1. Implementacja

a. Osobnik

Osobnik to obiekt klasy Subject. Klasa ta przechowuje informacje dotyczące konkretnego rozwiązania połączeń na płytce. Wszystkie linie przechowywane są w liście obiektów klasy Line. Każda z takich ścieżek składa się z segmentów opisywanych przez kierunek (poziomy lub pionowy) oraz zwrot. Zwrot segmentu jest wartością liczbową całkowitą.

Klasa Subject udostępnia podstawowe funkcjonalności i operatory związane z osobnikami. Między innymi znajdują się tam funkcje tworzące nowego osobnika. Copy tworzy osobnika na podstawie innego podanego w argumencie osobnika. Crossover tworzy osobnika na podstawie dwóch osobników i działa jako operator krzyżowania. Random tworzy osobnika posiadającego linie, które są złożone z podanej ilości losowych segmentów oraz dwóch końcowych segmentów. Shortest tworzy osobnika posiadającego linie łączące punkty w najkrótszy sposób.

Pozostałe metody to mutate, będący operatorem mutacji oraz test sprawdzający czy osobnik spełnia połączenia między punktami.

b. Populacja

Klasa Population przechowuje dane dotyczące danej populacji, takie jak lista osobników i lista ocen tych osobników. Klasa umożliwia tworzenie obiektu na 2 sposoby. Pierwszy tworzy populację osobników losowych, druga tworzy osobników posiadających najkrótsze ścieżki.

Metoda test sprawdza czy wszystkie osobniki spełniają łączenia między punktami. Metoda selectRoulette i selectTournament są operatorami selekcji i zwracają index wybranego osobnika. Metoda evaluate wpisuje do listy wyników aktualną ocenę przechowywanych osobników. Metoda getWorstFitness, getBestFitness oraz getAverageFitness zwracają odpowiednio najgorszy wynik, najlepszy wynik oraz średni wynik w tej populacji. Ostatnia metoda best zwraca index najlepszego osobnika.

c. Operator mutacji

Każde wywołanie algorytmu korzysta z dwóch operatorów mutacji na raz. Ponieważ operator A jest specyficznym przypadkiem operatora B to postanowiłem wykorzystać obydwa operatory jednocześnie. Przy wywoływaniu mutacji osobnika dla losowej ścieżki wywoływana jest metoda mutate_complex. Metoda ta wybiera jeden segment który zostanie przesunięty. Następnie wybierana jest część segmentu która zostanie przesunięta. Następnie losowana jest długość segmentu, który zostanie przesunięty. Przykład: wybrany został segment długości 5, losujemy część poprzedzającą, losujemy długość 3, segment zostaje podzielony na części długości 3 i 2, a część 3 zostaje przesunięta. Jeżeli wylosuje się długość równa długości segmentu to segment nie jest dzielony na dwa (występuje przypadek mutacji typu A). Losowanie długości odbywa się według prawa Benforda. Dzięki temu częściej występują mutacje krótkich odcinków.

d. Operator selekcji

W populacji zostały zaimplementowane dwa operatory selekcji. Pierwszy selectRoulette wypiera osobnika na podstawie metody ruletki. Operator selectTournament

pobiera jeden parametr będący liczbą od 0 do 1. Liczba ta oznacza procent populacji który zostanie wylosowany do turnieju.

e. Operator krzyżowania

Krzyżowanie odbywa się na poziomie klasy Subject. Metoda crossover pobiera dwóch osobników oraz parametr stosunku genów. Parametr ten określa jaką część genów nowy osobnik odziedziczy po “matce” a jaką po “ojcu”.

f. Testowanie

Metody testowe sprawdzają poprawność rozwiązania badając czy osobnik łączy ze sobą wszystkie podane mu punkty. Wychodzenie poza krawędź, krzyżowanie się ścieżek oraz inne błędy są uwzględniane na poziomie oceny osobników.

2. Wyniki

Jako problemu badawczego użyłem zadania testowego numer 3. Na tych danych odpaliłem 11 modeli. Każdy z nich zmieniał jeden z parametrów aby sprawdzić jego wpływ na wyniki. Każdy z 11 modeli został odpalony 10 razy a zebrane dane zostały uśrednione. Zebrane dane umieszczone są w pliku Raport.xlsx.

NAZWA DANYCH	DANE 1	DANE 2	DANE 3	DANE 4	DANE 5	DANE 6	DANE 7	DANE 8	DANE 9	DANE 10	DANE 11
ROZMIAR POPULACJI	250	250	250	250	500	10	250	250	250	250	-
LICZBA POKOLEN	50	50	50	50	10	500	50	50	50	50	-
PRAWDOPODOBIENSTWO KRZYZOWANIA	0.8	0.8	0.3	0.8	0.8	0.8	0.8	0.8	0.8	0.8	-
PRAWDOPODOBIENSTWO MUTACJI	0.3	0.3	0.8	0.3	0.3	0.3	0.3	0.3	0.3	0.3	-
TYP SELEKCJI	TURNIEJ	TURNIEJ	TURNIEJ	RULETKA	TURNIEJ	TURNIEJ	TURNIEJ	TURNIEJ	TURNIEJ	TURNIEJ	-
WIELKOSC TURNIEJU	0.3	0.8	0.3	-	0.4	0.4	0.4	0.4	0.4	0.4	-
STOSUNEK KRZYZOWANIA	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.5	-
ILOSC LOSOWYCH SEGMENTOW POCZATKOWYCH	3	3	3	3	3	3	20	6	0	3	3
NAJLEPSZY OSOBNIK	6590	6790	6620	6870	6610	9080	9150	7180	6590	6610	11140
ŚREDNIA KOŃCOWA	7166	7471	7878	7534	7107	11524	13062	8954	6736	7445	35227.14
ODCHYLENIE STANDARDOWE	515.07669	482.39921	941.54979	452	488.3656	1353.0425	3509.367	1570.224	358.2792	763.4953	586.3554

3. Wnioski

Na podstawie przeprowadzonych badań nad implementacją algorytmu zauważyłem, że:

- Powiększenie rozmiaru populacji wpływa bardzo korzystnie na wyniki algorytmu. Jest to spowodowane większą różnorodnością populacji i mniejszym prawdopodobieństwem utknięcia w minimum lokalnym.
- Wpływ ilości iteracji ma mały wpływ na wyniki końcowe. Należy dostosować ją do innych parametrów.
- Operator selekcji turnieju jest znacznie lepszy od ruletki. Pozwala on tak samo jak większa populacja zmniejszyć ryzyko utknięcia w minimum lokalnym.
- Zbytnie zwiększenie rozmiaru turnieju wpływa niekorzystnie na wyniki algorytmu.
- Zwiększenie prawdopodobieństwa mutacji daje dobre wyniki ale wprowadza spory chaos w tych wynikach, co w niektórych przypadkach może wpłynąć negatywnie na wyniki.
- Stosunek krzyżowania nie wpływa wynik końcowy ale wprowadza w wynikach większą entropię.
- Najlepsze wyniki uzyskałem ograniczając ilość losowych segmentów w ścieżkach do 0. Sądzę, że jest to ściśle powiązane z problemem, który rozwiązywałem. Przy

bardziej skomplikowanej płytce optymalna ilość segmentów losowych mogłaby się różnić.

- Dane 11 to proporcjonalna ilość zupełnie losowych osobników nie będących częścią algorytmu genetycznego. Można zauważyć że nawet najgorszy przypadek algorytmu genetycznego jest lepszy od losowania przypadkowych osobników.