Emilia Pawlaszek, 241279
GRUPA PN/17:05

Wrocław, 16 grudnia 2019

Sprawozdanie z projektu przedmiotu „Projektowanie Efektywnych Algorytmów”

Rok akad. 2019/2020, kierunek: INF

Etap 2. Algorytmy lokalnego przeszukiwania.

Spis treści

1 Wstęp **2**

2 Przeszukiwanie z zabronieniami (Tabu Search) **2**

3 Implementacja algorytmu **3**

4 Badania **3**
4.1 Pomiary dla TSP. 4
4.2 Pomiary dla ATSP. 5
4.3 Pomiary dla SMALL. 6
4.4 Porównanie Tabu Search z Brute Force i Dynamic Programming. 8
4.5 Zmiana długości kadencji. 8

5 Wnioski **9**

1 Wstęp

W etapie 2 projektu zostały kontynuowane prace nad rozwiązaniem asymetrycznego problemu komiwojażera (problem ten został dokładnie opisany w sprawozdaniu do etapu 1). W poprzednim etapie należało zaimplementować algorytmy dokładne. Algorytmy takie zawsze znajdują rozwiązanie optymalne, jednakże charakteryzują się niską wydajnością dla dużych instancji problemu. W tym etapie należało zaimplementować algorytmy lokalnego przeszukiwania. Nie dają one gwarancji znalezienia optymalnego rozwiązania, ale są dużo wydajniejsze - działają szybciej dla dużych instancji problemu.

2 Przeszukiwanie z zabronieniami (Tabu Search)

Algorytm tabu search jest algorytmem przeszukiwania lokalnego. Oznacza to, że rozwiązań optymalnych szukamy w najbliższym sąsiedztwie, na przykład poprzez zamianę kolejności sąsiadujących wierzchołków. Sprowadza się to do znalezienia minimum lokalnego (czyli ścieżki o najmniejszym koszcie) w danym sąsiedztwie.

Aby uniknąć sprawdzaniu tych samych rozwiązań wykorzystuje się listę tabu (listę ruchów zakazanych). Do listy dodawane są ostatnio wybrane rozwiązania. Jeśli jakiś ruch znajduje się już na liście tabu to nie zostaje wykonany.

Możliwe jest również określenie długości listy tabu - kadencji. Dany ruch będzie się znajdował na liście tabu tylko przez określoną przez nas kadencję. Jeśli kadencja będzie zbyt krótka, zwiększy się ryzyko wpadnięcia w cykl w okolicach lokalnego minimum. Jeśli będzie za długa - pogorszy się jakość rozwiązań ze względu na brak dokładniejszego przeszukiwania sąsiedztwa. Zaleca się aby kadencja była zależna od wielkości instancji - im większy rozmiar tym dłuższe kadencje.

Z kryterium aspiracji korzysta się wtedy, gdy chcemy sprawdzić czy w otoczeniu rozwiązania zostało znalezione dobre rozwiązanie będące na liście tabu. Jeśli dane rozwiązanie będzie spełniać nasz warunek to będzie brane jako kolejne, mimo że jest na liście ruchów zakazanych.

W algorytmie można też zaimplementować strategię dywersyfikacji, tzn. jeśli w obrębie danego sąsiedztwa, w zadanej ilości iteracji, nie zostanie znalezione rozwiązanie lepsze niż globalne - generowane jest nowe rozwiązanie, w sąsiedztwie którego będą szukane kolejne rozwiązania. Funkcja zdarzeń krytycznych przyjmuje wartość true jeżeli przez k kolejnych iteracji nie zostało znalezione lepsze rozwiązanie lub algorytm wykonał k iteracji od nowego rozwiązania startowego. Funkcja ta uruchamia procedurę generowania nowego rozwiązania początkowego.

Sposoby generowania rozwiązania startowego:

- sposób naturalny (połączenie kolejnych wierzchołków),
- algorytm zachłanny,
- losowe,
- inne.

Warunek zakończenia algorytmu:

- wykonanie zadanej liczby iteracji,
- uzyskanie wyniku optymalnego,
- wyczerpanie czasu.

