AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa inżynierska

Biblioteka rankingowa w Java Script
Java Script Ranking Library

Autor: *Małgorzata Wesołowska*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr Konrad Kułakowski*

Kraków, 2015

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję mojemu promotorowi, dr Konradowi Kułakowskiemu, za możliwość realizacji pracy dyplomowej pod jego kuratelą. Również słowa podziękowania należom się moim rodzicom i kolegom z roku, którzy zawsze mnie wspierali i pomagali dojść do tego momentu.

Spis treści

1. Wprowadzenie

Każdy z Nas codziennie staje przed koniecznością wyboru. Zarówno w życiu prywatnym jak i zawodowym. Maturzysta musi rozsądnie wybrać sposród wachlarza propozycji jakie oferują im uczelnie. Do głowy rodziny należy podjęcie decyzji, który samochód kupi, czy też z którym dostawcą elektryczności podpisze umowę. Prezes firmy staje przed wyborem potencjalnego partnera biznesowego. Ja i moi koledzy z roku byliśmy, bądź niebawem będziemy, oceniani jako potencjalni pracownicy, na podstawie życiorysu, listu motywacyjnego, doświadczenia,oraz tego jak wypadliśmy na rozmowie rekrutacyjnej.

Już w czasach antycznych greccy uczeni zdawali sobie sprawę, że istotne są nie tylko rangowanie opcji i podejmowanie decyzji, ale również sama klasyfikacja problemu. Grecki filozof Epikur podzielił ludzkie potrzeby na dwie kategorie: podstawowe (np. dążenie do odczuwania przyjemności) i zbyteczne (np. pragnienie nieśmiertelności). Twórca epikureizmu chciał w ten sposób ułatwić antycznym odnalezienie wewnętrznego spokoju.

W czasach współczesnych klasyfikacja przewija się na porządku dziennym. Patolog, dokonujac diagnozy, przydziela chorobę do odpowiedniej grupy, dzięki czemu jest w stanie przepisać pacjentowi odpowiednią terapię. Pięciostopniowa klasyfikacja Saffira-Simpsona dokonuje podziału hugaranów i cyklonów zgodnie z ich szybkością wiatru, ciśnieniem powierzchniowym oraz wysokością fali (źródło: [Wal01]).

Powyższe przykłady pokazują, że złożone problemy decyzyjne pojawiają się bardzo często. Niezależnie czy dokonujemy pojedynczego wyboru, czy ustalamy listę priorytetową, niejednokrotnie musimy wziąć pod uwagę więcej niż jedno kryterium. Przecież dokonując zakupu domu nie kierujemy się jedynie jego ceną.

Ideały nie istnieją, tak też niezmiernie trudno jest znaleźć perfekcyjne rozwiązanie pasujące do wszystkich wymogów. Dlatego konieczne jest znalezienie najoptymalniejszego wyjścia z danej sytuacji. W celu rozwiązania tego problemu decydent może skorzystać z metod naiwnych jak na przykład suma ważona. W rzeczywistości ta metoda jest nieodpowiednia, ponieważ zakłada linearność danych, co nie odzwierciedla stanu realnego. Nie możemy założyć, że czterogodzinna wycieczka do Disneylandu jest dwukrotnie lepsza od dwóch dwugodzinnych. Z

pomocą spieszą nam przeróżne metody wiekokryterialnej analizy decyzyjnej.

1.1. Cel pracy

Obecnie mamy na rynku znaczącą liczbę metod zaimplementowanych, wdrożonych i gotowych do rozwiązywania wielokryterialnych problemów. Ich rozwój wciąż trwa, a wzrost publikacji naukowych związanych z MCDA stale rośnie. Ekspansja następuje między innymi ze względu na wydajność naukowców i rozwój poszczególnych metod dla różnych rodzajów problemów napotkanych w MCDA. Dostępne oprogramowanie (wliczając w to: arkusze zawierające odpowiednio zaimplementowane metody obliczeniowe, rozwiązania ad hoc, webowe czy na smartfony) sprawiło, że metody wielokryterialnej analizy decyzyjnej stały się bardziej dostępne i przyczyniły się do wzrostu użycia metod MCDA wśród naukowców jak i użytkowników. [WJ11]

Celem poniższej pracy jest implementacja algorytmów rankingowych wspomagających metody wielokryterialnej analizy decyzyjnej w języku JavaScript. Poprzez realizację tego projektu zrozumienie metod rankingowych jak i samego pojęcia MCDA będzie znacznie łatwiejsze. Praca pokrywa tylko podstawowe pojęcia związane z tym tematem, dlatego też zainteresowanych odsyłam do źródeł z bibliografii.

1.2. Zawartość pracy

Praca dyplomowa składa się z pięciu rozdziałów. Podzieliłam je na trzy główne części:

1. Teoria
2. Implementacja
3. Wnioski

W rodziale 2 przedstawiłam podstawowe informacje dotyczące metod wielokryterialnej analizy decyzyjnej. Poruszyłam tematykę podejmowania złożonych decyzji oraz formowania problemu. Opisałam zestawienie istniejących już na rynku metod. W ostatniej sekcji podjęta jest problematyka odpowiedniego doboru metod do zaistnialego problemu.

Na przykładzie metody AHP (Analytic Hierarchy Process, rozdział 3) zaprezentowałam zasadę działania algorytmów rankingowych. Do realizacji tej sekcji użyłam przykładów z książki dr Ishizakiego - darmowe oprogramowanie *MakeItRational*. Bezpłatna wersja pozwala na dogłębniejsze zapoznanie się z głównymi ideami metod poprzez praktyczne ćwiczenia.

Rozdział 4 opisuje proces realizacji projektu. Wspomniałąm o samym języku javascript jak i wszystkich narzędziach i bibliotekach wspomagających moją pracę. Korzystałam także z kilku narzędzi zarządzania projektem, a samą pracę pisałam w LaTeXie. Powstały też proste testy (5) w celu prezentacji otrzymanych wyników.

Ostatnia część (rodział 6) zawiera podsumowanie całości projektu, wnioski jakie się nasuwają po pracy nad zagadnieniem metod MCDA oraz problemy jakie napotkałam podczas procesu implementacji algorytmów rankingowych.

2. Wstęp teoretyczny

2.1. Wielokryterialna analiza decyzyjna

Metody wielokryterialnej analizy decyzyjnej (MCDA) są rozwijane, by wspomóc decyzentów w ich unikalnym i spersonalizowanym procesie decyzyjnym. Metody MCDA oferują bezpieczne techniki pozwalające odszukanie kompromisowego rozwiązania. Wyraźnie umieszczają osobę podejmującą decyzję w środku procesu. Nie są one algorytmami zautomatyzowanymi, które wiodą do tego samego rozwiązania dla każdego decydenta, ale zwracają subiektywną informację [IA13]. Subiektywna informacja, znana także jako informacja o preferencjach, dostarczana jest przez decydenta i wiedzie do kompromisowego rozwiązania problemu.

Wielokryterialna analiza decyzyjna jest dyscypliną obejmująca takie dziedziny jak matematyka, zarządzanie, informatyka, psychologia, nauki społeczne i ekonomia. Jej aplikacja może być jeszcze szersza, ponieważ MCDA może zostać użyte do rozwiązania każdego problemu, gdzie istnieje wyraźna potrzeba podjęcia jednoznacznej decyzji. Decyzje te mogą być zarówno taktyczne jak i strategiczne, w zależności od perspektywy czasowej oraz konsekwencji (patrz tabela 2.1).

Tablica 2.1: Kategorie problemów decyzyjnych

Decyzja	**Czas**	**Oryginalność**	**Ustrukturyzowanie**	**Automatyzacja**
Strategiczna	Długoterminowe	Nowe	Niskie	Niska
Taktyczna	Średniotermiowa	Adaptacyjne	Średnie	Średnia
Operacyjna	Krótkoterminowe	Powszechne	Wysokie	Wysoka

2.1.1. Problem podejmowania decyzji

Każdego dnia ludzie stawiają czoła niezliczonej ilości różnych decyzji. Ponad 30 lat temu ktoś zdecydował się podzielić je na grupy. Bernard Roy [B.81] zidentyfikował cztery główne typy decyzji:

1. Problem wyboru – Celem jest wyselekcjonowanie pojedynczego, najlepszego rozwiązania lub redukcja zbioru rozwiązań do takich, które są w tym samym stopniu odpowiednie dla danego problemu. Przykładem może być dobieranie zespołu do konkretnego projektu.

