Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki i Telekomunikacji

KATEDRA INFORMATYKI

DOKUMENTACJA TECHNICZNA

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. Dziedzina problemu

1.1. Metoda obliczania wyników wyborów

1.1.1. Metoda Bordy

Niech v będzie głosem nad zbiorem kandydatów C. Wynik według Bordy kandydata c w v jest równy $\beta(i) = C - i$, gdzie i- pozycja kandydata w ciągu v. Wynik c w wyborach jest sumą wyników c u każdego z wyborców

1.1.2. Metoda k-Borda

Rozszerzenie metody Bordy. Wynik, zamiast dla jednego kandydata, obliczany jest dla ciągu kandydatów. f_{kB}- funkcja zadowolenia z komitetu. Ciąg $(i_1, \ldots, i_k)$- ciąg pozycji kandydatów

Przyklad $C = c_1, c_2, c_3, c_4$ - zbiór kandydatów, $v = (c_2, c_1, c_4, c_3)$ - głos Niech $k = 2$ (wybory 2 spośród 4) $w = (c_4, c_3)$ Najpierw określamy pozycje kandydatów z komitetu w w v: $pos_v(w) = (3, 4)$, zatem wynik komitetu w dla głosu v wynosi $f_{kB}(3, 4) = (3)+(4) = ||C||-3)+(||C||-4) = 1+0 = 1$

1.1.3. Uogólnienie - system ℓ_p Borda

Zanim wprowadzone zostanie pojęcie uogólnionego systemu k-Borda warto przypomnieć wzór na normę ℓ_p

Norma ℓ_p

$\ell_p(x_1, x_2, \ldots, x_n) = \sqrt[p]{x_1^p + x_2^p + \ldots + x_n^p}$,
Wówczas, w uogólnionej wersji metody k-Borda, funkcja zadowolenia f_{kB} zostaje uzależniona również od parametru p z powyższego wzoru. Norma liczona jest z wyników według Bordy, $\beta(i)$. Wzór uogólniony funkcji zadowolenia przyjmuje zatem postać:
$f_{\ell_p B}(p, (i_1 1, \ldots, i_k)) = p[(i1)]p + [(i2)]p + ... + [(ik)]p$
Systemy k-Borda i Cahmberlin'a-Courant'a są szczególnymi przypadkami zdefiniowanego powyżej systemu ℓ_p- Borda:
Dla $p = 1, l_1$
$f_{\ell_p B}(1, (i_1, \ldots, i_k)) = \beta(i_1) + \beta(i_2) + \ldots + \beta(ik) = f_{kB}(i_1 1, \ldots, i_k)$
Dla $p = \infty, l_\infty = max$
$f_{\ell_p B}(\infty, (i_1, \ldots, i_k)) = \lim_{p\to\infty} \sqrt[p]{\beta\left[(i_1)\right]^p + \beta\left[(i_2)\right]^p + \ldots + \left[\beta(i_k)\right]^p} = \max \beta(i_1), \beta(i_2), \ldots, \beta(i_k) = \beta(i_1) = f_{CC}$

1.2. Format danych wejściowych

Pojedynczy plik składa się z następującego formatu:

```
<liczba kandydatów>
1, <nazwa kandydata>
2, <nazwa kandydata>
...
<liczba kandydatów>, <nazwa kandydata>
<liczba głosujących>, <liczba głosów policzonych>, <liczba unikalnych
głosów>
<liczba powtórzeń głosu>, <głos>
...
<liczba powtórzeń głosu>, <głos>
```

2. Architektura Django

$Django$ to framework webowy napisany w $Pythonie$. Dostarcza wysokopoziomowych abstrakcji pozwalających na szybkie i wygodne pisanie przejrzystych aplikacji.

Jego architektura koncepcyjnie przypomina wzorzec architektoniczny $Model - View - Controller$, jednak jak przyznają sami twórcy [ref], $Django$ nie do końca wpasowuje się w klasyczne ujęcie MVC.

$Django$ dostarcza mapowania obiekto-relacyjnego, dzięki którym całość modeli można ująć w $Pythonie$. Wygodny ORM zazwyczaj wystarcza do obsługi bazy danych, jednak zawsze istnieje możliwość użycia bezpośrednio SQL.

Widoki w $Django$ spełniają dwojaką funkcję - służą zarówno przekazaniu danych do wyświetlenia, jak i ich modyfikacji. W wyświetleniu danych użytkownikowi pośredniczą szablony ($templates$), które ópakowują"przekazane dane do postaci $HTMLa$, który może wyświetlić przeglądarka internetowa. Dzięki temu wybór danych, jakie mają zostać pokazane użytkownikowi jest oddzielony od samego sposobu ich prezentacji.

Za kontroler z klasycznego MVC można uznać sam framework dostarczający wspomnianej obsługi bazy danych czy mapowania adresów URL do poszczególnych widoków.

3. Opis modułów

3.1. Moduł administracji kont

3.1.1. Opis ogólny

Moduł odpowiedzialny za zarządzanie kontami użytkowników. Umożliwia czynności logowania, wylogowania oraz rejestracji. Nie współpracuje z innymi modułami. Korzysta jedynie z bazy danych w celu weryfikacji użytkowników.

3.1.2. Komponenty programowe

Komponenty programowe dla tego modułu znajdują się w pakiecie $ecs.accounts$

Typ komponentu	Komponenty	Wykorzystanie
widoki	klasa $LoginView$	logowanie
	klasa $RegisterView$	rejestracja
szablony	$login.html$	logowanie
	$register.html$	rejestracja
formularze	klasa $LoginForm$	logowanie
	klasa $RegistrationForm$	rejestracja

3.2. Moduł zarządzania wyborami

3.2.1. Opis ogólny

Moduł odpowiedzialny za administrację wyborami i ich wynikami. Umożliwia podstawowe operacje na wyborach i ich wynikach oraz nawigację między nimi. Pozwala na wyświetlenie listy wszystkich wyborów i ich wyników, stworzenie nowych wyborów lub wyniku, czy usunięcie wyborów. W celu wykonania niektórych zadań współpracuje z modułem obliczania wyników wyborów oraz modułem wizualizacji wyborów i ich wyników. Jest to najbardziej rozbudowany moduł.