3 Implementacja algorytmu

Poniżej znajduje się algorytm tabu search w postaci pseudokodu.

```
wybierz lub wylosuj punkt startowy x_0 ∈ X
x_opt ← x_0
tabu_list ← ∅
repeat
   x_0 ← AspirationPlus(x_0)
   if f(x_0) > f(x_opt) then
      x_opt ← x_0
   zweryfikuj tabu_list
   ∀element ∈ tabu_list do
      − − kadencja_i
      if kadencja_i = 0 then
         usuń element(atrybut_i, kadencja_i) z tabu_list
   if CriticalEvents() then
   x_0 ← Restart() (Dywersyfikacja)
   if f(x_0) > f(x_opt) then
      x_opt ← x_0
until warunek zakończenia
```

4 Badania

Obliczenia zastały wykonane na laptopie z procesorem i5-7200, kartą graficzną NVIDIA GeForce 940, 8GB RAM i DYSK SSD.

Algorytm jako dane wejściowe przyjmuje:

1. liczbę iteracji,
2. długość kadencji,
3. wartość aspiracji (wynik dla którego omijamy tabu),
4. maksymalny licznik aspiracji (iteracje do nowego rozwiązania początkowego).

Pomiary wykonywane były jeden raz, nie warto było robić np 10 powtórzeń i wyciągać średniej, bo wynik i tak za każdym razem wychodzi taki sam dla tych samych danych wejściowych.

Wartości optymalne, przetestowane empirycznie dla tego programu to kolejno: liczba iteracji, 10, -30, 10.

4.1 Pomiary dla TSP.

Tablica 1: Pomiary dla TSP (1000, 10, -30, 10).

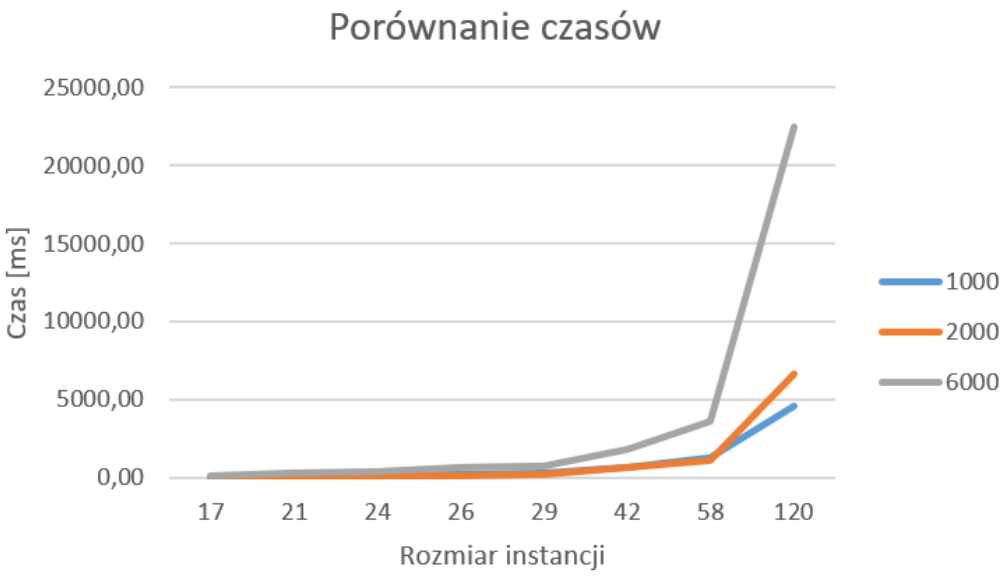
-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
17	2085	2399	15,06	24,54
21	2707	3554	31,29	72,96
24	1272	1677	31,84	147,70
26	937	992	5,87	248,92
29	1610	2104	30,68	367,14
42	699	699	0,00	710,38
58	25395	37572	47,95	1314,87
120	6942	15090	117,37	4578,29
	średni błąd:		**35,01**	

Tablica 2: Pomiary dla TSP (2000, 10, -30, 10).