2. Problem sortowania – Opcje są sortowane w uporzadkowanych i predefiniowanych grupach nazywanych kategoriami. Celem jest przegrupowanie opcji o podobnym zachowaniu lub charakterystyce. Przykładowo - ocena okresowa pracowników może być wystawiona jako „ponad przecięta", „przeciętna" lub „poniżej przeciętnej". Bazując na takiej klasyfikacji, można podjąć odpowiednie środki. Metody sortujące są przydatne do powtarzalnych i automatycznych zastosowań. Mogą one również być stosowane jako wstępne badanie w celu ograniczenia liczby możliwości, które należy uwzględniać w następnym etapie.

3. Problem rangowania – Alternatywy są szeregowane od najlepszej do najgorszej w rozumieniu porównań rankingowych wyników itp. Kolejność może być cząstkowa, jeśli bierzemy pod uwagę nieporównywalne rozwiązania. Typowym przykładem może być wybór uczelni. Rankingu dokonujemy wtedy na podstawie kilku kryteriów, np. jakość kształcenia, kosztó dojazu lub życia, perpektyw zatrudnienia w swoim zawodzie.

4. Problem opisowy – Celem jest opis rozwiązań i ich konsekwencji. Zazwyczaj jest to robione w pierwszej kolejności, by zrozumieć naturę tematu.

Do wyżej wymienionego zestawienia zostały zaproponowane dwa nowe:

1. Problem eliminacji – Bana e Costa [WJ11] rozwinął gałąź problemów sortowania o problem emilinacji.

2. Problem projektowy – Celem jest zidentyfikowanie nowej akcji, która spełni wymagania i aspiracje decydenta. [R.92]

Do listy można dodać „problem aktywizacji", którego celem jest wyłonienie preferowanych parametrów dla konkretnej metody MCDA. Co więcej, kiedy problem dotyczy kilku decydentów, musi zostć użyta adekwatna grupa metod.

Istnieje wiele innych problemów, często łączących ze sobą kilka wspomnianych w tym rozdziale. Jednakże, ta praca skupia się tylko i wyłącznie na problemach rankingowych.

2.1.2. Metody MCDA

By uporać się z problemami opisanymi w poprzedniej sekcji, rozwinięto doraźne metody. Tabelka 2.2 prezentuje metody z wyróżnieniem jakich problemów decyzyjnych dotyczą. Istnieje wiele więcej metod niż te niżej wymienione, te jednak są najpopularniejsze. Ja skupiłam się w kolejnym rozdziale na konkretnej metodzie - AHP.

Tablica 2.2: Problemy i metody MCDA

Problemy wyboru	**Problemy rankingowe**	**Problemy sortowania**	**Problemy opisowe**
AHP	AHP	AHPSort	
ANP	ANP		
MAUT/UTA	MAUT/UTA	UTADIS	
MACBETH	MACBETH		
PROMETHEE	PROMETHEE	FlowSort	GAIA, FS-Gaia
ELECTRE I	ELECTRE III	ELECTRE-Tri	
TOPSIS	TOPSIS		
Goal Programming			
DEA	DEA		
Wielometodyczna platforma wspierająca różne metody MCDA			

Wielu naukowców oraz firm komercyjnych rozwijało różne oprogramowania na przestrzeni ostatnich lat, by pomóc użytkownikom sformułować oraz rozwiązać ich problemy decyzyjne. W kolejnym rozdziale w przykładzie użyto oprogramowania *MakeItRational*. Dla zobrazowania ilości dostępnych rozwiązań w tabeli 2.3 znajduje się niepełne zestawienie istniejących oprogramowań.

2.1.3. Dobór metod

Biorąc pod uwagę ilość dostępnych metod MCDA, użytkownik postawiony jest przed żmudnym zadaniem wyboru odpowiedniego narzędzia decyzyjnego i wybór ten niejednokrotnie jest ciężki do uzasadnienia. Żadna z metod nie jest idealna, czy też odpowiednia dla wszystkich problemów. Każda z nich ma swoje ograniczenia, szczegółowość, hipotezy, przesłanki i perspektywy. Cytując Roy'a i Bouyssou:

„Choć duża różnorodność procedur MCDA może być postrzegana jako zaleta, może być również ich słabością. Do tej pory nie było możliwości podjęcia decyzji, czy jedna metoda ma większy sens niż inna w określonej sytuacji problemowej. Systematyczna analiza aksjomatów procedur decyzyjnych i algorytmów jest jeszcze do wykonania." ([B.81]).

Tablica 2.3: Oprogramowania MCDA

Problemy	**Metody MCDA**	**Oprogramowanie**
Rankingowe, opisowe, wyboru	PROMETHEE -GAIA	Decision Lab,D-Sight, Smart Picker Pro,Visual Promethee
Rankingowe, wyboru	PROMETHEE	DECERNS
	ELECTRE	Electre IS, Electre III-IVUTA Right Choice, UTA+,DECERNS
	UTA	Right Choice, UTA+, DECERNS
	AHP	MakeItRational, ExpertChoice, Decision Lens, HIPRE 3+, RightChoiceDSS, Criterium, EasyMind, Questfox,ChoiceResults, 123AHP,DECERNS
	ANP	Super Decisions, Decision Lens
	MACBETH	M-MACBETH
	TOPSIS	DECERNS
	DEA	Win4DEAP, EfficiencyMeasurement System, DEASolver Online, DEAFrontier, DEA-Solver PRO, FrontierAnalyst
Wyboru	Goal Programming	-
Sortowania, opisowe	FlowSort - FS-GAIA	Smart Picker Pro
Sortowania	ELECTRE-Tri	Electre Tri, IRIS
	UTADIS	-
	AHPSort	-

Istnieje wiele sposobów doboru odpowiedniej meotody MCDA do rozwiązania specyficznych problemów. Można brać pod uwagę wymagane informacje wejściowe, czyli dane i parametry metody wraz z konsekwencjami wysiłku modelowania, jednocześnie miejąc na uwadze na dane wyjściowe oraz ich szczegółowość. 2.4

Jeśli funkcja użytkowa dla każdego kryterium jest znana, należy wybrać MAUT. Jednakże konstrukcja takiej funkcji jest wysoce wymagająca i jeśli jest to zbyt czasochłonne i trudne – istnieją inne rozwiązania. Dobrym zastępstwem w takim przypadku są porównania parami pomiędzy kryteriami i rozwiązaniami. AHP oraz MACBETH wspierają takie rozwiązania. Róż-

Tablica 2.4: Wymagane wejście dla metod rankingowych i sortowania

Wejście	Wysiłek	Metoda	Wyjście
Funkcja użytkowa	Bardzo wysoki	MAUT	
Porównania parami na skali ilorazowej i współzależności		ANP	Kompletny ranking z ocenami
Porównania parami na skali podziałowej i współzależności		MACBETH	
Porównania parami na skali ilorazowej		AHP	
Obojętność, preferencje i warunek stopu		ELECTRE	Częściowy i kompletny ranking z ocenami
Obojętność i próg preferencji		PROMETHEE	Częściowy i kompletny ranking z ocenami
Idealna opcja z ograniczeniami		Goal Programming	Wykonalne rozwiązanie z przybliżonymi ocenami
Idealna i najgorzsza opcja		TOPSIS	Kompletny ranking z przybliżonymi ocenami
Nie wymagane żadne wejście	Bardzo niski	DEA	Częściowy ranking z przybliżonymi ocenami

nica polega na tym, że porównania dla AHP są rozpatrywane dla skali ilorazowej[1], natomiast dla metody MACBETH - na skali z podziałką[2]. Osoba podejmująca decyzję musi wiedzieć, która skala lepiej nadaje się do osiągnięcia swoich preferencji. Wadą tych rozwiązań jest fakt, że potrzeba dużej ilości informacji.Skupiając się na porównaniach parami, będących tematem tej pracy, nie będę opisywać procesu doboru kolejnych metod.

Wysiłek modelowania określa bogactwo danych wyjściowych. Zaletą definiowania funkcji użytkowych jest to, że rozwiązania problemu decyzyjnego mają ocenę globalną. Na podstawie tego wyniku, można porównać wszystkie możliwości i klasyfikować je od najlepszej do najgorszej. Dozwolone są równe pozycje w rankingu. Tak właśnie zdefiniowany jest kompletny

[1]Skala ilorazowa (także: skala stosunkowa) – stosunki między dwiema jej wartościami mają interpretację w świecie rzeczywistym. Przykłady: temperatura w kelwinach, napięcie elektryczne, inflacja, bezrobocie

[2]Skala interwałowa (przedziałowa) – różnice między dwiema jej wartościami dają się obliczyć i mają interpretację w świecie rzeczywistym, jednak nie ma sensu dzielenie dwóch wartości zmiennej przez siebie. Innymi słowy określona jest jednostka miary, jednak punkt zero jest wybrany umownie.Przykłady: daty, temperatura w stopniach Celsjusza

ranking. To podejście jest określane jako podejście pełnej agregacji, kiedy jakiś zły wynik jednego kryterium można rekompensować dobrym - z innego kryterium.

