n_{ij} = \frac{\min r_j}{r_{ij}}$

Poznámka 1: Pro $i = 1, 2, \ldots, m$ a $j = 1, 2, \ldots, n$.
Poznámka 2: Zápisem $\max r_j$ ($\min r_j$) je myšlena maximální (minimální) hodnota sloupce j.

Tabulka 6: Normalizační metody

4 Metody vícekriteriální analýzy variant

Hlavní zdroje pro vytvoření této kapitoly jsou [1] a [3].

Rozhodovací metody slouží pro zpracování prvků modelu a vytvoření požadovaného cíle. Konkrétní využité prvky modelu jsou varianty, kritéria a jejich váhy. U některých rozhodovacích metod jsou využity i typy kritérií nebo je nutné použití dodatečných informací či omezení. Pro vytvoření cíle je použita matematika, práce s maticemi a vektory, určení vzdáleností v metrických prostorech, vytvoření hierarchické struktury, určení dominance variant a tak dále.

Definice 9 (Rozhodovací metoda)
Rozhodovací metoda využívá zpracovaných prvků modelu, na základě kterých určí požadovaný cíl modelu. Některé typy metod využívají i dodatečné informace pro postup při práci s informacemi.

Byla navržena spousta metod, jejich verzí a modifikací tak, aby vyhovovaly potřebám různých typů problémů. Například rozhodovací metoda ELECTRE má

minimálně sedm verzí, kde každá z verzí je v určitém smyslu vylepšením některé z předchozích. Některé pouze určují vzájemnou dominanci variant. Jiné dokáží i určit pořadí variant. Je tedy vhodné nastudovat klady a zápory různých metod a jejich modifikací a vybírat tak, aby rozhodovací metoda vhodně pracovala s konkrétním problémem.

Rozhodovací metody mají různé vlastnosti, podle kterých je lze dělit (např. podle způsobu určení cíle, podle počtu rozhodovatelů, podle způsobu strukturalizace problému, ...). Metody se mohou dělit podle typů informací, se kterými pracují na normální, stochastické (náhodné) nebo fuzzy.[9] Dalším způsobem dělení je například dělení podle toho, jestli rozhodovací metody vůbec využívají váhy kritérií při určování cíle. V mé práci se soustředím na typ metod využívající normální informace a váhy kritérií.

V následujících podkapitolách popisuji metody, které jsou implementovány v balíčku z praktické části. U každé metody uvedu zdroje použité pro vytvoření dané podkapitoly, stručnou historii vzniku, kroky výpočtu, typ výsledku (cíl) a nejčastější oblasti využití. U některých metod uvádím normalizaci v nultém kroku, protože přímo nesouvisí s výpočtem metody.

Nechť máme pro každou metodu zadaný model **M** vícekriteriální analýzy variant. Prvky modelu **M** jsou značeny následovně:

- $C = \{C_1, C_2, \ldots, C_n\}$ množina n kritérií. Váha kritéria C_j je značena w_j.
- $A = \{A_1, A_2, \ldots, A_m\}$ množina m variant a $A_{m\times n}$ označuje matici variant. Ohodnocení varianty A_i podle kritéria C_j se značí r_{ij} (stejně jako u matice variant).

4.1 Metoda váženého součtu

Metoda váženého součtu ([3]), anglicky „The Weighted Sum Model“ (WSM) nebo „Simple Additive Weighting“ (SAW), je jednou z nejjednodušších a nejpoužívanějších metod využívající model VAV. Metodu poprvé definovali Churchman a Ackoff roku 1945 pro problém výběru portfolia [16].

Cílem této metody je uspořádání variant na základě skóre, které je variantám přiřazeno. Postup pro vytvoření uspořádání je získán následujícím postupem.

0. Pro normalizaci matice variant $A_{m\times n}$ použijeme normalizační metodu Max-min.
1. Vypočítáme vektor $\vec{s} = (s_1, s_2, \ldots, s_m)$ určující hodnocení varianty následovně:

$$s_i = \sum_{j=1}^{n} n_{ij} w_j, \;\; i = 1, 2, \ldots m\,,$$

[9] Metody pracující s fuzzy informacemi jsou zpracovány například v knize [15].

kde s_i značí skóre varianty A_i podle metody WSM.

2. Nakonec podle vektoru skóre $\vec{s}$ určíme pořadí variant. Pokud máme za cíl maximalizovat skóre (tj. čím více, tím lépe), pak varianta s největším skóre bude první v pořadí, varianta s druhým největším skóre bude druhá, atd. Naopak pokud je cílem minimalizovat skóre (tj. čím méně, tím lépe), pak varianta s nejmenší hodnotou skóre bude v pořadí první, ...

Normalizační metodu Max-min, použitou při výpočtu, jsem vybral na základě studie [16]. Metoda váženého součtu se uplatňuje v oblasti vodního hospodářství, obchodu a finančního řízení [17].

4.2 Metoda váženého součinu

Metoda váženého součinu ([3]), anglicky „The Weighted Product Model“ (WPM), je metoda velmi podobná metodě váženého součtu. Metoda byla poprvé zmíněna v publikaci „Dimensional analysis“ autorem P. W. Bridgmanem roku 1922 a následně znovu v „Executive decisions and operations research“ autory D. W. Millerem a M. K. Starrem roku 1963.

Cílem této metody je sestavení uspořádání variant na základě skóre. Postup pro vytvoření uspořádání je získán následujícím postupem.[10]

0. Normalizujeme matici variant $A_{m\times n}$. Dostaneme tak matici $N_{m\times n}$.

1. Vypočítáme vektor $\vec{s} = (s_1, s_2, \ldots, s_m)$ určující hodnocení varianty následovně:

$$s_i = \prod_{j=1}^{n} (n_{ij})^{w_j}, \ \ i = 1, 2, \ldots m\,,$$

kde s_i značí skóre varianty A_i podle metody WPM.

2. Nakonec podle vektoru skóre $\vec{s}$ určíme pořadí variant. To provedeme stejným postupem jako u metody váženého součtu.

Metoda váženého součinu není v praxi moc využívaná. Z toho důvodu žádná studie zkoumající nejlepší normalizační metodu pro metodu váženého součinu nevznikla.

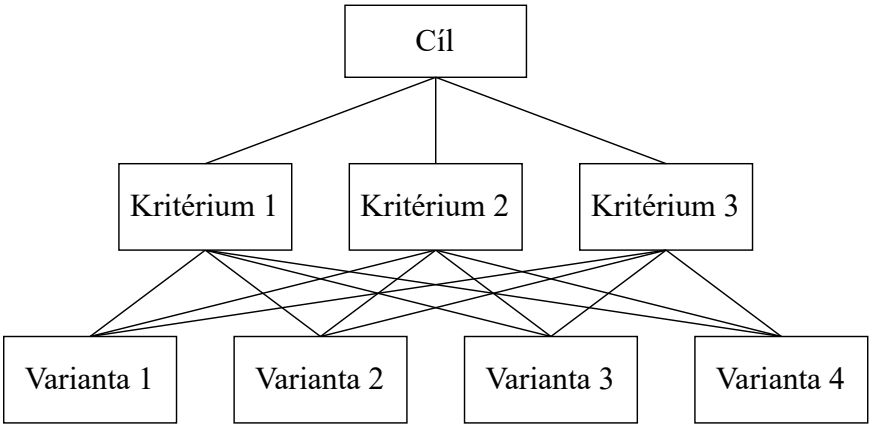
[10] Metoda váženého součinu má dva způsoby výpočtu uspořádání variant. Ve své práci použiji ten, co vypočítá skóre variant. Druhý způsob si mohou případní zájemci najít v [3].

4.3 Analytický hierarchický proces

AHP ([3], [2]) je proces vytvořený Thomasem L. Saatym. AHP umožňuje zahrnout do rozhodovacího procesu objektivní i subjektivní myšlenky rozhodovatele, a navíc je vhodný pro skupinovou práci [2].

Metoda vytváří hierarchickou strukturu, aby zjednodušila práci s daným problémem. Hierarchická struktura obsahuje několik úrovní. Obvykle jsou tyto úrovně 3. Na první úrovni je vždy cíl, následuje úroveň s kritérii a nakonec úroveň s variantami. Další možná úroveň je třeba úroveň s rozhodovateli, kterou bychom umístily mezi cíl a kritéria. Příklad hierarchické struktury je zobrazen na Obrázku 1. Vidíme, že daný problém má čtyři varianty, tři kritéria a požadovaný cíl analýzy.

