OAST - projekt 2

Przedmiot	OAST	Projekt	2
Zespół	Imię	Nazwisko	Nr albumu
-	Kacper	Murat	300581
	Dominik	Topyła	300522

1. Opis implementacji

W pliku main.py utworzono 4 klasy: Link, Path, Demand i Network (reprezentujące odpowiednio łącze, ścieżkę, żądanie, sieć). Klasa Network zawiera w sobie listy obiektów typu Link, Path, Demand. Klasa Demand zawiera w sobie listę obiektów typu Path. W funkcji main znajdują się parametry wejściowe programu (Punkt 3.) oraz uruchamiana jest symulacja (funkcja simulation()). W zależności od wyboru użytkownika program rozwiązuje problem DAP (ang. Demand Allocation Problem) lub DDAP (ang. Dimensioning and Demand Allocation Problem) metodą Ewolucyjną lub metodą Bruteforce. Po uruchomieniu symulacji program odczytuje z pliku dane wejściowe (funkcja Network.parse()) i tworzy sieć. Następnie w zależności od wybranej metody wywoływana jest funkcja Network.bruteForce() lub Network.evolution() rozwiązująca wybrany problem wywołaną metodą.

Implementacja algorytmu ewolucyjnego

Zaimplementowano algorytm ewolucyjny według strategii (μ + λ), przy czym μ = λ. Na początku generowana jest populacja o liczności μ. Dla każdego chromosomu obliczana jest funkcja celu. Chromosomy z największą funkcją celu mają największą szansę na reprodukcję. Reprodukcja polega na generowaniu populacji o liczności λ. Przy czym zastosowano zasadę, że μ razy losowany jest ze zwracaniem chromosom, który zostanie przypisany do tymczasowej populacji. W efekcie populacja tymczasowa ma taką samą liczność jak populacja początkowa. Na populacji tymczasowej wykonywane są operacje krzyżowania i mutacji (z określonymi w parametrach wejściowych prawdopodobieństwami). W wyniku tych operacji tworzona jest populacja potomna. Dla każdego chromosomu z populacji obliczana funkcja celu. Populacja potomna dodawana jest do bazowej. Z populacji bazowej wybieranych jest μ chromosomów z najlepszą wartością funkcji celu. Otrzymana populacja jest populacja bazową w następnym kroku pętli.

Implementacja algorytmu Brute Force

Algorytm Brute Force polega na sukcesywnym sprawdzaniu wszystkich możliwych rozwiązań w poszukiwaniu tych najlepszych. Metoda ta jest skuteczna, gdyż zawsze pokazuje idealne rozwiązanie, jednak nie jest efektywna, gdyż dla złożonych problemów wymaga dużego czasu obliczeń.
Algorytm ze względu na szybkość działania został zaimplementowany w sposób iteracyjny zamiast rekurencyjnego. W efekcie w niezbyt wydajnym Pythonie dla pliku 'net4.txt' program znajduje wszystkie rozwiązania w zaledwie kilka sekund. Podejście iteracyjne nie zajmuje pamięci komputera przez wywoływanie wielopoziomowo zagnieżdżonych funkcji. Efekt osiągnięto przez zastosowanie zasady licznika zarówno przy generowaniu kolejnych kombinacji przepływów w ścieżkach dla jednego żądania (Demand.nextFlowDistribution()), jak i generując kombinacje zestawów żądań

(Network.bruteForce()). Sama zasada działania licznika opiera się na inkrementowaniu najmniej znaczącego znaku do czasu osiągnięcia przez niego wartości maksymalnej. Następnie znak ten jest zerowany, co towarzyszy inkrementacji o jedną jednostkę kolejnego najmniej znaczącego znaku. W ten sposób po osiągnięciu maksymalnej wartości przez najbardziej znaczący znak wykonane zostały wszystkie możliwe kombinacje wartości na wszystkich znakach.

2. Instrukcja uruchomienia programu

Do uruchomienia programu wymagane jest zainstalowane narzędzie Python oraz biblioteki numpy i scipy. Ważne aby program źródłowy znajdował się w tym samym katalogu co katalogi input (z plikami wejściowymi) i output (do którego zapisywane są wyniki). Program uruchamiany jest przy pomocy polecenia:

```
python main.py
```

Parametry wejściowe programu znajdują się w liniach 441-451:

```
seed = 'random'              # ['random'|(int)]
inputFileName = 'net4.txt'
outputFileName = 'result.txt'
problem = 'DDAP'              # ['DAP'|'DDAP']
method = 'Brute Force'             # ['Brute Force'|'Evolution']
crossoverProbability=0.75
mutationProbability=0.05
numberOfChromosomes=4
stopConditionType='generations'
   #['time'|'generations'|'mutations'|'bestForN']
stopConditionValue=5
saveAllBFSolutions=True
```

Zostały one opisane w poniższej tabeli:

Zmienna	Typ	Opis
seed	string/int	Ziarno generatora liczb losowych. W przypadku podania całkowitej wartości uruchamia program z podanym ziarnem. Opcjonalnie wartość ‘random’ spowoduje wykonanie programu z losowym ziarnem.
inputFileName	string	Nazwa pliku wejściowego znajdującego się w katalogu ‘input/’
outputFileName	string	Nazwa pliku wyjściowego który zostanie zapisany w katalogu ‘output/’
problem	string	Nazwa rozważanego problemu. Dopuszczalne wartości: ‘DAP’, ‘DDAP’
method	string	Wykorzystywana metoda. Dopuszczalne wartości: ‘'Brute Force' i 'Evolution', odpowiednio dla algorytmu Brute Force i algorytmu ewolucyjnego

crossoverProbability	float	Prawdopodobieństwo wystąpienia krzyżowania w algorytmie ewolucyjnym.
mutationProbability	float	Prawdopodobieństwo wystąpienia mutacji w algorytmie ewolucyjnym.
numberOfChromosomes	int	Liczebność populacji startowej w algorytmie ewolucyjnym.
stopConditionType	string	Warunek zakończenia programu w algorytmie ewolucyjnym. Dopuszczalne wartości: - 'time' - program zatrzyma się po czasie określonym przez 'stopConditionValue' wyrażonym w sekundach - 'generations' - program zatrzyma się po wystąpieniu liczby generacji określonej przez 'stopConditionValue' - 'mutations' - program zatrzyma się po wystąpieniu liczby mutacji określonej przez 'stopConditionValue' - 'bestForN' - program zatrzyma się, jeśli w kolejnych X generacjach najlepsza wartość funkcji celu się nie zmieni, gdzie X określany jest przez 'stopConditionValue'
stopConditionValue	int	Wartość po której osiągnięciu program skończy działanie.
saveAllBFSolutions	bool	Zmienna decycująca czy w algorytmie Brute Force do pliku mają zostać zapisane wszystkie rozwiązania. Dopuszczalne wartości: True lub False.

3. Wyniki

Algorytm BruteForce działa poprawnie i szybko. Niestety dla dużych sieci, nie może wykonać się w skończonym czasie. Niestety algorytm ewolucyjny zdawał się nie działać prawidłowo, co determinowało decyzję o zostawieniu wyników do czasu obrony projektu.