3.2.2. Komponenty programowe

Typ komponentu	Komponenty	Wykorzystanie
widoki	klasa $ElectionListView$	wyświetlenie listy wszystkich wyborów
	klasa $ElectionCreateView$	stworzenie wyborów
	klasa $ElectionDeleteView$	usunięcie wyborów
	klasa $ElectionDetailView$	wyświetlenie informacji szczegółowych o danych wyborach
	klasa $ResultCreateView$	stworzenie wyniku dla danych wyborów - wyświetlenie formularza określającego parametry wyniku
	klasa $ResultDetailsView$	wyświetlenie danego wyniku danych wyborów
	klasa $ResultDeleteView$	usuwanie pojedynczego wyniku wyborów
szablony	$election_list.html$	wyświetlenie listy wszystkich wyborów, usunięcie wyborów - strona sukcesu wyświetlana po usunięciu wyborów
	$election_create.html$	stworzenie wyborów
	$election_details.html$	stworzenie wyborów - strona sukcesu wyświetlana po stworzeniu wyborów, wyświetlenie informacji szczegółowych o danych wyborach
	$election_delete.html$	usunięcie wyborów
	$result_create.html$	stworzenie nowego wyniku wyborów - strona z formularzem
	$result_details.html$	wyświetlenie danego wyniku danych wyborów
	$result_delete.html$	potwierdzenie usunięcia rezultatu
formularze	klasa $ElectionForm$	stworzenie wyborów
	klasa $ResultForm$	stworzenie wyniku dla danych wyborów - formularz określający parametry wyniku

3.3. Moduł wizualizacji wyborów i wyników wyborów

3.3.1. Opis ogólny

Moduł odpowiedzialny za stworzenie wykresu $2D$ wizualizującego wybory i jego wyniki. Wizualizacja dotyczy tylko wyborów wygenerowanych z rozkładu normalnego. Wyborcy i kandydaci są reprezentowani jako punkty na płaszczyźnie. Punkty reprezentujące wyborców i kandydatów mają na wykresie odmienne kolory. Na wykresie wyników wyborów punkty reprezentujące zwycięzców wyborów są powiększone. Moduł współpracuje z modułem zarządzania wyborów, który zleca mu zadanie wizualizacji wyborów lub jego wyników. W celu wykonania zadania moduł wizualizacji wyborów i wyników wyborów pobiera dane z bazy danych.

3.3.2. Komponenty programowe

Typ komponentu	Komponenty	Wykorzystanie
widoki	klasa $ScatterChartMixin$	pobranie danych potrzebnych do wygenerowania wykresów (punkty, tytuł wykresu, serii danych), dziedziczy po klasie $View$
	klasa $ElectionChartView$	wizualizacja wyborów, klasa dziedziczy po klasie $ScatterChartMixin$, pobranie współrzędnych kandydatów i wyborców
	klasa $ResultChartView$	wizualizacja wyników wyborów, klasa dziedziczy po klasie $ScatterChartMixin$, pobranie współrzędnych kandydatów, wyborców oraz zwycięzców

3.4. Moduł generacji i wczytywania wyborów

3.4.1. Opis ogólny

Moduł odpowiedzialny za generację wyborów z rozkładu normalnego oraz wczytywanie wyborów z pliku formatu *.soc*. Moduł współpracuje z modułem zarządzania wyborami, któremu zleca po wykonaniu swoich zadań, stworzenie i wysłanie użytkownikowi odpowiedniej strony internetowej. Moduł zapewnia wygenerowanie wyborów z rozkładu normalnego według wskazanych parametrów oraz wali-

dację danych przy wczytywaniu wyborów z pliku. Po stworzeniu wyborów moduł komunikuje się z bazą danych w celu utrwalenia wyborów.

3.4.2. Komponenty programowe

Typ komponentu	Komponenty	Wykorzystanie
Widoki	klasa $ElectionLoadDataFormView$	wczytanie wyborów z pliku
	klasa $ElectionGenerateDataFormView$	generacja wyborów z rozkładu normalnego
Szablony	$election_load_data.html$	strona z formularzem do wskazania pliku
	$election_generate_data.html$	strona z formularzem do wskazania parametrów wyborów i rozkładu normalnego
	$election_details.html$	strona wyświetlana po poprawnym wczytaniu danych z pliku
Formularze	klasa $ElectionLoadDataForm$	wczytanie z pliku

3.5. Moduł obliczania wyników wyborów

3.5.1. Opis ogólny

Moduł odpowiedzialny za obliczanie wyników wyborów. Zapewnia różne algorytmy do wykonania zadania. Użytkownik ma wybór między algorytmem genetycznym, dwoma algorytmami zachłannymi oraz algorytmem typu brute-force. Moduł współpracuje z modułem zarządzania wyborami, który zleca mu wykonanie zadania.

3.5.2. Komponenty programowe

Wszystkie komponenty programowe dotyczące modułu obliczania wyników wyborów zawierają się w pakiecie $ecs.elections.algorithms$.
Klasy odpowiedzialne za poszczególne algorytmy:

- $BruteForce$ - odpowiedzialna za algorytm typu brute-force
- $GreedyAlgorithm$ - odpowiedzialna za algorytm zachłanny zależny od parametru p
- $GreedyCC$ - odpowiedzialna za algorytm zachłanny niezależny od parametru p
- $GeneticAlgorithm$ - odpowiedzialna za algorytm genetyczny

3.6. Moduł URL Resolver

3.6.1. Opis ogólny

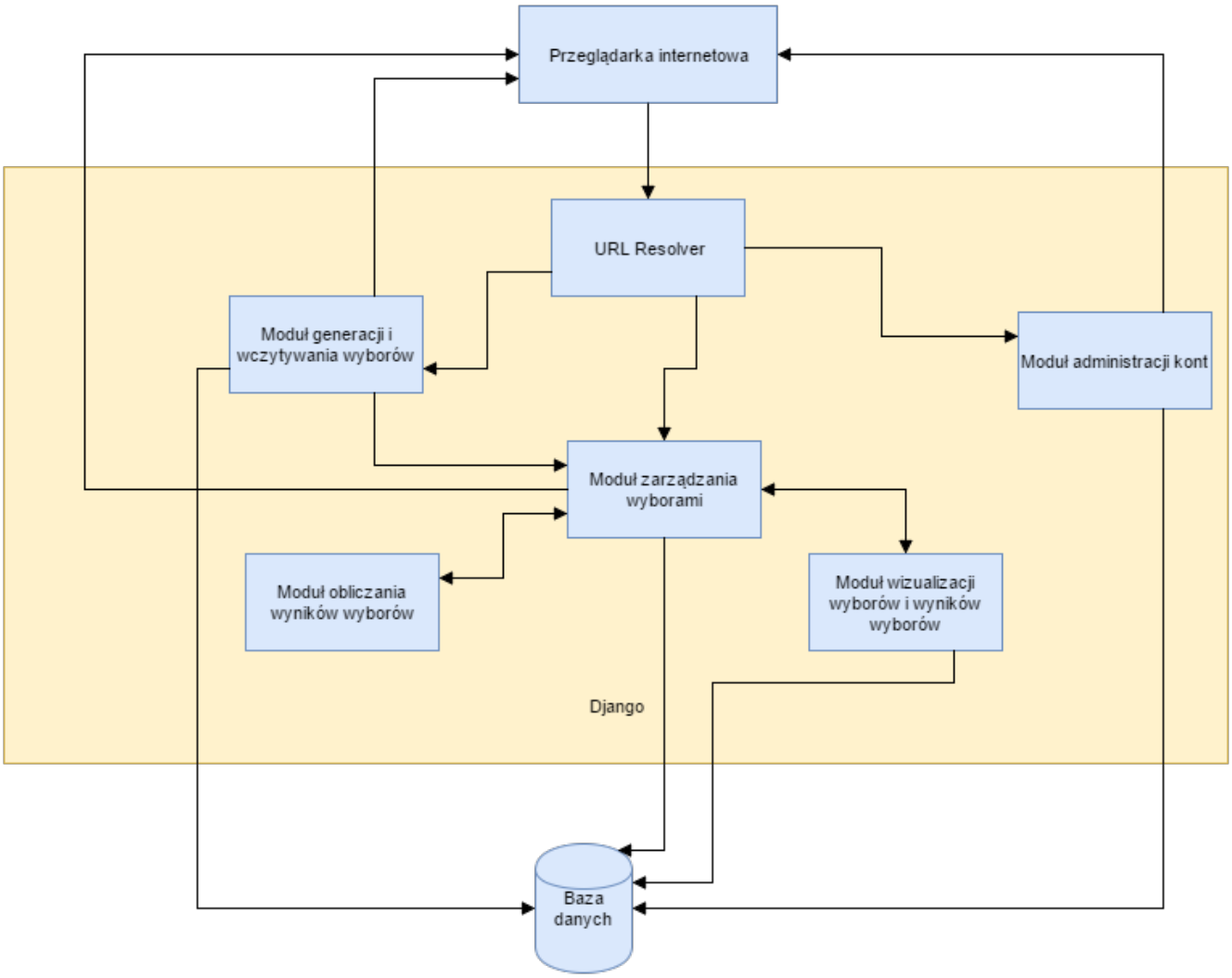
Moduł odpowiedzialny za przekazywanie żądań otrzymywanych przez klienta do odpowiednich modułów.

