Implementacje dwóch algorytmów optymalizacyjnych

- algorytm genetyczny,
- algorytm symulowanego wyżarzania

Program składa się z 10 klas.

Klasa Main -> Uruchamia wątek klient oraz oczekuje na połączenie przy pomocy ServerSocket na porcie 1996.

Klasa Client -> reprezentuje stronę klienta który przy pomocy protokołu wysyła prośbę do serwera.

init() -> inicjuje połączenie z ograniczeniem max. 2 sek. na poprawne połączenie z serwerem.

sendPopulation() ->tworzy tablice chromosomów oraz wysyła do serwera chromosomy przy pomocy klasy pomocniczej odpowiedzialnej za tworzenie łańcucha.

offClient()-> zamyka połączenia/strumienie.

Klasa RequestBuilder -> klasa tworzy string odpowiedni do komunikacji z serwerem.

LENGTH: ->długość tablicy

CHROMOSOME: ->wartość chromosomów

Klasa ServerThread -> obsługuje klienta

init() -> pobranie strumieni z gniazda.

checkClientRequest() -> pobiera String od klienta przy użyciu klasy pomocniczej odczytuje informacje zawarte w protokole i zwraca tablice chromosomów. Przekazuje tablice do klas reprezentujących algorytmy. Wynik algorytmu wyświetla pomocniczo na ekranie.

offThread() -> zamyka połączenia/strumienie.

Klasa ParseRequest - > klasa pomocnicza do odczytu informacji z protokołu.

RequestNumber -> typ wyliczeniowy do określenia kolejność przesłanych informacji

getInfo() -> przy pomocy wyrażenia regularnego dzieli napisy

Klasa Chromosome -> model chromosomu

toString()-> pomocniczo do debugowania

Klasa ChartServerThread -> wyświetla wykres przesłany z klas reprezentujących algorytmy.

Klasa GeneticAlgorithm

openConetionWithChart() -> nawiązuje połączenie z serwerem wykresów .

sort() -> sortujemy chromosomy względem wartość funkcji rosenbrocka.

printChromosomes() - > wysyła chromosomy do serwera wykresów.

calculateTotalInverse() -> odwracamy liczbę zwróconą przez funkcje celu i sumujemy całość dla wszystkich chromosomów oraz sprawdzamy czy wartość wyjścia jest osiągana tj. funkcja celu chromosomu = 0.

{

calculateProbability() -> prawdopodobieństwo wylosowania obliczymy odwracając liczbę z funkcji celu, teraz liczba o najmniejszej wartość jest największa = ma największą szanse na wylosowanie.

chooseChromosome() -> wybieramy chromosom z wcześniej zadanym prawdopodobieństwem.

swapChromosomes() ->w tablicy wcześniej wybranych chromosomów następuję mieszanie ich parami.

mutation() -> umożliwia mutowanie chromosomom z zadanym prawdopodobieństwem.

}

offConetionWithChart() -> kończy połączenie z czatem.

Klasa SimulatedAnnealing

openConetionWithChart() -> jw.

{

printChromosomes() -> jw.

calculatePurpose() -> wyświetla obecne chromosomy na konsole.

changeChromosomeValue()-> chromosomy są edytowane o losową wartość oraz sprawdza się czy nowy chromosom jest lepszy przy pomocy funkcji celu, jeżeli jest to zamieniany z starym chromosomem. Nowy chromosom me jeszcze szanse na zamianę mimo bycia gorszym, szansa tym większa im wyższa temperatura.

cooledDown() -> zmniejszamy temperaturę o zadana ilość, szansa na zamianę na gorsze chromosomy maleje.

}

offConetionWithChart() -> jw.