Sprawozdanie

Algorytmy Metaheurysyczne - Labolarorium

Radosław Wojtczak, Witold Karaś

1 Wstęp:

1.1 Algorytm Tabu Search

Algorytm Tabu Search jest metaheurystyką opierającą się na przeszukiwaniu przestrzeni stworzonej ze wszystkich możliwych rozwiązań, za pomocą sekwencji ruchów. Istotą ów algorytmu jest istnienie ruchów niedozwolonych (czyli ruchów tabu). Dzięki swojej strukturze algorytm unika ciągłego rozpatrywania jednego optimum lokalnego.
Nasza implementacja przewiduje dwa sposoby determinowania sąsiedztwa: poprzez ruchy **swap** oraz **invert**. Ruchy **swap** polegają na zamienianiu pozycjami dwóch miast z danej permutacji, natomiast ruchy typu **invert** polegają na odwróceniu kolejności w jakiej występują wszystkie miasta w danej permutacji między dwoma losowo wybranymi indeksami.

Złożoność obliczeniowa:
Główna pętla algorytmu odbywa się aż do momentu osiągnięcia kryterium zatrzymującego pracę. W naszej implementacji jest to liczba iteracji, która jest podawana przez użytkownika w momencie uruchomienia programu. Uznajemy więc, iż liczba iteracji jest liczbą stałą. Następnie odbywa się generowanie sąsiedztwa dla obecnej permutacji. Generowanie sąsiedztwa odbywa się w pętli for, której liczba iteracji jest podawana przez użytkownika w momencie uruchomienia programu.Na potrzeby ów analizy ustalmy, iż użytkownik generuje liczbę sąsiadów równą liczbie miast dla danej instancji (jest to zalecane rozwiązanie dla większych instancji). Po dokonaniu powyższego założenia zauważamy, iż pojedyncza realizacja funkcji generującej sąsiadów wykonuje się w czasie liniowym (podobnie jak w poprzednim zadaniu, wbudowana funkcja *reverse* w języku python wykonuje się w **czasie liniowym**, poza ów funkcją wykonywane są jedynie takie operacje jak losowanie, o stałym czasie realizacji, czy kopiowanie tablicy, które również jest realizowane w czasie liniowym). Ostatecznie wnioskujemy, iż złożoność obliczeniowa funkcji *get neighbors* wynosi $O(n^2)$. Następnie dochodzi do porównania wyników otrzymanych przez funkcję *fitness* (która jest odwrotnością funkcji kosztu) dla każdego z wygenerowanych sąsiadów. Biorąc pod uwagę, iż liczba sąsiadów jest liniowo zależna od liczby miast oraz funkcja *cost* wykonuje się w

czasie liniowym, dokonanie sprawdzenia, czy któryś z sąsiadów zwraca lepszy wynik niż aktualna permutacja również odbywa się w złożoności $O(n^2)$. Wszystkie następne operacje wykonywane w głonej pętli to operacje przypisania bądź porównania, które odbywają się w czasie stałym. Dodatkowo funkcja *shuffle* również odbywa się w czasie liniowym.
Wniosek: Funkcja *tabu* realizowana jest w czasie $O(n^2)$.
Złożoność pamięciowa:
Algorytm w swoim procesie często manewruje permutacjami miast, stąd wiemy, iż złożoność pamięciowa jest liniowo zależna od liczby miast. Jedynym wyjątkiem jest lista sąsiadów (gdzie w zależności od wprowadzonych przez użytkownika danych trzymamy odpowiednią liczbę permutacji) oraz lista tabu, która również zależy od wprowadzonych danych. Oznaczając jako k większą z wartości sąsiedztwa oraz pojemności listy tabu stwierdzamy, iż złożoność pamięciowa funkcji *tabu* wynosi $O(k * n)$

2 Przypadek Testowy 1 - Algorytm genetyczny - zależność PRD od ilości iteracji

2.1 Cel:

W tej części zostaną ze sobą porównane PRD rozwiązania algorytmu genetycznego w zależności od liczby iteracji.