AHP využívá párového porovnání pro vytvoření matic párových porovnání. Ty jsou následně zpracovány pomocí metody vlastního vektoru. Nakonec je vypočítáno skóre všech variant. Kroky AHP pro výpočet skóre jsou dány následujícím postupem.



Obrázek 1: Příklad hierarchické struktury AHP

1. Vytvoříme hierarchickou strukturu podle zadaného problému. První úroveň obsahuje cíl analýzy. Následuje úroveň kritérií a poslední úroveň bude obsahovat varianty.

2. Vytvoříme matici variant a váhový vektor

   Nejprve pro každé kritérium vytvoříme n matic párových porovnání, kde každá označuje párové porovnání všech m variant na základě daného kritéria. Následně na matice párových porovnání použijeme metodu vlastního vektoru. Získáme tak ohodnocení variant podle kritérií a z něj sestavíme matici variant.

   Pro získání váhového vektoru použijeme Metodu párových porovnání uvedenou v sekci 3.2.1.

3. Vypočítáme vektor $\vec{s} = (s_1, s_2, \ldots, s_m)$ skóre pomocí vzorce

$$s_i = \sum_{j=1}^{n} n_{ij} w_j, \ i = 1, 2, \ldots m\,,$$

kde s_i značí skóre varianty A_i podle metody AHP.

4. Nakonec podle vektoru skóre $\vec{s}$ určíme pořadí variant. To provedeme stejným postupem jako u metody váženého součtu.

WSM a AHP používají stejný vzorec pro výpočet skóre, ale AHP využívá relativní hodnoty namísto skutečných. Proces AHP se obvykle využívá ve firemní a veřejné politice, politické strategii a plánování [17].

4.4 TOPSIS

TOPSIS ([1], [3]) je akronymem pro „Technique for Order Preference by Similarity to Ideal Solution“.[11] Tato rozhodovací metoda byla vytvořena autory K. Yoonem a C. L. Hwangem roku 1981 v knize [3]. V dalších letech na ni pracovali a vyvíjeli ji.

Základní myšlenkou metody je, že varianty by měly mít nejmenší vzdálenost od ideální varianty $I \in A$ a největší vzdálenost od bazální varianty $B \in A$. Pro výpočet vzdáleností se využívá Euklidovská metrika. Metoda přiřazuje každé variantě číselné skóre, na základě kterého je sestaveno pořadí variant. Skóre lze vypočítat pomocí níže zmíněného postupu.

1. Zkonstruujeme normalizovanou matici variant $A_{m\times n}$ pomocí Vektorové normalizační metody. Dostaneme tak matici $N_{m\times n}$.

2. Vytvoříme váženou normalizovanou matici $Y_{m\times n}$ tak, že každý sloupec matice $N_{m\times n}$ vynásobíme vahou kritéria příslušící danému sloupci.

$$Y = \begin{pmatrix} w_1 n_{11} & w_2 n_{12} & \ldots & w_n n_{1n} \\ w_1 n_{21} & w_2 n_{22} & \ldots & w_n n_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_1 n_{m1} & w_2 n_{m2} & \ldots & w_n n_{mn} \end{pmatrix}$$

3. Určíme ideální variantu $I = (i_1, i_2, \ldots, i_n)$ a bazální variantu $B = (b_1, b_2, \ldots, b_n)$ podle následujících vzorců

$$i_j = \begin{cases} \max n_j & \text{pro } C_j \text{ maximalizační} \\ \min n_j & \text{jinak} \end{cases}$$

[11] Český překlad akronymu TOPSIS je „Technika pro preferenci pořadí podle podobnosti s ideálním řešením“.

$$b_j = \begin{cases} \min n_j & \text{pro } C_j \text{ maximalizační} \\ \max n_j & \text{jinak} \end{cases}$$

pro $j = 1, 2, \ldots, n$.

4. Určíme vzdálenosti variant od ideální I a bazální B varianty, kde d_i^+ značí vzdálenost varianty A_i od varianty I a d_i^- značí vzdálenost varianty A_i od varianty B. Vzdálenosti d^+ a d^- určíme pomocí vzorců (1) a (2) pro výpočet vzdálenosti v Euklidovské metrice.

$$d_i^+ = \sqrt{\sum_{j=1}^{n} (y_{ij} - i_j)^2} \tag{1}$$

$$d_i^- = \sqrt{\sum_{j=1}^{n} (y_{ij} - b_j)^2} \tag{2}$$

pro $i = 1, 2, \ldots, m$.

5. Vypočítáme relativní vzdálenost $R = (r_1, r_2, \ldots, r_n)$ k ideální variantě pomocí vzorce

$$r_i = \frac{d_i^-}{d_i^+ + d_i^-}, \ i = 1, 2, \ldots, m\,.$$

Relativní vzdálenost nám určuje skóre každé varianty.

6. Nakonec určíme uspořádání podle relativní vzdálenosti R. Pokud máme za cíl maximalizovat, pak varianta A_i s největším r_i je určena jako nejlepší, následuje varianta s druhou největší hodnotou R, atd. Pro minimalizaci je postup analogický.

Metodu Vektorové normalizace jsem zvolil z toho důvodu, že byla autory zadána v krocích výpočtu, a proto ji uvádím v prvním kroku. Dále byla ve studii [14] byla prokázána jako nejkonzistentnější ve srovnání s běžně používanými metodami. TOPSIS se využívá zejména v oblastech inženýrství, logistiky, výrobních systémů, obchodu a marketingu [17].

4.5 VIKOR

VIKOR ([2], [15]), celým názvem „VlseKriterijumska Optimizacija I Kompromisno Resenje“, je metoda využívající se pro práci s konfliktními kritérii, u kterých je zapotřebí kompromisu.[12] Využití principu kompromisu a popis jeho myšlenky byl popsán v článku [18]. Následně roku 1980 posloužily tyto myšlenky pro vytvoření základu metody VIKOR v článku [19].

[12] Název metody VIKOR se anglicky překládá „Multi-criteria optimization and compromises solution“ což v českém jazyce znamená „Vícekriteriální optimalizace a kompromisní řešení“.

Metoda má za cíl nalezení nejlepší varianty či skupiny variant. Stejně jako u metody TOPSIS mají mít preferované varianty nejmenší vzdálenost od ideální varianty $I \in A$ a největší vzdálenost od bazální varianty $B \in A$. Postup metody VIKOR pro výpočet nejlepší varianty nebo skupiny variant je dán následujícím postupem.

0. Normalizujeme matici variant $A_{m\times n}$. Dostaneme tak matici $N_{m\times n}$.

1. Určíme ideální variantu $I = (i_1, i_2, \ldots, i_n)$ a bazální variantu $B = (b_1, b_2, \ldots, b_n)$ podle následujících vzorců

$$i_j = \begin{cases} \max n_j & \text{pro } C_j \text{ maximalizační} \\ \min n_j & \text{jinak} \end{cases}$$

$$b_j = \begin{cases} \min n_j & \text{pro } C_j \text{ maximalizační} \\ \max n_j & \text{jinak} \end{cases}$$

   pro $j = 1, 2, \ldots, n$.

2. Výpočet n-tice $S = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ nazvané „utility“ a n-tice $R = (\rho_1, \rho_2, \ldots, \rho_n)$ nazvané „regret“ pomocí vzorců

$$\sigma_i = \sum_{j=1}^{n} w_j \left[\frac{i_j - n_{ij}}{i_j - b_j} \right],$$

$$\rho_i = \max \left(w_j \left[\frac{i_j - n_{ij}}{i_j - b_j} \right] \right),$$

   pro $i = 1, 2, \ldots, m$.

