Optymalizacja Kombinatoryczna - Projekt 1

Wykonali:
Filip Rosiak - 151799 - filip.rosiak@student.put.poznan.pl
Jakub Stefański - 151876 - jakub.stefanski.1@student.put.poznan.pl

Data wręczenia sprawozdania: …….

I. Informacje podstawowe

Temat projektu: Problem III - Capacitated Vehicle Routing Problem with Time Windows (problem marszrutyzacji z ograniczeniami pojemności i oknami czasowymi)

Opis problemu: problem decyzyjny polegający na wyznaczeniu optymalnych tras przewozowych dla pewnej ściśle określonej liczby środków transportu, której zadaniem jest obsłużenie zbioru klientów znajdujących się w różnych punktach przy zachowaniu ograniczeń. Kryterium optymalizacji jest całkowity koszt transportu (wyrażony odległościowo, cenowo lub czasowo).

Kod źródłowy - język C++.
Środowisko - Linux.

Obsługa program:

- sugerowana kompilacja: *g++ -std=c++11 cvrptw.cpp*
- uruchamianie programu: *./a.out <plik_wejściowy> <plik_wyjściowy> <ilość klientów do wczytania(-1 aby wczytać wszystkich klientów)>*

II. Rozwiązania problemu

a) algorytm zachłanny

Algorytm na początku każdej iteracji głównej pętli while za punkt startowy obiera magazyn, a za następnego klienta wybiera domyślnie pierwszego klienta zapisanego w wektorze klientów. Następnie, sprawdza po kolei wektor w celu znalezienia klienta, którego jest w stanie obsłużyć i który najlepiej spełnia warunki funkcji *najlepszy_lokalnie_klient.* W przypadku znalezienia następnego klienta dodajemy go do obecnej trasy i następuje jego obsługa tzn. aktualizujemy zmienne związane z obsługą tras i usuwamy go z wektora dostępnych klientów.

W przeciwnym przypadku, tj. kiedy nie znaleziono lepszego klienta od domyślnego i nie może zostać on obsłużony w tej trasie następuje podsumowanie trasy i rozpoczęcie następnej. Alternatywą na zakończenie trasy jest skończenie się ładunku w ciężarówce. Cykl ten wykonuje się aż w liście klientów zostanie sam magazyn.

Główną zaletą tego algorytmu jest jego szybkość (złożoność O(n2)), a nie jakość jego rozwiązań.

```
194         int nastepny_klient = 1; // pomijamy magazyn (klienci[0] = magazyn), przypisujemy pierwszego klienta z listy
195         for (int i = 2; i < (int)klienci.size(); i++) {  // szukanie najlepszego lokalnego następnego klienta
196             if (czy_dojedzie(obecny_klient, klienci[i], czas) &&
197                 zdazy_wrocic(obecny_klient, klienci[i], czas) &&
198                 czy_wystarczy(klienci[i], ladownosc) &&
199                 najlepszy_lokalnie_klient(obecny_klient, klienci[i])
200                 < najlepszy_lokalnie_klient(obecny_klient, klienci[nastepny_klient])) {
201                 nastepny_klient = i;    // przypisanie zalezionego lepszego klienta
202             }
203         }
204 
205         // gdy nie znajdzie nasteępnego pasującego klienta i piewszy z listy(ustawiany domyślnie) nie może być obsłużony,
206         // przypisuje wartość -1, która reprezentuje brak następnego klienta
207         if (nastepny_klient == 1 &&
208             (!czy_dojedzie(obecny_klient, klienci[1], czas) ||
209                 !zdazy_wrocic(obecny_klient, klienci[1], czas) ||
210                 !czy_wystarczy(klienci[1], ladownosc))) {
211             nastepny_klient = -1;
212         }
```

Krótki opis funkcji występujących w programie:

dodaj_klienta() - dodaje klienta do globalnego wektora klientów

wczytanie_pliku() - wczytywanie danych z pliku wejściowego

czy_wystarczy() - sprawdza, czy ciężarówce wystarczy towaru na konkretnego klienta

czy_dojedzie() - sprawdza czy ciężarówka zdąży dojechać do klienta(lub magazynu) przed końcem jego obsługi

zdazy_wrocic() - sprawdza czy ciężarówka zdąży wrócić do magazynu po obsłużeniu danego klienta

najlepszy_lokalnie_klient() - ocenia najlepszego następnego klienta(w tym przypadku szuka najbliższego klienta od obecnej lokalizacji)

czy_dopuszczalne() - sprawdza, czy da się obsłużyć każdego klienta niezależnie, służy do walidacji problemu

zachlanny() - implementacja wyżej opisanego algorytmu zachłannego w programie