Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki i Telekomunikacji

KATEDRA INFORMATYKI

DOKUMENTACJA PROCESOWA

TOMASZ KASPRZYK, DANIEL OGIELA, JAKUB STĘPAK

SYSTEM OBLICZAJĄCY WYNIKI WYBORÓW DLA UOGÓLNIENIA SYSTEMU K-BORDA

PROMOTOR:
dr hab. inż. Piotr Faliszewski

Kraków 2016

Spis treści

1. Cele projektu

Celem projektu jest stworzenie systemu obliczającego wyniki wyborów dla uogólnienia opisanego poniżej systemu k-Borda. Stworzona aplikacja webowa ma pozwalać użytkownikowi na szybkie definiowanie wyborów, jak również importowanie istniejących danych, w celu uzyskania wyników. Jako wybory rozumiemy nie tylko te, w których głosujący wyłaniają swoich przedstawicieli w organach władzy - swoje preferencje można także określić dla, na przykład, filmów jakie chcemy obejrzeć podczas seansu z przyjaciółmi, w jaką grę chcielibyśmy zagrać, jaką restaurację wybrać na rodzinne spotkanie itp. Ustalenie preferencji to jedno, ale o tym, kto lub co zostanie wybrane na podstawie preferencji wszystkich głosujących, decyduje również system wyborczy. Określając odpowiednie parametry aplikacja ma za zadanie ilustrować jak, w zależności od parametrów, zmieniają się wyniki wyborów, przy tym samym zestawie preferencji. W pierwszej kolejności użytkownik określa listę preferencji wyborców. Określając swoje preferencje każdy wyborca porządkuje kandydatów w kolejności od najlepszego do najgorszego. Lista preferencji może zostać wygenerowana losowo, zaimportowana z odpowiednio sformatowanego pliku lub stworzona ręcznie. Na wprowadzonych do systemu listach preferencji i ustalonej liczbie k wyboraców użytkownik może wielokrotnie uruchamiać algorytm wyłaniania zwycięzców. Przed każdym uruchomieniem algorytmu ustalany jest parametr p normy ℓ_p, który w głównej mierze (pomijając listę preferencji wszystkich głosujących) decyduje o wyniku obliczeń.

2. Studium wykonalności

2.1. Opis stanu istniejącego

Tworzony system nie posiada swojego pierwowzoru.

2.2. Opis wymagań

2.2.1. Wymagania funkcjonalne

Ponieważ zastosowano ewolucyjny proces tworzenia oprogramowania, więc wymagania funkcjonalne zbierane były na bieżąco w trakcie trwania projektu. Dla kolejnych wersji systemu dodawano nowe wymagania, bądź nieco zmieniano już istniejące. Wymagania były dodawane lub zmienianie na bazie rozmów z klientem i rozwoju prac. Poniżej przedstawiono wymagania dla kolejnych wersji systemu oraz wymagania przedstawione na początku przez klienta.

Wymagania przedstawione przez klienta na pierwszym spotkaniu:

- Obliczanie wyników wyborów przez zaimplementowane w tym celu algorytmy heurystyczne. Definicja wyborów została ściśle określona.
- Przystępne przedstawienie wyników wyborów

Wymagania dla 1-szej wersji systemu:

- Wczytanie wyborów do systemu z pliku formatu $.soc$
- Obliczanie wyników wyborów przez algorytm typu brute-force
- Obliczanie normy ℓ_p

Wymagania dla 2-giej wersji systemu:

- Przeniesienie wcześniej dodanych funkcjonalności do aplikacji webowej
- Generowanie wyborów z rozkładu normalnego
- Logowanie użytkowników na swoje konto
- Tworzenie i usuwanie wyborów

- Wyświetlanie wyborów i ich wyników dla wyborów wygenerowanych z rozkładu normalnego

Wymagania dla 3-ciej wersji systemu:

- Obliczanie wyników wyborów algorytmem zachłannym
- Obliczanie wyników wyborów algorytmem genetycznym

Wymagania dla 4-tej wersji systemu:

- Wydajniejsze obliczanie wyników wyborów przez algorytm zachłanny i genetyczny
- Możliwość parametryzowania z poziomu użytkownika algorytmu genetycznego
- Obliczanie wyników wyborów za pomocą algorytmu zachłannego według zasady $Chamberlin - Courant'a$
- Poprawienie użyteczności systemu

2.2.2. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne nie zmieniały się zbytnio w trakcie rozwoju projektu.