2.2 Założenia:

Do badania tego przypadku została wykorzystana instancja **berlin52.tsp**. Dodatkowo współczynnik mutacji został ustalony na 0.15, współczynnik selekcji na 0.95. Ponad to wielkość populacji została ustalona na 10, 20, 30, 40. Badane iteracje były z zakresu 1000, 2000, 3000, ... 10000. Dla każdej wielkości populacji testy zostały przeprowadzone 10 razy.

2.3 Wyniki:

Poniższe tabele przedstawiają wyniki testów, odchylenie standardowe oraz błęd standardowy.

	10	20	30	40
1000	80.47	61.66	43.67	57.52
2000	62.40	48.94	39.31	45.41
3000	47.33	45.31	38.33	38.38
4000	36.40	39.02	32.11	32.29
5000	49.98	34.98	30.57	25.19
6000	34.92	33.88	32.87	30.54
7000	37.85	34.57	29.66	25.77
8000	39.34	34.57	25.27	27.23
9000	37.54	26.08	25.11	22.17
10000	38.83	28.65	24.14	22.52

Tabela 1: Wyniki to wartości PRD (wyrażone w procętach), uśrednione.

Odchylenie standardowe oraz błąd standardowy zostały obliczone według wzorów:
Odchylenie standardowe:

$$\sigma = \sqrt{\frac{\sum_{n=1}^{10}(\bar{x} - x_n)^2}{10}}$$

Błąd standardowe:

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{10}}$$

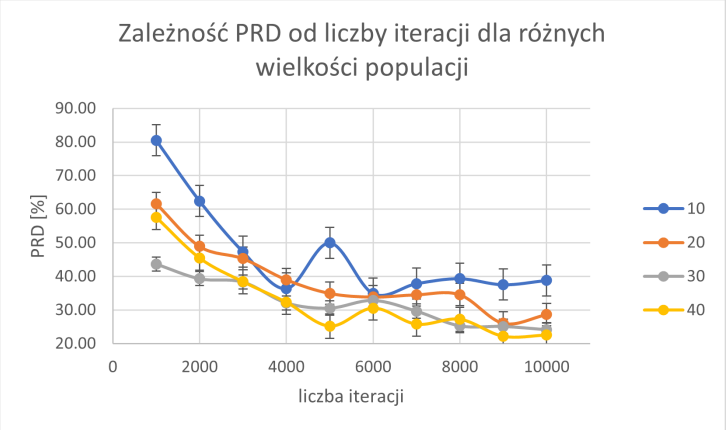
SD	10	20	30	40
1000	13.96	11.90	9.46	18.87
2000	18.91	11.07	8.42	16.18
3000	12.87	12.22	10.20	12.82
4000	15.65	16.56	9.80	10.56
5000	18.25	10.17	14.08	5.57
6000	11.27	10.79	7.59	8.99
7000	11.27	15.08	10.46	8.64
8000	13.86	9.02	9.84	9.10
9000	15.78	8.49	7.34	8.81
10000	17.20	9.70	8.36	7.42

Tabela 2: SD - Odchylenie standardowe

SE	10	20	30	40
1000	4.41	3.76	2.99	5.97
2000	5.98	3.50	2.66	5.12
3000	4.07	3.86	3.23	4.06
4000	4.95	5.24	3.10	3.34
5000	5.77	3.22	4.45	1.76
6000	3.56	3.41	2.40	2.84
7000	3.56	4.77	3.31	2.73
8000	4.38	2.85	3.11	2.88
9000	4.99	2.68	2.32	2.79
10000	5.44	3.07	2.64	2.35

Tabela 3: SE - Błąd standardowy

2.4 Wykresy:



Rysunek 1: Zależność PRD od liczby iteracji dla różnych wielkości populacji

Na wykresie przedstawione są średnie wartości PRD dla badanych danych. Uśrednione wartości PRD maleją logarytmicznie, wraz ze wzrostem liczby iteracji.

2.5 Wnioski:

Zgodnie z oczyekiwaniami, zwiększanie ilości iteracji zwiększa jakość rozwiązań jednak w okolicach 5000 iteracji zaczyna poprawa zaczyna być coraz mniejsza - zależniść nie jest liniowa, raczej logarytmiczna. Dodatkowo z wykresu można odczytać że zwiększanie rozmiaru populacji nie koniecznie zwiększa jakość rozwiązania dla jednakowej ilości iteracji.