3. Vypočítáme n-tici $Q = (q_1, q_2, \ldots, q_n)$ pomocí vzorce

$$q_i = v \left[\frac{\sigma_i - \min \sigma_i}{\max \sigma_i - \min \sigma_i} \right] + (1 - v) \left[\frac{\rho_i - \min \rho_i}{\max \rho_i - \min \rho_i} \right],$$

   pro $i = 1, 2, \ldots, m$. Kde v je hodnota z uzavřeného intervalu $\langle 0, 1 \rangle$ nazývána maximální užitek skupiny („maximum group utility“) a $(1-v)$ je nazývána individuální lítost („individual regret“). Za v se obvykle volí hodnota 0,5.

   Nyní budeme x. nejmenší hodnotu n-tice Q označovat Q^x.

4. Variantu A_α, které patří hodnota Q^1, nazveme jako nejlepší, pokud splňuje následující podmínky (a) a (b).

(a) **Podmínka 1**: Přijatelná výhoda („Acceptable advantage“)
Hodnoty Q^1 a Q^2 musí splňovat následující nerovnici

$$Q^2 - Q^1 \leq \gamma\,,$$

kde $\gamma = \frac{1}{m-1}$.

(b) **Podmínka 2**: Přijatelná stabilita („Acceptable stability“)
Variantě A_α musí patřit nejvyšší hodnoty v n-ticích S a R.

5. Pokud A_α splňuje obě podmínky, potom je zvolena jako výsledek.

   Pokud A_α splňuje pouze první podmínku, pak jsou za výsledek zvoleny varianty, kterým přísluší hodnoty Q^1 a Q^2.

   Pokud A_α nesplňuje žádnou z uvedených podmínek, pak jsou za výsledek zvoleny varianty, kterým přísluší hodnoty $Q^1, Q^2, \ldots, Q^\mu$. Kde Q^μ je nejmenší hodnota, pro kterou ještě platí podmínka

$$Q^\mu - Q^1 \leq \gamma\,.$$

Rozhodovateli je tedy vrácena nejlepší varianta nebo množina nejlepších variant. Metoda VIKOR se uplatňuje především v inženýrství [2].

4.6 ELECTRE

ELECTRE ([3], [2]) je akronymem pro „ÉLimination Et Choix Traduisant la REalité“.[13] Metoda byla poprvé představena B. Royem roku 1966 v [20]. V dalších letech postupoval vývoj a vznikaly nové verze této metody.

ELECTRE je založena na určení takzvaných vztazích převahy („outranking relations“), které se zjišťují párovým porovnáváním všech variací. Vztah převahy dvou variant A_i a A_j, označován $A_i \to A_j$, popisuje, že i když i. varianta kvantitativně nepřevažuje nad j. variantou, může rozhodovatel stále riskovat a považovat A_i za téměř jistě lepší než A_j [3].

První verze ELECTRE vrací za výsledek binární matici $O_{m\times m}$ vztahů převahy. Z matice $O_{m\times m}$ není vždy možné sestavit pořadí nebo určit nejlepší variantu, přirozeně proto vznikly verze, které toho byly schopny. Výpočet matice $O_{m\times m}$ vztahů převahy je dán následujícím postupem.

1. Normalizujeme matici variant $A_{m\times n}$ pomocí Vektorové normalizační metody. Dostaneme tak matici $N_{m\times n}$.

[13] Akronym ELECTRE lze přeložit do angličtiny jako „Elimination and Choice Translating Reality“ a do češtiny jako „Eliminace a volba překladání reality“.

2. Vytvoříme váženou normalizovanou matici $Y_{m\times n}$ tak, že každý sloupec matice $N_{m\times n}$ vynásobíme vahou kritéria příslušící danému sloupci.

$$Y = \begin{pmatrix} w_1 n_{11} & w_2 n_{12} & \dots & w_n n_{1n} \\ w_1 n_{21} & w_2 n_{22} & \dots & w_n n_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_1 n_{m1} & w_2 n_{m2} & \dots & w_n n_{mn} \end{pmatrix}$$

3. Sestrojíme matice množin $C_{m\times m}$ a $D_{m\times m}$ s pomocí párového porovnání všech variant dle následujících postupů

$$c_{kl} = \{j \mid y_{kj} \geq y_{lj}\},\ j = 1, 2, \dots, n\,, \tag{3}$$

$$d_{kl} = \{j \mid y_{kj} < y_{lj}\},\ j = 1, 2, \dots, n\,, \tag{4}$$

pro $k, l = 1, 2, \dots, m$.[14] Předchozí rovnice jsou uvažovány pro maximalizační typy kritérií. Pokud by kritérium C_j bylo minimalizačního typu, pak se musí změnit znaménka v rovnici (3) na $<$ a v rovnici (4) na $\geq$.

4. Z předchozích matic sestrojíme matice $X_{m\times m}$ a $\Delta_{m\times m}$. Matici $X_{m\times m}$ sestrojíme vzorcem

$$\chi_{kl} = \sum_{j\in c_{kl}} w_j\,,$$

pro $k, l = 1, 2, \dots, m$ a matici $\Delta_{m\times m}$ vzorcem

$$\delta_{kl} = \frac{\max(y_{kj} - y_{lj})_{j\in d_{kl}}}{\max(y_{kj} - y_{lj})_{\forall j}}\,, \tag{5}$$

pro $k, l = 1, 2, \dots, m$. V rovnici (5) je ve jmenovateli bráno maximum z rozdílů po prvcích dvou řádků k a l (dvou variant A_k a A_l). V čitateli chceme maximum z rozdílů, u kterých je j v množině d_{kl}.

5. Zajistíme prahové hodnoty $\hat{c}$ a $\hat{d}$. Buď tyto hodnoty zvolí rozhodovatel nebo je lze vypočítat jako průměr z hodnot matic X a Δ pomocí

$$\hat{c} = \frac{1}{m(m-1)} \sum_{(k=1\wedge k\neq l)}^{m} \sum_{(l=1\wedge l\neq k)}^{m} \chi_{kl}\,,$$

$$\hat{d} = \frac{1}{m(m-1)} \sum_{(k=1\wedge k\neq l)}^{m} \sum_{(l=1\wedge l\neq k)}^{m} \delta_{kl}\,.$$

[14] V literatuře se matice množin $C_{m\times m}$ a $D_{m\times m}$ nazývají „corcondance sets“ a „discordance sets“.

6. Určíme matice $F_{m\times m}$ a $G_{m\times m}$ následovně

$$f_{kl} = \begin{cases} 1 & \text{pokud } \chi_{kl} \geq \hat{c} \\ 0 & \text{pokud } \chi_{kl} < \hat{c} \end{cases},$$

$$g_{kl} = \begin{cases} 1 & \text{pokud } \delta_{kl} \geq \hat{d} \\ 0 & \text{pokud } \delta_{kl} < \hat{d} \end{cases}.$$

7. Vytvoříme binární matici $O_{m\times m}$ vztahů převahy z vzorce

$$o_{kl} = f_{kl} * g_{kl}\,.$$

   Z matice $O_{m\times m}$ lze vyčíst následující. Pokud $o_{kl} = 1$ pak varianta A_k je preferovaná před variantou A_l. Pokud má některý sloupec alespoň jeden prvek roven 1, pak je tento sloupec takzvaně „ELECTREally“ dominován příslušným řádkem (variantou) [3].

ELECTRE a její verze jsou převážně využity pro energetické, ekonomické, environmentální, vodohospodářské a dopravní problémy [17].

5 Praktická část

V této kapitole popisuji praktickou část této bakalářské práce, ve které jsem měl za cíl naprogramovat nástroj pro pomoc při práci s problémy vícekriteriální analýzy variant. Je mnoho způsobů a forem jak takový nástroj zpracovat. Ať už ve formě webové, desktopové či konzolové aplikace pro rozhodování.[15] nebo vytvoření knihovny, která s tímto typem problému umí pracovat. Rozhodl jsem se tento nástroj pojmout ve formě knihovny (balíčku), která by mohla být typu „open-source“.[16] Nejprve představím použité technologie a nástroje a následně uvedu popis celého balíčku.

5.1 Použité technologie a nástroje

Začnu s popisem použitých technologií a nástrojů, aby byly jasné důvody, proč jsem při vytváření balíčku postupoval tak, či onak. Uvedu jaký programovací jazyk jsem zvolil a následně ho popíšu. Zmíním mým kódem využité balíčky, i ty, které mi pomohly při jeho tvorbě. Nakonec krátce popíšu aplikace, které mi při vytváření celé práce pomohly.

