|  |  |
| --- | --- |
| 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

|  |  |  |
| --- | --- | --- |
| **Typ algorytmu** | **kroa** | **krob** |
| MSLS | 3928 (3538 -4639) | 3615 (3558-3742) |

# Tabela prezentująca czasy obliczeń eksperymentu obliczeniowego

|  |  |  |
| --- | --- | --- |
| **Typ algorytmu** | **kroa** | **krob** |
| Algorytm Ewolucyjny | 36714 (36296-37445) | 37053 (36384-37701) |
| Algorytm Ewolucyjny z LS | 33557 (32424-34446) | 33768 (32924-34613) |
| 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

|  |  |
| --- | --- |
| **Ewolucyjny bez LS – Kroa** | **Ewolucyjny bez LS – Krob** |
|  |  |

|  |  |
| --- | --- |
| **Ewolucyjny z LS – Kroa** | **Ewolucyjny z LS – Krob** |
|  |  |

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

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

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

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

# Wnioski

Algorytmy ewolucyjne pozwalają eksplorować przestrzeń rozwiązań w oryginalny dla nich sposób. Daje im to zdolność do znajdywania optimów, do których nie zawsze potrafiły dotrzeć wersje zwykłe (MSLS) ani nawet te z perturbacjami (ILSx).

Wersja algorytmu ewolucyjnego, w której po odbudowie nie wykonywaliśmy lokalnego przeszukiwania rzadko kiedy dawała na tyle dobre ścieżki, żeby kwalifikowały się one do zastąpienia nimi innego wyniku w populacji. Działo się tak ze względu na to, że odbudowa na bazie najbliższego sąsiada (mimo że umożliwiała wstawiać w dowolne miejsca cyklu) dawała za słabe wyniki. Z tego względu w wyznaczonym czasie populacja nie ewoluowała wystarczająco często, żeby zmiany były wyraźnie zauważalne.

Natomiast wersja, w której to lokalne przeszukiwanie na końcu występuje pozwalała na tyle efektywnie aktualizować populację, że ewolucja po pewnym czasie dawała wyraźne na wykresach efekty. Widać jak cykle z biegiem ewolucji coraz mniej na siebie nachodzą a zamiast tego alokują się coraz bardziej na rozłącznych przestrzeniach.

# 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 Lab5/Lab5.ipynb)