Paweł Biel

nr indeksu 225949

Sprawozdanie

**Problem komiwojażera metodą Tabu Search**

1. **Informacje teoretyczne o Tabu Search**

**Tabu Search jest metodą opartą na przeszukiwaniu przestrzeni rozwiązań, wykorzystując sąsiedztwo rozwiązań oraz na zapamiętywaniu   
ruchów(transformacji), rozwiązania i ich częstotliwości występowania. Taki zabieg pozwala na uniknięcie przeszukiwania minimów lokalnych, a** poszukiwania rozwiązań globalnych w rozsądnym czasie. Tabu Search jest metodą, która nie **daje gwarancji znalezienia optymalnego rozwiązania.**

1. **Informacje wstępne**

**W mojej implementacji algorytmu rozpoczynamy od wygenerowania rozwiązania metodą zachłanną, następnie z tego rozwiązania jest wybierane rozwiązanie sąsiednie po przez transformację typu swap (A,B), czyli zamiany miasta A z innym dowolnym miastem. Jeśli nowe rozwiązanie jest lepsze jest zapamiętywane, a wykonany ruch/transformacja oraz jej kadencja jest zapamiętywana na liście tabu, która ma stały rozmiar, przez co dana transformacja nie będzie mogła być wykorzystana przez pewną ilość iteracji, które określa kadencja.  
Jeśli w danym sąsiedztwie nie można znaleźć lepszego rozwiązania co świadczy o znalezieniu minimum lokalnego, zostaje wykorzystana strategia dywersyfikacji.   
Strategia dywersyfikacji polega na wygenerowaniu nowego losowego rozwiązania, a następnie kontynuowanie algorytmu od nowego rozwiązania.**

1. **Pseudokod**

Wybierz punkt startowy metodą zachłanną x0 ∈ X

xopt ← x0

tabu\_list ← ∅

repeat

znajdź x ∈ N′′(x0), dla którego x < x0

x0 ← x

if f(x0) > f(xopt) then

xopt ← x0

zweryfikuj tabu\_list

∀element ∈ tabu list do

− − kadencjai

if kadencjai = 0 then

usuń element(atrybuti , kadencjai) z tabu list

if CriticalEvents() then

x0 ← Restart() (Dywersyfikacja)

