***Przemysław Wojsław**, nr ind. 277251*

OAST - projekt
Implementacja algorytmu ewolucyjnego i brute force

1. Algorytmy

a) Brute force

W algorytmie siłowym celem jest stworzenie listy wszystkich możliwych rozwiązań problemu, a następnie wyszukanie wśród nich tego, który daje najlepszy wynik funkcji celu. Podstawą jest więc generacja zbioru rozwiązań. To wykonywane jest w dwóch krokach.

Pierwszy krok: generacja rozwiązań dla poszczególnych zapotrzebowań.

Dla każdego zapotrzebowania generowane są wszystkie możliwe kombinacje obciążania ścieżek w sposób analogiczny do przedstawionego na slajdzie 23 wykładu "OAST-OPT-W5-30-04-2020". Ścieżce o numerze "**curp**" jest przyznawane obciążenie w zakresie od 0 do "**lefth**", zaś pozostałe "**parth**" jest tą samą funkcją rozdzielane pomiędzy kolejne ścieżki. Pseudokod:

```
rec(demand.id, 1, demand.numberOfPaths, demand.volume)

Function rec(curd, curp, maxp, lefth)
Begin
        if curp == maxp:
                create solution_object
                solution_object.set_path(curp, lefth)
                # ścieżki są wkładane począwszy od ostatniej, więc później lista ścieżek jest
                        odwracana
                list_of_possible_solutions = solution_object  # zapoczątkowana zostaje lista
        else:
                for parth=0 to lefth:
                        sub_list_of_solutions = recFindFlows(curd, curp + 1, maxp, lefth - parth)
                        for each solution_object in sub_list_of_solutions:
                                solution_object.set_path(curp, parth)

                        list_of_possible_solutions += sub_list_of_solutions

        return list_of_possible_solutions
```

Drugi krok: złożenie rozwiązań pojedynczych zapotrzebowań w rozwiązania całości.

Wykonywany jest zwykły iloczyn kartezjański list rozwiązań poszczególnych zapotrzebowań, każda powstała w ten sposób kombinacja to jedno możliwe rozwiązanie. Dla każdego otrzymanego rozwiązania wyliczane jest obciążenie łączy i wartość funkcji celu (od razu DAP i DDAP).

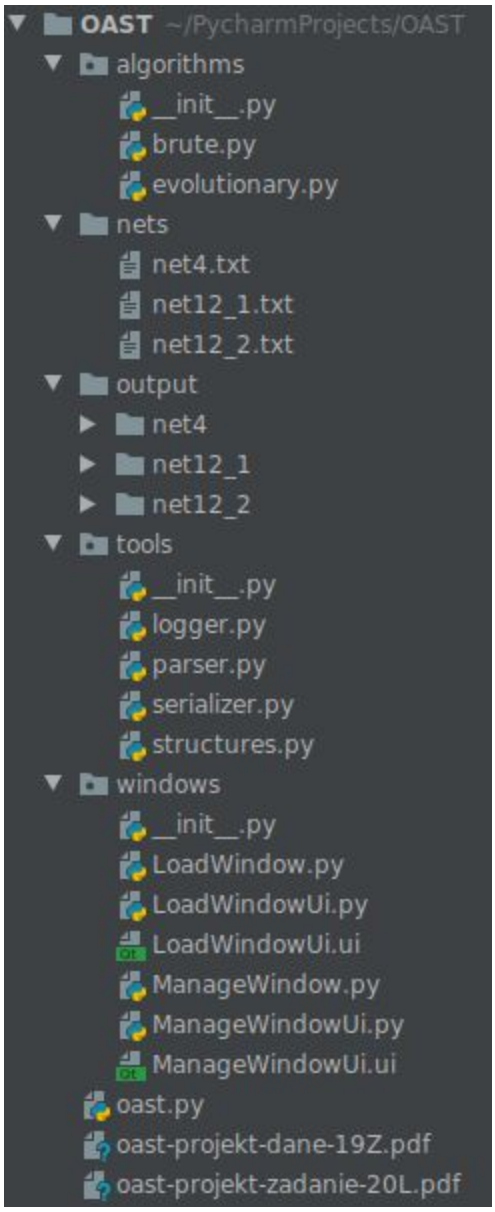
b) Algorytm ewolucyjny

Algorytm ewolucyjny w ogólnej postaci pokrywa się z zaproponowanym na stronie 8 dokumentu z wytycznymi projektu (“oast-projekt-zadanie-20L.pdf”). Można go podzielić na fazy:

- **Inicjalizacja** - losowo generowana jest początkowa populacja (zbiór możliwych rozwiązań) o zadanej wielkości. Tworzenie chromosomu odbywa się poprzez wygenerowanie wszystkich rozwiązań pojedynczych zapotrzebowań analogicznie jak przy *brute force*, a następnie z tej puli losowane są rozwiązania dla każdego chromosomu.
- **Wybór rodziców** - z populacji wybierana jest zadana liczba rodziców, chromosomów o najlepszej wartości funkcji celu.
- **Krzyżowanie** - rodzice są dzieleni na losowe pary, każda z nich ma taką samą zadaną szansę na to ich geny zostaną przemieszane. Powstaje dwójka potomstwa, gen każdego z nich ma po 50% szansy na bycie z poszczególnego rodzica.
- **Mutacja** - każde potomstwa ma zadaną szansę na mutację W takim wypadku losowy wybierany jest gen, z którego zabierana jest jednostka zapotrzebowania i przekazywana innemu, również losowemu genowi (za wyjątkiem sytuacji gdy jest tylko jeden gen, tzn. dane zapotrzebowanie miało tylko jedną możliwą ścieżkę)
- **Selekcja** - potomstwo dołączane jest do dotychczasowej populacji (z której zostali wybrani rodzice), po czym z populacji usuwane jest tyle najgorszych (pod wzgędem funkcji celu) chromosomów ile potomstwa zostało dodane (wielkość populacji nie zmienia się).

2. Implementacja

Projekt został zrealizowany w języku **Python** (wersja 3.6) z użyciem dodatkowej biblioteki **PyQt5** (na potrzeby GUI). Posiada następującą strukturę:



- **algorithms** - algorytmy
- brute force
- ewolucyjny

- **nets** - opisy analizowanych sieci

- **output** - wyniki i logi

Każda sieć zapisuje swoje wyniki we własnym folderze (który jest tworzony, jeśli go nie ma).

- **tools** - narzędzia do konkretnych zadań:

a) logger - do zapisu wyników z poszczególnych uruchomień programu
b) parser - do odczytu sieci z pliku .txt
c) serializer - do zapisu sieci w pliku .txt
d) structures - modele danych

- **windows** - okna na bazie PyQt5

a) LoadWindow - do wskazania pliku z siecią
b) ManageWindow - główne okno zarządzania programem.

- **oast.py** - główny plik programu

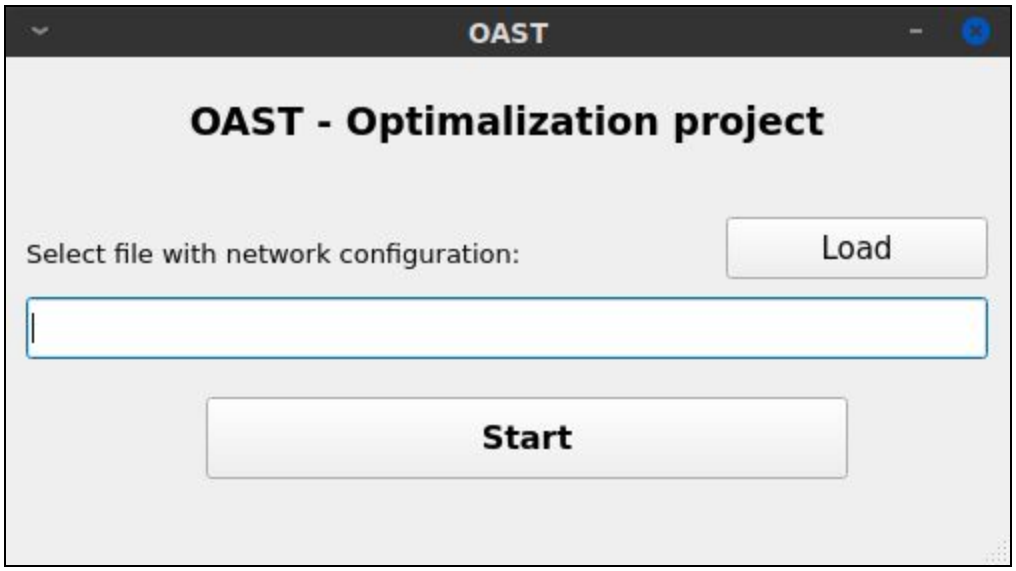
Modele danych jak i zapis rozwiązania w pliku są w konwencji zgodnej z przykładem na stronie 7 z wytycznych projektu (“oast-projekt-zadanie-20L.pdf”), z dodatkowym wstawieniem separatora (“-1”) i EOL w miejscach analogicznych do formatu wejściowego.