3 Przypadek Testowy 2 - Algorytm genetyczny - zależność PRD od wskaźnika mutacji

3.1 Cel:

W tej części zostaną ze sobą porównane PRD rozwiązania algorytmu genetycznego w zależności od wskaźnika mutacji.

3.2 Założenia:

Do badania tego przypadku została wykorzystana instancja **berlin52.tsp**. Dodatkowo współczynnik selekcji został ustalony na 0.95. Ponad to wielkość populacji została ustalona na 10, liczba iteracji to 10000. Badane współczynniki mutacji $m \in 0.05, 0.15, 0.25...0.95$. Dla każdej wielkości współczynnika mutacji test został przeprowadzony trzykrotnie.

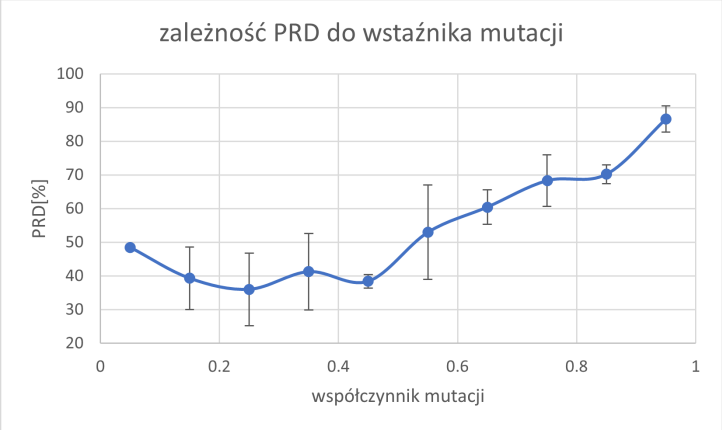
3.3 Wyniki:

Poniższa tabela przedstawia wyniki testów, odchylenie standardowe oraz błęd standardowy.

m	PRD	SD	SE
0.05	48.47	1.54	0.89
0.15	39.30	16.07	9.28
0.25	36.01	18.63	10.76
0.35	41.24	19.61	11.32
0.45	38.39	3.52	2.03
0.55	52.92	24.27	14.01
0.65	60.43	8.97	5.18
0.75	68.26	13.21	7.62
0.85	70.17	4.90	2.83
0.95	86.56	6.82	3.94

Tabela 4: PRD - wyrażone w procętach [%], SD - Odchylenie standardowe, SE - Błąd standardowy

3.4 Wykresy:



Rysunek 2: zależność PRD od współczynnika mutacji

Na wykresach przedstawione są średnie wartości PRD dla badanych danych. Uśrednione wartości PRD maleją logarytmicznie, wraz ze wzrostem liczy iteracji.

Odchylenie standardowe oraz błąd standardowy zostały obliczone według wzorów:
Odchylenie standardowe:

$$\sigma = \sqrt{\frac{\sum_{n=1}^{10}(\bar{x} - x_n)^2}{10}}$$

Błąd standardowe:

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{10}}$$

3.5 Wnioski:

Zmiana współczynnik mutacji przynosi niejednoznaczne rezultaty, przez wbudowaną w mutację losowość. Można jednak zauważyć pewne zależności. Dla współczynnika mutacji 0.05 odchylenie standardowe wyników jest najmniejsze co świadczy stałości rozwiązania przy niskim poziomie mutacji. Przy szansie na mutację 0.25 algorytm osiągnął minimum w testach, jednak odchylenie standardowe było duże. W punkcie 0.45, na wykresie pojawia się minimum lokalne, w tym punkcie odchylenie standardowe znów jest niskie. po przekroczeniu 0.5 algorytm daje coraz gorsze wyniki co ma związek z częstymi mutacjami - bardziej zaczyna przypominać k-random.

4 Przypadek Testowy 2 - Algorytm genetyczny - zależność PRD od użytej metody selekcji

4.1 Cel:

W tej części zostaną ze sobą porównane PRD rozwiązania algorytmu genetycznego w zależności użytej metody selekcji.