5.1.1 Programovací jazyk Python

Na úplném začátku jsem musel popřemýšlet jaký programovací jazyk by se na vytvoření požadovaného nástroje nejvíce hodil. Po nastudování a zvážení všech informací o možný jazycích jsem zvolil programovací jazyk Python z několika důvodů. Tím hlavním byl ten, že pro jazyk Python existuje obrovské množství balíčků, které se pro tento nástroj hodí. Výhodou bylo i to, že jazyk Python jsem již znal a uměl jsem s ním pracovat. Další důvody jasně vyplývají z výhod a popisu jazyka.

Programovací jazyk Python je vysokoúrovňový multiparadigmatický programovací jazyk vytvořil G. V. Rossum v roce 1991. Python je jeden z nejpopulárnějších programovacích jazyků současnosti a to hlavně z důvodu jeho velmi jednoduché syntaxe a obrovského množství uživateli vytvořených balíčků.[17] Python používá dynamické typování, obsahuje garbage collector (automatická správa paměti) a REPL.[18]

Python je vyvíjen jako open-source software pod licencí OSI, což znamená že jeho zdrojový kód je otevřený a volně použitelný pro osobní a komerční účely.[19] Je dostupný na všech běžných operačních systémech Windows, macOS a Linux.

[15] Software, který napomáhá skupině či jednotlivcům učinit složitá rozhodnutí se nazývá „Decision-making software“.

[16] Open-source je označení pro otevřený software, což je software s otevřeným zdrojovým kódem, který je volně šiřitelný.

[17] Informaci o popularitě jazyka Python jsem odvodil na základě statistik ze stánky Github a dotazníků, které byly podané uživatelům stránky Stack Overflow.

[18] REPL je zkratka pro „read-eval-print-loop“. REPL je interaktivní programovací prostředí programovacího jazyka. Více informací lze dočíst například na wikipedii.

[19] Pro více informací k licenci OSI doporučuji navštívit oficiální stránky OSI.

V poslední hlavní verzi jazyka 3.11, byla přidána nová funkcionalita a tato verze nabízí výrazné zrychlení oproti předchozím verzím.

Soubory obsahující zdrojový kód (definice a příkazy) jazyka Python se nazývají moduly (zakončeny koncovkou „.py“). Ty lze importovat do jiných modulů. Python obsahuje i vestavěnou knihovnu standardních modulů, které umožňují například práci se souborovým systémem, práci pro síťovou a meziprocesorovou komunikaci nebo moduly pro kompresi a archivaci dat.[20]

Moduly se mohou shlukovat do balíčku či knihoven. Pojmy knihovna a balíček se často zaměňují, protože neexistují přesně dané definice. Obvykle se pod pojmem Python balíček myslí kolekce modulů, které poskytují určitou funkcionalitu. U pojmu Python knihovna se obvykle myslí větší balíček obsahující spoustu dalších balíčků a modulů.

Oficiálním repozitářem balíčků pro jazyk Python je stránka PyPI („Python Package Index“), která obsahuje přes 400 tisíc projektů. Přídavné balíčky lze instalovat pomocí správce balíčků pip. Balíčky se instalují do globálního prostředí, kde jsou potom dostupné pro všechny moduly, které globální prostředí využívají. Pro vytváření projektů se však vždy hodí vytvořit izolované Python prostředí, pro lepší organizaci a správu balíčků, které projekt využívá. To lze pomocí nástroje virtualenv, který jsem ve své práci využil.

Pro vylepšení jazyka Python vznikl rejstřík zvaný PEP obsahující spoustu návrhů pro vylepšení funkcionality stávajícího jazyka, psaní kódu, a podobně. Rád bych zmínil návrh PEP 8 a PEP 20. PEP 8 je stylová příručka pro kód jazyka Python a zavádí pravidla, jakým stylem tento kód psát a stylizovat. PEP 20 zase obsahuje filozofii, ve formě krátkého rčení, kterou je doporučené se řídit při vytváření, psaní a návrhu kódu. Dodržování návrhů PEP 8 a PEP 20 napomáhá přehlednosti a čitelnosti kódu a dokonce umožňuje předejít vzniku chyb.[21] Ve svém balíčku jsem několik z těchto návrhů dodržoval, abych zajistil větší kvalitu kódu.

5.1.2 Využité balíčky

Jak už jsem zmínil, balíčků pro jazyk Python je spousta, proto jsem měl širokou škálu možností. Především jsem se soustředil na ty, které mi zjednoduší práci při psaní kódu a ty, které umí efektivně pracovat s maticemi nebo strukturalizovat data.

Balíčky dělím na dva typy. První typ balíčků, který označuji „vývojové balíčky“. Ty mi pomohly při udržení přehlednosti kódu a přípravě balíčku na exportování. Druhým typem jsou balíčky přímo využité mým kódem umožňující pracovat s lineární algebrou, statistikou či celkově s daty.

Za vývojové balíčky jsem zvolil:

[20] Knihovna standardních modulů jazyka Python lze najít v oficiální dokumentaci na webových stránkách python.org.

[21] Vývojáři jazyka Python si na čitelnosti kódu zakládají, proto vznikl pojem „pythonovský kód“ (z anglického „pythonic code“) označující přehledný kód dodržující stylistická pravidla PEP.

- **pycodestyle** (dříve nazvaný pep8) je jednoduchý konzolový nástroj umožňující kontrolu Python kódu podle pravidel PEP8. Obsahuje i nastavení pro případné změny nebo vynechání určitých pravidel PEP8.
- **Black** je nástroj umožňující formátovat kód Python podle návrhu PEP8. Nástroj Black má pomoci programátorovi odprostit se od potřeby stylizovat kód, aby se mohl více soustředit na psaní kvalitního kódu, a zároveň zachovat určitou stylistiku a kvalitu.
- **pdoc** umožňuje automaticky vygenerovat dokumentaci na základě komentářů kódu. Dokumentace je vytvořena jako webová stránka, která poskytuje informace o všech poskytnutých modulech, funkcích a třídách.
- **Setuptools** je knihovna pro usnadnění vytváření distribučních balíčku pro export do online repozitářů.[22] Umožňuje vytvořit soubor s informacemi, které jsou následně využity repozitáři nebo při instalaci balíčku koncovým uživatelem.

Balíčky využívané mým kódem jsou:

- **NumPy** je jednou z nejpopulárnějších Python knihoven pro datovou vědu. Používá se pro práci s numerickými daty a umožňuje provádět různé matematické operace s poli. Obsahuje části pro operace lineární algebry, použití Fourierovy transformace, statistické operace, pro náhodné simulace a další.

  NumPy poskytuje vývojáři maticové datové struktury ve formě tříd. Ty na rozdíl od základních polí z jazyka Python mají pevnou velikost a obsahují pouze jeden typ prvků. To zajistí lepší, rychlejší a bezpečnější práci s velkými daty. NumPy kód je také více čitelný, protože poskytuje základní operace pro práci s maticemi jako součet hodnot či násobení prvků matice, tudíž není potřeba používat standardní cykly nebo tvořit vlastní metody.

  NumPy tvoří jádro ekosystému knihoven zaměřených na datovou vědu. NumPy API se používá v knihovnách pro kvantové výpočty, statistické výpočty, zpracování signálů, matematickou analýzu, astronomické procesy a pro mnoho dalších knihoven.[23]
- **Pandas** je další z velmi oblíbených a hojně využívaných knihoven. Využívá se pro analýzu a manipulaci s velkými daty. Obsahuje i nástroje pro čtení a zápis dat s různými formáty.

  Stejně jako NumPy poskytuje vývojáři třídy. Hlavní třídy balíčku Pandas jsou DataFrame a Series, které indexují pole a maticové datové struktury. Indexy jsou následně využity pro agregace, transformace a manipulace s daty.

[22] Distribuční balíček je typ balíčku, který obsahuje soubor s informacemi jako je verze, autor, licence a podobně a je používán pro šíření na internetu. Koncový uživatel si následně balíček může stáhnout a nainstalovat.

[23] Pojem API je zkratkou pro „Application Programming Interface“. Označuje vrstvu softwarového rozhraní poskytující data, služby a funkce jiným programátorům.