Wymagania produktowe

- Szybkość i dokładność wykonywanych obliczeń - obliczanie wyników wyborów w systemie $k - Borda$ jest czasochłonne. Zadaniem projektowym jest opracowanie algorytmów, które pozwolą na szybsze otrzymywanie wyników wyborów kosztem ich przybliżenia. Stworzone algorytmy powinny być użyteczne dla możliwie dużych danych.

Wymagania organizacyjne

- Dotrzymanie terminów na przedstawienie poszczególnych elementów pracy - realizacja projektu inżynierskiego odbywa się według trybu ustalonego przez władze uczelni. Pierwszym aspektem są specjalnie wydzielone przedmioty - $PracowniaProjektowa1$ i $PracowniaProjektowa2$ - które wspomagają i motywują studentów do wykonywania poszczególnych części projektu w wyznaczonym czasie. Na zajęciach seminaryjnych przedstawiano w określonych terminach kolejno: Wizję produktu, Studium wykonalności i prezentację kolejnych prototypów. Wyznaczono również termin na oddanie dokumentacji i przedstawienie produktu.

2.3. Strategia testowania

Poprawność działania podstawowych funkcjonalności zapewnianych przez system gwarantują testy jednostkowe. Pisane były one na bieżąco po dodawaniu kolejnych możliwości i udogodnień systemu.

W celu zagwarantowania poprawności działania wcześniej dodanych funkcjonalności, po wykonaniu i wdrożeniu nowych funkcjonalności systemu, skorzystano z praktyki ciągłej integracji. Zrealizowano ją za pomocą serwisu Travis CI. Serwis zapewnia automatyczne wykonanie przygotowanych testów jednostkowych, po każdej wprowadzonej zmianie do repozytorium kodu źródłowego

2.4. Aspekt technologiczny

Do stworzenia oprogramowania wykorzystano język $Python$ 2.7. System wykonany jest w formie aplikacji internetowej. Do zrealizowania aplikacji webowej wykorzystano framework dla języka programowania $Python$ - $Django$ 1.9. $Django$ dostarcza w pełni funkcjonalny system uwierzytelniania, zapewnia obsługę kont, grup oraz uprawnień użytkowników. Oprócz zaoszczędzenia czasu na realizacji powyższych, nieistotnych dla realizacji tematu projektu, funkcjonalności systemu, głównym powodem było również doświadczenie w tworzeniu aplikacji z użyciem $Django$ posiadane przez jednego z członków zespołu realizującego niniejszy projekt.

3. Analiza ryzyka

3.1. Identyfikacja zagrożeń

Spośród wielu czynników mogących mieć wpływ na przedłużanie się czasu wykonania poszczególnych elementów systemu za najważniejsze można uznać:

1. Problem ze stworzeniem satysfakcjonującego algorytmu heurystycznego. Zaprojektowanie i implementacja algorytmów heurystycznych to najważniejsze i największe wymaganie dla tego projektu. W związku z tym, realizację tego wymagania może utrudnić wiele nieprzewidzianych czynników.

2. Potencjalny problem z czasem trwania obliczania normy ℓ_p dla dużych wartości parametru p . Jest to jeden z czynników mogących wpłynąć na stworzenie wydajnego algorytmu heurystycznego. Jeżeli ten problem wystąpi, utrudni to stworzenie satysfakcjonującego algorytmu heurystycznego.

3. Koordynacja prac zespołu. Praca w trzyosobowej grupie może nieść ze sobą zarówno korzyści jak i problemy. Konieczny jest wyraźny podział zadań - pomocne okażą się narzędzia do kontrolowania podziału pracy.
   Należy liczyć się również z problemami z synchronizacją czasową. Różne plany zajęć członków zespołu (różne przedmioty obieralne i związane z tym obciążenie czasowe w danej części semestru), praca zawodowa członków zespołu, konieczność odbycia staży wakacyjnych, pobyt poza Krakowem (brak możliwości spotkania się we trójkę).