Metody rankingowe są oparte na porównaniach par. Oznacza to, że opcje są porównywane parami za pomocą stopnia rankingowego lub preferencji. Stopnie te odzwierciedlają o ile bardziej preferowaną jest dana alternatywa w porównaniu z inną opcją. Możliwe jest by jakieś opcje były niemożliwe do porównania. Jeśli dwa rozwiązania mają odmienne profile ich porównanie może być skomplikowane: pierwsza opcja bardziej pasuje dodanego zestawu kryteriów, a druga - do innego zestawu wymogów. Właściwość ta oznacza, że nie zawsze można uzyskać kompletny ranking, co jest orkeślane w dolnej części tabelki jako częściowy ranking. Nieporównywalność jest konsekwencją nierekompensacyjnych aspektów tych metod. W obliczu problemu decyzyjnego, ważne jest, aby określić pożądany typ wyjścia już na początku.

3. Analytic Hierarchy Process

Ten rozdział wyjaśnia teorię metody Analytic Hierarchy Process (sekcja 3.1). Zagadnienia teoretyczne poparto, którkim sprawozdaniem z testowania oprogramowania *MakeItRational*. Pomaga ono uporządkować problemy oraz obliczyć priorytety rozwiązań wykorzystując AHP (sekcja 3.2).

3.1. Istotne pojęcia AHP

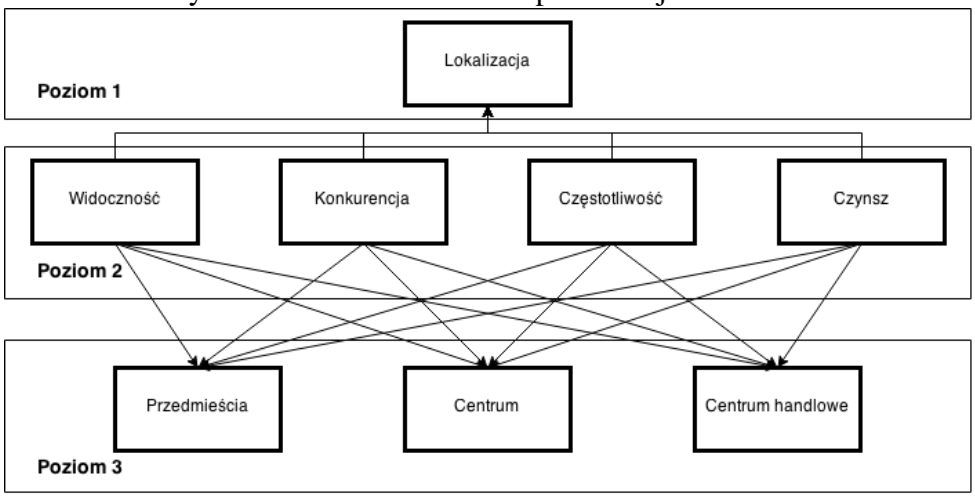
Metoda AHP została opracowana przez Saaty'ego ([WJ11]).Jest szczególnie przydatną metodą, gdy decydent nie jest w stanie skonstruować funkcji użyteczności, w innym wypadku zalecana jest metoda MAUT ([IA13]). Korzystając z AHP użytkownik musi wypełnić cztery kroki, aby uzyskać ranking alternatyw. Tak jak w przypadku każdej innej metody MCDA, problem musi mieć najpierw strukturę tak jak opisano to w sekcji 3.1.1. Następnie wyniki, a raczej ich priorytety są obliczane na podstawie porównań parami informacji dostarczonych przez użytkownika (3.1.2). Decydent nie musi dostarczać numerycznej oceny, zamiast tego słowna aprobata, bardziej podobna do naszego codziennego życia, będzie w zupełności wystarczająca. Istnieją jeszcze dwa kolejne kroki, które mogą zostać realizowane: sprawdzenie zgodności (3.1.3) oraz analiza wrażliwości wyników (3.1.4). Obie czynności są opcjonalne, ale zalecane jako potwierdzenie trwałości wyników. Kontrola spójności jest wspólna dla wszystkich metod opartych na porównaniach parami, jak AHP. Oprogramowanie *MakeItRational* ułatwia analizę wrażliwości.

3.1.1. Konstrukcja problemu

Metoda AHP opiera się na regule „dziel i zwyciężaj". Problemy wymagające zastosowania metod MCDA są zazwyczaj złożone i korzystnie jest, aby je rozdzielić i rozwiązać jeden podproblem na raz. Ten podział odbywa się w dwóch etapach procesu decyzyjnego podczas:

– strukturyzacji problemu oraz

Rysunek 3.1: Podstawowa reprezentacja hierachii.



– wydobywania priorytetów za pomocą porównań parami.

Problem jest skonstruowany zgodnie z hierarchią (Rysunek 3.1), gdzie górny element jest celem decyzji. Drugi poziom hierarchii przedstawia kryteria, a najniższy poziom reprezentuje alternatywy. W bardziej złożonych hierarchiach może być dodanych więcej poziomów. Dodatkowe poziomy reprezentują podkryteria. W każdym przypadku, są co najmniej trzy poziomy hierarchii.

Przejdźmy do przykładu, żeby lepiej zobrazować poszczególne kroki metody AHP. Załóżmy, że chcemy otworzyć sklep i zastanawiamy się nad jego lokalizacją. Mamy do wyboru:

– Lokal w Galerii Krakowskiej, gdzie zawsze jest pełno potencjalnych klientów, sklepy promują swoje produkty wielkimi wystawami w oknach, a koszta wynajmu jak na takie warunki są rozsądne.

– Mniejszy sklep znajduje się na Rynku Głównym. Jest to miejsce spotkań młodzieży i zawsze widać tłumy turystów. Ze względu na zabytkową lokalizację czynsz jest dość wysoki.

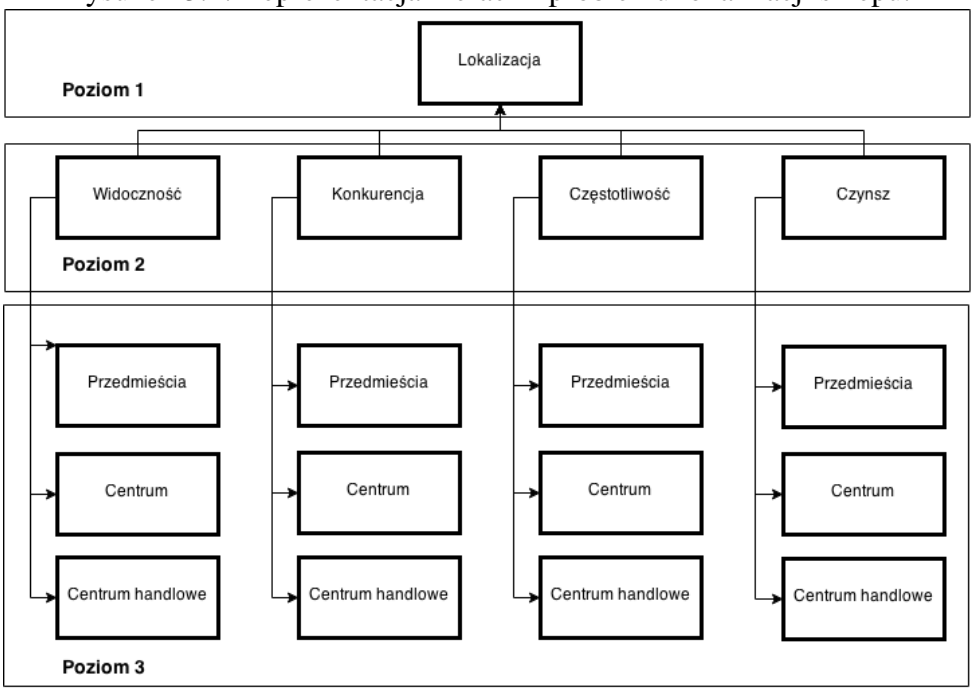
– Ostatecznie moglibyśmy otworzyć sklep na przedmieściach w powstającej dzielnicy. W tej okolicy mimo małej konkurencji ciężko jest przyciągnąć klientów, przez co koszta wynajmu jak na razie są niskie.