3.6.2. Komponenty programowe

Przyporządkowania żądań użytkownika do odpowiednich modułów (adresów URL do widoków) znajdują się w pliku $ecs.elections.urls.py$.

4. Współpraca modułów i innych komponentów

4.1. Diagram komunikacji



Rysunek 4.1: Diagram komunikacji

4.2. Ścieżki przejść

4.2.1. Logowanie

1. Użytkownik za pomocą przeglądarki internetowej wysyła żądanie logowania.

2. Moduł URL Resolver kieruje żądanie do modułu administracji kont.

3. Moduł administracji kont przesyła do przeglądarki stronę z formularzem do zalogowania (pole $Username$ i $Password$).
4. Użytkownik wypełnia formularz i za pomocą przeglądarki internetowej wysyła uzupełniony formularz.
5. Moduł URL Resolver przekazuje żądanie do modułu administracji kont.
6. Moduł administracji kont przeprowadza uwierzytelnienie użytkownika.
7. W zależności od rezultatu uwierzytelnienia podjęte są trzy możliwe akcje:
   - jeżeli uwierzytelnienie powiodło się, Moduł administracji kont loguje użytkownika i wysyła do przeglądarki stronę z wiadomością o sukcesie logowania
   - jeżeli podana nazwa użytkownika nie istnieje w bazie, Moduł administracji kont wysyła do przeglądarki internetowej stronę z formularzem do zalogowania (pole $Username$ i $Password$) i z informacją o nieistnieniu podanej nazwy użytkownika. Następuje przejście do kroku nr 4
   - jeżeli podana nazwa użytkownika istnieje w bazie, ale podane hasło jest nieprawidłowe, Moduł administracji kont wysyła do przeglądarki internetowej stronę z formularzem do zalogowania (pole $Username$ i $Password$) i z informacją o błędnym haśle. Następuje przejście do kroku nr 4

4.2.2. Wylogowanie

1. Użytkownik za pomocą przeglądarki internetowej wysyła żądanie wylogowania.
2. Moduł URL Resolver kieruje żądanie do modułu administracji kont.
3. Moduł administracji kont wylogowuje użytkownika i przesyła do przeglądarki stronę domową aplikacji.

4.2.3. Rejestracja

1. Użytkownik za pomocą przeglądarki internetowej wysyła żądanie zarejestrowania użytkownika.
2. Moduł URL Resolver kieruje żądanie do modułu administracji kont.
3. Moduł administracji kont przesyła do przeglądarki stronę z formularzem do rejestracji (pola $E-mail$, $Username$, $Password$, $Firstname$ i $Lastname$).
4. Użytkownik wypełnia formularz i za pomocą przeglądarki internetowej wysyła uzupełniony formularz.
5. Moduł URL Resolver przekazuje żądanie do modułu administracji kont.
6. Moduł administracji kont przeprowadza walidację danych.

7. W zależności od rezultatu walidacji:

   - jeżeli się powiodła, Moduł administracji kont tworzy konto nowego użytkownika, zapisuje je do bazy danych, loguje użytkownika do nowo utworzonego konta i przesyła do przeglądarki stronę z wiadomością o sukcesie logowania
   - w przeciwnym wypadku, zostaje wysłana do przeglądarki strona z formularzem do rejestracji wraz z informacją o znalezionym problemie. Następuje przejście do kroku nr 4

4.2.4. Wyświetlenie listy wszystkich wyborów

1. Użytkownik klikając przycisk $Yourelections$ na stronie, wysyła za pomocą przeglądarki internetowej żądanie wyświetlenia listy swoich wyborów.

2. Moduł URL Resolver kieruje żądanie do modułu zarządzania wyborami (widok $ElectionListView$).

3. Moduł zarządzania wyborami kieruje zapytanie do bazy danych o obiekty reprezentujące wybory danego użytkownika.

4. Po pobraniu z bazy potrzebnych danych moduł zarządzania wyborami za pomocą szablonu $election\_list.html$ tworzy stronę i wysyła do przeglądarki internetowej użytkownika.

4.2.5. Stworzenie wyborów

1. Użytkownik klikając przycisk $Newelection$ na stronie, wysyła za pomocą przeglądarki internetowej żądanie stworzenie nowych wyborów.

2. Moduł URL Resolver kieruje żądanie do modułu zarządzania wyborami (widok $ElectionCreateView$).

3. Widok posiada zdefiniowany szablon $election\_create.html$ i formularz $ElectionForm$. Wykorzystując te komponenty tworzy i wysyła do użytkownika stronę z formularzem.

4. Użytkownik wypełnia formularz (pola $Name$ i $Commiteesize$) i wysyła uzupełniony formularz.

5. Moduł URL Resolver kieruje żądanie do modułu zarządzania wyborami (widok $ElectionCreateView$).

6. Widok wykorzystując klasę formularza pobiera dane z formularza, tworzy nowy obiekt wyborów i zapisuje go do bazy.

7. Widok wykorzystując szablon $election\_details.html$ tworzy stronę i wysyła ją do przeglądarki użytkownika.

4.2.6. Usunięcie wyborów

1. Użytkownik klikając ikonę kosza na stronie, wysyła za pomocą przeglądarki internetowej żądanie usunięcia wyborów.

2. Moduł URL Resolver kieruje żądanie do modułu zarządzania wyborami (widok $ElectionDeleteView$).

3. Widok posiada zdefiniowany szablon $election\_delete.html$, który wykorzystuje do stworzenia i wysłania strony potwierdzającej akcję użytkownika.

4. Użytkownik potwierdza chęć usunięcia wyborów przez naciśnięcie przycisku $Yes, deletethis$. (W przypadku wyboru przycisku $No, cancel$ akcja nie zostaje przeprowadzona a użytkownik zostaje przeniesiony na stronę z listą swoich wyborów).

5. Moduł URL Resolver kieruje żądanie do modułu zarządzania wyborami (widok $ElectionDeleteView$).

6. Widok usuwa wybory z bazy danych i wykorzystując szablon $election\_list.html$ tworzy i wysyła stronę do przeglądarki internetowej użytkownika.