3.2. Analiza zagrożeń

Zagrożenie	**Prawdopodobieństwo**	**Konsekwencje**	**Strategia**
Problem ze stworzeniem satysfakcjonującego algorytmu heurystycznego	Duże	Poważne	Szybkie stworzenie bazy systemu i skupienie się na głównym zadaniu projektu
Czas trwania obliczania normy ℓ_p dla dużych p	Duże	Poważne	Zaoszczędzenie czasu działania algorytmu poprzez minimalizację operacji na bazie danych
Koordynacja prac zespołu - problem z przydziałem zadań i czasem ich wykonania	Duże	Znośne	Zastosowanie systemu przydziału zadań i częste wspólne wewnętrzne spotkania

Tablica 3.1: Analiza zagrożeń

4. Przyjęta metodyka pracy

4.1. Tworzenie oprogramowania

Proces tworzenia oprogramowania dla tego projektu inżynierskiego posiadał więcej lub mniej cech wielu metod tworzenia oprogramowania. Najwięcej wspólnego miał z ewolucyjnym procesem tworzenia oprogramowania. Można tak uznać głównie ze względu na początkowo ogólne, niedoprecyzowane wymagania klienta. Właściwie jedynym wymaganiem dla systemu było stworzenie algorytmów heurystycznych do obliczania wyborów zdefiniowanych według ściśle określonych zasad. Cała otoczka systemu pozostała w gestii członków zespołu. W trakcie kolejnych etapów projektu często kontaktowano się z klientem i wymagania doprecyzowywano.

Wątpliwość co do pełnego uznania tworzenia tego systemu jako ewolucyjnego procesu tworzenia oprogramowania, może pojawić się przejściach między kolejnymi wersjami systemu. Część tych przejść, to rzeczywiście powstanie zupełnie nowej wersji systemu jak na przykład przejście z aplikacji konsolowej do aplikacji webowej. Część przejść dotyczyła jednak dodania do aktualnej wersji systemu nowych funkcjonalności (bardzo poważnie ulepszających system, lecz wciąż bardziej uzupełniających poprzednią wersję niż tworzącą nową) i miało to cechy przyrostu w przyrostowym modelu tworzenia oprogramowania. Ze względu na klarowność opisu przebiegu prac i zgodność przytłaczającej liczby cech, zdecydowano uznać tworzenie tego systemu jako proces ewolucyjnego tworzenia oprogramowania.

Wykorzystano tworzenie badawcze, które polega na częstym kontakcie z klientem w celu ciągłego badania i weryfikowania wymagań systemu. Opracowano pierwotną wersję systemu, a następnie udoskonalano go w wielu wersjach, aż do uzyskania ostatecznej wersji. Każda kolejna wersja systemu dodawała nową funkcjonalność, bądź przyspieszała działanie wcześniej zrealizowanych funkcjonalności.

4.2. Podział prac

Wykonane czynności przez poszczególnych członków zespołu:

Tomasz Kasprzyk:

- Czynność 1

Daniel Ogiela:

- Czynność 1

Jakub Stępak:

- Czynność 1

4.3. Komunikacja z managerem i klientem

W trakcie prac nad systemem odbywały się spotkania z managerem i klientem. Spotkania z managerem miały formę zajęć seminaryjnych, na których przedstawiano kolejne prezentacje dotyczące pracy inżynierskiej. Przedstawiono wstępną wizję projektu, studium wykonalności oraz prototyp systemu. Oprócz wymienionych artefaktów na zajęciach z managerem regularnie zdawano raport z aktualnego postępu prac.

Spotkania z klientem odbywały się rzadziej i miały formę sprawozdań z aktualnego stanu prac. Klient na podstawie sprawozdań sugerował kolejne czynności i ulepszenia systemu, nad którymi powinni się skupić członkowie zespołu inżynierskiego.

4.4. Wykorzystane narzędzia do zarządzania projektem

Do zarządzania wykonywanymi w ramach projektu zadaniami wykorzystano aplikację internetową $Trello$. Narzędzie służyło zarówno do określania zadań, przydzielania ich do konkretnych członków zespołu, jak i do gromadzenia istotnych dla realizacji projektu informacji. Narzędzie umożliwiało określenie typu zadania oraz wskazanie, na którym etapie wykonywania zadania znajduje się dany członek zespołu.

Do przechowywania kodu źródłowego oprogramowania wykorzystano serwis internetowy $GitHub$. Dostęp do repozytorium kodu źródłowego na każdym etapie procesu tworzenia oprogramowania miał zarówno manager i klient.

W celu zdalnego kontaktowania się pomiędzy członkami zespołu korzystano z oprogramowania $Slack$. Komunikator ten, oprócz podstawowej funkcjonalności prowadzenia konwersacji, pozwala również na transfer plików, przesyłanie fragmentów sformatowanego kodu w postaci $code\ snippets$ oraz wyszukiwanie powyższych w całej historii konwersacji. Korzystano również z mechanizmu botów, m.in. $reminderw$ (cyklicznych przypomnieniach, np. o $standupach$) a także informatorach o zmianach stanu repozytorium czy $build$ serwera.

