Górka Bartosz 127228
Zimniak Kajetan 127229

Algorytmy ewolucyjne i metaheurystyki

Sprawozdanie 4

1. Opis etapu projektu

Celem kolejnego etapu projektu była implementacja dwóch algorytmów przeszukiwania lokalnego. Pierwszym z nich był `Multiple Start Local Search`. Został on uruchomiony 10-cio krotnie, każdorazowo realizujący 100 operacji przeszukiwania lokalnego ze zrandomizowanym przydziałem punktów do grup.

Drugi algorytm to `Iterated Local Search`, który został przygotowany w trzech wersjach. Pierwsza realizuje małą perturbację polegającą na przeniesieniu 2% punktów pomiędzy grupami w sposób zrandomizowany. Dwie pozostałe realizują dużą perturbację z przeniesieniem 30% punktów. W fazie `Destroy` zostają wytypowane i usunięte punkty. Jedna z wersji realizuje to w sposób losowy, druga natomiast wybiera najbardziej oddalone od reszty grupy punkty. Kolejną fazą jest `Repair` z przydziałem punktu do grupy, dla której najmniej pogarsza wartość funkcji celu.

Liczba grup pozostała bez zmian na poziomie 20. Podobnie jak w poprzednich etapach, funkcja celu została zdefiniowana jako `minimalizacja średniej odległości wszystkich par obiektów umieszczonych w ramach tej samej grupy`.

W rozdziale 2 zaprezentowano pseudokody przygotowanych rozszerzeń algorytmu, natomiast w rozdziale 3 wyniki działania. Ostatni rozdział dotyczy wizualizacji najlepszych uzyskanych rozwiązań.

2. Pseudokody

2.1. Multiple Start Local Search

```
W pętli wykonaj iterację aż osiągniesz limit iteracji lokalnego przeszukiwania {
    Zainicjalizuj losowy przydział punktów do grup

    W zależności od parametru metody, wykonaj przeszukiwanie lokalne z wykorzystaniem
    algorytmu Greedy bądź Steepest

    Jeżeli osiągnięto lepsze rozwiązanie niż do tej pory najlepsze {
        Zapisz wartość funkcji celu
        Zapisz uzyskany przydział punktów do grup
    }
}
```

W naszym przypadku procedura opisana wyżej powtarzana jest 10-cio krotnie, za każdym razem zapamiętujemy uzyskany wynik i przydział punktów do grup.

2.2. Iterated Local Search

```
Wyznacz liczbę punktów do przeniesienia w zależności od użytej perturbacji
Wykonuj dopóki nie przekroczono limitu czasu wykonania {
    Dokonaj perturbacji rozwiązania startowego

    Z wykorzystaniem algorytmu Steepest wyznacz optimum lokalne

    Jeżeli wyznaczone rozwiązanie jest lepsze niż dotychczas najlepsze {
        Zapamiętaj wartość rozwiązania jako nowe najlepsze
        Za rozwiązanie startowe przypisz uzyskane rozwiązanie
    }
}
```

Algorytm wykorzystuje metodę perturbacji. Jej dokładna realizacja została przedstawiona w kolejnych podrozdziałach, aby nie powielać opisu.

2.3. Mała perturbacja

```
Wykonuj aż do przeniesienia zakładanej liczby punktów {
    Wyznacz losowo grupę z której przenieść punkt
    Wyznacz grupę docelową inną niż startowa

    Wybierz punkt z grupy

    Przenieś go pomiędzy grupami
}
```

2.4. Duża perturbacja losowa

```
W fazie Destroy usuń zakładaną liczbę punktów z grup (wybierz punkty losowo)

Dla każdego usuniętego punktu {
    Wyznacz grupę dla której najmniej pogarsza wartość funkcji celu

    Dodaj punkt do wyznaczonej grupy
}
```

2.5. Duża perturbacja heurystyczna

```
Wyznacz sumę odległości każdego punktu od innych w ramach tej samej grupy
Posortuj listę zgodnie z malejącą sumą odległości

W fazie Destroy usuń zakładaną liczbę punktów z grup
iterując na przygotowanej liście punktów
```

```
Dla każdego usuniętego punktu {
    Wyznacz grupę dla której najmniej pogarsza wartość funkcji celu

    Dodaj punkt do wyznaczonej grupy
}
```

3. Wyniki eksperymentów obliczeniowych

W tabeli 1 zaprezentowano wyniki eksperymentów obliczeniowych. W przypadku `Multiple Start Local Search` podany czas dotyczy obliczeń dotyczy minimalnej/maksymalnej/średniej wartości z 10 uruchomień algorytmu. Dla algorytmu iteracyjnego, wartości czasu obliczeń dotyczą całego procesu (od uruchomienia do wyczerpania limitu czasowego). W algorytmach nie wykorzystano cache ani ruchów kandydackich. Cały proces testowania algorytmów został wykonany 10-cio krotnie, aby możliwe było wskazanie wartości średniej.