-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
17	2085	2399	15,06	42.9195
21	2707	3554	31,29	99.9533
24	1272	1637	28,69	128.054
26	937	1011	7,90	185.602
29	1610	2032	26,21	223.042
42	699	699	0,00	641.726
58	25395	37572	47,95	1128.57
120	6942	15090	117,37	6705.36
	średni błąd:		**34,31**	

Tablica 3: Pomiary dla TSP (6000, 10, -30, 10).

-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
17	2085	2399	15,06	146.527
21	2707	3500	29,29	355.676
24	1272	1588	24,84	394.41
26	937	1000	6,72	686.861
29	1610	2104	30,68	740.867
42	699	699	0,00	1879.27
58	25395	37572	47,95	3626.8
120	6942	15090	117,37	22509.8
	średni błąd:		**33,99**	



Rysunek 1: Wykres zależności czasu od liczby iteracji dla TSP.

4.2 Pomiary dla ATSP.

Tablica 4: Pomiary dla ATSP (1000, 10, -30, 10).

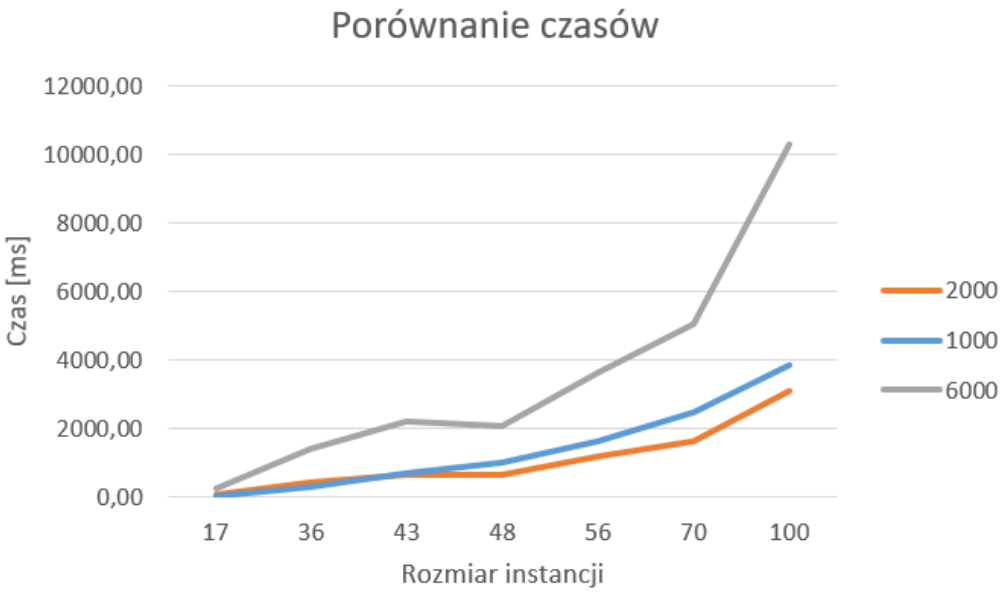
-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
17	39	87	123,08	40,32
36	1473	2031	37,88	292,02
43	5620	5713	1,65	686,32
48	14422	20134	39,61	1015,50
56	1608	2210	37,44	1621,01
70	38673	42806	10,69	2460,14
100	36230	62271	71,88	3858,34
323	1326	1706	28,66	65674,70
	średni błąd:		**43,86**	

Tablica 5: Pomiary dla ATSP (2000, 10, -30, 10).

-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
17	39	51	30,77	69,52
36	1473	1890	28,31	432,85
43	5620	5704	1,49	662,22
48	14422	20134	39,61	644,90
56	1608	2215	37,75	1167,82
70	38673	42806	10,69	1628,60
100	36230	62339	72,06	3099,82
323	1326	1705	28,58	119347,00
	średni błąd:		**31,16**	

Tablica 6: Pomiary dla ATSP (6000, 10, -30, 10).

-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
17	39	53	35,90	221,08
36	1473	2081	41,28	1405,00
43	5620	5714	1,67	2176,54
48	14422	19706	36,64	2081,71
56	1608	2215	37,75	3613,51
70	38673	42806	10,69	5031,45
100	36230	62380	72,18	10290,60
323	1326	1705	28,58	
	średni błąd:		**33,09**	



Rysunek 2: Wykres zależności czasu od liczby iteracji dla ATSP.