$Pycharm$, IDE firmy $JetBrains$. W wersji dla studentów udostępniony jest pakiet zaawansowanych funkcjonalności, w szczególności integracja z frameworkiem $Django$. Z poziomu IDE można również aktualizować repozytorium (opcje dostępne w zakładce VCS) W celu przepisywania dokumentacji z $GoogleDocs$ do $LaTeX$ korzystano z edytora $TexMaker$. Funkcja $WYSIWYG$ znacznie przyspiesza proces tworzenia poprawnie sformatowanego dokumentu.

Serwisy $Travis\ CI$ oraz $Coveralls.io$ służyły odpowiednio zdalnemu budowaniu aplikacji i uruchamianiu testów oraz ciągłej integracji zmian wprowadzanych do głównego repozytorium. Serwisy te połączono dla wygody ze $Slackiem$ zespołu projektowego poprzez wykorzystanie wspomnianych wyżej botów.

4.5. Weryfikacja wyników projektu

Jakość obliczanych wyników wyborów weryfikowano za pomocą testów porównawczych oraz oceny wizualnej. Testy porównawcze polegały one na porównywaniu wyników generowanych przez różne algorytmy heurystyczne. Porównanie dotyczyło czasów działania algorytmów oraz zadowoleń wyborców z wyniku wypracowanych przez algorytmy. Ocena wizualna dotyczyła ocenienia wykresów wyników wyborów generowanych przez algorytmy heurystyczne. Sprawdzano czy w miarę wzrostu parametru p punkty reprezentujące zwycięskich kandydatów stopniowo rozpraszały się na mapie spektrum poglądów. Ocena wizualna dotyczyła tylko wyborów generowanych z rozkładu normalnego - tylko dla tych wyborów, wyborcy i kandydaci są reprezentowani jako punkty na płaszczyźnie.

5. Przebieg prac

5.1. Harmonogram

Ze względu na ewolucyjny charakter tworzenia oprogramowania, szczegółowe zadania były wyznaczane na bieżąco. Dostosowywano je do aktualnych wymagań. Poniżej przedstawiono orientacyjny harmonogram określony na początku pracy, który przewidywał czas ukończenia najważniejszych elementów systemu.

- Maj 2016
  - Implementacja algorytmu typu brute-force
  - Praca nad algorytmem pozwalającym na szybsze wykonywanie obliczeń
- Lipiec 2016
  - Implementacja pierwszych wersji algorytmów heurystycznych
  - Stworzenie interfejsu
  - Importowanie danych z $PrefLib$
  - Symulowanie wyborów z zastosowaniem rozkładu normalnego
- Październik 2016
  - Przyspieszenie algorytmów heurystycznych
- Listopad 2016
  - Przygotowanie dokumentacji
  - Udoskonalenie interfejsu

5.2. Opis zebrania podstawowych wymagań klienta i podjęcie decyzji projektowych

5.2.1. Czas trwania

Marzec 2016

5.2.2. Zadania wyznaczone w tej fazie

Głównym zadaniem w tej fazie procesu tworzenia oprogramowania było zebranie podstawowych wymagań klienta i ich dokładne przeanalizowanie. Na bazie specyfikacji wymagań należało wybrać odpowiednie technologie do realizacji produktu.

5.2.3. Opis przebiegu prac

W celu wykonania wyżej wymienionych zadań odbyto spotkanie z przyszłym klientem. Na spotkaniu poruszono zagadnienia dotyczące obszaru zainteresowań klienta, które były bezpośrednio związane z tematem projektu inżynierskiego. Klient przekazał podstawowe wymagania odnośnie przyszłego systemu. Na bazie uzyskanych informacji i materiałów przystąpiono do dokładnego przestudiowania dziedziny problemu. Po wnikliwej analizie podjęto główne decyzje projektowe. Przygotowano również prezentację na zajęcia z managerem opisującą zadany projekt inżynierski.

5.2.4. Wynik

- Lista początkowych wymagań funkcjonalnych i niefunkcjonalnych systemu
- Wybór technologii $Python\ 2.7$
- Opis projektu inżynierskiego w postaci prezentacji Wizji produktu

5.3. Opis kolejnych wersji systemu

5.3.1. Wersja 1.

Czas trwania

Kwiecień 2016

Cele wyznaczone do osiągnięcia

Jednym z celów do osiągnięcia w pierwszej wersji systemu (niespełniającej wszystkich wymagań) było zaimplementowanie wczytywania do systemu preferencji wyborców otrzymanych z pliku odpowiedniego formatu. Drugim ważnym celem była implementacja algorytmu typu brute-force do obliczania wyników wyborów. Implementacja wymienionych zadań poza dodaniem funkcjonalności systemowi miała posłużyć dookreśleniu wymagań systemu i poznaniu specyfiki rozwiązywanego problemu. Algorytm typu brute-force miał również przydać się w późniejszej fazie projektu do testów porównawczych docelowego algorytmu dla małych rozmiarów danych wejściowych.