Ve své práci používám i moduly z vestavěné knihovny standardních modulů Python. Těmito moduly jsou: **json** pro práci se soubory typu JSON; **typing** k statickému typování parametrů a návratových hodnot; funkce modulu **math** a funkce pro zpracování cest z modulu **pathlib**.

5.1.3 Nástroje

Při práci jsem využil několik nástrojů, které bych rád stručně popsal. Tyto nástroje mi pomohly při vypracování praktické a teoretické části, zaznamenání potřebných poznámek a zálohování mé práce.

Visual Studio Code je textový editor zdrojového kódu pro operační systémy macOS, Windows a Linux, který je vyvíjený firmou Microsoft. Obsahuje spoustu nástrojů a technologií, které napomáhají při tvorbě projektů a při samotném programování. Rád bych vyzdvihnul zvýraznění syntaxe nazvané IntelliSense, automatické doplňování kódu, integrovanou podporu GIT a možnost přidat spousty rozšíření. Editor jsem využil jak na vytvoření praktické části, tak i na napsání písemné části.

Obsidian je textový editor pro vytváření poznámek využívající značkovací jazyk Markdown. Použití jazyka Markdown je velmi snadné, proto se při psaní lze více zaměřit na text. Obsidian umožňuje propojit poznámky pomocí takzvaných odkazů a tím organizovat související témata. Aplikaci jsem využil pro vytvoření seznamu úkolů a psaní poznámek z knih a článků.

Github je služba určená pro vytváření privátních i veřejných projektů a zálohování jeho verzí. Služba díky systému Git dokáže efektivně pracovat s verzemi projektu. Dále nabízí spoustu nástrojů pro správu projektu a pomoci při jeho vývoji. Github jsem využil pro zálohování mé práce.

5.2 Popis balíčku

V této podkapitole podrobně popíšu vytvořený balíček. Nejprve uvedu strukturu celého balíčku a následně uvedu konkrétní implementaci a API. Nakonec popíšu možnosti práce s problémy VAV za pomoci balíčku. Pro balíček jsem zvolil název „mymcdm“.[24]

5.2.1 Struktura

Strukturu balíčku jsem udělal co nejlogičtější a nejsrozumitelnější, aby bylo snadné případné rozšíření. Pro přehlednější popis, strukturu rozdělím na tři části. První část obsahuje modul pro instalaci balíčku a soubor s popisem o něm. Část druhá obsahuje pomocné metody pro postup při určování rozhodnutí, dále konzolovou část aplikace a ostatní logiku a funkcionalitu balíku. V třetí části jsou implementované rozhodovací metody, normalizační metody a metody určení vah.

[24]Název „mymcdm“ vyšel ze spojení anglických slov „my“ a „mcdm“ (v překladu „moje vícekriteriální rozhodování“).

První část obsahuje soubory *README.md* a *setup.py*. Soubor *README.md* popisuje balíček a to ve formátu Markdown. Modul *setup.py* (využívá balíčku Setuptools) obsahuje instalační instrukce a slouží pro nainstalování balíčku.

Druhá část balíčku obsahuje hlavní logiku a funkcionalitu celého balíčku. Modul *main.py* obsahuje pomocnou metodu, která je užitečná při procesu určování požadovaného cíle. Funkce pro čtení a zápis souborů jsem vložil do modulu *inout.py*. Celá konzolová část balíčku je obsažena v modulu *cli.py*. V této části jsou i obsaženy pomocné moduly ve složce *utils*, které obsahují kontrolu vstupních dat, funkce pro „orámování“ vah a variant, vlastní slovníkové typy a pomocné metody pro výpočet.[25]

Třetí část obsahuje všechny metody popsané v teoretické části. Balíček obsahuje tři složky („podbalíčky“) a to složku *methods* obsahující rozhodovací metody, složku *normalization* obsahující normalizační metody a složku *weighting* obsahující metody určení vah. Metody v příslušných složkách jsou dělené do modulů, kde každý modul obsahuje funkci obsahující výpočet dané metody. Balíček nyní obsahuje šest normalizačních metod, šest rozhodovacích metod a sedm metod pro vážení. Všechny tyto metody jsou popsány v teoretické části.

Přidání dalších metod je snadné a intuitivní. Stačí přidat modul s novou metodou do příslušné složky a importovat funkci z modulu do souboru *___init___.py*, který každá složka obsahuje. Funkce se následně zpřístupní pro importování. Takže například pro importování Vektorové normalizační metody zadáme následující kód.

```
1   from mymcdm.normalization import vector
```

Zdrojový kód 1: Ukázka importování normalizační metody

U normalizačních a rozhodovacích metod je vloženo ověření vstupních argumentů. Kontroluje se většinou správnost rozměrů matic a vektorů. Například probíhá kontrola, jestli počet sloupců matice odpovídá velikosti váhového vektoru. U rozhodovacích metod je kontrolováno, aby součet vah byl roven 1. Ověřuji jen ta nejdůležitější pravidla. Snažil jsem se ověření nedělat příliš striktní, aby to případné uživatele neodradilo od použití.

5.2.2 API

Balíček by měl být přívětivý a snadný na použití ostatními programátory. Snažil jsem se udělat práci s implementovanými funkcemi jednoduchou a přívětivou bez vynechání základních principů práce s modelem VAV. Proto všechny funkce pracují buď s maticovými objekty z balíčku Numpy nebo s objekty DataFrame a Series z balíčku Pandas. Dále některé novodobé nástroje na editaci zdrojového kódu mají technologie, které zobrazují dokumentaci balíčku nebo napoví-

[25] Orámováním nazývám proces, při kterém vytvořím Pandas objekt třídy DataFrame nebo Series, kterému přiřadím označení řádků či sloupců matic nebo vektorů.

dají podle daného typu. Snažil jsem se těchto technologií využít pro ještě snazší práci s mým balíčkem.

Rozhodovací metody požadují, aby matice variant byl objekt třídy DataFrame z balíčku Pandas. K tomu jsem vytvořil metodu zvanou *frame_alternatives* (z podbalíků *utils*), která jako vstup vezme matici variant (volitelně i názvy variant) a vrátí DataFrame. Tento proces nazývám orámování matice variant. Ostatní metody z modulu *framing.py* slouží pro orámování vah nebo vytvoření rozhodovací matice.

Výsledky implementovaných metod, až na ELECTRE, vrací objekt třídy Series. Series je opět třída z balíčku Pandas, která označuje orámovaný vektor. Výsledek je tedy sloupcový vektor označující skóre variant. Metoda ELECTRE jako jediná vrací binární matici vztahů převahy, která je navrácena jako objekt třídy DataFrame. S objekty třídy Series a DataFrame lze dále efektivně pracovat. Například lze snadno vybrat hodnoty na základě podmínky nebo různě sjednocovat ostatní objekty DataFrame či Series.

Pomocná metoda *decision* vrací objekt třídy *Result* z modulu *utils/types.py*, který využívá modul *TypedDict* z knihovny standardních modulů Python. Modul *TypedDict* nabízí možnost slovníku přidat typy a tím zpřehlednit práci se slovníkovým typem dat. Třída *Result* je definována Zdrojovým kódem 2.[26] Jak lze v kódu vidět, třída obsahuje všechny informace a data, která jsou během práce s problémem použity.

```
1  class Result(TypedDict):
2      decision: DataFrame
3      alternatives: DataFrame
4      weights: Series
5      criteria_type: NDArray
6      n_method: str | None
7      s_method: str
8      path: Path | None
```

Zdrojový kód 2: Ukázka slovníkového typu Result

Pro lepší práci s daty jsem do balíčku implementoval modul obsahující funkce pro čtení a zápis dat. Funkce *load_data* umožňuje podle typu načíst a zpracovat požadovaná vstupní data. Pokud se v souboru najde klíč „format“, který se rovná hodnotě „matrix“ proběhne načtení variant, vah a typů kritérií. Pokud je hodnota klíče „format“ rovna „pairwise“, proběhne načtení matic porovnání a jejich zpracování pomocí metody vlastního vektoru. Uživatel tak může načítat požadovaná data a na jejich základě vytvořit rozhodnutí.