3. Instrukcja uruchomienia

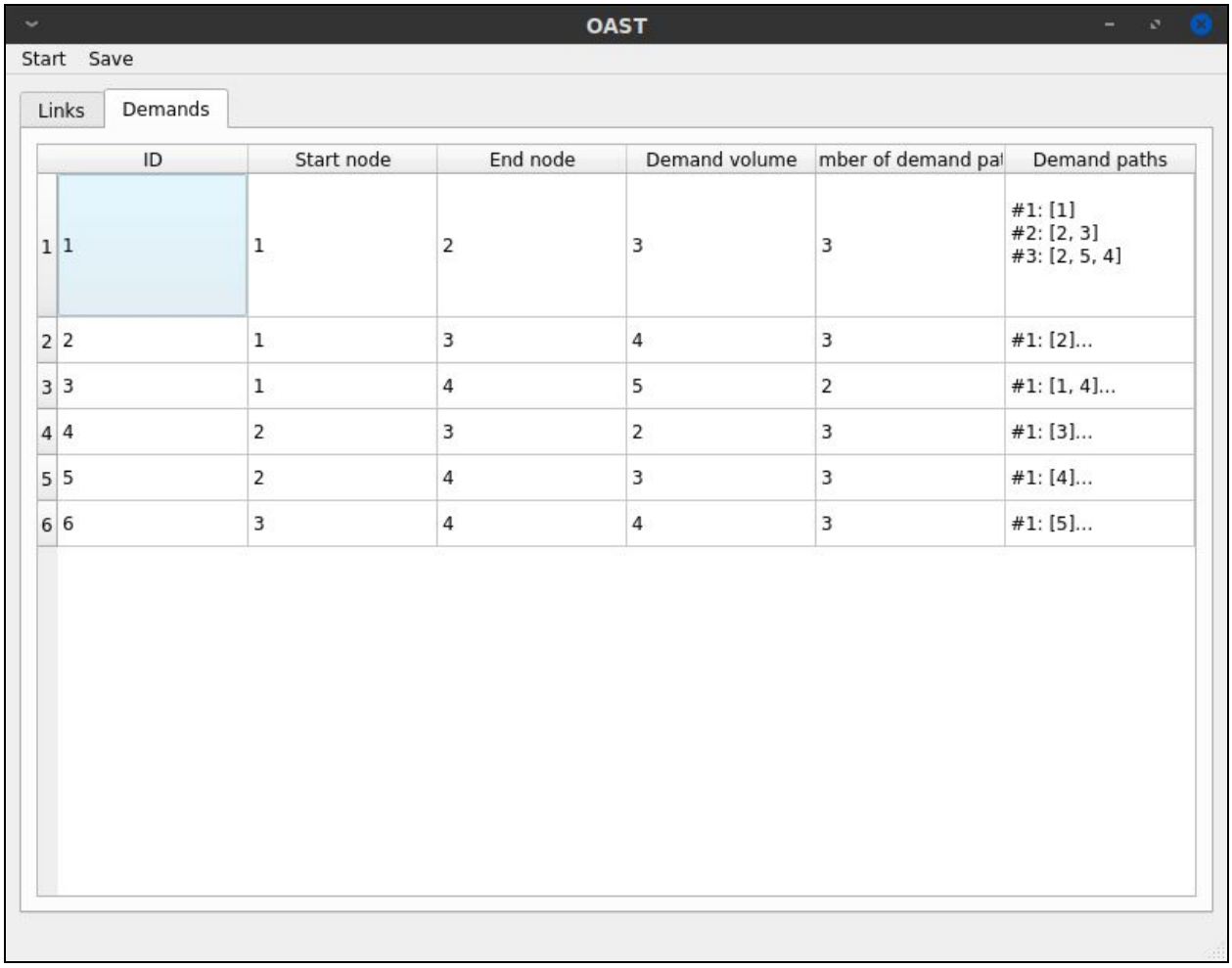
Do poprawnego działania programu konieczne jest posiadania biblioteki **PyQt5**. Program uruchamiany jest z poziomu głównego folderu projektu komendą:

python3 oast.py

Po jej wykonaniu pojawia się okno wyboru sieci. Należy kliknąć przycisk “**Load**” i wybrać odpowiedni plik, a następni nacisnąć “**Start**”.

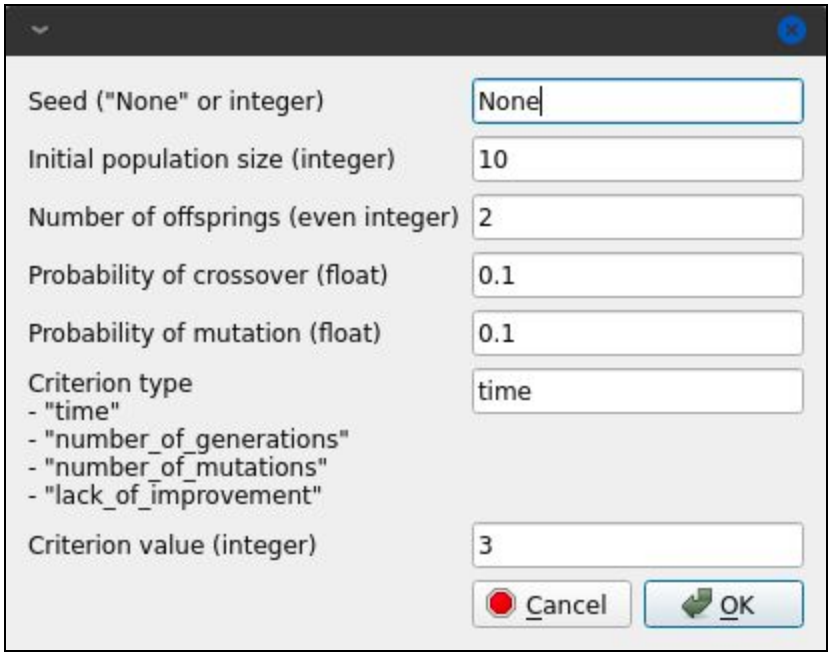


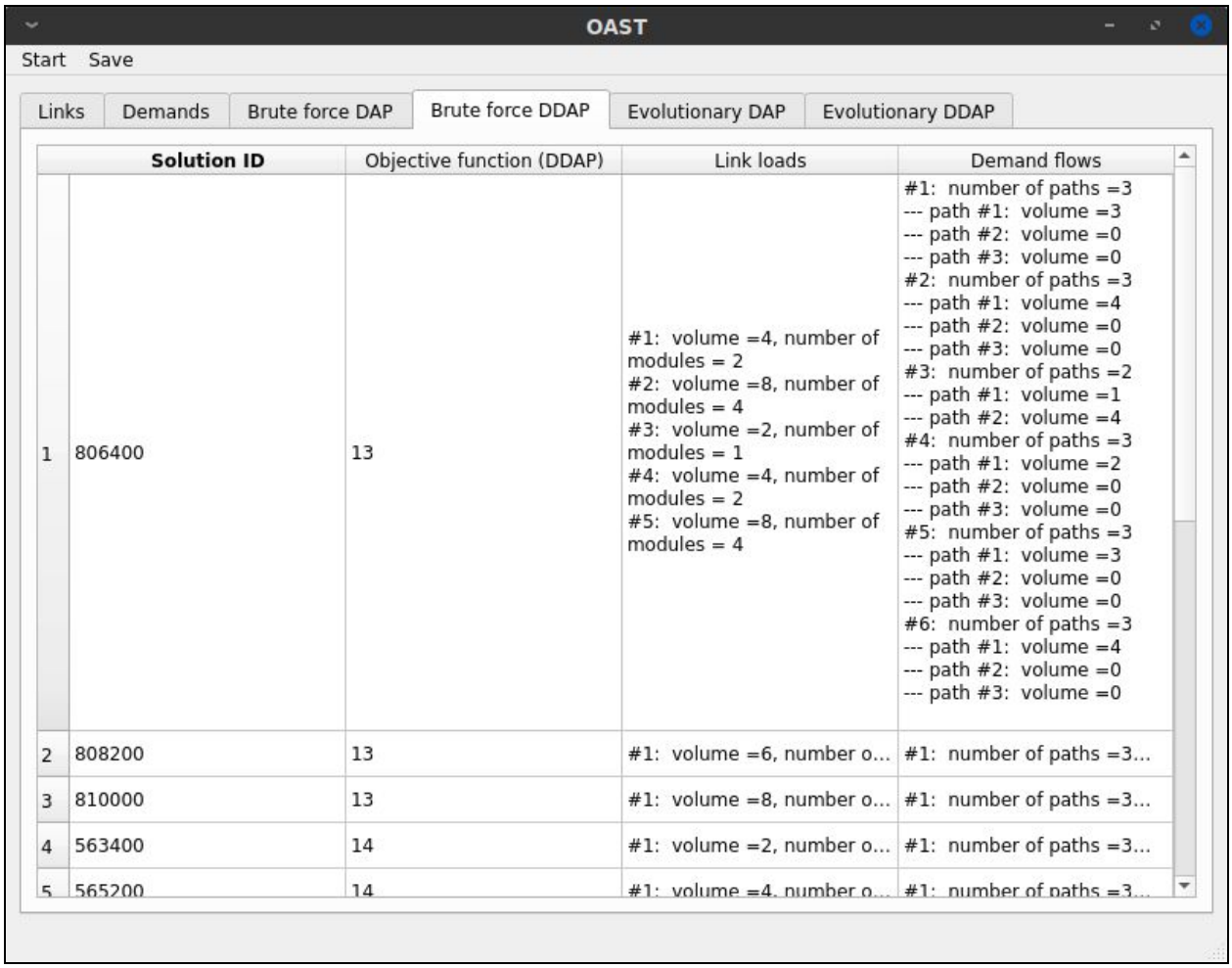
Pojawi się nowe okno zarządzania. Przełączając się pomiędzy kartami “**Links**” i “**Demands**” można zapoznać się z danymi odczytanymi z pliku. Aby zobaczyć wszystkie ścieżki w kolumnie “Demand paths” należy ręcznie rozszerzyć wiersz nakierowując kursor na linię rozdzielającą wiersze po lewej stronie (przy ich numerach porządkowych). Okno można powiększyć do rozmiarów ekranu (poprzez dwukrotne kliknięcie w górny pasek z nazwą “OAST”) lub dowolnego innego (poprzez odpowiednie działanie kursorem).



Aby rozpocząć wykonywanie algorytmu należy nacisnąć odpowiednią opcję z menu "**Start**" nad tabelami. Algorytm *brute force* rozpocznie się automatycznie, zaś przy wyborze ewolucyjnego pojawi się okienko pomocnicze do wpisania odpowiednich parametrów. W nawiasach podane są typu przyjmowanej wartości. **Kryterium** ustala się wpisując jeden z podanych tekstów (z podkreśleniami) i niżej wartość liczbową (sekund czy generacji). Kliknięcie "**OK**" rozpoczyna algorytm (a "Cancel" odwołuje operację). Algorytmy ewolucyjne można powtarzać z różnymi parametrami (ale plik z najlepszym rozwiązaniem jest nadpisywany).