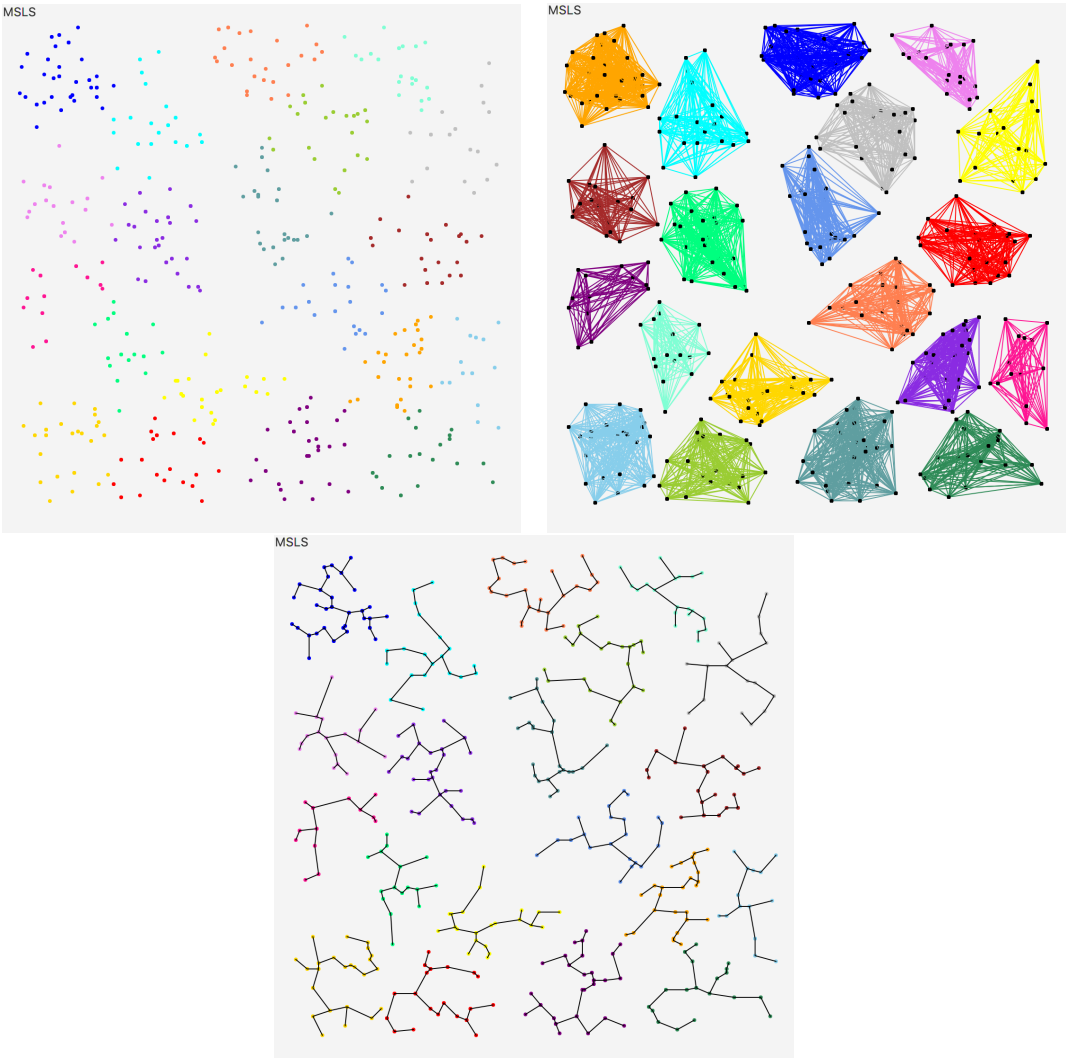
Tabela 1. Wyniki eksperymentów obliczeniowych