Výsledek analýzy může uložit pomocí funkce *save_result* (nebo v decision metodě při zadání argumentu „save“ a „folder“), která přijímá objekt třídy *Result* a ten uloží. Poté je možno i výsledek analýzy načíst opět pomocí funkce

[26] Ve Zdrojovém kódu 2 jsem záměrně vynechal dokumentaci třídy z důvodu zabrání velkého prostoru textu.

load_data, pokud zadáme cestu k uloženému výsledku. Tyto metody používám i pro konzolovou část balíčku.

Načítat a ukládat lze pouze soubory typu JSON.[27] Formát JSON však nedovoluje zadat čísla ve formátu zlomku. Kdyby se čísla musela zadávat přesně, docházelo k nepřesnostem, a tomu je nutné předejít. Implementoval jsem proto funkci *replace_fractions*, která dokáže převést matici textových řetězců reprezentující zlomky na desetinná čísla, která jsou přesná na šestnáct desetinných míst.

Parametrům funkcí jsem vložil typové označení a „docstring“. Typové označení odpovídá požadovaným hodnotám, které je nutné zadat pro správné fungování funkce. Neslouží pro kontrolu vstupů. Docstring jsou speciální komentáře slouží pro dokumentaci funkcí a modulů. U funkcí jsem většinou uváděl základní popis, informace o parametrech a výstupech. Označení a docstring jsou využity editory zdrojového kódu pro nápovědu při psaní. Na základě docstring jsem vygeneroval podrobnou dokumentaci pomocí balíčku pdoc, která lze najít v elektronických datech této práce.

5.2.3 Práce s problémy VAV

Příklad zdrojového kódu při použití balíčku „mymcdm“ a využití jeho konzolové části lze najít v příloze B.

Nástroj lze použít jak ve formě Python balíčku pomocí API, tak i využitím konzole. Užití v konzoli je spíše zaměřeno na menší problémy, zatímco využití API balíčku je zaměřeno na všechny typy problémů.

Konzolová část balíčku umožňuje uživateli načíst matici variant a váhy kritérií, normalizovat varianty a nakonec určit cíl modelu. Uživatel může v příkazové řádce projít celým procesem práce s modelem VAV, až na možnost určení vah kritérií. Ty musí být zadány předem. V konzoli stačí použití příkaz „mymcdm“, který vypíše nápovědu pro použití konzolové části. Tato část je spíše zaměřena na práci s menšími problémy, protože nenabízí tak velkou kontrolu při práci s daty. Data se dají jen uložit, vypsat a načíst.

Balíček poskytuje dva způsoby práce s problémy VAV. Prvním způsob je vytvoření postupu z implementovaných metod. Uživatel si všechny požadované metody importuje do svého modulu a následně si může celý proces naprogramovat sám. Rozhodovací proces může být přizpůsoben různým potřebám a uživatel má větší kontrolu nad výsledky funkcí.

Druhý způsob je jednodušší pro uživatele s menší znalostí teorie VAV. Uživatel může využít pomocné metody nazvané *decision*, kterou lze najít v modulu „main.py“. Metoda má několik povinných a volitelných parametrů. Povinné parametry jsou: matice variant, váhový vektor, kódové označení normalizační a rozhodovací metody. Volitelnými parametry jsou: vektor typů kritérií, místo

[27] JSON vychází ze zkratky „JavaScript Object Notation“. JSON je datový formát využívaný v IT pro přenos dat. JSON byl odvozen z programovacího jazyka JavaScript a jeho přípona je *json*.

uložení a příznak, zda se má výsledek a informace o procesu uložit. Jediné, co této metodě chybí, je možnost určení vah. Ten jsem se rozhodl v metodě *decision* neobsáhnout a požaduji, aby byly váhy kritérií předem určeny.

Závěr

V teoretické části této práce je podrobně popsán model vícekriteriální analýzy variant (2) společně s postupem určování rozhodnutí. V dalších kapitolách jsou popsána kritéria a metody pro určení vah (3.2), normalizace a normalizační metody (3.3) a následně metody pro určení cíle rozhodnutí (4).

Výstupem praktické části je balíček vytvořen v programovacím jazyce Python. Ten byl zpracován na základě poznatků a metod z teoretické části. Jsou v něm implementovány všechny zmíněné metody pro normalizaci, určení vah a rozhodutí. Navíc zde byla přidána spousta nástrojů pro usnadnění práce s problémy vícekriteriální analýzy variant, jakými jsou funkce pro načítání a ukládaní dat nebo objekty pro jejich strukturalizaci. Podrobný popis balíčku je uveden v praktické části (5) textu práce.

Práce se dá zlepšit a rozšířit mnoha způsoby. Je možné přidat další rozhodovací metody. Například takové, které pracující s fuzzy daty nebo se rozhodují bez vah kritérií. Pokud by byl balíček „mymcdm“ zveřejněn jako open-source, bylo by vhodné zlepšit dokumentaci a vytvořit testování funkcí. Konzolová část balíčku by se dala rozšířit, a nebo na jejím základě vytvořit rozhodovací software.

Conclusions

In the theoretical part of this thesis, the Multiple Attribute Decision Making model (2) is described in detail together with the decision determination procedure. The following chapters describe the criteria and methods for determining weights (3.2), normalization and normalization methods (3.3), and then methods for determining the decision objective (4).

The output of the practical part is a package created in the Python programming language. It was developed based on the knowledge and methods from the theoretical part. It implements all the mentioned methods for normalization, determination of weights and decision. In addition, many tools have been added to help working with Multiple Attribute Decision Making problems, such as functions for loading and storing data or objects for structuring them. A detailed description of the package is given in the practical section (5) of the thesis.

The thesis can be improved and extended in many ways. Additional decision methods can be added. For example, ones that work with fuzzy data or make decisions without criteria weights. If the "mymcdm" package were released as open-source, it would be appropriate to improve the documentation and create testing functions. The console part of the package could be extended, or decision software could be developed based on it.

A Ukázka práce s modelem VAV

V této příloze ukazuji práci s modelem VAV na hypotetickém příkladu: „Problém výběru mobilního telefonu v cenovém rozmezí 8 – 12 tisíc korun.“. Varianty a informace o nich jsou čerpány z internetu. Níže je popsán postup podle kroků popsaných v kapitole 2.2.

Krok 1: Vytvoření struktury modelu VAV
Mám problém výběru mobilního telefonu a chci seřadit telefony podle mnou zvolených kritérií. Cílem může být sestavení žebříčku variant podle množiny variant. Problém má následující vlastnosti: jeho aplikace bude jednorázová; varianty musí být v cenovém rozmezí 8 – 12 tisíc Kč; při určování dat a rozhodování by mělo být možno zahrnout subjektivní názor.

Krok 2: Vytvoření kritérií
Zvolím mnou požadovaná kritéria podle kterých budu zařízení hodnotit. V seznamu lze vidět kritéria, která jsem vybral společně s jejich označením. Ostatní vlastnosti telefonu mě nezajímají, a tudíž je nebudu brát jako hledisko hodnocení.

- Úhlopříčka displeje (C_1) – Nechci velké zařízení, čím menší tím lepší. Kritérium C_1 je tedy typu minimalizačního.
- Velikost paměti RAM (C_2) – Chci, aby mělo zařízení velkou operační paměť. Typ kritéria C_2 je maximalizační.
- Hodnocení procesoru (C_3) – Rychlost je pro mě také důležitá. Procesor nelze vybírat pouze na základě frekvence, proto jsem našel stránku nanoreview.net, která obsahuje bodové hodnocení procesoru. Typ kritéria C_3 je také maximalizační.
- Cena (C_4) – Je i vhodné zvolit cenu, protože se zařízení cenově dosti liší. Kritérium C_4 je typu minimalizačního.

Krok 3: Zvolení a ohodnocení variant
Nyní vyberu mobilní telefony, které budou reprezentovat varianty modelu. Mobilní telefony vybírám z internetového obchodu alza.cz, kde jsem vybral prvních pět ze seznamu „nejprodávanější“ v kategorii „mobily“. Snažil jsem se zamezit duplicitnímu výběru modelu. Do výběru se dostaly zařízení z Tabulky 7.

Nyní určím matici variant podle informací získaných z internetového obchodu. Matice variant je dána následující Tabulkou 8, pro přehlednost jsem přidal označení sloupců.

