Sprawozdanie z projektu PEA

Tabu search

12/14/2017

Prow adzący *Jarosław Rudy*

Wstęp

Przeszukiwanie tabu (Tabu search, TS)

> Metaheurystyka (algorytm) stosow ana do rozw iązyw ania problemów optymalizacyjnych. Wykorzystyw ana do otrzymyw ania rozw iązań optymalnych lub niew iele różniących się od niego dla problemów z różnych dziedzin (np. planow anie, planow anie zadań). Tw órcą algorytmu jest Fred Glover.
> Podstaw ow ą ideą algorytmu jest przeszukiw anie przestrzeni, stw orzonej ze w szystkich możliw ych rozw iązań, za pomocą sekw encji ruchów. W sekw encji ruchów istnieją ruchy niedozw olone, ruchy tabu. Algorytm unika oscylacji w okół optimum lokalnego dzięki przechow yw aniu informacji o spraw dzonych już rozw iązaniach w postaci listy tabu (TL).

Opis implementacji

```
for (unsigned i = 0; i < actualResult.size()-2; ++i)
{
    for (unsigned j = i+1; j < actualResult.size()-1; ++j)
    {
        if(isInTabu(i,j)){break;}
        std::swap(actualResult[i],actualResult[j]);
        actualResult.back()=actualResult.front();
        //----------------------------------------
        // 1) Get length
        // 2) Check if path is worth  following
        auto length = getPathLength(actualResult);
        if(currentElement[2]>length) {currentElement={i,j,length};}
        //----------------------------------------
        std::swap(actualResult[j],actualResult[i]);
        actualResult.back()=actualResult.front();
    }
}
```

```
//-------------------------------------------
// 1) if is better than actually known best, then acknowledge it!
// 2) if is better than actually known aspiration, then take it!
if(best > currentElement[2])
{
    ...
    tabuSize_++;
    ...
}
else if(aspirationValue > currentElement[2])
{
    ...
}
else
{
    tabuSize_--;
}
//-------------------------------------------
```

W mojej implementacji w ykorzystuje dw ie pętle 'for', które poruszają się w zdłuż istniejącego 'vector', w ypełnionego miastami w edług algorytmu chciw ego. Dodatkow o jest tam napisana metoda 'getPathLength()', która po kolei spraw dza (miasto po mieście), długość aktualnie znanej ścieżki.

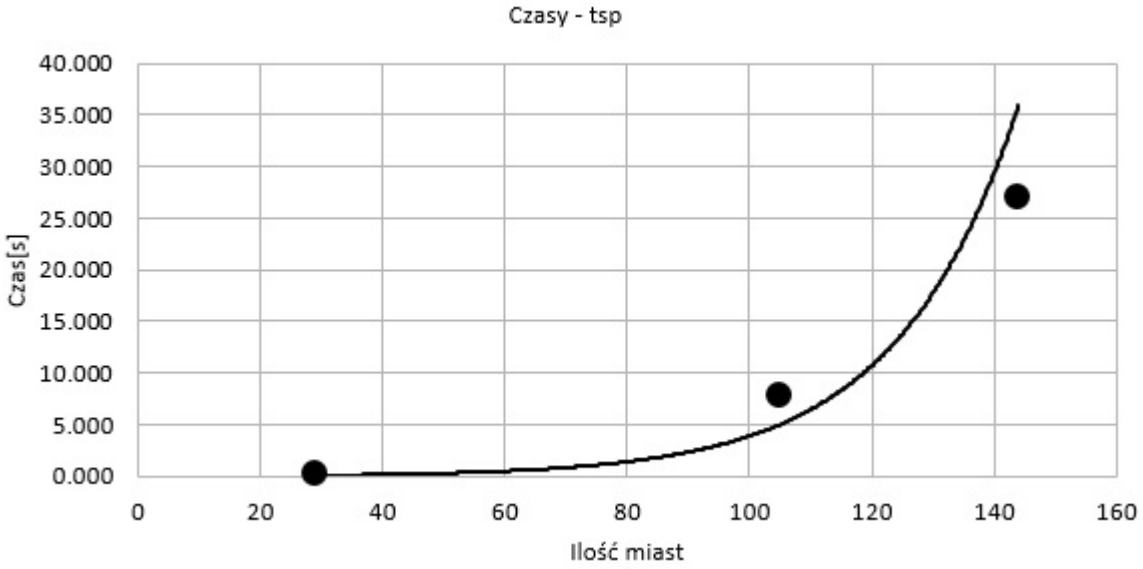
Jeżeli znajdzie jakiekolw iek rozw iązanie w arte rozw ażania, zapisze je w 'currentElement', który jest obiektem 'std::array 'std::list' jest listą tabu, która jest nieskończona i przechow uje w ew nątrz siebie elementy