4.2.7. Wyświetlenie informacji szczegółowych dla danych wyborów

1. Użytkownik klikając na link wyborów na stronie z listą swoich wyborów, wysyła za pomocą przeglądarki internetowej żądanie wyświetlenia informacji szczegółowych dla danych wyborów.

2. Moduł URL Resolver kieruje żądanie do modułu zarządzania wyborami (widok $ElectionDetailView$).

3. Widok pobiera z bazy danych listę głosujących oraz listę wyników wyborów.

4. Jeżeli wybory zostały wygenerowane z rozkładu normalnego, widok przekierowuje działanie programu do modułu wizualizacji wyborów i wyników wyborów (widok $ElectionChartView$). Jeżeli wybory nie zostały wygenerowane z rozkładu normalnego, następuje przejście do kroku nr 6

5. Widok pobiera z bazy danych obiekty będące punktami $2D$ reprezentującymi wyborców i kandydatów. Ustawia parametry wykresu a następnie wysyła wszystkie dane z powrotem do widoku $ElectionDetailView$ umożliwiając stworzenie wykresu.

6. Widok wykorzystując szablon $election\_details.html$ tworzy i wysyła stronę do przeglądarki użytkownika.

4.2.8. Stworzenie nowego wyniku dla danych wyborów

1. Użytkownik klikając na link $Addnewresult$ na stronie ze szczegółowymi informacjami o danych wyborach, wysyła za pomocą przeglądarki internetowej żądanie stworzenia nowego wyniku dla danych wyborów.
2. Moduł URL Resolver kieruje żądanie do modułu zarządzania wyborami (widok $ResultCreateView$).
3. Widok wykorzystując formularz $ResultForm$ i szablon $result\_create.html$ tworzy oraz wysyła do użytkownika stronę z formularzem do określenia parametrów wyniku (pola: parametr p do obliczania normy oraz typ algorytmu).
4. Użytkownik wypełnia formularz i wysyła go do systemu.
5. Moduł URL Resolver kieruje żądanie do modułu zarządzania wyborami (widok $ResultCreateView$).
6. Widok wczytuje dane z formularza, pobiera z bazy danych obiekt reprezentujący wybory a następnie przekierowuje działanie programu do modułu obliczania wyników wyborów przekazując konieczne parametry.
7. Moduł obliczania wyników wyborów liczy zwycięzców wyborów za pomocą odpowiedniego algorytmu. Następnie przekazuje sterowanie z powrotem do widoku $ResultCreateView$ przekazując wyniki.
8. Widok zapisuje wynik do bazy danych i przechodzi do ścieżki *Wyświetlenie informacji szczegółowych dla danych wyborów*.

4.2.9. Wyświetlenie szczegółowych informacji o danym wyniku danych wyborów

1. Użytkownik klikając na link danego wyniku wyborów na stronie ze szczegółowymi informacjami o danych wyborach, wysyła za pomocą przeglądarki internetowej żądanie wyświetlenia szczegółowych informacji o danym wyniku dla danych wyborów.
2. Moduł URL Resolver kieruje żądanie do modułu zarządzania wyborami (widok $ResultDetailsView$).
3. Widok pobiera z bazy danych informacje o zwycięzcach danych wyborów.
4. Jeżeli wybory zostały wygenerowane z rozkładu normalnego, widok przekierowuje działanie programu do modułu wizualizacji wyborów i wyników wyborów (widok $ResultChartView$). Jeżeli wybory nie zostały wygenerowane z rozkładu normalnego, następuje przejście do kroku nr 6
5. Widok pobiera z bazy danych obiekty będące punktami $2D$ reprezentującymi wyborców, kandydatów i zwycięzców. Ustawia parametry wykresu a następnie wysyła wszystkie dane z powrotem do widoku $ResultDetailsView$ umożliwiając stworzenie wykresu.

6. Widok wykorzystując szablon $result\_details.html$ tworzy i wysyła stronę do przeglądarki użytkownika.

4.2.10. Wczytanie wyborów z pliku .soc

1. Użytkownik klikając na przycisk $Loaddatafromfile$ na stronie ze szczegółowymi informacjami o danych wyborach, wysyła za pomocą przeglądarki internetowej żądanie wczytania wyborów z pliku $.soc$.
2. Moduł URL Resolver kieruje żądanie do modułu generacji i wczytywania wyborów (widok $ElectionLoadDataFormView$).
3. Widok wykorzystując formularz $ElectionLoadDataForm$ i szablon $election_load_data.html$ tworzy oraz wysyła do użytkownika stronę z formularzem do wybrania pliku, z którego będą wczytywane wybory.
4. Użytkownik wskazuje plik i wysyła uzupełniony formularz do systemu.
5. Moduł URL Resolver kieruje żądanie do modułu generacji i wczytywania wyborów (widok $ElectionLoadDataFormView$).
6. Widok parsuje plik wejściowy i waliduje dane. W przypadku poprawnego wczytania pliku wejściowego następuje zapis wczytanych obiektów do bazy danych. Za pomocą szablonu $election\_details.html$ zostaje stworzona i wysłana strona do przeglądarki użytkownika.

4.2.11. Generacja wyborów z rozkładu normalnego

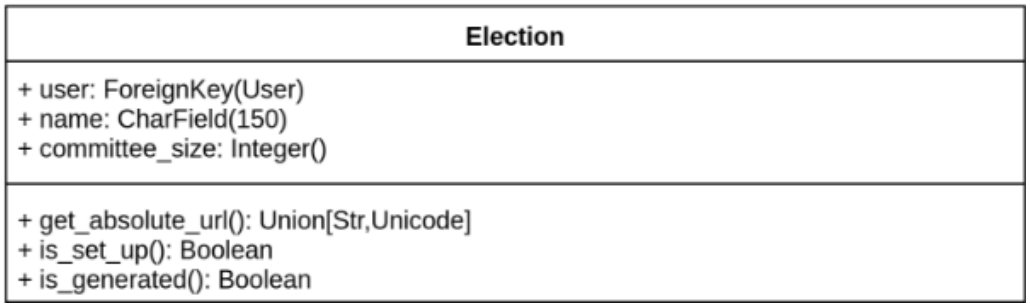
1. Użytkownik klikając na przycisk $Generateelectionfromnormaldistribution$ na stronie ze szczegółowymi informacjami o danych wyborach, wysyła za pomocą przeglądarki internetowej żądanie wygenerowania wyborów za pomocą rozkładu normalnego.
2. Moduł URL Resolver kieruje żądanie do modułu generacji i wczytywania wyborów (widok $ElectionGenerateDataFormView$).
3. Widok wykorzystując formularz $ElectionGenerateDataForm$ i szablon $election\_generate\_data.html$ tworzy oraz wysyła do użytkownika stronę z formularzem do określenia parametrów wyborów oraz i rozkładu normalnego.
4. Użytkownik uzupełnia formularz i wysyła go do systemu.
5. Moduł URL Resolver kieruje żądanie do modułu generacji i wczytywania wyborów (widok $ElectionGenerateDataFormView$).
6. Widok wczytuje dane z formularza i wykorzystując generator, generuje według wczytanych parametrów kandydatów, wyborców oraz liczy ich preferencje. Dane zostają zapisane do bazy. Następnie program przechodzi do 2-go punktu ścieżki *Wyświetlanie informacji szczegółowych dla danych wyborów*.