Cecha	MSLS	ILS small perturbation	ILS big perturbation	ILS big perturbation heuristic
Wartość minimalna funkcji celu	26.37	26.44	26.37	26.65
Wartość maksymalna funkcji celu	26.38	27.78	26.74	27.08
Wartość średnia funkcji celu	26.37	27.03	26.51	26.82
Wartość minimalna czasu obliczeń [sec]	38.50	38.55	38.54	38.54
Wartość maksymalna czasu obliczeń [sec]	43.57	43.58	43.64	43.57
Wartość średnia czasu obliczeń [sec]	40.21	40.26	40.26	40.23

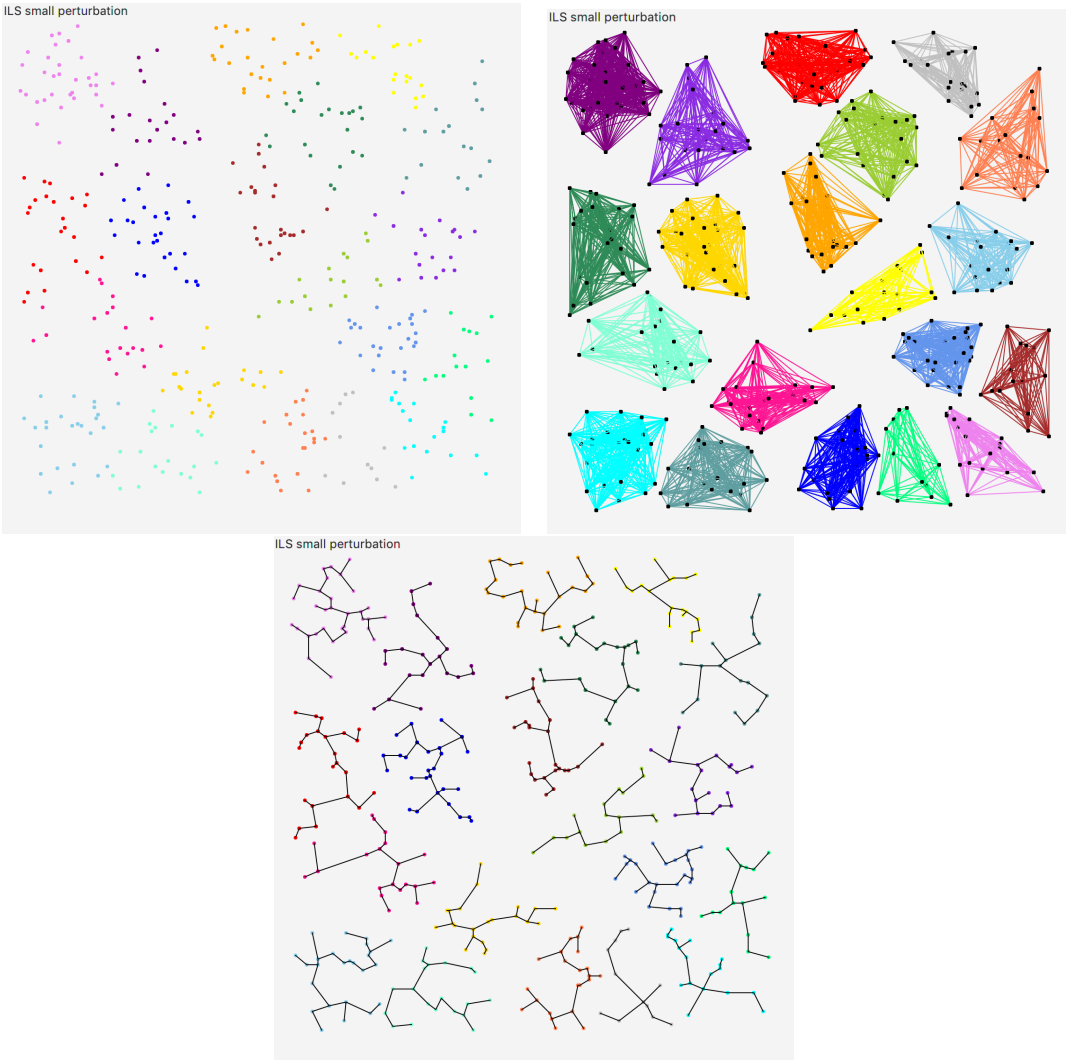
Algorytm MSLS osiągał średnio najlepsze wyniki. Porównywalnie sprawdził się ILS z dużymi perturbacjami. Zgodnie z oczekiwaniami, wprowadzanie niewielkich zmian w przydziale do grup nie wystarczało, aby znacząco oddalić się od minimów lokalnych i ostatecznie algorytm z małymi perturbacjami nie osiągnął dobrych wyników. W porównaniu z poprzednimi testowanymi algorytmami, stworzone w ramach czwartego etapu projektu działają o wiele dłużej. Spowodowane jest to oczywiście ich charakterystyką, tj. wykorzystują wielokrotne uruchomienie lokalnego przeszukiwania.