Musimy wziąć pod uwagę kilka istotnych aspektów. W tabelce poniżej (3.1) znajduje się zebrany zestaw kryteriów, według których moglibyśmy przeglądać i wybierać dostępne alternatywy.

Tablica 3.1: Kryteria przy wyborze lokalizacji sklepu.

Kryterium	Opis
Widoczność	Prawdopodobieństwo, że przechodzień zauważy sklep.
Konkurencja	Ilość konkurencyjnych sklepów w okolicy.
Częstotliwość	Średnia ilość klientów w sklepach tej branży w okolicy.
Czynsz	Średni koszt wynajmu lokalu na metr kw.

Rysunek 3.2: Reprezentacja hierachii problemu lokalizacji sklepu.



Rysunek 3.2 przedstawia jak taka hierarchia mogłaby wyglądać dla tego przykładu. Ma ona trzy poziomy, czyli wymagane minimum, żeby roziwązać problem z pomocą AHP. Inne podkryteria mogłyby zostać rozważone (np. konkurencję można podzielić na bezpośrednią i pośrednią. Do pierwszej zaliczalibyśmy sklepy z tej samej brażny, do drugiej – pozostałe sklepy, które mogłyby rozpraszać potencjalnych klientów). By nie utrudniać sobie sprawy nie dodawałam większej ilości poziomów.

Każdy poziom ma niższy priorytet niż jego bezpośredni sąsiad powyżej. By odpowiednio rozwiązać do problemu musimy wiedzieć co jest dla Nas najważneijsze. W tym celu nalezy zadać sobie właściwe pytania. Na przykład, w celu priorytetowego traktowania kryterium poziomu 2 w odniesieniu do celu „lokalizacja sklepu", odpowiednie byłoby pytanie: „Jakie kryterium jest najważniejsze przy wyborze lokalizacji sklepu i do jakiego stopnia?". Z drugiej strony, możliwości w poziomie 3 muszą mieć priorytet w odniesieniu do każdego kryterium w pozio-

mie 2. W tym przypadku, odpowiednie byłoby pytanie: „Jaką alternatywę zaleca się, by spełnić dane kryterium i w jakim stopniu?”.

W prezentowanym przypadku, wymagane jest pięć różnych klasyfikacji według priorytetów:

– Cztery lokalne priorytetyzacje w odniesieniu do każdego kryterium.

– Priorytetyzacja dla kryteriów.

Gromadzenie priorytetyzacji lokalnej i kryterialnej prowadzi do globalnej priorytetyzacji.

3.1.2. Obliczanie priorytetów

Priorytet jest wynikiem, który określa ważność danej alternatywy bądź kryterium w decyzji. Po fazie strukturyzującej wielokryterialny problem, trzy rodzaje priorytetów muszą być obliczone:

– Priorytety kryteriów - znaczenie każdego z kryteriów (w stosunku do odgórnego celu).

– Lokalne priorytety alternatyw - znaczenie rozwiązania względem konkretnego wymogu.

– Globalne priorytety alternatyw - dwa pierwsze są pośrednimi wynikami stosowanymi do obliczenia priorytetów globalnych. Globalne priorytety alternatyw plasują wyniki respektując wszystkie wymogi, a co za tym idzie główny cel.

Tablica 3.2: Fundamentalna skala Saaty’ego ([Var10])

Stopień	Opis
1	Równie ważne
2	Słabo
3	Stosunkowo ważne
4	Stosunkowo ważne +
5	Silnie ważne
6	Silnie ważne +
7	Bardzo ważne
8	Bardzo, bardzo ważne
9	Ekstremalnie ważne

Dwa pierwsze priorytety obliczane są przy pomocy tej samej techniki porównań parami. Z metody tej często korzystają psychologowie. Podjęli się oni pewnego eksperymentu. podsuwając kotu dwie miski z jedzeniem oryginalnym i tańszą podróbką, naukowcy obserwowali

preferencje zwierzęcia. Psychologowie twierdzą, że łatwiejsze i bardziej precyzyjne jest wyrażenie preferencji między tylko dwoma alternatywami niż jednocześnie wśród wszystkich opcji. Porównania parami zazwyczaj oceniane jest w podstawowej skali 1-9. Znaczenia poszczególnych stopni są w tabeli 3.2. Psychologowie wskazują, że mniejsza skala, powiedzmy 1-5, nie daje tego samego poziomu szczegółowości zbioru danych oraz że decydent byłby zagubiony w skali np. od 1-100., ponieważ trudno jest odróżnić wynik 46 do 47. W praktyce, nie ma stałej reguły i inne skale są także stosowane. Porównania zbierane są w macierz (przykładowa w tabeli 3.3).

Tablica 3.3: Macierz porównań

	Widoczność	**Konkurencja**	**Częstotliwość**	**Czynsz**
Widoczność	1	1/4	1/5	2
Konkurencja	4	1	1/2	1
Częstotliwość	5	2	1	4
Czynsz	1/2	1	1/4	1

Mamy tutaj macierz 4x4. Porównania na głównej przekątnej wynoszą 1, ponieważ to porównanie kryterium z samym sobą. Łatwo zauważyć, że górny trójkąt jest odwrotnością dolnego trójkąta (np. widoczność jest 1/4 tak ważna, jak konkurencja, a konkurencja jest 4 razy ważniejsza niż widoczność).

Precyzja wymaga więcej wysiłku, zwłaszcza gdy istnieje duża liczba kryteriów lub alternatyw. Liczbę niezbędnych porównań dla każdej macierzy porównania obliczamy ze wzoru: $\frac{n^2-n}{2}$. Dla przykładu n = 4, dlatego ilość koniecznych porównań wynosi (16 - 4)/2 = 6. Chociaż potęga jest zmniejszona o n i podzielona przez 2, wymagana liczba porównań może być bardzo wysoka.Wysiłek potrzebny do wykonania macierzy jest czasochłonny i może być zniechęcający.

Z macierzy porównań program wyliczy priorytety lokalne i kryterialne, które zostaną na końcu użyte do wyznaczenia priorytetów globalnych.

3.1.3. Kontrola stopnia zgodności

Gdy macierz jest uzupełniona, można wykonać sprawdzanie spójności w celu wykrycia ewentualnych sprzeczności w jej wpisach. Po przeprowadzeniu kilku kolejnych udanych porównaniach parami, może się okazać, że są one sprzeczne względem siebie. Przyczynami tych sprzeczności mogą być, na przykład, niejasno zdefiniowane problemy, niewystarczająca ilość informacji wejściowych, niepewne informacje lub brak koncentracji. Załóżmy, że ekspert, jako przykład podaje następujące porównania parami:

– Sklep w galeria jest 2 razy bardziej widoczny od sklepu w centrum miasta.

– Lokal w centrum jest 3 razy bardziej widoczny niż ten na przedmieściach.

– Sklep na przedmieściach są 4 razy bardziej widoczne niż ten w centrum.

Trzecie twierdzenie jest sprzeczne, jak wynika z pierwszych dwóch twierdzeń: galeria jest sześć razy bardziej widoczna niż przedmieścia (2 x 3). Ludzka natura jest jednak często nieprzewidywalna, np drużyna piłkarska z końca tabeli może pokonać liderów z tej samej grupy. By zagwarantować naszej niepewności pewien realizm AHP pozwala na 10% rozbieżności w porównaniu ze średnią niepewnością wyliczoną z 500 losowo wypełnionych macierzy ([IA13]). Obliczenia wskazują, czy macierz musi zostać ponownie przeanalizowane z uwagi na jej wysoką niezgodność.

