Piotr Wodzyński
225968

Projektowanie efektowych algorytmów.
Projekt 2
Algorytm poszukiwania lokalnego z zakazami

1. Wstęp teoretyczny

Algorytm poszukiwania lokalnego z zakazami (z ang. Tabu Search) to algorytm metahuerystyczny, którego działanie opiera się na iteracyjnym przeszukiwaniu przestrzeni rozwiązań, wykorzystując sąsiedztwo pewnych elementów tej przestrzeni oraz zapamiętując przy tym przeszukiwaniu ostatnie ruchy, dopóki nie zostanie spełniony warunek końcowy. Ruchy zapisywane są w postaci atrybutów przejścia (parametry opisujące jednoznacznie wykonany ruch) na liście tabu. Obecność danego ruchu na liście tabu jest tymczasowa (zazwyczaj na określoną liczbę iteracji). Dzięki zastosowaniu takiego rozwiązania jak lista tabu można zmniejszyć ryzyko powstawania zapętleń i zmusić algorytm do przeszukiwania nowych obszarów które są wybierane losowo.

2. Opis zaimplementowanego algorytmu w pseudokodzie

2.0 Wytłumaczenie oznaczeń funkcji

-c_{xy} - koszt przejścia z wierzchołka y do wierzchołka x

-tab_{xy} - wartość listy tabu dla parametrów x i y

-c$_path_i$ - kolejne wierzchołki grafu w ścieżce dla pojedynczej iteracji

-t$_path_length$ - tymczasowa zmienna zawierająca długość ścieżki

-b_path_i - kolejne wierzchołki grafu w ścieżce o najkrótszej znalezionej długości

-numberOfIteratrions - zdefiniowana początkowo maksymalna ilość iteracji

-lifetime - zdefiniowana początkowo kadencja

-xy - zmienna tymczasowa zapisująca wierzchołek x i y

-maxCriticalEvenets - zdefiniowana początkowo maksymalna ilość powtórzeń swapów bez poprawy wyników, po osiągnięciu tego limitu następuje nowe losowanie

-criticalEvent - zmienna tymczasowa przechowująca ilość zamian miejsc bez poprawy wyniku

2.1 Pseudokod:

```
stwórz_losowy(c_path)
b_path = c_path
for(i = 0; i<numberOfIterations; ++i){
    t_path_length= length(c_path) //długosc ścieżki c_path
```

xy = find_best_change(c_path) //funkcja ta znajduje najlepszą możliwą pojedynczą zamianę kolejności wierzchołków w ścieżce pomijając te pary wierzchołków które są na liście tabu - tab_{xy} >0

swap(c_path,xy) //funkcja ta zamienia wierzchołki w ścieżce

tab_{xy} = lifetime

decrement(tab) //funkcja ta zmniejsza każdy element w liście tabu o jeden

```
if(t_path_length < length(c_path)){
        ++criricalEvents
}
else{
        criticalEvents = 0
}

if(criticalEvents >= maxCriticalEvents){
        stwórz_losowy(c_path)
        clear(tab) // czyszczenie listy tabu
}

if(długość(c_path)<długość(b_path){
        b_path = c_path
}

}
```

3. Badania dla różnych instancji

3.1 Badania dla pliku gr17.tsp - dla 17 miast w grafie symetrycznym

Czas operacji [ms]	Liczba iteracji	L. Powtórzeń bez poprawy	Czas życia w liście tabu	Obliczona droga	Błąd względny
3	5	2	2	2833	35,87529976019
7	50	5	5	2468	18,36930455635
6	50	1	10	2129	2,110311750599
20	500	10	10	2120	1,678657074340
20	500	1	10	2095	0,479616306954
90	5000	100	100	2254	8,105515587529
90	5000	10	10	2221	6,522781774580
98	5000	10	1	2085	0

Dla porównania w programowaniu dynamicznym czas rozwiązania wynosił 912ms jednak jego rozwiązanie wynosiło dokładną najlepiej znaną wartość - 2085. Z powyższej tabeli wynika, że dla małej liczby wierzchołków grafu zmniejszenie liczby wykonania swapów bez poprawy do jednego jest rozwiązaniem, którego funkcja celu jest najmniejsza. Wynika to z faktu, iż prawdopodobieństwo wylosowania najkrótszej drogi jest większe niż prawdopodobieństwo dojścia do najkrótszej drogi za pomocą swapów wylosowanej ścieżki, jednak dotyczy to tylko małych instancji, w których liczba wszystkich kombinacji ścieżki nie jest na tyle duża aby prawdopodobieństwo losowego trafienia było niemalże zerowe. Ponadto widać, że liczba iteracji ma tutaj olbrzymie znaczenie.

3.2 Badania dla średniej instancji 48 miast grafu symetrycznego (plik gr48.tsp)

Czas operacji [ms]	Liczba iteracji	L. Powtórzeń bez poprawy	Czas życia w liście tabu	Obliczona droga	Błąd względny
10	5	2	2	15120	199,6432818073
28	50	5	5	7227	43,22235434007
28	50	1	10	7851	55,58858501783
153	500	10	10	6017	19,24296472453
167	500	1	10	6139	21,66072136345
1084	5000	100	100	5702	13,00039635354
1169	5000	10	10	6152	21,91835116924
1186	5000	1	10	5880	16,52794292508
1103	5000	10	100	5597	10,91954022988
7958	50000	10	100	5351	6,044391597304
7958	50000	1	100	5765	14,24891002774
6707	50000	10	10	6921	37,15814506539
5209	50000	10	1000	5835	15,63614744351

Jak widać na powyższej tabeli stosowana wcześniej „taktyka” nie jest już skuteczna. Ze względu na bardzo dużą możliwość kombinacji ścieżki losowe trafienie optymalnej ścieżki jest bardzo mało prawdopodobne. W takim wypadku widzimy już skuteczność zastosowanego algorytmu który zmniejsza długość ścieżki. Warto tutaj zwrócić uwagę na dostosowanie liczby powtórzeń, a także czasu życia w liście tabu. Jak widać dla 48 miast współczynnikami, dla których funkcja celu jest najmniejsza to ustawienie czasu życia w liście tabu na 100, natomiast liczby powtórzeń bez poprawy na 10. Ponadto widać, że zwiększenie liczby iteracji zmniejsza błąd względny.