4. Wizualizacja najlepszych rozwiązań

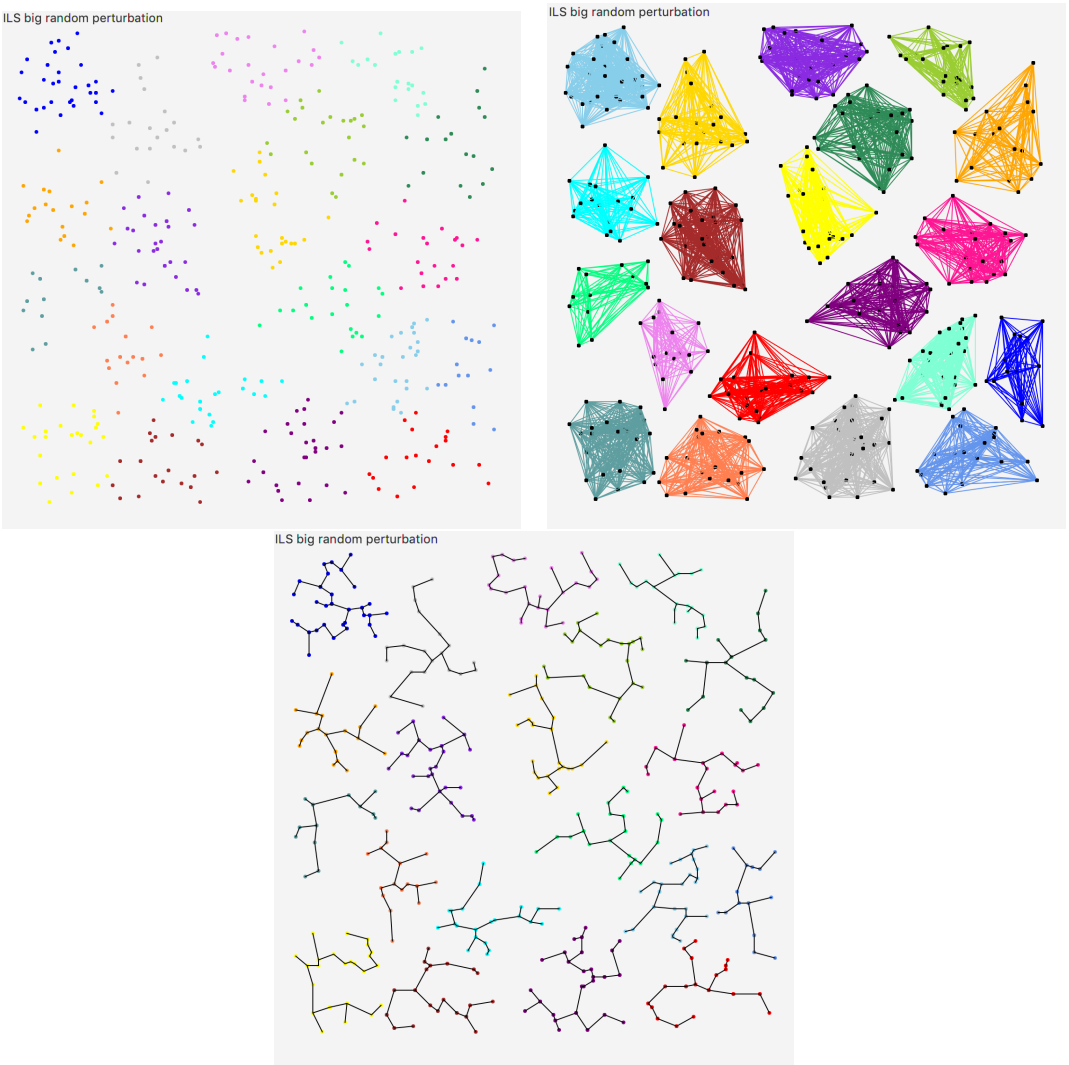
Podobnie jak w poprzednich etapach projektu, również w tym zastosowano wizualizację najlepszych rozwiązań na trzy sposoby. Pierwszy z nich to zaprezentowanie samych grup punktów, bez jakichkolwiek powiązań. Drugim sposobem jest prezentacja zgodna z funkcją celu, czyli zaprezentowanie powiązań pomiędzy punktami w ramach grupy. Ostatni sposób wykorzystuje minimalne drzewo rozpinające, które w przejrzysty sposób prezentuje przydział punktów do grup.