if f(x0) > f(xopt) then

xopt ← x0

until

warunek zakończenia **gdzie:** N′′(x) = {y∣y ∈ N(x) ∧ y ∉ tabu list}

**CriticalEvents() zachodzi gdy w danym sąsiedztwie nie zostaną znalezione lepsze rozwiązania.**

**Pseudokod jest zaczerpnięty z prezentacji Dr.Kapłona oraz lekko zmodyfikowana na potrzeby mojej implantacji.**

1. **Wyniki**
   1. **Zależność jakości wyniku od ilości iteracji i wielkości kadencji oraz ich wykresy dla macierzy asymetrycznych**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| br17 |  | n = 17 | Optymalny wynik = 39 | | |  |  |  |  |  |
| Wynik | Kadencja | | | |  | Czas [s] | Kadencja | | | |
| iteracje | 7 | 1/3\*n | 1/2\*n | n |  | iteracje | 7 | 1/3\*n | 1/2\*n | n |
| 1000 | 39 | 40 | 39 | 39 |  | 1000 | 0,469815 | 0,426413 | 0,481939 | 0,595177 |
| 5000 | 39 | 39 | 39 | 39 |  | 5000 | 2,531224 | 2,133405 | 2,143641 | 2,922634 |
| 10000 | 39 | 39 | 39 | 39 |  | 10000 | 4,396844 | 4,463901 | 4,453239 | 5,667621 |
|  |  |  |  |  |  |  |  |  |  |  |
| ftv33 |  | n = 33 | Optymalny wynik = 1286 | | |  |  |  |  |  |
| Wynik | Kadencja | | | |  | Czas [s] | Kadencja | | | |
| iteracje | 7 | 1/3\*n | 1/2\*n | n |  | iteracje | 7 | 1/3\*n | 1/2\*n | n |
| 1000 | 1557 | 1582 | 1576 | 1570 |  | 1000 | 1,696787 | 1,964093 | 2,163003 | 3,048858 |
| 5000 | 1554 | 1545 | 1540 | 1527 |  | 5000 | 8,223318 | 9,522301 | 11,62681 | 15,50268 |
| 10000 | 1535 | 1516 | 1568 | 1506 |  | 10000 | 17,94442 | 19,3188 | 22,71231 | 28,66663 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| ftv47 |  | n = 47 | Optymalny wynik = 1776 | | |  |  |  |  |  |
| Wynik | Kadencja | | | |  | Czas[s] | Kadencja | | | |
| iteracje | 7 | 1/3\*n | 1/2\*n | n |  | iteracje | 7 | 1/3\*n | 1/2\*n | n |
| 1000 | 2289 | 2289 | 2249 | 2289 |  | 1000 | 3,48238 | 4,577493 | 4,902551 | 7,010385 |
| 5000 | 2256 | 2281 | 2267 | 2286 |  | 5000 | 18,29853 | 21,60025 | 24,17101 | 34,43291 |
| 10000 | 2177 | 2250 | 2226 | 2289 |  | 10000 | 35,70766 | 43,71711 | 49,59652 | 66,80184 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ftv70 |  | n = 70 | Optymalny wynik = 1950 | | |  |  |  |  |  |
| Wynik | Kadencja | | | |  | Czas | Kadencja | | | |
| iteracje | 7 | 1/3\*n | 1/2\*n | n |  | iteracje | 7 | 1/3\*n | 1/2\*n | n |
| 1000 | 2521 | 2483 | 2462 | 2433 |  | 1000 | 9,384739 | 11,4054 | 15,47678 | 19,57893 |
| 5000 | 2483 | 2458 | 2433 | 2433 |  | 5000 | 47,2106 | 58,35714 | 69,14207 | 94,93941 |
| 10000 | 2457 | 2446 | 2435 | 2421 |  | 10000 | 93,48388 | 118,2576 | 142,1674 | 190,3743 |

* 1. **Zależność jakości wyniku od ilości iteracji i wielkości kadencji oraz ich wykresy dla macierzy symetrycznych**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| gr17 |  | n = 17 | Optymalny wynik = 2085 | | |  |  |  |  |  |
| Wynik | Kadencja | | | |  | Czas | Kadencja | | | |
| iteracje | 7 | 1/3\*n | 1/2\*n | n |  | iteracje | 7 | 1/3\*n | 1/2\*n | n |
| 1000 | 2088 | 2087 | 2087 | 2087 |  | 1000 | 0,386967 | 0,445523 | 0,471853 | 0,529912 |
| 5000 | 2085 | 2085 | 2085 | 2085 |  | 5000 | 2,15015 | 2,118382 | 2,143816 | 2,885492 |
| 10000 | 2085 | 2085 | 2085 | 2085 |  | 10000 | 4,209146 | 4,10512 | 4,438006 | 5,696868 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| gr24 |  | n = 24 | Optymalny wynik = 1272 | | |  |  |  |  |  |
| Wynik | Kadencja | | | |  | Czas | Kadencja | | | |
| iteracje | 7 | 1/3\*n | 1/2\*n | n |  | iteracje | 7 | 1/3\*n | 1/2\*n | n |
| 1000 | 1366 | 1352 | 1342 | 1367 |  | 100. | 0,816154 | 0,940293 | 1,020889 | 1,352122 |
| 5000 | 1345 | 1318 | 1312 | 1319 |  | 5000 | 4,586984 | 4,834269 | 5,113441 | 6,696294 |
| 10000 | 1283 | 1303 | 1299 | 1327 |  | 10000 | 8,746042 | 9,496561 | 10,54375 | 13,25248 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| gr48 |  | n = 48 | Optymalny wynik = 5046 | | |  |  |  |  |  |
| Wynik | Kadencja | | | |  | Czas | Kadencja | | | |
| iteracje | 7 | 1/3\*n | 1/2\*n | n |  | iteracje | 7 | 1/3\*n | 1/2\*n | n |
| 1000 | 5948 | 5920 | 5948 | 5893 |  | 1000 | 5,113429 | 5,080506 | 5,791205 | 7,806751 |
| 5000 | 5744 | 5948 | 5910 | 5810 |  | 5000 | 23,10587 | 26,6453 | 31,37407 | 41,58839 |
| 10000 | 5894 | 5748 | 5941 | 5912 |  | 10000 | 43,09115 | 54,70738 | 59,84727 | 81,70223 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| gr120 |  | n = 120 | Optymalny wynik = 6942 | | |  |  |  |  |  |
| Wynik | Kadencja | | | |  | Czas | Kadencja | | | |
| iteracje | 7 | 1/3\*n | 1/2\*n | n |  | iteracje | 7 | 1/3\*n | 1/2\*n | n |
| 1000 | 8789 | 8930 | 8854 | 8872 |  | 1000. | 50,12874 | 71,40623 | 58,36713 | 152,5193 |
| 5000 | 8789 | 8854 | 8885 | 8854 |  | 5000. | 232,0503 | 405,3416 | 465,5788 | 543,8683 |
| 10000 | 8789 | 8854 | 8885 | 8821 |  | 10000. | 475,2038 | 677,0778 | 811,8655 | 948,414 |