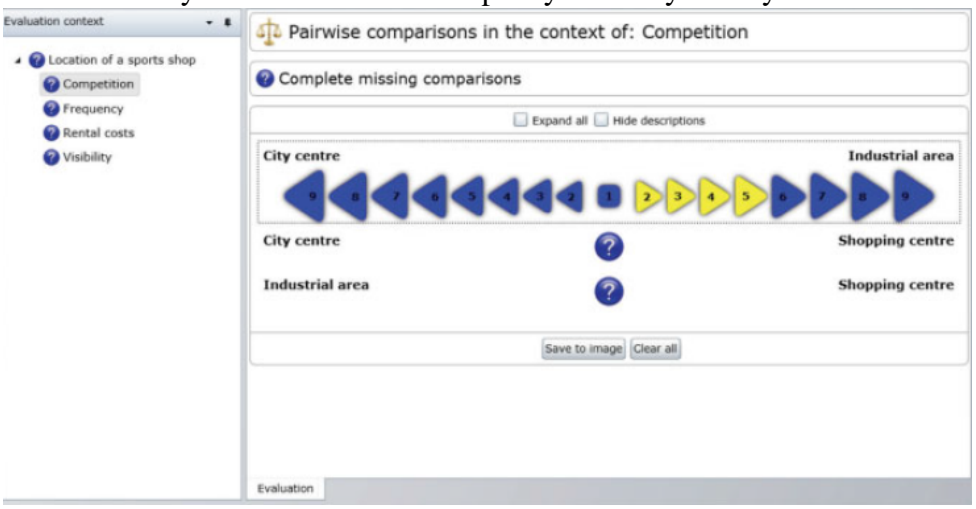
3.1.4. Analiza wrażliwości wyników

Ostatnim krokiem procesu decyzyjnego jest analiza czułości, gdy dane wejściowe są nieco zmodyfikowane, aby obserwować wpływ na wyniki. Skomplikowane modele decyzyjne są zazwyczaj często dość słabo zdefiniowane. Analiza wrażliwości umożliwia wyłonienie różnych scenariuszy, które składają się z różnych rankingów. Wtedy mogą być potrzebne dalsze rozmowy, aby osiągnąć konsensus. Jeżeli ranking się nie zmienia, wyniki są uważane za niezawodne - w przeciwnym razie są one wrażliwe. Analiza wrażliwości w *MakeItRational* odbywa się poprzez zmianę wagi kryteriów i obserwację ich wpływu na globalny prirytet alternatyw.

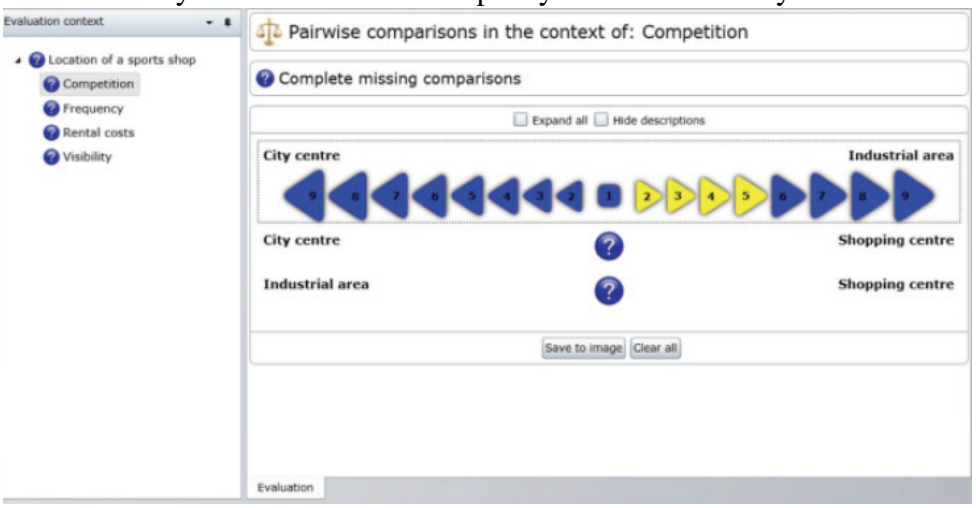
3.2. Przykład programu AHP: MakeItRational

Dostępne oprogramowania AHP w znacznym stopniu przyczyniły się do rozpowszechnienia metody. Oprogramowanie zawiera intuicyjny interfejs użytkownika, automatyczne obliczanie priorytetów i niespójności i udostępnia kilka sposobów przetwarzania analizy wrażliwości. Poniżej znajdują się zrzuty ekranu oprogramowania *MakeItRational*.

Rysunek 3.3: Obliczanie priorytetów kryterialnych.



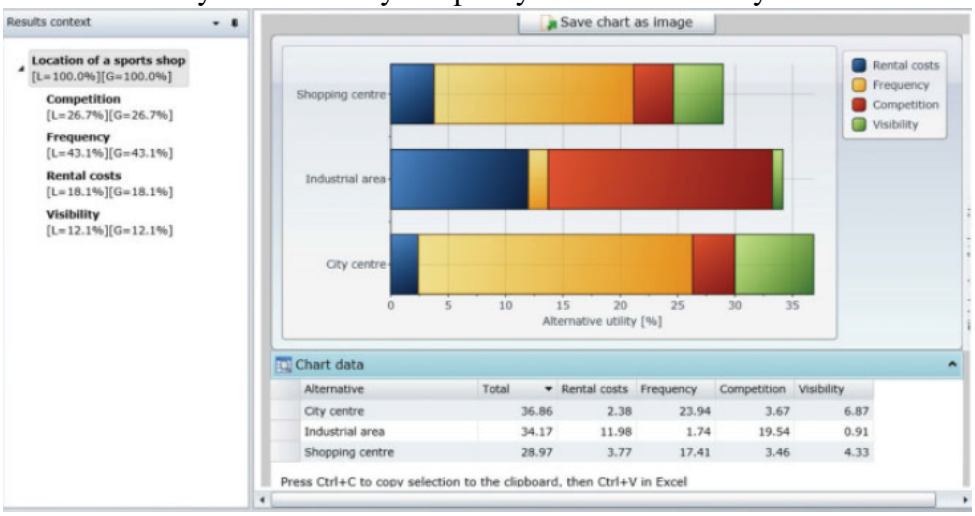
Rysunek 3.4: Obliczanie priorytetów dla alternatyw.



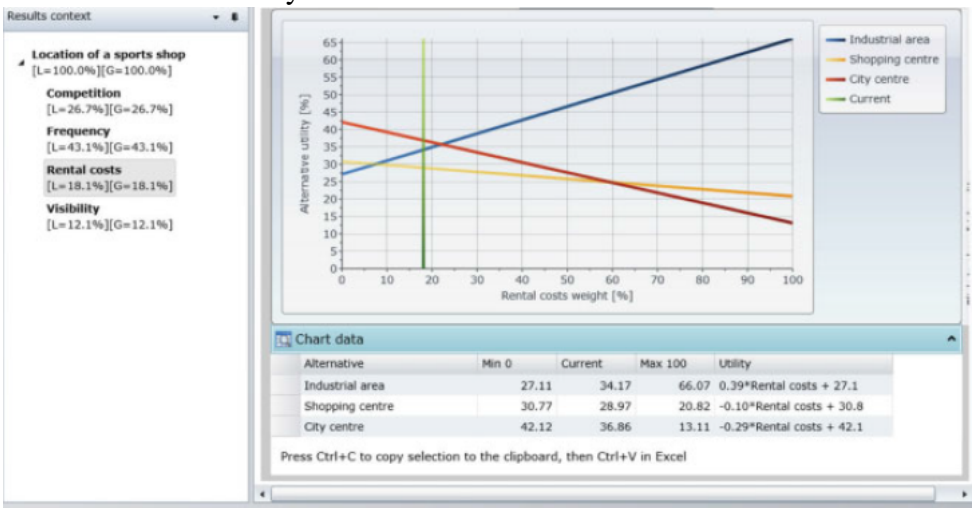
Przy tych dwóch krokach, program od razu zakomunikuje nam jeśli wprowadzane priorytety będą niezgodne w sensie AHP.

Mamy do dyspozycji kilka rozdzajów wykresów (słupkowy, kołowy, liniowy). Wyraźnie tutaj widać, że alternatywą wybraną dla naszego problemu została trzecia opcja.

Rysunek 3.5: Wyniki priorytetów dla alternatyw.



Rysunek 3.6: Analiza wrażliwości.



4. Realizacja pracy

W rozdziale tym przedstawiono opis procesu realizacji projektu. Począwszy od analizy dostarczonych przez promotora materiałów, zapoznaniu się z technologiami użytymi do ich powstania (sekcja 4.1), poprzez dobór odpowiednich narzędzi przed przystąpieniem do pracy (sekcja 4.2), a skończywszy na weryfikacji powstałego kodu poprzez porównanie wyników z pierwowzorem (rozdział 5).

4.1. Analiza biblioteki

Na stronie promotora tej pracy, doktora Konrada Kułakowskiego, można znaleźć gotową paczkę Pairwise Comparison Package.