Krok 4: Zpracování informací
Nyní zpracuji všechna získaná data.

- Jelikož varianty žádné nečíselné hodnoty nemají, nebudu převádět žádné hodnoty.

Označení varianty	**Název telefonu**
A_1	Samsung Galaxy A34 5G 6GB/128GB
A_2	Samsung Galaxy A54 5G 8GB/128GB
A_3	Xiaomi Redmi Note 12 Pro+ 5G 8GB/256GB
A_4	Google Pixel 6a 5G 6GB/128GB
A_5	Samsung Galaxy S20 FE 5G 128GB

Tabulka 7: Vybrané varianty z příkladu

	C_1	C_2	C_3	C_4
A_1	6,6	6	56	9499
A_2	6,4	8	72	11999
A_3	6,67	8	56	11999
A_4	6,1	6	78	9990
A_5	6,5	6	68	11999

Tabulka 8: Matice variant z příkladu

- Pro normalizaci variant jsem zvolil metodu Max-min. Normalizuji matici variant pomocí vybrané metody. Například prvek $n_{11} = 0{,}1228$, protože musím zvolit vzorec pro minimalizační kritérium u metody Max-min, kde $\max r_1 = 6{,}67$ (maximum z 1. sloupce) a $\min r_1 = 6{,}1$. Po dosazení do vzorce mám

$$n_{ij} = \frac{6{,}67 - 6{,}6}{6{,}67 - 6{,}1}.$$

  Výsledek normalizace je zobrazen v Tabulce 9. Nyní jsou všechna kritéria pro normalizovanou matici variant maximalizačního typu. Pro přehlednost jsem čísla v tabulce zaokrouhlil na čtyři desetinná místa.

	C_1	C_2	C_3	C_4
A_1	0,1228	0	0	1
A_2	0,4737	1	0,7273	0
A_3	0	1	0	0
A_4	1	0	1	0,8036
A_5	0,2982	0	0,5455	0

Tabulka 9: Normalizovaná matice variant z příkladu

- Pro určení vah kritérií jsem vybral subjektivní metodu určení vah párových porovnání. Nejprve určím matici párových porovnání. Tuto matici ukazuje Tabulka 10. Například rovnost $c_{13} = 5$ znamená, že kritérium hodnocení procesoru je silně preferováno oproti kritériu úhlopříčka displeje.

	C_1	C_2	C_3	C_4
C_1	1	1/3	1/5	1/3
C_2	3	1	1/5	1/3
C_3	5	5	1	3
C_4	3	3	1/3	1

Tabulka 10: Matice párových porovnání z příkladu

Z matice párových porovnání vypočítám váhový vektor pomocí metody vlastního vektoru. Dostanu tak vektor

$$\vec{w} = (0{,}0736, 0{,}1293, 0{,}5495, 0{,}2476)\,,$$

kde jsem čísla opět zaokrouhlil na čtyři desetinná místa. Poměr konzistence $CR = 0{,}0734$ což znamená, že určení intenzity důležitosti bylo konzistentní.

Krok 5: Použití rozhodovací metody
Pro jednoduchost jsem vybral rozhodovací metodu váženého součtu. Nyní vypočítám skóre variant. Skóre každé varianty je následující:

$$A_1 = 0{,}2567, A_2 = 0{,}5638, A_3 = 0{,}1293, A_4 = 0{,}8221, A_5 = 0{,}3217\,,$$

kde například varianta $A_1 = 0{,}2567$, protože

$$A_1 = 0{,}1228 * 0{,}0736 + 0 * 0{,}1293 + 0 * 0{,}5495 + 1 * 0{,}2476\,.$$

Na základě scóre určíme uspořádání, které je $A_4 > A_2 > A_5 > A_1 > A_3$.

Krok 6: Využití výsledku rozhodnutí
Nejlepší možná varianta, dle vybraných kritérií a vypočítaných vah, je varianta A_4. Výsledné uspořádání variant je tedy výsledek, na základě kterého se mohu rozhodnout při výběru telefonu.

B Ukázka použití balíčku

V této příloze předvedu možné použití knihovny „mymcdm“. Nejprve uvedu použití API balíčku a do kontrastu ukáži použití metody *decision* na stejném příkladu. Nakonec představím práci s konzolovou částí. Následující Zdrojový kód 3 využívá API balíčku „mymcdm“.

```
1  import numpy as np
2
3  from mymcdm.normalization import vector
4  from mymcdm.utils import framing
5  from mymcdm.methods import topsis
6
7  types = [True, False, True, False]
8  weights = np.array((0.20, 0.15, 0.40, 0.25))
9  alternatives = np.array(
10     [
11         (30, 20, 10, 20),
12         (25, 20, 15, 30),
13         (25, 25, 5, 10),
14         (10, 30, 20, 30),
15         (30, 10, 30, 10),
16     ]
17 )
18
19 alternatives, types = vector(alternatives, types)
20
21 weights = framing.frame_criterions(weights, c_types=types)
22 alternatives = framing.frame_alternatives(alternatives, a_types=
       types)
23
24 result = topsis(alternatives, weights, types)
25
26 decision_matrix = framing.make_decision_matrix(alternatives,
       weights)
27
28 print(decision_matrix, result, sep= 2 * "\n")
```

Zdrojový kód 3: Ukázka použití balíčku

Ve Zdrojovém kódu 3 na řádcích 1 – 5 importuji potřebné metody z balíčku „mymcdm“ a importuji knihovnu NumPy. Následně si do proměnných *types*, *weights*, *alternatives* uložím vstupní hodnoty problému. Na řádku 19 pomocí vektorové normalizační metody normalizuji matici variant uloženou v proměnné *alternatives*. Na řádcích 21 a 22 orámuji váhy a normalizované varianty. Rozhodnutí proběhne na řádku 24, kde používám metodu TOPSIS pro vytvoření skóre variant. Nakonec vytvořím rozhodovací matici a společně ji s výsledkem rozhodnutí vypíšu. Spuštění modulu se Zdrojovým kódem 3 a výpis z příkazové řády je zobrazen na Obrázku 2.

```
(.benv)          @Marek-MacBook-Air tests % python3 test.py
Crits.       C1+       C2+       C3+       C4+
            0.20      0.15      0.40      0.25
Alts.
A1       0.534522  0.593862  0.246183  0.591752
A2       0.445435  0.593862  0.369274  0.387628
A3       0.445435  0.492327  0.123091  0.795876
A4       0.178174  0.390792  0.492366  0.387628
A5       0.534522  0.796931  0.738549  0.795876

Alts.
A1    0.338035
A2    0.388220
A3    0.316458
A4    0.464902
A5    1.000000
Name: score, dtype: float64
```

Obrázek 2: Výpis konzole na základě ukázkového kódu

Pokud bych chtěl naopak využít pomocnou metodu *decision*, napsal bych Zdrojový kód 4. Na něm lze vidět, že použití metody *decision* zredukuje množství potřebného kódu, narozdíl od přístupu, který využívá ostatních metod balíčku. Ve Zdrojovém kódu 4 se na základě zadaných argumentů vypočítá a uloží návratová hodnota metody *decision*. Návratová hodnota metody *decision* je objekt třídy *Result*, který obsahuje výsledek rozhodnutí, normalizovanou matici variant, váhy kritérií a další hodnoty.