Opis przebiegu prac

Prace rozpoczęto od implementacji wczytywania wyborów z pliku z rozszerzeniem $.soc$. Następnie dodano liczenie normy ℓ_p oraz algorytmu typu brute-force pozwalającego na obliczanie wyników wyborów dla prostych danych. Z czasem stwierdzono, że wczytywanie danych z pliku jest niewystarczające dla testowania dużych danych, stąd rozpoczęto prace nad generowaniem wyborów z rozkładu normalnego. Zdecydowano o możliwości przedstawiania wyborców i kandydatów jako punkty na płaszczyźnie. Preferencje wyborcze danego wyborcy zdecydowano ustalać na podstawie odległości euklidesowych punktu przedstawiającego tego wyborcę do punktów przyporządkowanych kandydatom do komitetu. W dalszych pracach dodano również walidację poprawności pliku wejściowego oraz zaobserwowano ko-

nieczność zaprojektowania obiektowej struktury danych występujących w systemie. W związku z tym stworzono diagram klas.

Wynik

- Pierwsza wersja systemu uruchamiana z poziomu konsoli $Pythona$ przyjmująca za argumenty parametr p konieczny do obliczania normy ℓ_p oraz plik z rozszerzeniem $.soc$.
- Dodane funkcjonalności: wczytywanie wyborów z pliku z rozszerzeniem $.soc$, algorytm typu brute-force, walidacja danych z pliku wejściowego
- Rozpoczęcie prac nad generowaniem własnych wyborów
- Diagram klas
- Podjęcie decyzji o wykonaniu systemu w postaci aplikacji webowej i wykorzystaniu do tego frameworka $Django$ 1.9

5.3.2. Wersja 2.

Czas trwania

Maj 2016

Cele wyznaczone do osiągnięcia

Głównym celem dla tej wersji systemu było przeniesienie zrealizowanych wcześniej funkcjonalności do stworzonej na bazie frameworka $Django$ 1.9 aplikacji webowej. Celem było również dokończenie prac nad generacją własnych wyborów, w szczególności dodanie możliwości generowania preferencji w obiektowo zorientowany sposób.

Opis przebiegu prac

Prace rozpoczęto od stworzenia szkieletu aplikacji webowej generowanego przez skrypt $Django$ 1.9. Następnie dodawano kolejne funkcjonalności systemu: możliwość rejestracji i logowania się użytkowników, tworzenie wyborów, usuwanie wyborów czy wyświetlanie dla danego użytkownika stworzonych wyborów. Stworzono modele pozwalające na zapisanie obiektów w bazie danych. Równolegle dodano możliwość generowania wyborów, które odzwierciedlały obiektową strukturę danych. Na bazie tego i stworzonych funkcjonalności w poprzedniej wersji systemu dodano do aplikacji webowej możliwości: wczytywania wyborów z pliku, walidację danych z pliku wejściowego, generowanie wyborów z rozkładu normalnego oraz wyświetlanie preferencji wszystkich wyborców. Do kodu w repozytorium dodano ciągłą integrację za pomocą narzędzia $Travis\ CI$. W dalszej kolejności zajęto się umożliwieniem w aplikacji webowej obliczania wyników wyborów algorytmem typu brute-force. Równolegle rozpoczęto pracę nad wyświetlaniem wyborców i kandydatów reprezentowanych jako punkty na wykresie $2D$ (dla wyborów generowanych z rozkładu normalnego). Po ukończeniu tych zadań dodano funkcjonalność wyświetlania wyników wyborów na wykresie $2D$. W międzyczasie skonfigurowano wdrożenie systemu

na platformie $Heroku$. W ramach zajęć seminaryjnych wykonano studium wykonalności, w którym podjęto szereg decyzji projektowych.