`www.kulakowski.org/doku.php?id=profi:research:pcpackage`

Zawiera ona zaimplementowane podstawowe funkcjonalności metod rankingowych. Można wyliczyć stopień zgodności, czy też obliczyć przybliżone wartości nieznanych alternatyw przy pomocy zestawu referencyjnego. Paczka została zaprojektowana i napisana w języku Wolfram. Na stronie znajduje się krótka dokumentacja oparta na przykładach oraz link do repozytorium zawierającym aktualny kod, jak również skompresowany plik .tgz. „Pairwise Comparisons Package for Wolfram Mathematica" działa wraz z programem Wolfram Mathematica, dostępnym również na darmowej licencji dla użytkowników RaspberryPI [AGH14].

Twórcą języka Wolfram jak i samego programu *Mathematica* jest brytyjski naukowiec Stephen Wolfram [Wal01]. Założyciel firmy Research Wolfram wydał w roku 1988 swój komercyjny system obliczeń symbolicznych i numerycznych. *Mathematica* wciągu ostatnich 20 lat zdobyła liczne grono zwolennników i rozwijana jest po dziś dzień. Charakteryzują ją wysoka wydajność, szerokie możliwości wizualizacji i prezentacji danych oraz przenośność. Używa ona własnego języka funkcyjnego nazwanego za nazwiskiem autora (i właściciela firmy) - *Wolfram language*. Wolfram zaprojektował najogólniejszy jak to tylko możliwe język z wyszczególnieniem obliczeń symbolicznych, programowania funkcyjnego oraz programowania opartego na regułach. Został zbudowany do reprezentowania dowolnych struktur i danych. Język jest bar-

dzo obszerny, zahaczając o wiele, często specjalistycznych, dziedzin. Na przykład, zawiera on wbudowane funkcje do tworzenia i uruchamiania maszyny Turinga, tworzenia grafiki i dźwięku, analizy modeli 3D i rozwiązywania równań różniczkowych. Ma również dużą ilość dokumentacji.

Język Wolfram oraz Mathematica są dostępne w pakiecie z oprogramowaniem systemu zainstalowanego na każdym Raspberry Pi. Edison firmy Intel, wprowadzony na targach CES 2014, również integruje się językiem Wolfram [Upt14] [Wal01].

4.2. Wybrane narzedzia

4.2.1. Narzędzia zarządzania projektem

Przygotowując się do implementacji konieczne było dobranie odpowiedniego zestawu narzędzi wspomagających zarządzanie projektami. Zdecydowałam się na realizację samej pracy w LaTeXu. Korzystałam z edytora online *ShareLaTeX* [Upt15], a kod przechowywałam lokalnie oraz na repozytorium Github [TPW15].

1. LaTeX to oprogramowanie do zautomatyzowanego składu tekstu, a także związany z nim język znaczników, służący do formatowania dokumentów tekstowych i tekstowo-graficznych (na przykład: broszur, artykułów, książek, plakatów, prezentacji, a nawet stron HTML). W istocie LaTeX nie jest samodzielnym środowiskiem programistycznym, a jedynie zestawem makr stanowiących nadbudowę dla systemu składu TeX, automatyzujących wiele czynności związanych z procesem poprawnego składania tekstu.

2. ShareLaTeX jest edytorem LaTeX online, który pozwala na współpracę w czasie rzeczywistym i kompilowaniu projektów online do formatu PDF. W przeciwieństwie do podobnych usług, ShareLaTeX wymaga rejestracji, a wszystkie projekty są z natury prywatne. Po wykupieniu oferty Premium, strona oferuje bardziej zaawansowane funkcje, takie jak więcej niż dwóch współpracowników, historia zmian i synchronizacja z Dropbox i GitHub. ShareLaTeX wykorzystuje Node.js, napisane jest w CoffeeScript, z danymi przechowywanymi w MongoDB i Redis. ShareLaTeX jest w stanie zintegrować R oraz ma zintegrowany pakiet Knitr.

3. GitHub jest internetowym serwisem hostingowym przeznaczonym dla projektów programistycznych wykorzystujących system kontroli wersji Git. Stworzony został przy wykorzystaniu języków Ruby on Rails i Erlang. Github udostępnia darmowy hosting programów open source oraz płatne prywatne repozytoria. Kilka z dostępnych funkcji githuba to: bugtracker, pull request, fork, statystyki czy wiki.

4.2.2. Technologie i biblioteki

Założeniem projektu była realizacja projektu w języku javascript przy wykorzystaniu biblioteki matematycznej wspomagającej obliczenia i wspierającej rachunki macierzowe. Po przetestowaniu kilku dostępnych rozwiazań doszłam do wniosku, że najlepszym rozwiązaniem będzie stosunkowo młoda biblioteka math.js [dJ13]. W trakcie realizacji projektu wystąpiło kilka przeszkód, które utwierdziły mnie w przekonaniu, że jedna biblioteka będzie niewystarczająca na moje potrzeby. Rozwiązaniem mogły być dalsze poszukiwania biblioteki matematycznej, skorzystanie z dwóch bibliotek lub napisanie części funckjonalności od zera. Przeszukując fora i blogi tematyczne dowiedziałam, się że JS nie wspiera matematyki tak mocno jak inne języki, przez co znalezienie idealnej biblioteki może być niemożliwe. Uznałam także, że nie ma sensu wymyślać na nowo koła dlatego sięgnęłam po numeric.js, która uzupełnia funkcjonalności mathjs o kilka obliczenia na macierzach [Loi13]. Moim IDE jest WebStorm.

1. Javascript to dynamiczny język programowania komputerowego. Najczęściej jest używany jako część przeglądarek internetowych, który pozwana na interakcję z użytkownikiem po stronie klienta, kontrolować przeglądarkę, komunikację asynchroniczną i zmian w treści dokumentu, która jest wyświetlana. Jest również stosowany w programowaniu sieciowym po stronie serwera z środowiskami wykonawczymi, takimi jak Node.js, tworzeniu gier i tworzeniu aplikacji mobilnych i stacjonarnych.

2. Math.js jest obszerną biblioteką matematyczną dla JavaScript i Node.js. Zawiera elastyczny parser wyrażeń i oferuje zintegrowane rozwiązanie do pracy z liczbami, dużymi liczbami, liczbami zespolonymi, jednostkami i macierzami. Jest kompatybilny z wbudowaną biblioteką Math JavaScriptu. Obsługuje łańcuchy operacji, pochodzi z dużym zestawem wbudowanych funkcji i stałych. Działa na każdym silniku JavaScript i może być używany jako aplikacja wiersza poleceń. Jest łatwo rozszerzalny. Biblioteka znajduja się na licencji apache: `http://www.apache.org/licenses/LICENSE-2.0`

3. Numeric.js to biblioteka numeryczna Javascript pozwala na wykonywanie skomplikowanych obliczeń numerycznych w czystym JavaScript w przeglądarce, jak i poza nią. Numeric.js jest biblioteką, która oferuje wiele przydatnych funkcji dla algebry liniowej.

4. Na środowisko programistyczne wybrałam produkt firmy JetBrains, autorów popularnego IDE Intellij. WebStorm to lekkie, ale potężne IDE, doskonale wyposażone do kompleksowego rozwoju po stronie klienta i po stronie serwera wraz z Node.js. Zdecydowałam się na korzystanie z 30-dniowej wersji testowej [tea15].

5. Do testów wykorzystałam darmowy framework Bootstrap firmy Twitter. Jest to potężny framework do budowania w pełni funkcjonalnych i reponsywnych stron na przeglądarki oraz urządzenia mobilne. Same funkcjonalności biblioteki sprawdzałam za pomocą wbudowanej w Chrome konsoli javascript [Twi15].

4.3. Implementacja

Realizacja pracy polegała na zagłebieniu się w tematykę metod rankingowych, dokładnym zrozumieniu zasad jakimi się kierują. Następnie konieczne było przeniesienie istniejących funkcji z paczki doktora Kułakowskiego do javascript. Poniżej przedstawiłam listę dostępnych funkcjonalności wraz z ich opisem. Następnie umieściłam kilka przykłądowych funkcji w obu językach dla

1. PrincipalEigenValue(M) – zwraca największą wartość własną macierzy.
2. PrincipalEigenVector(M) – zwraca wektor własny dla najwięszej wartości własnej macierzy.
3. SaatyIdx(M) – zwraca współczynnik nieścisłości Saaty'ego wyliczony z macierzy.
4. EigenvalueRank(M) – zwraca przeskalowany wektor własny macierzy, tak by suma jego elementów równa była 1.
5. GeometricRank(M) – zwraca ranking podany jako średnia geometryczna rzędów macierzy.
6. GeometricRescaledRank(M) – zwraca przeskalowany ranking, w taki sposób, że suma wejść równa jest 1.
7. HREConstantTermVectorEntry(M,v) – zwraca stały, określony wektor dla metody HRE. M to macierz porónań, a v to wektor znanych alternatyw, gdzie nienzane alternatywy mają wartość 0.
8. HREMatrix(M,v) – zwraca macierz dla metody HRE. Razem z porpzednią funkcją b=HREConstantTermVectorEntry(M,v) tworzą równanie liniowe Au = b, gdzie u to częściowy wektor rankingowy. M to macierz porównań, a v to wektor znanych alternatyw, gdzie nienzane alternatywy mają wartość 0.
9. HREPartialRank(M,v) – zwraca wartość nieznanych alternatyw. M to macierz porównań, a v to wektor znanych alternatyw, gdzie nieznane alternatywy mają wartość 0.