5. Struktura obiektowa przechowywanych danych

5.1. Opis modeli wykorzystywanych w ORM

Poniżej zamieszczono opis klas wykorzystywanych przez $Django$ do odwzorowania obiektowej architektury systemu informatycznego na bazę danych.

5.1.1. Election



Rysunek 5.1: Obiekt mapowany na tabelę wyborów

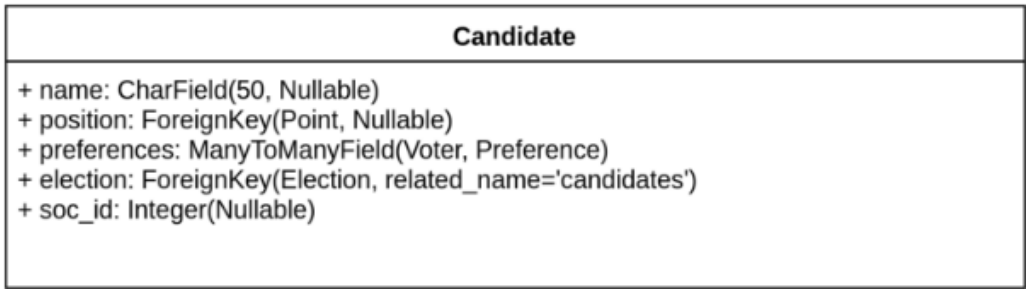
Pola:

- $user$ - użytkownik aplikacji
- $name$ - nazwa wyborów
- $committee\_size$ - rozmiar zwycięskiego komitetu

Metody:

- $get\_absolute\_url()$ - zwraca adres URL, który daje dostęp do danych wyborów z poziomu dokumentu $.html$
- $is\_set\_up()$ - zwraca prawdę jeśli dla danych wyborów zostali określeni kandydaci i głosujący lub fałsz w przeciwnym przypadku

5.1.2. Candidate

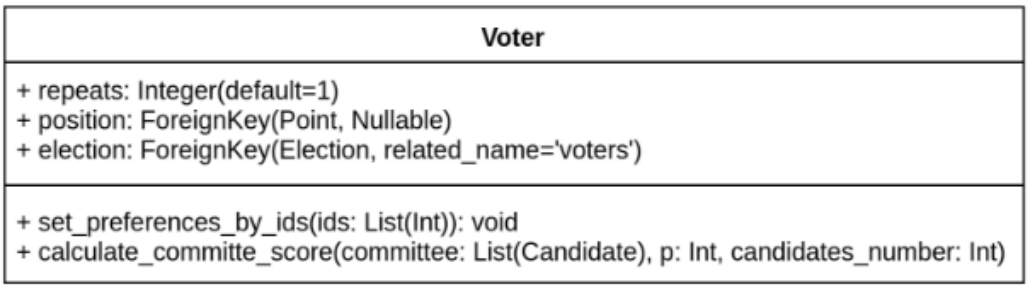


Rysunek 5.2: Obiekt mapowany na tabelę z danymi kandydata

Pola:

- $name$ - nazwa kandydata
- $position$ - położenie kandydata w układzie współrzędnych, pole opcjonalne dla kandydatów generowanych z rozkładu normalnego
- $preferences$ - pole mapowane na relację wiele do wielu łączącej kandydata i głosującego w tabeli $Preference$
- $election$ - wybory, do których należy kandydat
- soc_id - identyfikator kandydata w odpowiadającym wyborom pliku w formacie $.soc$, pole opcjonalne dla wyborów parsowanych z pliku

5.1.3. Voter



Rysunek 5.3: Obiekt reprezentujący głos w wyborach

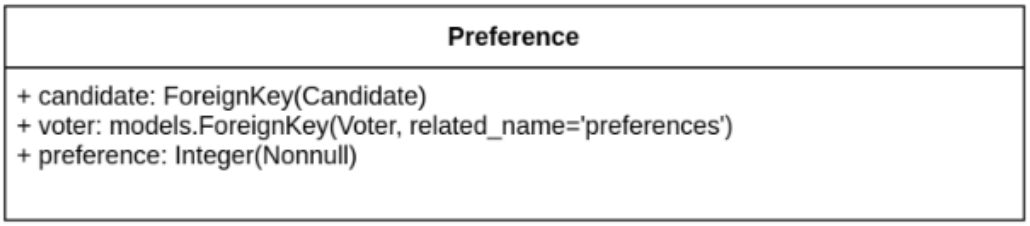
Pola:

- $repeats$ - liczba powtórzeń głosów w wyborach
- $position$ - współrzędne wyborcy na wykresie, pole opcjonalne dla wyborów generowanych z rozkładu normalnego
- $election$ - wybory, do których należy głosujący

Metody:

- $set\_preferences\_by\_ids()$ - metoda wykorzystywana przy parsowaniu plików w formacie *.soc*. Na podstawie ciągu identyfikatorów wyborców pobranego z pliku tworzy ciąg kandydatów ułożony według preferencji wyborcy. Pobrane dane zapisuje w bazie danych w postaci rekordu w tabeli $Preference$.
- $calculate\_committe\_score(committee : List(Candidate),\ p : Int,\ candidates\_number : Int)$ - oblicza wynik komitetu $committee$ na podstawie parametru p i liczby kandydatów $candidates\_number$. Wynik mnożony jest przez liczbę powtórzeń głosu.

5.1.4. Preference

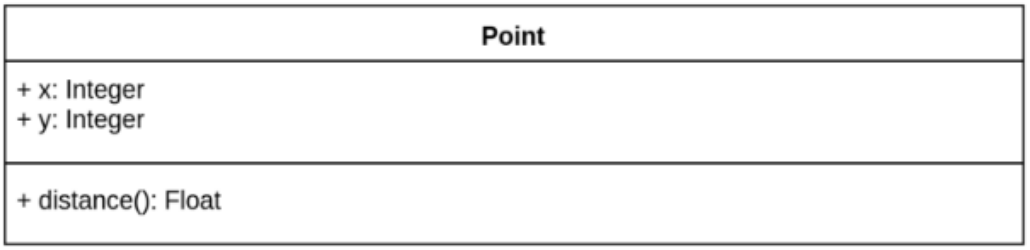


Rysunek 5.4: Obiekt reprezentujący pozycję kandydata w liście preferencji głosującego

Pola:

- $candidate$ - kandydat związany z preferencją
- $voter$ - głosujący związany z preferencją
- $preference$ - pozycja preferencji na liście preferencji głosującego

5.1.5. Point



Rysunek 5.5: Obiekt przechowujący współrzędne wyborcy lub kandydata w układzie współrzędnych.