Wynik

- Aplikacja webowa wdrożona na platformę $Heroku$
- Przeniesione funkcjonalności z poprzedniej wersji systemu do aplikacji webowej
- Dodane nowe funkcjonalności: rejestracja i logowanie użytkowników, generacja wyborów z rozkładu normalnego, tworzenie oraz usuwanie wyborów, wyświetlanie dla danego użytkownika stworzonych wyborów, wyświetlanie kandydatów i wyborców reprezentowanych jako punkty oraz wyników wyborów na wykresach $2D$ (dla wyborów generowanych z rozkładu normalnego)
- Studium wykonalności

5.3.3. Wersja 3.

Czas trwania

Koniec maja 2016 - czerwiec 2016

Cele wyznaczone do osiągnięcia

Głównym celem wyznaczonym dla tej wersji systemu było stworzenie pierwszych wersji dwóch algorytmów heurystycznych. Pierwszym z nich jest algorytm genetyczny. Drugim z nich jest algorytm zachłanny. Ponadto zdecydowano się skupić nad operacjach odczytu i zapisu do bazy danych w celu potencjalnego ograniczenia liczby ich wykonania.

Opis przebiegu pracy

Rozpoczęto od ograniczenia liczby operacji odczytu preferencji wyborczych z bazy danych przy wykonywaniu algorytmu typu brute-force. Zredukowano liczbę tych operacji poprzez jednokrotne wczytanie do pamięci preferencji wyborczych każdego wyborcy. Następnie jeszcze bardziej zredukowano liczbę odwołań do bazy danych poprzez wczytywanie preferencji danego wyborcy do pamięci od razu przy wczytywaniu z bazy danych tego wyborcy. Po wykonaniu zadania redukcji operacji odczytu danych z bazy danych, przystąpiono do równoległej realizacji algorytmu genetycznego i algorytmu zachłannego. W ramach zajęć seminaryjnych wykonano prezentację, w której pokazano funkcjonalność aktualnej wersji systemu.

Wynik

- Dodanie funkcjonalności obliczania wyników wyborów za pomocą algorytmu genetycznego i zachłannego
- Czas działania algorytmu zachłannego wyraźnie krótszy od algorytmu typu brute-force
- Czas działania algorytmu genetycznego porównywalny z czasem działania algorytmu typu brute-force

- Redukcja operacji odczytu danych z bazy danych przy obliczaniu wyników wyborów algorytmem typu brute-force

5.3.4. Wersja 4. - końcowa

Czas trwania

Sierpień 2016 - Grudzień 2016

Cele wyznaczone do osiągnięcia

Najważniejszym celem dla tej wersji systemu jest maksymalne przyspieszenie działania algorytmu genetycznego i zachłannego. Kolejnym zadaniem jest ułatwienie porównywania rezultatów obliczeń (w szczególności: czasu wykonywania obliczeń oraz wyniku zwycięskiego komitetu) dla różnych algorytmów. Ponadto skupiono się na ulepszeniu dynamicznego wyświetlania wyników wyborów.

Opis przebiegu pracy

Algorytmy heurystyczne przyspieszano głównie poprzez eliminację niepotrzebnych operacji na bazie danych. Zmniejszało to czas dostępu do danych i tym samym czas trwania algorytmów. Dla ułatwienia procesu optymalizacji algorytmów konieczne okazało się umożliwienie parametryzacji algorytmów (w szczególności genetycznego) z poziomu interfejsu webowego. Ułatwienie porównywania rezultatów wiązało się z aktualizacją wyglądu listy rezultatów. Dla realizacji celu ulepszenia wyświetlania wyników postanowiono zamienić dotychczasową listę z rozwiązaniami, na suwak pozwalający na szybsze przełączania pomiędzy wykresami wyników. Ważnym aspektem zadania jest odpowiednie posortowanie wyników - najpierw względem algorytmów, następnie według parametru p (tak aby parametr p kolejnego wyniku był nie mniejszy niż poprzedniego).

Wynik

- Dodanie wersji algorytmu zachłannego dla szczególnego przypadku - algorytm $GreedyCC$
- Zastąpienie rozwijanej listy do wyboru wykresów suwakiem
- Dodanie informacji o wyniku punktowym dla każdego elementu listy rezultatów
- Dodanie możliwości usuwania pojedynczego rezultatu

5.4. Problemy napotkane w czasie realizacji projektu

Jednym z napotkanych problemów podczas realizacji systemu był niezadowalający czas liczenia normy ℓ_p . W związku z tym należało znaleźć miejsca w algorytmie, które zaoszczędzałyby czas trwania obliczeń. Znaleziono miejsca, w których program niepotrzebnie wiele razy sięga do bazy danych, zamiast sięgać do pamięci w celu odczytania potrzebnych danych.