Podczas wykonywania algorytmów okno programu jest wyszarzone i niemożliwe jest przeprowadzanie żadnych działań aż do zakończenia algorytmu. Pojawiają się wtedy dodatkowe karty z wynikami - 30 najlepszych rozwiązań dla metody *brute force*, a dla ewolucyjnej 30 ostatnich generacji. Kolumna "**Solution ID**" zawiera unikalny numer porządkowy rozwiązania (dla *brute force*) lub numer generacji (dla ewolucyjnego).


Seed ("None" or integer)
None
Initial population size (integer)
10
Number of offsprings (even integer)
2
Probability of crossover (float)
0.1
Probability of mutation (float)
0.1
Criterion type
- "time"
- "number_of_generations"
- "number_of_mutations"
- "lack_of_improvement"
time
Criterion value (integer)
3
Cancel
OK


OAST
Start
Save
Links
Demands
Brute force DAP
Brute force DDAP
Evolutionary DAP
Evolutionary DDAP
Solution ID
Objective function (DDAP)
Link loads
Demand flows
1
806400
13
#1: volume =4, number of modules = 2
#2: volume =8, number of modules = 4
#3: volume =2, number of modules = 1
#4: volume =4, number of modules = 2
#5: volume =8, number of modules = 4
#1: number of paths =3
--- path #1: volume =3
--- path #2: volume =0
--- path #3: volume =0
#2: number of paths =3
--- path #1: volume =4
--- path #2: volume =0
--- path #3: volume =0
#3: number of paths =2
--- path #1: volume =1
--- path #2: volume =4
#4: number of paths =3
--- path #1: volume =2
--- path #2: volume =0
--- path #3: volume =0
#5: number of paths =3
--- path #1: volume =3
--- path #2: volume =0
--- path #3: volume =0
#6: number of paths =3
--- path #1: volume =4
--- path #2: volume =0
--- path #3: volume =0
2
808200
13
#1: volume =6, number o...
#1: number of paths =3...
3
810000
13
#1: volume =8, number o...
#1: number of paths =3...
4
563400
14
#1: volume =2, number o...
#1: number of paths =3...
5
565200
14
#1: volume =4, number o...
#1: number of paths =3...

W czasie wykonywania algorytmu w terminalu z którego uruchomiony został program pojawiają się logi dotyczące wyników algorytmu. Takie same logi są zapisywane w folderze “**output**” w odpowiednim podkatalogu. Tam też zapisywane jest najlepsze rozwiązanie w podanej wcześniej konwencji. Oprócz tego można zapisać dowolne rozwiązanie (po wykonaniu algorytmu *brute force*) lub najlepsze rozwiązanie wybranej generacji (po wykonaniu algorytmu ewolucyjnego). W tym celu należy kliknąć “**Save**” w górnym menu, a następnie podać “Solution ID” (czyli odpowiednio numer porządkowy rozwiązania lub numer generacji). Wybór opcji typu “best” spowoduje powtórne zapisanie najlepszego wyniku (np. Jeśli przypadkiem usunęło się poprzedni).

4. Wyniki

Wszystkie wyniki można znaleźć w folderze “**output**”. Pliki typu “*best.txt” zawierają opis najlepszego rozwiązania zgodnie z konwencją na stronie 7 wytycznych projektu (“oast-projekt-zadanie-20L.pdf”) zmodyfikowana o dodatkowy separator (“-1”). Dodatkowe dane (np. parametry użytego algorytmu ewolucyjnego) są zawarte w plikach z logami (“logs_[date].txt”). Pliki z logami nie są nadpisywane, ale pliki z najlepszym rozwiązaniem tak (np. przy powtórnym uruchomieniu algorytmy ewolucyjnego z innymi parametrami).