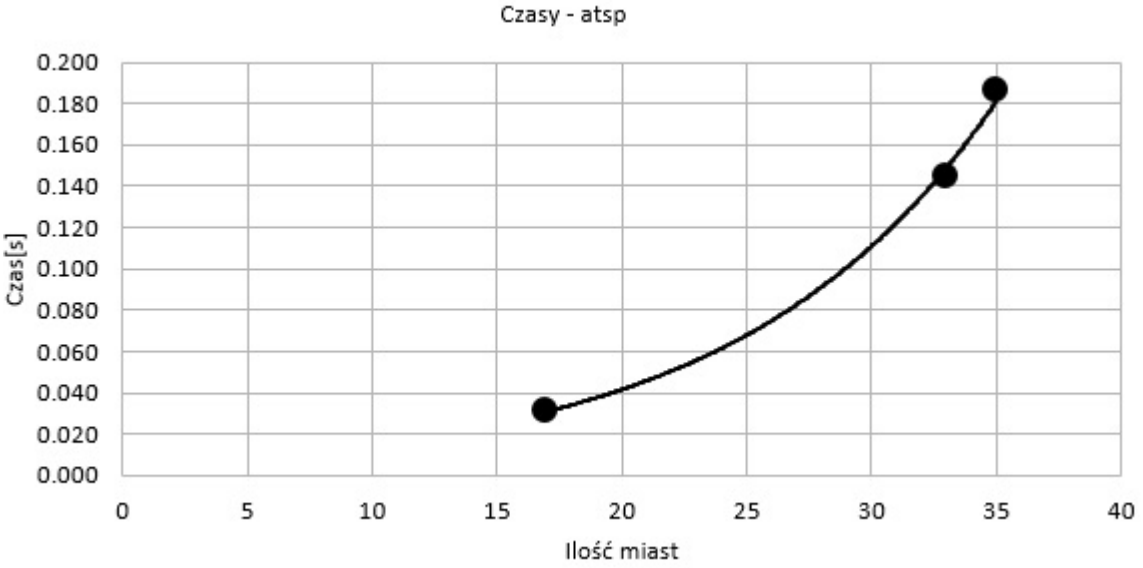
- [0] -> start point
- [1] -> end point
- [2] -> value

zmienna 'tabuSize_', jest w yznacznikiem, jaki zakres listy tabu jest dostępny dla programu (w yznacza tabu list).