4.3 Pomiary dla SMALL.

Tablica 7: Pomiary dla SMALL (1000, 10, -30, 10).

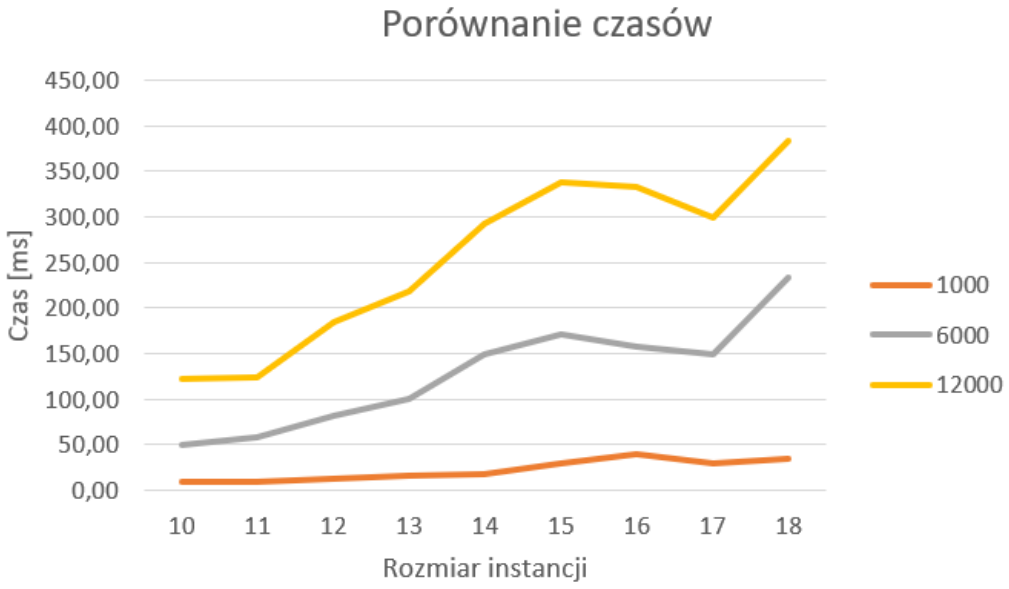
-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
10	212	343	61,79	9,77
11	202	234	15,84	9,85
12	264	264	0,00	13,19
13	269	269	0,00	16,46
14	125	381	204,80	17,46
15	291	457	57,04	30,26
16	156	378	142,31	40,10
17	2085	2399	15,06	29,44
18	187	441	135,83	33,98
	średni błąd:		**70,30**	

Tablica 8: Pomiary dla SMALL (6000, 10, -30, 10).

-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
10	212	563	165,57	50,43
11	202	234	15,84	57,75
12	264	264	0,00	82,27
13	269	269	0,00	100,42
14	125	202	61,60	149,87
15	291	405	39,18	171,80
16	156	384	146,15	157,35
17	2085	2399	15,06	149,87
18	187	417	122,99	233,65
	średni błąd:		**62,93**	

Tablica 9: Pomiary dla SMALL (12000, 10, -30, 10).

