|  |  |
| --- | --- |
| Sprawozdanie 5 – Hybrydowy algorytm ewolucyjny | |
| **Imie Nazwisko** | **Nr indexu** |
| Patryk Jedlikowski  Mikołaj Sienkiewicz | 136723  136309 |

# Opis zadania

# Zadanie polegało na zaimplementowaniu hybrydowego algorytmu ewolucyjnego i porównaniu go z metodami MSLS i ILSx z poprzedniego tygodnia. Algorytm ewolucyjny opierał się na populacji reprezentatywnej o wielkości 20, z której losowaliśmy rodziców i dokonując rekombinacji generowaliśmy nowe rozwiązanie.

# Pseudokod zaimplementowanego algorytmu ewolucyjnego:

Funkcja evolve:

Populacja = []

Powtórz 20 razy:

    Do populacji dodaj wynik Greedy Search Inicjalizowanego Heurystycznie

Powtarzaj:

    Wylosuj z Populacja dwóch rodziców;

    Dziecko=Rodzic1

    Dla każdego wierzchołka w Dziecko:

        usuń jeżeli oboje sasiedzi nie sa tacy sami jak u Rodzic2

    Dziecko=rebuild\_cycle\_heuristic(Dziecko,zbiór danych)

    Jeżeli robić LS:

        Wykonaj Greedy Search na dziecku

    Jeżeli Dziecko > Najgorszy reprezentant populacji ORAZ W populacji nie ma wyniku dziecka:

        Zastap najgorszego reprezentata dzieckiem

    Jeżeli populacja nie zmieniła się od 25 iteracji:

        Zastap najgorszego reprezentacja nowym wynikiem z Greedy\_Search

Dopóki nie skończy się czas

Funkcja rebuild\_cycle\_heuristic odbudowuje Podany na wejściu cykle algorytmem Najbliższego Sasiada, w taki sposób że wierzchołki mogą być uzupełnione miedzy dowolne dwa wybranego cyklu (jeżeli ten cykl nie ma już 100 wierzchołków).

## 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

#### MSLS (Greedy search):

Powtórz 100 razy:

Wygeneruj rozwiązanie startowe x (start\_cycle\_heuristic)

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.

#### ILS1 - Iteracyjne przeszukiwanie lokalne z niewielką perturbacją

Ustaw zmienną N, która określa liczbę wierzchołków poddanych perturbacji

Wygeneruj rozwiązanie startowe x (start\_cycle\_random)

Powtarzaj:

    Wykonaj lokalne przeszukiwanie LS (greedy)

    Jeśli poprawiło się globalne rozwiązanie to zapamiętaj nowy najlepszy cykl, w przeciwnym razie przywróć do zmiennej cycle najlepszy wygenerowany cykl

    Dokonaj perturbacji N wierzchołków

Dopóki nie upłynął czas

Zwróć najlepszy znaleziony cykl (zmienna cycle)

#### ILS2 – Iteracyjne przeszukiwanie z większa perturbacją

Ustaw zmienną N, która określa liczbę wierzchołków poddanych perturbacji w jednym cykly(default=10)

Wygeneruj rozwiązanie startowe x (start\_cycle\_random)

Powtarzaj:

Wykonaj lokalne przeszukiwanie LS (greedy)

Jeśli poprawiło się globalnie najlepsze rozwiązanie to zapamiętaj nowy najlepszy cykl, wprzeciwnym razieprzywróć do zmiennej cycle najlepszy wygenerowany cykl

Usuń N wierzchołków z każdego cyklu

Odbuduj cykle metodą najbliższego sąsiada

Dopóki nie upłynął czas

Zwróć najlepszy znaleziony cykl (zmienna cycle)

#### ILS2a – Iteracyjne przeszukiwanie z większa perturbacją

Ustaw zmienną N, która określa liczbę wierzchołków poddanych perturbacji w jednym cykly(default=10)

Wygeneruj rozwiązanie startowe x (start\_cycle\_random)

Wykonaj lokalne przeszukiwanie LS (greedy)

Powtarzaj:

Jeśli poprawiło się globalnie \\ najlepsze rozwiązanie to zapamiętaj nowy najlepszy cykl, wprzeciwnym razieprzywróć do zmiennej cycle najlepszy wygenerowany cykl

Usuń N wierzchołków z każdego cyklu

Odbuduj cykle metodą najbliższego sąsiada

Dopóki nie upłynął czas

Zwróć najlepszy znaleziony cykl (zmienna cycle)

# Tabela prezentująca wartości funkcji celu eksperymentu obliczeniowego

|  |  |  |
| --- | --- | --- |
| **Initialization type** | **kroa** | **krob** |
| MSLS | 3928 (3538 -4639) | 3615 (3558-3742) |

# Tabela prezentująca czasy obliczeń eksperymentu obliczeniowego

|  |  |  |
| --- | --- | --- |
| **Initialization type** | **kroa** | **krob** |
| Algorytm Ewolucyjny |  |  |
| Algorytm Ewolucyjny z LS |  |  |
| MSLS | 37610 (35918-38123) | 38110 (37117-38935) |
| ILS1 | 35809 (34802- 36878) | 35993 (33800- 38032) |
| ILS2 | 34470 (33097 – 36391) | 36365 (35729-36990) |
| ILS2a | 37137 (34294 – 38487) | 37809 (36583-39257) |

# Wizualizacje najlepszych rozwiązań

|  |  |
| --- | --- |
| **MSLS – Kroa** | **MSLS– Krob** |
|  |  |

|  |  |
| --- | --- |
| **ILS1 – Kroa** | **ILS1 – Krob** |
|  |  |

|  |  |
| --- | --- |
| **ILS2 – Kroa** | **ILS2 – Krob** |
|  |  |

|  |  |
| --- | --- |
| **ILS2a – Kroa** | **ILS2a – Krob** |
|  |  |

# Wnioski

Lokalne przeszukiwanie w wersji zachłannej okazało się podejściem szybszym niż metoda w wersji stromej. Różnica jest szczególnie zauważalna, gdy algorytm rozpoczynał od rozwiązania losowego. Z tego względu zdecydowaliśmy się oprzeć perturbacje o lokalne przeszukiwanie w wersji greedy.

Perturbacje są w stanie znacząco skrócić wynikową ścieżkę w porównaniu do wersji Multi Start Local Search. Zwraca to uwagę na fakt jak istotny dla wyniku jest tak naprawdę sposób rozdystrybuowania tych ostatnich wierzchołków, które często są brane do cyklu nie dlatego że są idealne z punktu widzenia funkcji celu, ale dlatego że tylko one zostały. Wykorzystując perturbacje jesteśmy w stanie tę zależność trochę zmitygować. Nie budujemy tylko rozwiązania, ale dodatkowo staramy się wielokrotnie przearanżować pewną część cyklu rozszerzając w ten sposób zdolności lokalnego przeszukiwania.

Prawdopodobnie w algorytmach z większą perturbacją można by uzyskać jeszcze lepsze wyniki parametryzując liczbę usuwanych i odbudowywanych z powrotem wierzchołków. Nie mniej przez ograniczenia czasowe i długość wykonywania się algorytmu skupiliśmy się na opracowaniu jednej wersji dla liczby usuwanych wierzchołków równej 10% - czyli 20 co wciąż przyniosło drobne poprawy.

# 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 Lab4/Lab4.ipynb)