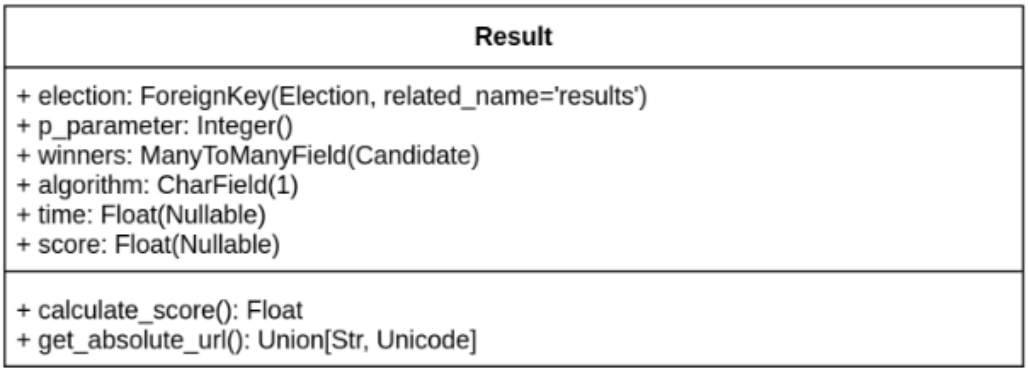
Pola - współrzędne w układzie kartezjańskim:

- $x : Int$
- $y : Int$

Metody:

- $distance(other : Point) : Float$ - metoda zwracająca odległość punktu od punktu $other$ w metryce euklidesowej w przestrzeni R^2

5.1.6. Result

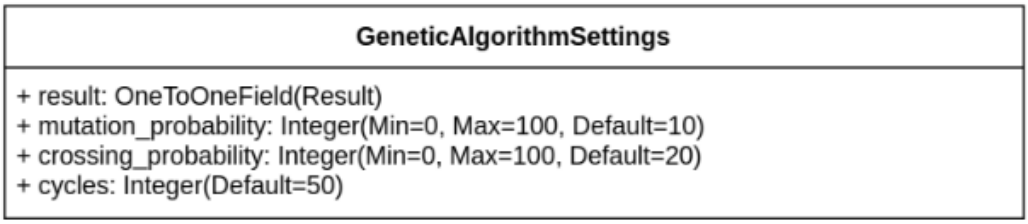


Rysunek 5.6: Obiekt mapowany na tabelę z danymi wyników wyborów: rodzaj algorytmu, parametry wywołania algorytmu, otrzymany wynik punktowy komitetu oraz czas działania algorytmu

Pola:

- $election$ - wybory, dla których obliczono rezultat
- $p\_parameter$ - parametr p normy ℓ_p
- $winners$ - zwycięski komitet
- $algorithm$ - identyfikator algorytmu
- $time$ - czas wykonania obliczeń
- $score$ - wynik punktowy zwycięskiego komitetu

5.1.7. GeneticAlgorithmSettings

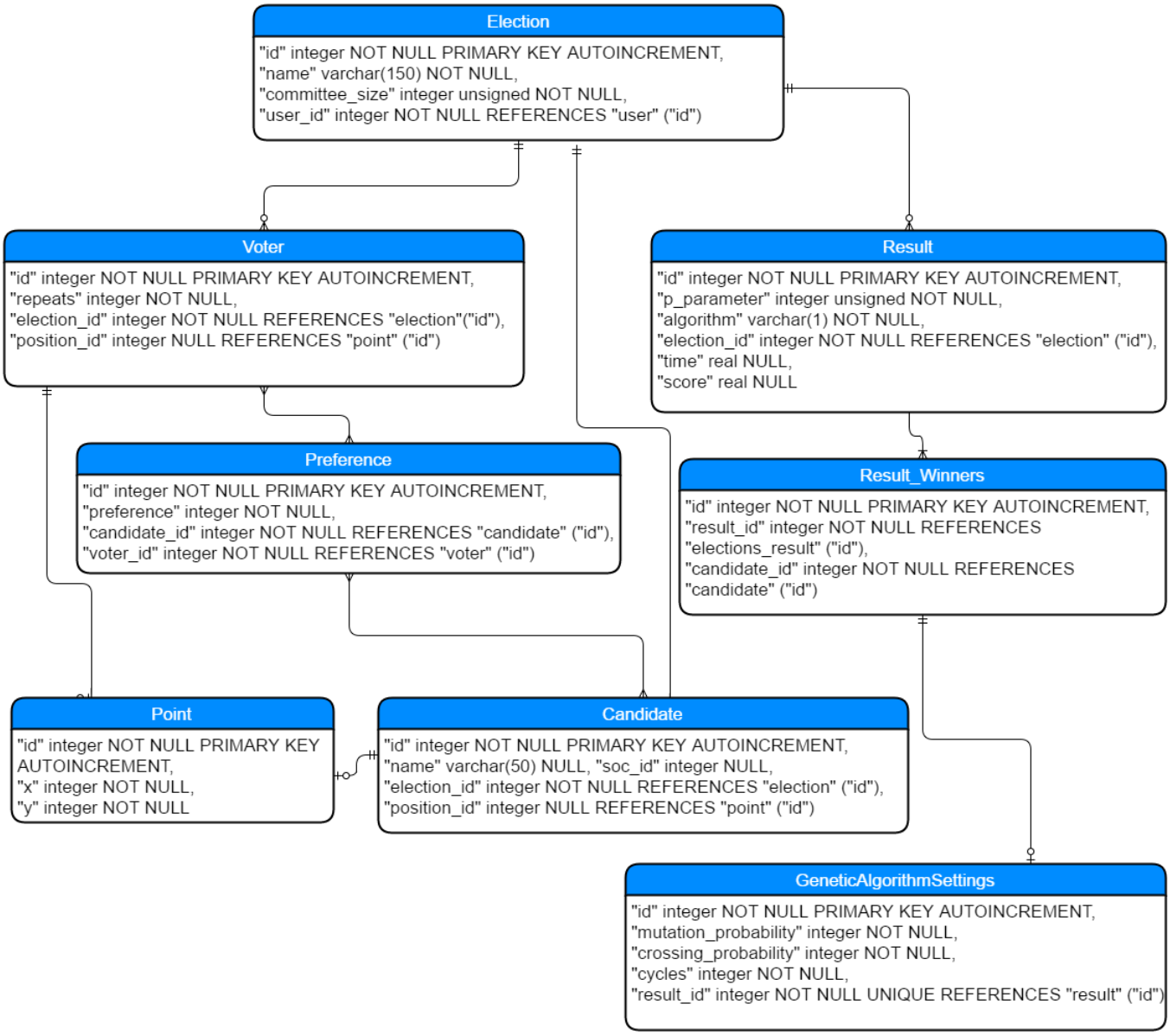


Rysunek 5.7: Klasa mapowana na tabelę gromadząca dane specyficzne dla konfiguracji algorytmu genetycznego. Wykorzystywana do zapisania specyficznych dla algorytmu genetycznego parametrów wywołania.

