Mateusz Sobieszek
Michał Szmaja

RAPORT KOŃCOWY

PROJEKT Z ALGORYTMÓW HEURYSTYCZNYCH

1. Temat projektu

Przygotować algorytm poszukujący optymalnej sieci kolejowej pomiędzy miastami z danego zbioru. Ponieważ do sieci kolejowej trzeba dostarczyć elektryczność, rozwiązanie powinno uwzględniać położenie elektrowni - im dalej są one od najbliższych torów, tym większy koszt budowy sieci.

2. Realizacja celu projektu

Do osiągnięcia celu projektu posłużyliśmy się algorytmem ewolucyjnym. Program realizujący algorytm został napisany w języku Python. Szczegółowy opis przebiegu oraz charakterystyka algorytmu znajdują się w specyfikacji zadania. Wprowadzone w rozwiązaniu zmiany zostały opisane poniżej.

3. Zmiany w stosunku do specyfikacji

W trakcie pisania rozwiązania nastąpiły zmiany, które wymieniono poniżej.

- Przekształcenie funkcji celu do postaci:

$$F(x,y) = \frac{1}{x \cdot k_1 + y \cdot k_2}$$

gdzie:
x - sumaryczna długość trakcji kolejowej,
k1 - koszt jednostki trakcji kolejowej,
y - sumaryczna długość sieci elektroenergetycznych,
k2 - koszt jednostki sieci elektroenergetycznej.

Stąd też algorytm dąży do maksymalizacji funkcji celu.

- W przypadku krzyżowania krawędzie posiadają połączenia z elektrowniami nie mają pierwszeństwa w stosunku do innych krawędzi, jednak każde kolejne połączenie z elektrownią zwiększa o pewien procent prawdopodobieństwo wyboru krawędzi przy generowaniu potomka. Procent ten jest równy odwrotności długości połączenia energetycznego pomnożony przez iloczyn kosztu jednostki trakcji i kosztu jednostki linii energetycznej. Przy wyznaczaniu wartości jeżeli długość połączenia z elektrownią jest krótsza od jedności przyjmujemy, że wynosi ona 1 aby zmniejszyć wpływ bardzo bliskich miastu (lub leżących w tym samym punkcie) elektrowni. Bez wymienionego usprawnienia takie elektrownie w wysokim stopniu zaburzyłoby rozkład prawdopodobieństw między krawędziami.

4. Opis metaheurystyki oraz jej parametrów

Poniżej opisano parametry wejściowe dla pojedynczego wykonania algorytmu:

- położenie miast - współrzędne punktów na płaszczyźnie reprezentujące położenie miast
- położenie elektrowni - współrzędne punktów na płaszczyźnie reprezentujące położenie elektrowni
- wielkość populacji - liczba całkowita reprezentująca liczbę osobników w populacji
- liczba najlepszych członków populacji, którzy zostaną poddani sukcesji - reprezentowana przez liczbę całkowitą
- liczba iteracji - reprezentowana przez liczbę całkowitą
- prawdopodobieństwo mutacji - zawierające się w przedziale [0.0, 1.0]
- koszt jednostki trakcji - reprezentowany przez liczbę zmiennoprzecinkową
- koszt jednostki linii energetycznej- reprezentowany przez liczbę zmiennoprzecinkową

5. Testy oraz wyniki

W celu uzyskania wyników przebiegu algorytmu wykonano szereg testów. Przypadki testowe, które uwzględniono:

- losowe położenie miast i elektrowni (zestawy TC_003, TC_004)
- równomierne rozmieszczenie miast (zestawy TC_005, TC_006)
- rozmieszczenie miast w skupiskach (zestawy TC_007, TC_008)
- rozmieszczenie miast na krańcach przestrzeni (zestawy TC_010, TC_011)

Dla każdego z wymienionych przypadków opracowano po dwa zestawy danych reprezentujące położenie miast oraz elektrowni. Następnie dla jednego z zestawów przeprowadzono kalibrację w ramach sprawdzenia w jaki sposób parametry wejściowe wpływają na generowane wyniki. Poniżej zaprezentowano argumenty wywołania, które posłużyły ocenie wpływu parametrów na uzyskiwaną funkcję celu:

Rozmiar populacji	Ilość zachowanych osobników	Liczba iteracji	P-ństwo mutacji	Koszt trakcji	Koszt linii energe.	Liczba powtórzeń testu
10	9	500	0.5	1.0	0.5	15
20	10	500	0.5	1.0	0.5	15
30	10	500	0.5	1.0	0.5	15
40	10	500	0.5	1.0	0.5	15
50	10	500	0.5	1.0	0.5	15
20	18	500	0.5	1.0	0.5	15
20	16	500	0.5	1.0	0.5	15
20	14	500	0.5	1.0	0.5	15
20	12	500	0.5	1.0	0.5	15
20	10	500	0.5	1.0	0.5	15
20	8	500	0.5	1.0	0.5	15
20	6	500	0.5	1.0	0.5	15

20	4	500	0.5	1.0	0.5	15
20	2	500	0.5	1.0	0.5	15
20	10	50	0.5	1.0	0.5	15
20	10	100	0.5	1.0	0.5	15
20	10	200	0.5	1.0	0.5	15
20	10	500	0.5	1.0	0.5	15
20	10	1000	0.5	1.0	0.5	15
20	10	2500	0.5	1.0	0.5	15
20	10	5000	0.5	1.0	0.5	15
20	10	500	0.1	1.0	0.5	15
20	10	500	0.2	1.0	0.5	15
20	10	500	0.3	1.0	0.5	15
20	10	500	0.4	1.0	0.5	15
20	10	500	0.5	1.0	0.5	15
20	10	500	0.6	1.0	0.5	15
20	10	500	0.7	1.0	0.5	15
20	10	500	0.8	1.0	0.5	15
20	10	500	0.9	1.0	0.5	15
20	10	500	0.5	0.5	0.5	15
20	10	500	0.5	1.0	0.5	15
20	10	500	0.5	2.0	0.5	15
20	10	500	0.5	5.0	0.5	15
20	10	500	0.5	50.0	0.5	15

Zaznaczone kolorem jasnoczerwonym komórki wyróżniają argumenty, które były zmieniane względem domyślnie dobranego zestawu parametrów (tutaj oznaczonego na zielono).