3.3. Badania dla dużej instancji 262 miast grafu symterycznego(plik gil262.tsp)

Czas operacji [ms]	Liczba iteracji	L. Powtórzeń bez poprawy	Czas życia w liście tabu	Obliczona droga	Błąd względny
234	5	2	2	24201	917,7039529015
2134	50	5	5	13069	449,5794785534
2006	50	1	10	12843	440,0756938603
13242	500	10	10	6123	157,4852817493
13052	500	1	10	5852	146,0891505466
103134	5000	100	100	4071	71,19428090832
114210	5000	10	100	4078	71,48864592094
114388	5000	1	100	4904	106,2237174095
124520	5000	50	100	3718	56,34987384356

Z powyższej tabeli wynika że wraz ze wzrostem liczby miast powinniśmy także zwiększyć liczbę powtórzeń bez poprawy w celu otrzymania lepszego wyniku. Niestety błąd względem najkrótszej znalezionej kiedykolwiek drogi przewyższa 56%. W celu zmniejszenia tego błędu można zwiększać liczbę iteracji, jednak wpływa to liniowo na czas wykonania operacji.

3.4 Badania dla małych instancji 17 miast w grafie asymetrycznym (br17.atsp)

Czas operacji [ms]	Liczba iteracji	L. Powtórzeń bez poprawy	Czas życia w liście tabu	Obliczona droga	Błąd względny
3	5	2	2	61	56,41025641025
7	50	5	5	51	30,76923076923
6	50	1	10	42	7,692307692307
21	500	10	10	42	7,692307692307
20	500	1	10	39	0
44	5000	100	100	41	5,128205128205
90	5000	10	10	44	12,82051282051
98	5000	1	10	39	0

Jak widać tendencja dla małych instancji się utrzymuje także dla grafów asymetrycznych. Tutaj częste losowanie sprawdza się dużo lepiej niż wykonywanie swapów. W przypadku programowania dynamicznego ten sam problem rozwiązano w przeciągu 930ms jednak jego wynik jest najkrótszą znaną do tej pory drogą.

3.5 Badanie dla średnich instancji 47 miast w grafie asymetrycznym (ftv47.atsp)

Czas operacji [ms]	Liczba iteracji	L. Powtórzeń bez poprawy	Czas życia w liście tabu	Obliczona droga	Błąd względny
13	5	2	2	5237	194,8761261261
26	50	5	5	2681	50,95720720720
28	50	1	10	2498	40,65315315315
149	500	10	10	2241	26,18243243243
153	500	1	10	2370	33,44594594594
1163	5000	100	100	2117	19,20045045045
1193	5000	10	10	2210	24,43693693693
1200	5000	1	10	2246	26,46396396396
1105	5000	10	100	2077	16,94819819819
7097	50000	10	100	2002	12,72522522522
7432	50000	1	100	2096	18,01801801801
7504	50000	10	10	2107	18,63738738738
3814	50000	10	1000	2190	23,31081081081

Jak widać tendencja dla średnich instancji dla grafu asymetrycznego także się utrzymuje. Ponownie współczynnikami dla których funkcja celu jest najmniejsza to ustawienie czasu życia w liście tabu na 100, natomiast liczby powtórzeń bez poprawy na 10.

3.6 Badanie dla dużych instancji (rbg323.atsp)

Czas operacji [ms]	Liczba iteracji	L. Powtórzeń bez poprawy	Czas życia w liście tabu	Obliczona droga	Błąd względny
431	5	2	2	5789	336,5761689291
3894	50	5	5	3387	155,4298642533
3781	50	1	10	3503	164,1779788838
26924	500	10	10	1774	33,78582202111
27628	500	1	10	1743	31,44796380090
285632	5000	100	100	1713	29,18552036199
277423	5000	10	100	1620	22,17194570135
266983	5000	1	100	1642	23,83107088989
280402	5000	50	100	1602	20,81447963800

Jak widać tendencja dla średnich instancji dla grafu asymetrycznego także się utrzymuje. Ponownie współczynnikami dla których funkcja celu jest najmniejsza to

ustawienie czasu życia w liście tabu na 100, natomiast liczby powtórzeń bez poprawy na 50.

4.Podsumowanie i wnioski

Podsumowując, najważniejszym czynnikiem wpływającym na dokładność wykonanego algorytmu jest liczba iteracji. Jednak zwiększenie liczby iteracji wpływa na wydłużenie czasu wykonywania algorytmu. Drugim czynnikiem wpływającym na czas wykonywania algorytmu jest liczba powtórzeń bez poprawy, a także czas życia w liście tabu. Te współczynniki powinny być dobierane w zależności od liczby miast - dla dużych instancji powinny być odpowiednio większe. Warto także wspomnieć o dużej losowości otrzymanych wyników, gdyż algorytm opiera się na losowaniu ścieżki, która jest następnie swapowana w celu zmniejszenia funkcji celu. Porównując natomiast tabu search do programowania dynamicznego widzimy znaczącą poprawę szybkości znajdywania rozwiązania kosztem dokładności znalezionego rozwiązania, która w programowaniu dynamicznym jest większa.