Pola:

- $result$ - rezultat algorytmu genetycznego
- $mutation\_probability$ - prawdopodobieństwo wystąpienia mutacji danego osobnika
- $crossing\_probability$ - prawdopodobieństwo wymiany puli genów pomiędzy parą osobników
- $cycles$ - liczba iteracji algorytmu genetycznego

5.2. Diagram ERD



Rysunek 5.8: Diagram ERD

6. Algorytmy

6.1. Algorytm zachłanny

6.1.1. Idea algorytmu

Idea algorytmu polega na dołączaniu kolejnego kandydata do zwycięskiego komitetu w każdej iteracji. W związku z tym główna pętla algorytmu ma tyle iteracji, ile liczy rozmiar zwycięskiego komitetu. Jeżeli któryś z kandydatów został wybrany w danej iteracji jest pewne, iż znajdzie się on w zwycięskim komitecie. Wybór kandydata w danej iteracji dokonywany jest w sposób zachłanny. Spośród kandydatów, którzy jeszcze nie zostali wybrani do zwycięskiego komitetu, dołącza do zwycięzców ten, który razem z wcześniej wybranymi kandydatami tworzy komitet o największym uznaniu wśród wyborców. Mając więc na danym etapie algorytmu komitet złożony z k zwycięzców, w kolejnej iteracji poszukiwany jest $k+1$-szy kandydat spośród pozostałych. Ma on stworzyć komitet liczący $k+1$ członków, który jest najlepszy wśród wszystkich innych możliwości.W każdej iteracji algorytmu wybierany jest kandydat, który najwięcej dodaje do aktualnego poziomu zadowolenia wyborców. Stąd można uznać opisany algorytm jako zachłanny.

6.1.2. Funkcja zadowolenia

Aktualny poziom zadowolenia wyborców na danym etapie algorytmu określany jest za pomocą uprzednio zdefiniowanego uogólnienia systemu $k - Bordy$. Satysfakcja pojedynczego wyborcy z komitetu liczącego na danym etapie algorytmu k zwycięzców,określana jest za pomocą funkcji:

$$f_{\ell_p - Borda}(i_1, i_2, ..., i_k) = \sqrt[p]{[\beta(i_1)]^p + [\beta(i_2)]^p + \ldots + [\beta(i_k)]^p}$$

Funkcja bazuje na normie ℓ_p oraz punktacji $k - Bordy$. Dokładne informacje na temat funkcji umieszczone są w rozdziale *Dziedzina problemu*. Ogólne zadowolenie wyborców z komitetu jest obliczane jako suma pojedynczych zadowoleń wszystkich wyborców. Algorytm działa w ten sposób, że przed wyborem pierwszego zwycięzcy, poziom zadowolenia wyborców wynosi 0, gdyż zadowolenie każdego pojedynczego wyborcy wynosi 0 - zadowolenie z komitetu liczącego 0 osób zgodnie z powyższą funkcją wynosi 0. W pierwszej iteracji wybierany jest pierwszy zwycięzca w ten sposób, żeby suma pojedynczych zadowoleń wszystkich wyborców z jednoosobowego komitetu była największa. W kolejnej iteracji dobierany jest do wybranego wcześniej komitetu jednoosobowego drugi zwycięzca w ten sposób, by suma pojedynczych zadowoleń wszystkich wyborców z dwuosobowego komitetu była największa. W drugiej iteracji algorytmu brane więc pod uwagę są tylko wybrane komitety dwuosobowe - takie, które składają się z wcześniej wybranego zwycięzcy i dobranego wciąż jeszcze walczącego o zwycięstwo kandydata. W trzeciej iteracji do wybranego wcześniej komitetu dwuosobowego dobierany jest trzeci zwycięzca w ten sposób, by suma pojedynczych zadowoleń wszystkich wyborców z trzyosobowego komitetu była

największa. Podobnie jak w drugiej iteracji, pod uwagę brane są tylko wybrane komitety trzyosobowe - takie, które składają się z wcześniej wybranego komitetu dwuosobowego i dobranego wciąż jeszcze walczącego o zwycięstwo kandydata. W kolejnych iteracjach algorytmu, kandydaci dobierani są w sposób analogiczny. Kandydaci dobierani są aż do osiągnięcia określonego rozmiaru komitetu. Poziom zadowolenia wyborców po każdej iteracji algorytmu rośnie.

6.1.3. Szczegóły implementacji

Algorytm zaimplementowany jest w klasie $GreedyAlgorithm$ znajdującej się w pakiecie $ecs.elections.algorithms.greedy\_algorithm$. Główna pętla algorytmu znajduje się w metodzie run. Przed główną pętlą programu do pamięci operacyjnej pobrane są obiekty z bazy danych reprezentujące kandydatów, wyborców oraz preferencje wyborców. Ponadto odpowiednio uzupełniane są struktury danych reprezentujące aktualne zadowolenie wyborców z wybranych kandydatów, aktualny skład zwycięskiego komitetu oraz kandydatów wciąż walczących o zwycięstwo, którzy stanowią wszystkich kandydatów bez tych, którzy już zostali wybrani do zwycięskiego komitetu. Początkowo poziom zadowolenia każdego wyborcy wynosi 0, skład zwycięskiego komitetu jest pusty, a kandydaci wciąż walczący o zwycięstwo stanowią zbiór wszystkich kandydatów.

W głównej pętli algorytmu znajdują się dwie wewnętrzne pętle. Bardziej zewnętrzna pętla iteruje po wciąż walczących o zwycięstwo kandydatach, bardziej wewnętrzna pętla iteruje po wszystkich wyborcach. Wskazane dwie wewnętrzne pętle mają na celu wyłonienie najlepszego kandydata spośród jeszcze niewybranych kandydatów do zwycięskiego komitetu. Dzieje się to poprzez obliczenie poziomu zadowolenia wyborców z komitetu złożonego z kandydatów już wcześniej wybranych (na początku komitet zwycięski jest pusty) wraz z dodatkowo dobranym kandydatem wcześniej niewybranym. Obliczany jest poziom zadowolenia dla każdego możliwego dobrania (iteracja po kandydatach jeszcze niewybranych) i ostatecznie w danej iteracji głównej pętli algorytmu do zwycięskiego komitetu, dołączany jest kandydat, który wraz z wcześniej wybranymi kandydatami stanowi największy poziom zadowolenia wyborców. W algorytmie najlepszy kandydat w danej iteracji dobierany jest w ten sposób, że wskazany jest aktualny lider spośród kandydatów jeszcze niewybranych wraz z poziomem satysfakcji wyborców z komitetu utworzonego z tego lidera i z wcześniej wybranych kandydatów. Jeżeli poziom satysfakcji wyborców z komitetu z innym niewybranym jeszcze kandydatem okaże się większy od tego z aktualnym liderem, aktualny lider zostaje zmieniony a poziom satysfakcji zostaje aktualizowany. Ostatecznie w danej iteracji głównej pętli algorytmu, do zwycięskiego komitetu dołączany zostaje lider po zakończeniu iteracji po wszystkich jeszcze niewybranych kandydatach. Ponieważ poziom zadowolenia z komitetu ustalany jest jako suma pojedynczych zadowoleń wszystkich wyborców, w celu obliczenia funkcji satysfakcji dla komitetu złożonego z wcześniej wybranych kandydatów wraz z dobranym kandydatem wcześniej niewybranym, następuje iteracja po wszystkich wyborcach. Dla każdego wyborcy oceniane jest pojedyncze zadowolenie z komitetu i dodawane do sumy zadowolenia wszystkich wyborców. Pojedyncze zadowolenie liczone jest w ten sposób, że wyciągana jest punktacja badanego kandydata według punktacji k-Bordy, a następnie liczona jest norma ℓ_p z aktualnego zadowolenia

danego wyborcy (z wcześniej wybranych kandydatów) i punktów $k - Bordy$ przyznanych dla badanego kandydata. Po wybraniu zwycięzcy, ostatnim krokiem w głównej pętli algorytmu jest aktualizacja struktur danych reprezentujących aktualny stan algorytmu. Do zwycięskiego komitetu zostaje dołączony wybrany w danej iteracji kandydat, przy jednoczesnym usunięciu go ze zbioru kandydatów jeszcze niewybranych. Aktualizowane zostają również pojedyncze satysfakcje wszystkich wyborców w oparciu o wybranego w danej iteracji kandydata, który w tym momencie staje się zwycięzcą.