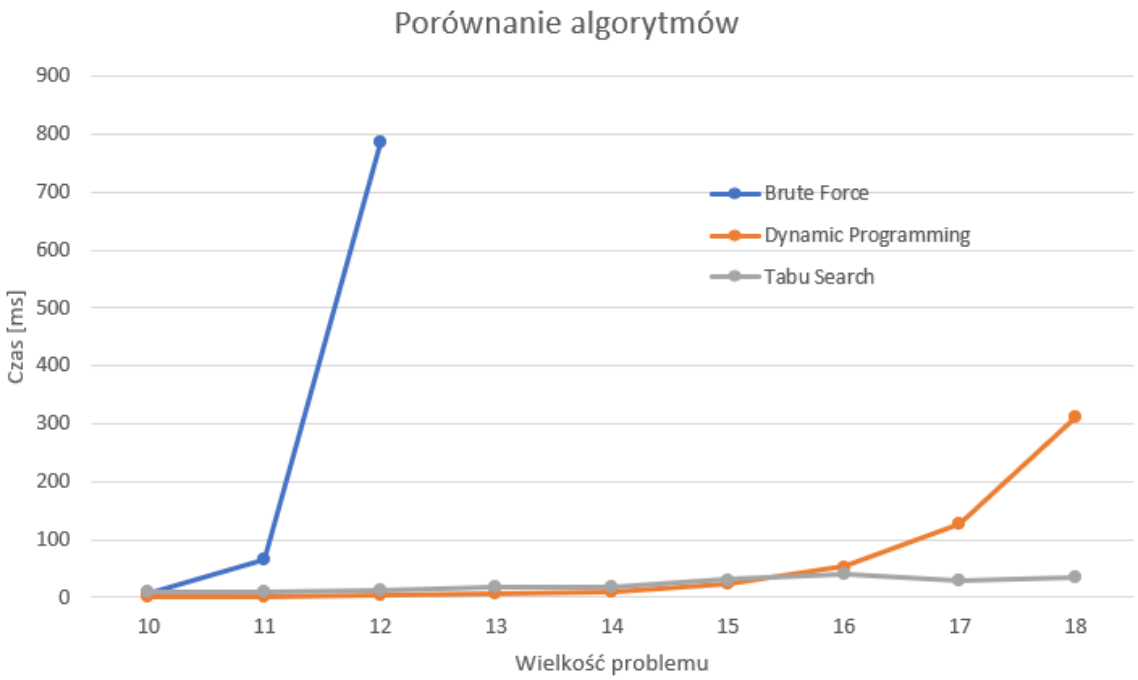
-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
10	212	563	165,57	123,07
11	202	234	15,84	123,66
12	264	264	0,00	184,70
13	269	269	0,00	217,66
14	125	202	61,60	292,11
15	291	429	47,42	338,13
16	156	288	84,62	332,81
17	2085	2399	15,06	299,94
18	187	417	122,99	383,17
	średni błąd:		**57,01**	



Rysunek 3: Wykres zależności czasu od liczby iteracji dla SMALL.

-

4.4 Porównanie Tabu Search z Brute Force i Dynamic Programming.



Rysunek 4: Wykres zależności czasu od liczby iteracji dla zbioru SMALL.

4.5 Zmiana długości kadencji.

Tablica 10: Pomiary dla TSP (2000, 30, -30, 10).

-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
17	2085	2945	41,25	45.0069
21	2707	3759	38,86	75.3112
24	1272	1858	46,07	127.77
26	937	1216	29,78	176.958
29	1610	2155	33,85	200.444
42	699	699	0,00	641.179
58	25395	42830	68,66	1050.23
120	6942	11879	71,12	6619.83
	średni błąd:		**41,20**	

Tablica 11: Pomiary dla ATSP (2000, 30, -30, 10).

-Rozmiar-	—Best–	–Wynik–	-Błąd [%]	—Czas–
17	39	130	233,33	65.9141
36	1473	1989	35,03	374.255
43	5620	5689	1,23	661.367
48	14422	19735	36,84	632.789
56	1608	2408	49,75	1137.91
70	38673	43260	11,86	1406.56
100	36230	60839	67,92	3237.36
323	1326	1651	24,51	
	średni błąd:		**57,56**	

5 Wnioski

Algorytm z podanymi danymi wejściowymi działa zdecydowanie lepiej dla dużych instancji - zauważalny jest dla nich dużo mniejszy błąd. Złym rozwiązaniem jest podawanie danych wejściowych „na sztywno", powinno się je uzależnić od rozmiaru i rodzaju problemu.
Zmiana długości listy tabu nie poprawia jakości otrzymywanych wyników.
Dla małych instancji (SMALL) lepiej zastosować algorytm dokładny Dynamic Programming. Daje nam on optymalne wartości, a jego czas działania jest porównywalny do czasu działania algorytmu Tabu Search. Jednak dla większych instancji (TSP, ATSP) lepiej zastosować Tabu Search, ponieważ dla tych instancji różnica czasów działania algorytmów jest już znacząca.

Literatura

[1] PEA, w5, Tabu Search (Poszukiwanie z zakazami).
`http://www.zio.iiar.pwr.wroc.pl/pea/w5_ts.pdf`