Nakonec ukáži použití konzolové části balíčku „mymcdm“. Příklad použití je zobrazen na Obrázku 3. Při zadání příkazu „mymcdm decision -v“ se spustí práce s modelem. Pokud uživatel předem nezadal parametry „-d“ a „-n“ s kódy příslušných metod, označující rozhodovací a normalizační metody, pak bude vyzván k jejich zadání. V příkladu jsem tyto parametry nezadal, a tudíž jsem byl vyzván k jejich zadání. Zvolil jsem normalizační metodu Max-min a rozhodovací metodu WSM. Uživatel je následně vyzván k zadání cesty k souboru s daty. Já zadal cestu *example.json*. Poté proběhne zpracování dat a určení výsledku rozhodnutí. Následně se vypíše výsledek a pokud uživatel zadal parametr „-v“, pak se vypíše i rozhodovací matice. Nakonec je uživatel dotázán, jestli požaduje uložit výsledek. Pokud ano, pak je vyžádána i cesta, výsledek se uloží, aplikace skončí. Pokud ne, pak aplikace skončí. V příkladu jsem nepožadoval uložení výsledku.

```
● (test)           @Marek-MacBook-Air library % mymcdm decision -v
Enter normalization method name.
MAXMIN

Enter decision method name.
WSM

Enter the path to the data file:
example.json

Decision matrix:
Crits.   C1+   C2+  C3+  C4+
        0.20  0.15 0.40 0.25
Alts.
A1      1.00  0.50  0.8  0.5
A2      0.75  0.50  0.6  1.0
A3      0.75  0.75  1.0  0.0
A4      0.00  1.00  0.4  1.0
A5      1.00  0.00  0.0  0.0

The result is:
        score  rank
Alts.
A1     0.7200     1
A2     0.7150     2
A3     0.6625     3
A4     0.5600     4
A5     0.2000     5

Do you want to save result ? [Y / N]:
N
```

Obrázek 3: Výpis konzole po použití příkazu *mymcdm decision*

```
1   import numpy as np
2   import mymcdm
3
4   types = [True, False, True, False]
5   weights = np.array((0.20, 0.15, 0.40, 0.25))
6   alternatives = np.array(
7      [
8         (30, 20, 10, 20),
9         (25, 20, 15, 30),
10        (25, 25, 5, 10),
11        (10, 30, 20, 30),
12        (30, 10, 30, 10),
13     ]
14  )
15
16  result = mymcdm.decision(
17       alternatives,
18       weights,
19       types,
20       "VECTOR",
21       "TOPSIS")
```

Zdrojový kód 4: Ukázka použití metody *decision*

C Obsah elektronických dat

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, a podobně.

docs/
: Složka obsahující dokumentaci modulů, funkcí a tříd balíčku „mymcdm“.

examples/
: Příklady využití balíčku „mymcdm“. V podsložce *data* lze najít příklady vstupních dat.

library/
: Obsahuje zdrojový kód balíčku „mymcdm“ společně s distribučními archivy v podložce *dist*.

readme.md
: Příručka k elektronickým datům práce. Obsahuje instrukce pro instalaci balíčku „mymcdm“, včetně všech požadavků pro jeho bezproblémový provoz.

Literatura

[1] Hwang, Ching-Lai; Yoon, Kwangsun. *Multiple Attribute Decision Making*. 1981. Dostupný také z: ⟨https://doi.org/10.1007/978-3-642-48318-9⟩.

[2] Uzun, Berna; Ozsahin, Ilker; Agbor, Valerie Oru; Ozsahin, Dilber Uzun. Theoretical aspects of multi-criteria decision-making (MCDM) methods. In. *Applications of Multi-Criteria Decision-Making Theories in Healthcare and Biomedical Engineering*. 2021, s. 3–40. Dostupný také z: ⟨https://doi.org/10.1016/b978-0-12-824086-1.00002-5⟩.

[3] Triantaphyllou, Evangelos. *Multi-criteria Decision Making Methods: A Comparative Study*. 2000. Dostupný také z: ⟨https://doi.org/10.1007/978-1-4757-3157-6⟩.

[4] Brožová, Helena; Houška, Milan; Šubrt, Tomáš. *Modely pro vícekriteriální rozhodování*. 2003. ISBN: 80-213-1019-7.

[5] 1000minds. *Multi-Criteria Decision Analysis (MCDA/MCDM)* [online]. 2023 [cit. 2023-3-27]. Dostupný z: ⟨https://www.1000minds.com/decision-making/what-is-mcdm-mcda⟩.

[6] Taherdoost, Hamed; Madanchian, Mitra. Multi-Criteria Decision Making (MCDM) Methods and Concepts. *Encyclopedia*. 2023, roč. 3, č. 1, s. 77–87. Dostupný také z: ⟨https://doi.org/10.3390/encyclopedia3010006⟩.

[7] Odu, Godwin Oghenewiroro. Weighting methods for multi-criteria decision making technique. *Journal of Applied Sciences and Environmental Management*. 2019, roč. 23, č. 8, s. 1449. Dostupný také z: ⟨https://doi.org/10.4314/jasem.v23i8.7⟩.

[8] Wang, Jiang-Jiang; Jing, You-Yin; Zhang, Chun-Fa; Zhao, Jun-Hong. Review on multi-criteria decision analysis aid in sustainable energy decision-making. *Renewable and Sustainable Energy Reviews*. 2009, roč. 13, č. 9, s. 2263–2278. Dostupný také z: ⟨https://doi.org/10.1016/j.rser.2009.06.021⟩.

[9] Zardari, Noorul Hassan; Ahmed, Kamal; Shirazi, Sharif Moniruzzaman; Yusop, Zulkifli Bin. *Weighting Methods and their Effects on Multi-Criteria Decision Making Model Outcomes in Water Resources Management*. 2015. Dostupný také z: ⟨https://doi.org/10.1007/978-3-319-12586-2⟩.

[10] Vafaei, Nazanin; Ribeiro, Rita Almeida; Matos, Luis Manuel. Data normalisation techniques in decision making: Case study with topsis method. *International Journal of Information and Decision Sciences*. 2018, roč. 10, č. 1, s. 19. Dostupný také z: ⟨http://dx.doi.org/10.1504/ijids.2018.090667⟩.

[11] Natalja, Kosareva; Aleksandras, Krylovas; Kazimieras, Zavadskas Edmundas. Statistical Analysis of MCDM Data Normalization Methods Using Monte Carlo Approach. The Case of Ternary Estimates Matrix. *Economic Computation And Economic Cybernetics Studies And Research*. 2018, roč. 52, č. 4/2018, s. 159–175. Dostupný také z: ⟨https://doi.org/10.24818/18423264/52.4.18.11⟩.

[12] Vafaei, Nazanin; Ribeiro, Rita Almeida; Camarinha-Matos, Luis Manuel. Normalization Techniques for Multi-Criteria Decision Making: Analytical Hierarchy Process Case Study. In. *Technological Innovation for Cyber-Physical Systems*. 2016, s. 261–269. Dostupný také z: ⟨https://doi.org/10.1007/978-3-319-31165-4_26⟩.

[13] Jahan, Ali; Edwards, Kevin L. A state-of-the-art survey on the influence of normalization techniques in ranking: Improving the materials selection process in engineering design. *Materials and Design*. 2015, roč. 65, s. 335–342. Dostupný také z: ⟨https://doi.org/10.1016/j.matdes.2014.09.022⟩.

[14] Çelen, Aydın. Comparative Analysis of Normalization Procedures in TOPSIS Method: With an Application to Turkish Deposit Banking Market. *Informatica*. 2014, roč. 25, č. 2, s. 185–208. Dostupný také z: ⟨https://doi.org/10.15388/informatica.2014.10⟩.

[15] Tzeng, Gwo-Hshiung; Huang, Jih-Jeng. *Multiple Attribute Decision Making*. 2011. Dostupný také z: ⟨https://doi.org/10.1201/b11032⟩.

[16] Vafaei, Nazanin; Ribeiro, Rita Almeida; Camarinha-Matos, Luis Manuel. Assessing Normalization Techniques for Simple Additive Weighting Method. *Procedia Computer Science*. 2022, roč. 199, s. 1229–1236. Dostupný také z: ⟨https://doi.org/10.1016/j.procs.2022.01.156⟩.

[17] Velasquez, Mark; Hester, Patrick Thomas. An analysis of multi-criteria decision making methods. *International Journal of Operations Research*. 2013, roč. 10, s. 56–66.

[18] Yu, Po Lung. A Class of Solutions for Group Decision Problems. *Management Science*. 1973, roč. 19, č. 8, s. 936–946. Dostupný také z: ⟨https://doi.org/10.1287/mnsc.19.8.936⟩.

[19] Duckstein, Lucien; Opricovic, Serafim. Multiobjective optimization in river basin development. *Water Resources Research*. 1980, roč. 16, č. 1, s. 14–20. Dostupný také z: ⟨http://dx.doi.org/https://doi.org/10.1029/WR016i001p00014⟩.

[20] Benayoun, Raphael; Roy, Bernard; Sussman, B. Manual de Reference du Programme Electre. *Direction Scientifque SEMA*. 1966, č. 25.