6.1.4. Pseudokod

Wejście: k – rozmiar komitetu, porządek preferencji m kandydatów dla n wyborców, p – parametr funkcji zadowolenia
Wyjście: zwycięski zbiór k kandydatów

Algorithm GREEDY(k, PREFERENCE_ORDER, p)

```
RESULT ← Ø
C ← zbiór wszystkich m kandydatów
V ← zbiór wszystkich n głosujących

for i ← 1 upto k
    for each c∈C/RESULT
        for each v∈V
            x ← liczba_punktów_w_porządku_preferencji(c, v)
            zadowolenieV ← wyciągnij_aktualne_zadowolenie(v)
            zadowolenieVwrazZKandydatemC ← ∛(zadowolenieV^p + x^p)
            zadowolenieZKandydataC += zadowolenieVwrazZKandydatemC

        if zadowolenieZKandydataC > zadowolenieZNajlepszegoDotychczasKandydata
            then
            prowadzącyKandydatWDanejIteracji ← c
            zadowolenieZNajlepszegoDotychczasKandydata
                ← zadowolenieZKandydataC

    uaktualnij_zadowolenia_wyborców(prowadzącyKandydatWDanejIteracji )

    RESULT ← RESULT ∪ prowadzącyKandydatWDanejIteracji

return RESULT
```

Rysunek 6.1: Pseudokod algorytmu zachłannego

6.2. Algorytm zachłanny według zasady Chamberlin-Courant'a

Algorytm ten nie aproksymuje wyników wyborów na bazie funkcji satysfakcji opisującej system $\ell_p - Borda$, ale na szczególnym przypadku tej funkcji gdy $p \to \infty$. W związku z tym parametr p nie wpływa na działanie algorytmu.

6.2.1. Idea algorytmu

Algorytm został zaproponowany przez Lu i Boutilier'a. Idea jest taka sama jak dla pierwszego algorytmu zachłannego. W każdej iteracji głównej pętli algorytmu do zwycięskiego komitetu dobrany zostaje jeden kandydat. Zwycięzca w danej iteracji zostaje dobrany według postępowania zachłannego. Wygrywa kandydat, który najwięcej dodaje do ogólnego poziomu zadowolenia wyborców. Różnica w stosunku do pierwszego algorytmu zachłannego polega na innej funkcji zadowolenia, na bazie której inaczej obliczane są pojedyncze zadowolenia wyborców i tym samym inaczej obliczane jest ogólne zadowolenie wyborców z danego komitetu, które jest sumą pojedynczych satysfakcji wszystkich wyborców. W regule $Chamberlin - Courant'a$ położony jest nacisk na regułę reprezentacji proporcjonalnej. Każdy wyborca reprezentowany jest przez jednego, wyraźnego kandydata i tylko punkty tego kandydata wpływają na wartość pojedynczego zadowolenia wyborcy z danego komitetu. Jest to szczególny przypadek uogólnienia systemu $k - Bordy$, w którym $p \to \infty$. Jednak dla odpowiednio dużych wartości parametru p opisana heurystyka dobrze sprawdza się dla uogólnienia systemu $k - Bordy$.

6.2.2. Funkcja zadowolenia

Aktualny poziom zadowolenia wyborców na danym etapie algorytmu określany jest za pomocą funkcji satysfakcji $Chamberlin - Courant'a$:

$$f_{CC}(i_1, i_2, ..., i_k) = \beta(i_1)$$

Jest ona skrajnym przypadkiem dla uogólnienia systemu $k - Borda$, kiedy $p \to \infty$. Zgodnie z tą funkcją, pojedyncze zadowolenie danego wyborcy z komitetu może wzrosnąć tylko w przypadku, gdy do komitetu dołączany zostaje kandydat, któremu dany wyborca przydzielił więcej punktów $k - Bordy$ od punktów aktualnego reprezentanta. Jeżeli taki kandydat zostanie dołączony do zwycięzców, to on staje się reprezentantem dla danego wyborcy w aktualnym stanie algorytmu. Jest to odmienna sytuacja od pierwszego algorytmu zachłannego, w którym pojedyncze zadowolenie danego wyborcy z komitetu rośnie wtedy, gdy nowo dobrany kandydat ma jakiekolwiek przydzielone punkty $k - Bordy$. W tamtym przypadku nowo dołączony zwycięzca nie musi być najlepszy spośród wszystkich aktualnie wybranych kandydatów do zwycięskiego komitetu, aby polepszyć indywidualne zadowolenie danego wyborcy i tym samym zbiorcze zadowolenie wyborców.

6.2.3. Szczegóły implementacyjne

Algorytm zaimplementowany jest w klasie $GreedyCC$ znajdującej się w pakiecie $ecs.elections.algorithms.greedy\_cc$. Główna pętla algorytmu znajduje się w metodzie run. Implementacja i kolejne kroki algorytmu są takie same lub bardzo podobne jak dla pierwszego algorytmu zachłannego zależnego od parametru p. Największa różnica jest w najbardziej wewnętrznej pętli iterującej po wszystkich wyborcach. Zaimplementowana jest tu funkcja zadowolenia pojedynczego wyborcy, różniąca się od analogicznej funkcji dla pierwszego algorytmu zachłannego. Badany kandydat zwiększa wartość pojedynczej satysfakcji danego wyborcy tylko wtedy, gdy ma on przydzielone u wyborcy najwięcej punktów $k - Bordy$ w porównaniu z dotychczasowymi zwycięzcami. W celu sprawdzenia

czy potencjalny zwycięzca w danej iteracji, zwiększa satysfakcję danego wyborcy, porównane są dwie wartości: punkty $k - Bordy$ przyznane przez wyborcę dla potencjalnego zwycięzcy i punkty $k - Bordy$ przyznane przez wyborcę dla najlepszego spośród kandydatów aktualnego zwycięskiego komitetu. Jeśli punkty dla badanego kandydata są większe od punktów najlepszego dotychczasowego zwycięzcy, dodatkowa satysfakcja dla badanego kandydata od danego wyborcy jest uzupełniania o różnicę między punktami $k - Bordy$ przydzielonymi dla badanego kandydata i punktami $k - Bordy$ przydzielonymi dla najlepszego dotychczas kandydata. W danej iteracji wygrywa kandydat, który wniósł najwięcej dodatkowej satysfakcji do ogólnego zadowolenia wyborców. Drugą istotną różnicą wspomnianą przy omawianiu pierwszej, jest sposób porównania w danej iteracji niewybranych jeszcze kandydatów w celu ustalenia najlepszego z nich. Polega on na zbadaniu jaką dodatkową satysfakcję do ogólnego poziomu zadowolenia wnosi badany kandydat i porównywaniu tych dodatkowych satysfakcji, a nie wartości ogólnego poziomu zadowolenia jak w przypadku pierwszego algorytmu.

6.3. Algorytm genetyczny