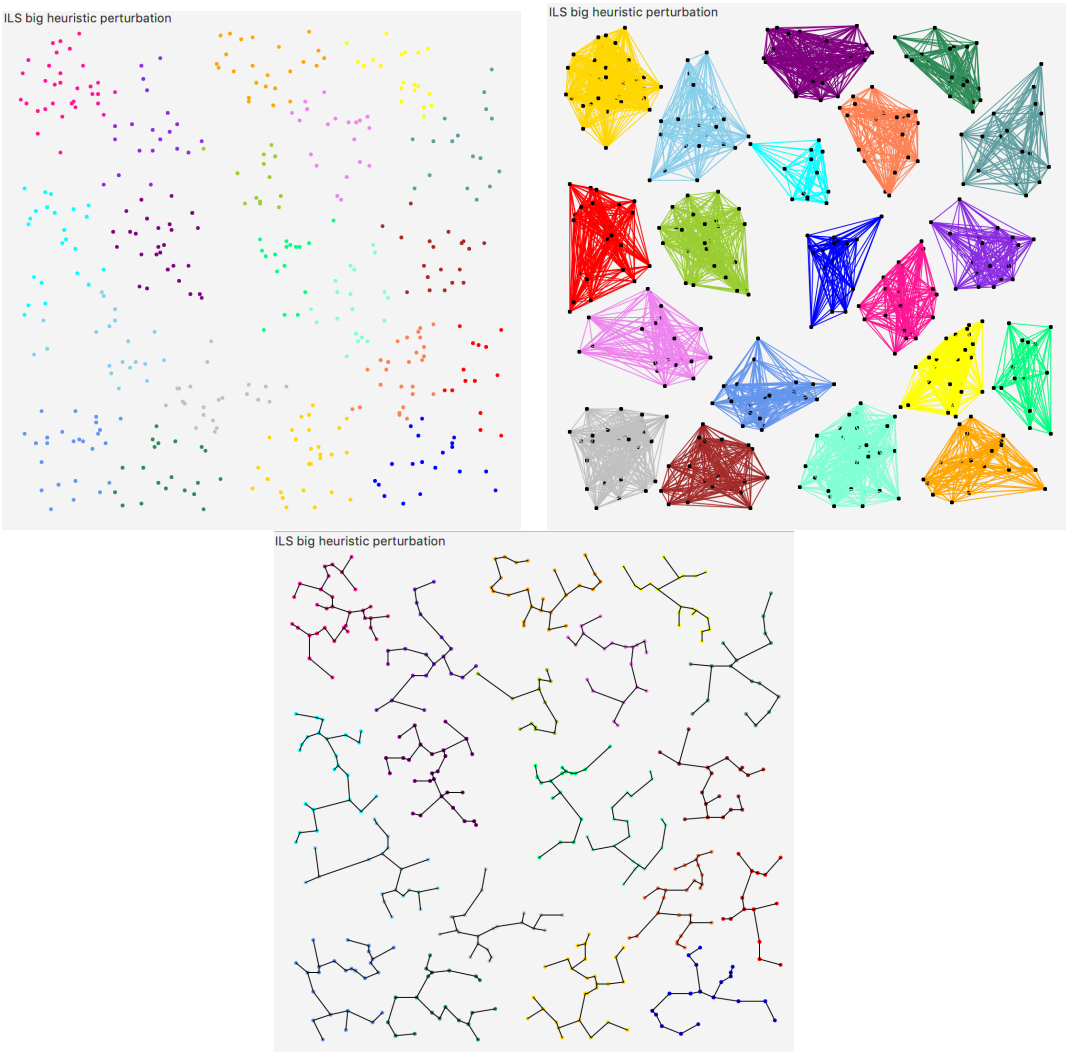
Rysunek 1. MSLS



Rysunek 2. ILS small perturbation



Rysunek 3. ILS big perturbation



Rysunek 4. ILS big perturbation heuristic