10. HREFullRank(M,v) – zwraca wartość dla znanych i nienznanych alternatyw. M to macierz porównań, a v to wektor znanych alternatyw, gdzie nienzane alternatywy mają wartość 0.

11. HRERescaledRank(M,v) – zwraca przeskalowany ranking z HREFullRank, tak by suma wejść równała się 1.

12. HREGeomConstantTermVector(M,v) – zwraca stały, określony wektor dla metody HRE. M to macierz porównań, a v to wektor znanych alternatyw, gdzie nienzane alternatywy mają wartość 0.

13. HREGeomMatrix(M,v) – zwraca macierz dla geometrycznej metody HRE. Razem z b=HREGeomConstantTermVector(M,v) tworzą równanie liniowe Au = b, gdzie u to częściowy wektor rankingowy. M to macierz porównań, a v to wektor znanych alternatyw, gdzie nienzane alternatywy mają wartość 0.

14. HREGeomIntermediateRank(M,v) – zwraca wartości nieznanych alternatyw przed podniesieniem ich do 10 potęgi. M to macierz porównań, a v to wektor znanych alternatyw, gdzie nienzane alternatywy mają wartość 0.

15. HREGeomPartialRank(M,v) – zwraca wartości nieznanych alternatyw. M to macierz porównań, a v to wektor znanych alternatyw, gdzie nienzane alternatywy mają wartość 0.

16. HREGeomFullRank(M,v) – zwraca wartość nieznanych alternatyw. M to macierz porównań, a v to wektor znanych alternatyw, gdzie nieznane alternatywy mają wartość 0.

17. HREGeomRescaledRank(M,v) – zwraca przeskalowany pełną listę rankingową dla HRE, w taki sposób, że suma wejść równa jest 1.

18. KoczkodajTriadIdx([a,b,c]) – zwraca trójkę niepewności Koczkodaja, czyli Min(Abs(1-c/ab),Abs(1-ab/c)).

19. KoczkodajIdx(M) – zwraca wartość niepewności dla macierzy M.

20. KoczkodajTheWorstTriad(M) – zwraca najgorszą trójkę w macierzy w rozumieniu niepewności Koczkodaja.

21. KoczkodajTheWorstTriad(M,n) – zwraca najgorsze trójki w macierzy w rozumieniu Koczkodaja.

22. KoczkodajConsistentTriad([a,b,c]) – zwraca najbliższą trójkę pewności.

23. KoczkodajImproveMatrixStep(M) – zwraca ulepszoną macierz, w której największa niepewność jest zastąpiona przez najbliższą pewną.

24. COP1ViolationList(M,r) – zwraca listę indeksów naruszających pierwszą zasadę zachowania porządku (COP, Bana e Costa i Vansnick). M jest macirzą, a r lista rankingową.

25. COP2ViolationList(M,r) – zwraca listę wskaźnikó, które naruszają drugą zasadę o zachowaniu porządku (COP, Bana e Costa i Vansnick). M jest macirzą, a r lista rankingową.

26. ErrorMatrix(M,mju) – oblicza macierz mającą dane wejściowe w postaci $e_{ij} = \frac{m_{ji} * mju(c_i)}{mju(c_j)}$. Kiedy macierz jest pewna wszystkie $e_{ij} = 1$.

27. LocalDiscrepancyMatrix(M,mju) – oblicza macierz z wejściami $d_{ij} = \max(e_{ij} - 1, \frac{1}{e_{ij}} - 1)$.

28. GlobalDiscrepancy(M,mju) – zwraca maksymalne wejście z poprzedniej funkcji.

29. RecreatePCMatrix(M) – odtwarza macierz „reciprocal" na podstawie macierzy trójkątnej górnej z M.

30. RankOrder(rankList) – wyświetla listę rankingową.

Poniżej przedstawiłam porównanie kilku funkcji, w celu pokazania różnic składniowych obu bibliotek. Całość znajduje się w pliku pcm.js a wyświetlam je w pliku index.html wspieranym przez style w bootstrapie, w części <*head*> załączyłam także obie biblioteki matematyczne.

Metoda obliczająca maksymalną wartość własną macierzy:

```
PrincipalEigenValue[matrix_] := Max[Abs[N[Eigenvalues[matrix]]]]
```

```
PCMethod.PrincipalEigenValue = function(M) {
    return math.max(numeric.eig(M).lambda.x);
};
```

Metoda do wyliczenia wektora własnego dla ww wartości własnej:

```
PrincipalEigenVector[matrix_] :=
 Extract[N[Eigenvectors[matrix]],
  First[First[
    Position[Abs[N[Eigenvalues[matrix]]],
     PrincipalEigenValue[matrix]]]]]
```

```
PCMethod.PrincipalEigenVector = function(M) {
    var eigval = PCMethod.PrincipalEigenValue(M);
    var values = numeric.eig(M).E.x;
    var i =0;
    do {
        if (values[i] = eigval){
            return (numeric.eig(M).E.x[i]);
        }
        i++;
    } while (i <= values.length || values[i-1] == eigval);
};
```

Funkcja wyznaczająca indeks niespójności Saaty'ego:

```
SaatyIdx[matrix_] := With[{n = First[Dimensions[matrix]],
alpha = PrincipalEigenValue[matrix]}, (alpha - n)/(n - 1)]
```

```
PCMethod.SaatyIdx = function(M){
    var n = M.length;
    var alfa = PCMethod.PrincipalEigenValue(M);
    return (alfa -n)/(n -1);
};
```

Przeskalonay wektor własny:

```
EigenvalueRank[matrix_] := With[{pev =
PrincipalEigenVector[matrix]}, (#/Apply[Plus, pev]) & /@ pev]
```

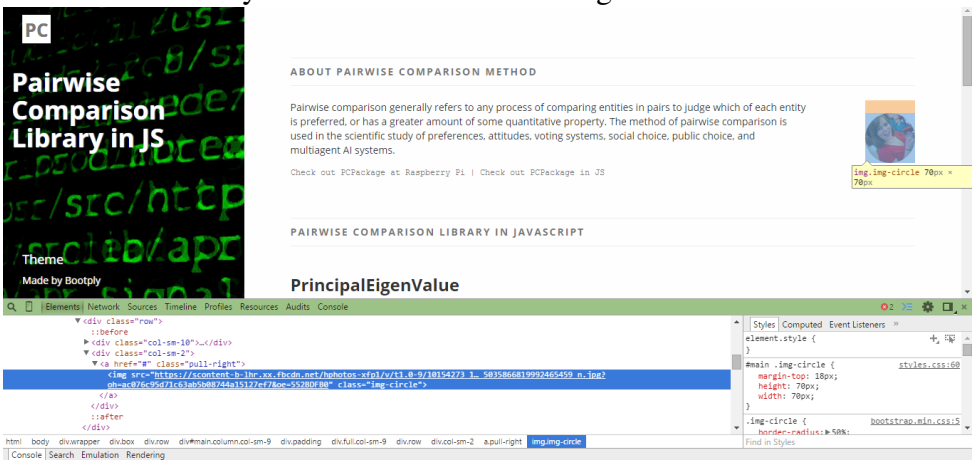
```
PCMethod.EigenvalRank = function(M){
    var eigvec = PCMethod.PrincipalEigenVector(M);
    var sum = math.sum(eigvec);
```

```
    for (var i=0; i < eigvec.length; i++ ){
        eigvec[i] = eigvec[i]/sum;
    }
};
```