Uzyskane wyniki dla zestawów TC_003, TC_005, TC_007, TC_010 zosty pokrótce opisane poniżej.

- Rozmiar populacji

  Niewielka liczba osobników (około 10) w populacji powoduje, że program potrzebuje wiele iteracji, aby osiągnąć rozwiązanie o wyższej wartości funkcji celu. Zwiększenie wartości tego parametru poprawia czas w jakim osiągane są nowe lepsze rozwiązania, ale jedynie do pewnej wartości granicznej (zwykle 20 - 30 osobników). Rozmiar populacji przekraczający tę wartość jedynie w znikomym stopniu wpływa na efektywność działania programu.

- Ilość zachowanych osobników

  Liczba zachowanych osobników oddziaływuje na efektywność działania algorytmu w całym zakresie swoich wartości. W przypadku, gdy zachowywane jest jedynie kilka najlepszych osobników (od 2 do 4), bądź prawie wszystkie najlepsze osobniki występują w następnej generacji program potrzebuje wiele czasu, aby ostatecznie znaleźć rozwiązanie bliskie optymalnemu. Najlepsze rezultaty program

osiąga dla wartości niewiele mniejszej niż połowa liczby wszystkich osobników (wahania w granicach 6 do 8).

- Liczba iteracji

  Zgodnie z przewidywaniami zwiększenie całkowitej liczby iteracji pozwala na znajdowanie coraz lepszych rozwiązań. Od momentu, w którym zostanie znalezione optymalne rozwiązanie (wartości wahają się w przedziale 450 do 500 iteracji) najlepszy osobnik dla każdej iteracji pozostaje stały.

- Prawdopodobieństwo mutacji

  Niska zadana wartość prawdopodobieństwa (od 0.1 do 0.2) mutacji sprawia, że program potrzebuje większej liczby iteracji, aby znaleźć lepsze rozwiązanie, gdy w danej chwili udało mu się otrzymać rozwiązanie będące optimum lokalnym. Zwiększenie tej wartości (około 0.4) znacząco poprawia przebieg programu. Wartości prawdopodobieństwa mutacji powyżej tej wartości utrudniają systematyczne osiągania lepszych wyników w ramach operacji krzyżowania.

- Koszt trakcji i linii energetycznej

  Przebieg programu jest spójny dla różnych wartości kosztów odcinków sieci i zależy jedynie od zadanych wartości pozostałych parametrów wymienionych powyżej.

W poniższej tabeli znajdują się szacowne wartości optymalne dla poszczególnych przypadków uzyskane w trakcie kalibracji parametrów algorytmu.

	TC_003	TC_005	TC_007	TC_010
Rozmiar populacji	30	30	20 - 30	30
Ilość zachowanych osobników	8	6 - 10	6 - 8	8
Liczba iteracji	450	450	450	500
P-ństwo mutacji	0.4	0.4	0.4 - 0.5	0.3 - 0.4
Koszt trakcji	-	-	-	-
Koszt linii energe.	-	-	-	-

Szczegółowe wykresy z przeprowadzonych testów (dla przypadku TC_003) można znaleźć w załączniku do raportu w podpunkcie 1.

Na podstawie wyników uzyskanych podczas kalibracji dobrano argumenty wywołania mające zapewnić najbardziej efektywne działanie algorytmu. Poniżej zaprezentowano wartości dla poszczególnych parametrów.

Rozmiar populacji	Ilość zachowanych osobników	Liczba iteracji	P-ństwo mutacji	Koszt trakcji	Koszt linii energe.	Liczba powtórzeń testu
40	8	500 - 1000	0.4	1.0	0.5	15
30	14	500	0.4	1.0	0.5	15

W każdym z wywołanych przebiegów algorytmu otrzymano takiego samego osobnika dla wszystkich przypadków testowych za wyjątkiem TC_010 gdzie udało się uzyskać 3 osobniki o identycznych wartościach funkcji celu. Rysunki przedstawiające wyniki wykonanych testów oraz najlepsze osobniki (funkcja celu, łączna długość sieci) znajdują się w załączniku do raportu w podpunkcie 2 oraz 3.

6. Podsumowanie

Na podstawie wyników przeprowadzonych testów (przedstawionych za pomocą wykresu funkcji przystosowania względem liczby iteracji oraz mapy sieci) można stwierdzić, że przebieg programu oraz generowane przez niego rozwiązania są optymalne. Jest to szczególnie widoczne na wykresie, który obejmuje uśrednione wartości funkcji celu wraz z odchyleniami dla wielokrotnego wywołania programu rozwiązującego ten sam problem z takimi samymi zadanymi parametrami. Wraz ze wzrostem liczby przeprowadzonych iteracji odchylenia od wartości średniej maleją, a program wyraźnie zbiega do rozwiązania optymalnego.