|  |  |
| --- | --- |
| Sprawozdanie 6 – Testy globalnej wypukłości | |
| **Imie Nazwisko** | **Nr indexu** |
| Patryk Jedlikowski  Mikołaj Sienkiewicz | 136723  136309 |

# Opis zadania

# Zadanie polegało na zbadaniu występowania optimów lokalnych przez obliczenie wielu losowo startujących rozwiązań algorytmu przeszukiwania w wersji zachłannej. Dla każdego z rozwiązań należało policzyć 2 typy podobieństwa (wierzchołkowe i krawędziowe) względem najlepszego rozwiązania i względem średniej rozwiązań.

# Pseudokod zaimplementowanych funkcji obliczających podobieństwo:

### Podobieństwo wierzchołkowe

func vertexSim (cykle1, cykle2):

    Dla pierwszego cyklu z cykle1:

        match=Suma wierzchołków z wybranego cyklu takich samych co w pierwszym cyklu z cykle2

    jeżeli 100-match>match:

        zwróć (100-match)\*2

    W przecuwnym wypadku:

        zwróć match\*2

### Podobieństwo krawędziowe

func edgeSim (cykle1,cykle2):

    Iteru po cyklach z cykle1:

        Dla wybranego cyklu sprawdz ile takich samych krawędzi jest w cykle2

    Zwróć sumę znalezionych krawędzi

### Policz oba podobieństwa (krawędziowe i wierzchołkowe) względem najlepszego rozwiązania

func simWithBest(cyclesTable,distanceTable):

    bestCycles=znajdz takie cykle z cyclesTable, który mają najmniejszy dystans w distanceTable

    vertexSimBest=[], edgeSimBest=[]

    Dla każdej pary cykle,dystans z (cyclesTable,distanceTable):

        vertexSimBest.append([dystans,vertexSim(cykle,bestCycles)])

        edgeSimBest.append([dystans,edgeSim(cykle,bestCycles)])

    zwróć vertexSimBest i edgeSimBest

### Policz średnie podobieństwa (wierzchołkowe i krawędziowe) dla każdego z przypadków

func simAverage(cyclesTable,distanceTable):

    averageVertSim=[]

    averageEdgeSim=[]

    Dla każdego cykle,dystans z (cyclesTable,distanceTable):

        tymczasoweVertexSim=[]

        tymczasoweEdgeSim=[]

        Dla każdego cykle2 z cyclesTable:

            tymczasoweVertexSim.append(vertexSim(cykle,cykl2))

            tymczasoweEdgeSim.append(edgeSim(cykle,cykl2))

        averageVertSim.append([dystans,średnia\_z(tymczasoweVertexSim)])

        averageEdgeSim.append([dystans,średnia\_z(tymczasoweEdgeSim)])

    zwróć averageEdgeSim,averageVertSim

## Pozostałe powtórnie wykorzystane funkcje

Poniższe funkcje zostały zaimplementowane już w ramach poprzednich laboratoriów i teraz ponownie wykorzystane. Z tego względu pseudokody zostały bez zmian. Funkcje te definiują:

* Ruchy wewnątrz i między trasowe (change edges i change vertices)
* Lokalne przesukiwanie typu zachłannego (greedy search)
* Multi Start Local Search oraz różne warianty podejścia Iterative Local Search

#### Change Vertices (ruch między trasowy)

funkcja zmiana\_wierzchołków ([] cykle, Int wierzchołek\_A, Int wierzchołek\_B, Bool ruch\_wewnętrzny):

jeżeli ruch\_wewnętrzny:

Znajdz wierzchołek A i B w jednym cyklu i zamień miejscami

w przeciwnym wypadku:

Znajdz wierzchołek A w jednym a B w drugim cyklu i zamień miejscami

#### Change Edges (ruch wewnątrztrasowy)

funkcja zmiana\_krawędzi(cykle, początekKrawędzi1, początekKrawędzi2):

    znajdź cykl, krórego dotyczy zmiana

    znajdź indeks początków krawędzi w cuklu

    podmień krawędzie pomiędzy początkami wybranych krawędzi obracając kierunek pierwotnego cyklu

#### Start\_cycle\_random

cykl\_pierwszy = wybierz losowo 100 liczb z zakresu (0-199)

cykl\_drugi = pozostałe liczby z zakresu (0-199) nie wybrane w cykl\_pierwszy

Zwróć cykl\_pierwszy, cykl\_drugi

#### Greedy search:

Wygeneruj rozwiązanie startowe x (start\_cycle\_random)

Powtarzaj:

    Wygeneruj zbiór ruchów M(x) - [external\_moves,internal\_moves]

    dla każdego m należacego do M(x) w losowej kolejności:

        jeżeli f(m(x)) > f(x) to:

            x := m(x)

            zacznij pętle od nowa

dopóki nie znaleziono lepszego rozwiązania po przejrzeniu całego M(x)

gdzie M(x) to przestrzeń możliwych ruchów wewnątrz i między trasowych.

# Tabela prezentująca czasy obliczeń eksperymentu obliczeniowego

|  |  |  |
| --- | --- | --- |
| **Typ algorytmu** | **kroa** | **krob** |
| Greedy x 1000 | 38853 (35003-42587) | 39009 (35059- 43094) |

# Wizualizacje rozwiązań, podobieństw i wyniki korelacji

|  |  |
| --- | --- |
| **Greedy – Najlepszy wynik na Kroa** | **Greedy – Najlepszy wynik na Kroa** |
|  |  |

|  |  |
| --- | --- |
| **Średnie podobieństwo wierzchołkowe Kroa** | **Średnie podobieństwo wierzchołkowe Krob** |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Średnie podobieństwo krawędziowe Kroa** | **Średnie podobieństwo krawędziowe Krob** |
|  |  |

|  |  |
| --- | --- |
| **Podobieństwo wierzchołkowe względem najlepszego - Kroa** | **Podobieństwo wierzchołkowe względem najlepszego - Krob** |
|  |  |

|  |  |
| --- | --- |
| **Podobieństwo krawędziowe względem najlepszego - Kroa** | **Podobieństwo krawędziowe względem najlepszego - Krob** |
|  |  |

# Wnioski

Wykresy podobieństw względem dystansu wykazują korelację w momencie, gdy porównujemy je względem średniej. Fakt, że średnie podobieństwo rośnie przeciwnie proporcjonalnie do uzyskanego dystansu może wskazywać na jakieś lokalne optimum, do którego taki zachłanny algorytm inicjalizowany losowo wpada i które ciężko mu przełamać.

Co więcej znikome podobieństwo innych rozwiązań względem rozwiązania najlepszego (niezależnie od dystansu), oraz tym samym bardzo niska korelacja wskazuje na fakt, że znalezione najlepsze rozwiązanie mało ma do czynienia z lokalnym optimum do którego często wpadamy i które definiuje średnią strukturę (uwidocznioną na wykresach ze średnim podobieństwem).

# Kod programu

Repozytorium z całym kodem napisanym w jupyter\_notebook znajduje się na githubie pod linkiem

<https://github.com/mikolaj-sienkiewicz/PP_Inteligente_Metody_Optymalizacji> (plik Lab6/Lab6.ipynb)