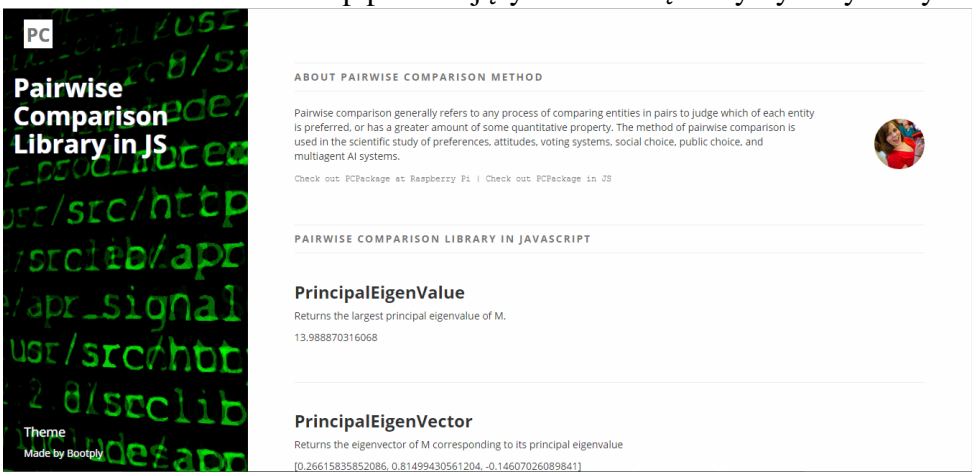
5. Testy

Chcąc zweryfikować poprawność wykonywanych zadań sięgnęłam po prosty szablon napisany dla bootstrapa i przystosowałam go do potrzeb projektu (rysunek 5.2). Dopilnować wyglądu pomogła mi wbudowana konsola dla Google Chrome (rysunek 5.1).

Rysunek 5.1: Konsola w Google Chrome.

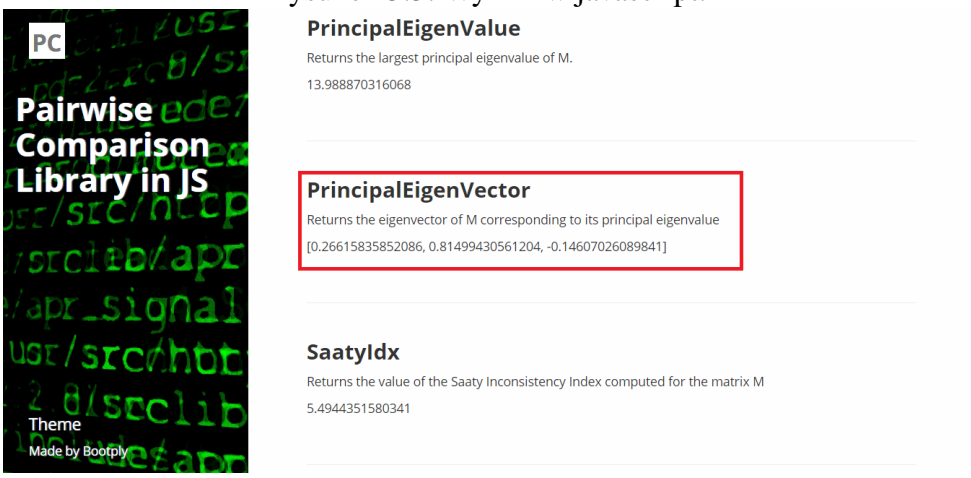


Rysunek 5.2: Szablon bootstrap pozwalający na kontrolę otrzymywanych wyników.

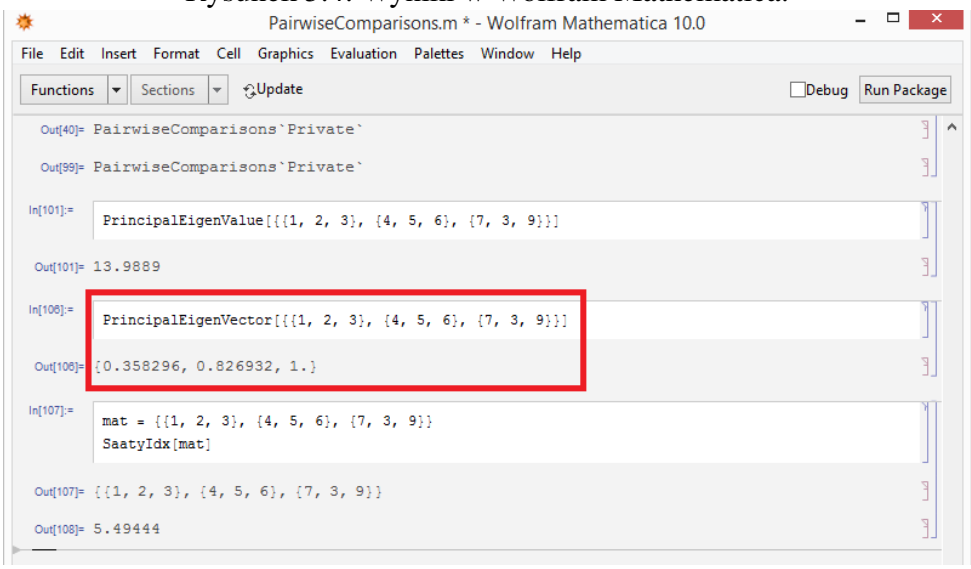


W obu przypadkach użyłam tej samej macierzy M. Wyświetlałam na stronie każdą funkcję i kontrolnie porównywałam je z wynikami jakie dawała Mathematica. W pewnym momencie wyniki zaczęły się różnić i po dogłębniejszej analizie doszłam do wniosku, że winą tych różnic jest inna precyzja reprezentacji danych (rysunek 5.3 i 5.4). Mathematica nastawiona jest na szczegółowe obliczenia symboliczne, matematyczne, tworzenie grafik 3D itd. Javascript natomiast z założenia jest językiem wprowadzonym na potrzeby stron WWW, mającym ułatwić interakcję użytkownika z zawartością strony, stąd mniejsze dopracowanie matematycznych aspektów.

Rysunek 5.3: Wyniki w javascript.



Rysunek 5.4: Wyniki w Wolfram Mathematica.



6. Podsumowanie i wnioski

Na koniec chciałabym krótko podsumować całość pracy nad biblioteką Pairwise Comparisons. Mimo dość gruntownego przestudiowania podstaw wieloktryterialnych metod decyzyjnych nadal pozostał problemem fakt doboru odpowieniej biblioteki dla javascript oraz sama reprezentacja liczb w tym języku. Odpowiednie dobranie koniecznych narzędzi na pewno przyspieszyło prace nad realizacją projektu. Sądzę, że z pomocą tych funkcji będzie można w przyszłości stworzyć aplikację, która w sposób bardziej wizualny i interaktywny pomagałoby studentom w nauce metod rankingowych, czy decydentom w priorytetyzowaniu alternatyw.

W przyszłości można poszukać lepszego rozwiązania bibliotek js lub zaimplementować brakujące funckje od podstaw. Używanie dwóch bibliotek w tym przypadku jest mało eleganckie. Inną kwestią jest fakt, że ten projekt powinien być traktowany jako materiał pokazowy, a do poważniejszych obliczeń nadal polecam wersję w języku Wolfram.

Chciałabym podziękować doktorowi Kułakowskiemu za cierpliwość i pomoc w realizacji tej pracy. Bez niego całość na pewno przysporzyłaby wielu kłopotów.

Bibliografia

[AGH14] AGH. *"By Example" introduction into the Mathematica PairwiseComparisons' Package*, 2014.

[B.81] Roy B. The optimisation problem formulation: Criticism and overstepping. *The Journal of the Operational Research Society*, 36(6):427–436, 1981.

[dJ13] Jos de Jong. Math.js, 2013.

[IA13] Nemery P. Ishizaka A. *Multi-criteria Decision Analysis: Methods and Software*. Wiley, New Delhi, 2013.

[Loi13] Sébastien Loisel. Numeric.js, 2013.

[R.92] Keeney R. *Value-Focused Thinking: A Path to Creative Decision Making*. MA: Harvard University Press, Cambridge, 1992.

[tea15] JetBrain team. Webstorm by jetbrain, 2015.

[TPW15] PJ Hyett Tom Preston-Werner, Chris Wanstrath. Github pairwise comparison library, 2015.

[Twi15] Twitter. Boostrap by twitter, 2015.

[Upt14] Liz Upton. Raspberry pi - blog, 2014.

[Upt15] Liz Upton. Sharelatex - online latex editor, 2015.

[Var10] Ricardo Vargas. Using the analytic hierarchy process (ahp) to select and prioritize projects in a portfolio, 2010.

[Wal01] Jimmy Wales. Wikipedia wolna encyklopedia, 2001.

[WJ11] Zionts S. Wallenius J. *Multiple Criteria Decision Making: From Early History to the 21st Century*. World Scientific, Singapur, 2011.