4.2 Założenia:

Do badania tego przypadku została wykorzystana instancja **berlin52.tsp**. Dodatkowo współczynnik selekcji został ustalony na 0.95. Ponad to wielkość populacji została ustalona na 10, liczba iteracji to 10000. Zmienne współczynniki mutacji $m \in 0.05, 0.15, 0.25...0.95$ oraz 2 metody selekcji. Dla każdej wielkości współczynnika mutacji test został przeprowadzony trzykrotnie.

4.3 Metody selekcji:

4.3.1 Ruletka:

Metoda selekcji przedstawicieli populacji polega na znormalizowaniu współczynnika dopasowania tak że suma wszystkich wynosi 1 odwrotnie proporcjonalnie do długości ścieżki. Następnie losowana jest liczba z przedziału $[0, 1)$. Do nowej populacji dopisywana jest jedna ścieżka aż nowa populacja będzie tego samego rozmiaru co poprzednia.

4.3.2 Odcięcie:

Metoda selekcji przedstawicieli populacji polega na posortowaniu populacji względem długości rozwiązania i odcięcie sparametryzowanej najgorszej części (np. 10%). aby uzupełnić populację najlepsze odobniki są doklejane po raz kolejny.

4.4 Wyniki:

Poniższa tabela przedstawia wyniki testów, odchylenie standardowe oraz błęd standardowy. Odchylenie standardowe oraz błąd standardowy zostały obliczone według wzorów:
Odchylenie standardowe:

$$\sigma = \sqrt{\frac{\sum_{n=1}^{3}(\bar{x} - x_n)^2}{3}}$$

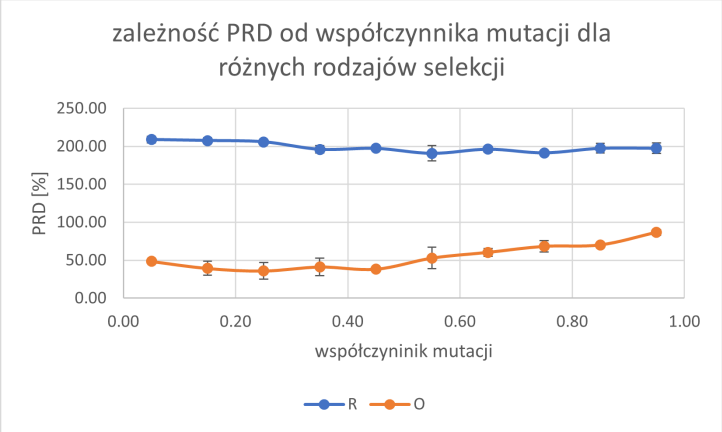
Błąd standardowe:

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{3}}$$

m	R	SD_R	SE_R	O	SD_O	SE_O
0.05	208.90	7.58	4.38	48.47	1.54	0.89
0.15	207.63	6.68	3.86	39.30	16.07	9.28
0.25	205.91	2.91	1.68	36.01	18.63	10.76
0.35	196.12	8.89	5.13	41.24	19.61	11.32
0.45	197.33	4.13	2.38	38.39	3.52	2.03
0.55	190.82	17.28	9.98	52.92	24.27	14.01
0.65	196.19	6.80	3.92	60.43	8.97	5.18
0.75	191.50	4.34	2.50	68.26	13.21	7.62
0.85	197.37	11.00	6.35	70.17	4.90	2.83
0.95	197.44	11.80	6.81	86.56	6.82	3.94

Tabela 5: m - współczynnik mutacji, R - PRD dla selekcji przez ruletkę, SD_R - odchylenie standardowe dla selekcji przez ruletkę, SE_R - Błąd standardowy dla selekcji przez ruletkę, O - PRD dla selekcji przez odcięcie, SD_O - odchylenie standardowe dla selekcji przez odcięcie, SE_O - Błąd standardowy dla selekcji przez odcięcie

4.5 Wykresy:



Rysunek 3: zależność PRD od współczynnika mutacji

Na wykresach przedstawione są średnie wartości PRD dla badanych danych.

4.6 Wnioski:

Selekcja przez odcięcie sprawdza się zdecydowanie lepiej niż selekcja metodą ruletki. W metodzie ruletki najsłabsze osobniki mają szansę przejść do krzyżowania gdzie w metodzie odcięcia są zawsze usuwane i krzyżują się tylko najlepsze cechy.