* 1. Wielkość błędu dla testowych instancji

|  |  |  |  |
| --- | --- | --- | --- |
| gr17 | 2085 | 2085 | 0% |
| br17 | 39 | 39 | 0% |
| gr24 | 1283 | 1272 | 1% |
| ftv33 | 1516 | 1286 | 18% |
| ftv47 | 2177 | 1776 | 23% |
| gr48 | 5744 | 5046 | 14% |
| ftv70 | 2421 | 1950 | 24% |
| gr120 | 8789 | 6942 | 27% |

* 1. **Wykres zależności czasu od ilości miast**

Dla macierzy asymetrycznych:

Dla macierzy symetrycznych:

Z racji że w poprzednim projekcie nie udało mi się uruchomić pomiarów dla macierzy   
symetrycznych, a bruteforce dla tak dużych instancji wykonuje się bardzo długo dlatego zamieszczam wykres tylko z zakresu obecnego projektu.

* 1. Wykres ilustrujący w których iteracjach wynik został poprawiony dla instancji gr120 i 1000 iteracji

1. **Wnioski**
   1. Czym większa ilość iteracji tym większa szansa na znalezienie lepszego rozwiązania, lecz wydłuża się czas wykonywania algorytmu.
   2. Na podstawie powyższych wykresów można stwierdzić, że kadencja ma wpływ w sposób losowy, to znaczy, że nie można zauważyć np. takiej tendencji że czym większa kadencja wynik staje się lepszy, lecz zależy to od instancji i trzeba dobrać odpowiednio parametry.
   3. Dla małych instancji o wielkościach 17 czy też 33 algorytm jest wstanie znaleźć najbardziej optymalne rozwiązanie oraz w dużo lepszym czasie niż metody B&B oraz BruteForce.
   4. Z wykresu można wywnioskować, że złożoność tabu search wynosi O(n3)
   5. Im większa instancja tym gorsze najlepsze uzyskiwane rozwiązanie (z największym błędem)
   6. Dla dużych instancji tabu search znajduje szybciej rozwiązanie,  
      z prawdopodobieństwem błędu, więc jeżeli nasz problem pozwala na założenie, że nasze wyniki nie są dokładne, ale są oszacowane i mieszczą się w zakresie dla nas dopuszczalnego błędu, to tabu search jest bardzo dobrą metodą na znalezienie takich w rozwiązań w szybkim czasie.
   7. Ten algorytm niestety przez większość czasu nie znajduję polepszenia wyniku, powodów może być kilka np. że startujemy z rozwiązania znalezionego zachłannie oraz z tego że generowanie nowego rozwiązania podczas CriticalEvent() jest losowe.
2. **Uwagi**

Poprawił bym funkcje CriticalEvent, aby nie losować jakiegokolwiek rozwiązania, lecz takiego które z może być obiecujące jeśli chodzi o poprawienie wyniku.

Poprawił bym też warunek stopu, aby algorytm nie pracował przez większość czasu bez sensu oraz może bym wymieszał metody szukania sąsiedztwa np. swap i invert.