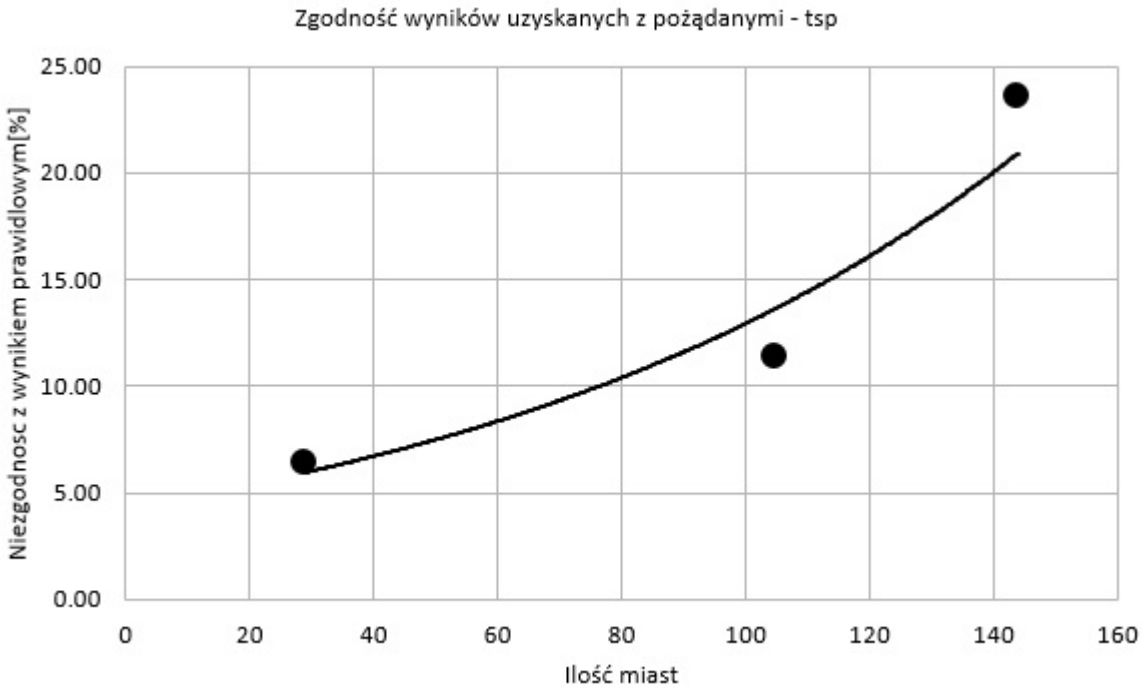
Wyniki

Czasy[s]					
br17.atsp	ftv33.atsp	ftv35.atsp	bays29.tsp	lin105.tsp	pr144.tsp
0.031	0.143	0.183	0.092	7.173	27.437
0.026	0.139	0.195	0.101	8.705	29.972
0.035	0.147	0.180	0.098	7.163	24.447
0.027	0.144	0.188	0.087	9.017	25.602
0.037	0.143	0.178	0.096	6.873	26.527
0.032	0.150	0.182	0.094	6.482	27.922
0.031	0.141	0.183	0.099	9.149	28.150
0.030	0.136	0.201	0.098	7.876	23.897
0.029	0.142	0.197	0.090	8.773	25.409
0.033	0.157	0.175	0.088	6.541	30.116
Średni czas[s]					
0.031	0.144	0.186	0.094	7.775	26.948
Koszty ścieżek					
39	1301	1522	2148	16016	72349
Prawidłowe koszty ścieżek					
39	1286	1473	2020	14379	58537


Czasy - tsp
40.000
35.000
30.000
25.000
20.000
15.000
10.000
5.000
0.000
Czas[s]
0
20
40
60
80
100
120
140
160
Ilość miast


Czasy - atsp
0.200
0.180
0.160
0.140
0.120
0.100
0.080
0.060
0.040
0.020
0.000
Czas[s]
0
5
10
15
20
25
30
35
40
Ilość miast


Zgodność wyników uzyskanych z pożądanymi - atsp
3.5
3
2.5
2
1.5
1
0.5
0
Niezgodnosc z wynikiem prawidlowym[%]
0
5
10
15
20
25
30
35
40
Ilość miast



Wnioski

Pomiary były prowadzone w łącznie z wykonaniem pojedynczego algorytmu chciwego, a limit czasu to 30s. Każdy test był uruchamiany na osobnym wątku i zapisywał wynik do własnego pliku (nie było nadpisywania). Każda mapa była testowana 50 razy, a wyniki przedstawione powyżej są ich uśrednieniem.

Jak widać wyniki nie są dokładne, i różnią się od 0% do 23.6% od najlepszego. W wyniku testów własnych, doszedłem do błędu poziomu 6% do 20% dla instancji większych (500 miast, 1000 miast oraz 5000 miast), jednak okres pomiaru był nadzwyczaj długi, i nie byłem w stanie przeprowadzić wystarczającej ich ilości, aby upewnić się do rzeczywistego wyniku.

Porównanie do B&B

Wyniki otrzymywane w **B&B**, były definitywnie więcej warte, ponieważ dawały rzeczywiście najlepszy wynik. Ich największym minusem był czas, w którym się wykonywały. Najmniejsza instancja w tym teście miała siedemnaście miast, a w teście **B&B** trzy. Jednak porównanie ich jest proste, ponieważ instancja 17 miast dała mi poprawny wynik (0% błędu), po zaledwie ~0.031s, a B&B musiałem oczekiwać czas bliski godzinie (na lepszym sprzęcie).

Zakładając, że mamy nieskończoną ilość czasu, definitywnie lepiej jest wybrać algorytm B&B, jednak w większości scenariuszy, **Tabu Search** będzie lepszym wyborem, ponieważ da nam przybliżoną wartość rozwiązania po niewielkim czasie, a najczęściej wystarczy nam tylko 'przybliżona' droga, aby podjąć dobrą decyzję.

Linki

https://pl.wikipedia.org/wiki/Przeszukiwanie_tabu

Student
Ormaniec Wojciech
226181