**PROJEKTOWANIE EFEKTYWNYCH ALGORYTMÓW**

PROJEKT

11/12/2021

252736 Hutnik Szymon

Simulated Annealing (4)

|  |  |
| --- | --- |
| Strona | Spis treści |
| 2 | Treść zadania |
| 3 | Opis metody |
| 4 | Opis algorytmu |
| 6 | Dane testowe |
| 7 | Procedura badawcza |
| 9 | Wyniki |
| 10 | Analiza |

1. Treść zadania

Opracować, napisać, zbadać rozwiązanie problemu komiwojażera w wersji optymalizacyjnej algorytmem *Simulated Annealing*.

Problem komiwojażera (eng. *Travelling Salesman Problem*) polega na znalezieniu minimalnego cyklu Hamiltona (przejście przez wszystkie wierzchołki tylko raz, startując i kończąc w tym samym punkcie) w pełnym grafie ważonym.

1. Opis metody

Metoda symulowanego wyżarzania polega na modyfikowaniu rozwiązania, aż temperatura spadnie do momentu „zastygnięcia” wyniku – jest to odniesieniem do zjawiska stygnięcia metali. Jeżeli spadek temperatury jest wystarczający wolny to cząsteczki tworzą równomierną strukturę. W termodynamice do opisania tego zjawiska wykorzystywany jest następujący wzór:

Gdzie: k jest stałą Boltzmanna.

Wzór ten, w trochę zmienionej formie, wykorzystywany jest do obliczenia prawdopodobieństwa wybrania rozwiązania gorszego. Algorytm symulowanego wyżarzania bazuje na metodzie iteracyjnej. Metoda ta zakłada, że ciągle ulepszamy istniejące rozwiązanie, aż w końcu nie będzie można go dalej poprawić. Przejście z jednego rozwiązania do drugiego polega na znalezieniu lepszego rozwiązania sąsiedniego, czyli takiego, który znajduje się w sąsiedztwie wcześniejszego. W algorytmie symulowanego wyżarzania istnieje jednak możliwość wyboru gorszego rozwiązania z pewnym prawdopodobieństwem. Szansa na wybór gorszego rozwiązania jest zależna od m.in. temperatury, która maleje wraz z czasem wykonywania programu. Taki mechanizm sprawia, że podczas wykonywania algorytmu będzie możliwość „wyjścia” z minimum lokalnego, czyli zastąpienie minima lokalnego rozwiązaniem gorszym.

1. Opis algorytmu

Rozwiązanie zaimplementowano w postaci programu opisanego przez poniższe diagramy:

Diagram

Description automatically generated

Rysunek 1: Ogólny diagram czynności programu

Najpierw inicjalizowane są zmienne. Po wczytaniu danych z konsoli następuje uruchomienie właściwej części algorytmu, następnie wypisywany jest wynik oraz czas wykonania właściwego algorytmu.

Diagram

Description automatically generated

Rysunek 2: Szczegółowy diagram czynności algorytmu

1. Dane testowe

Dane, na których była badana efektywność algorytmu pochodzą ze zbioru udostępnionego przez dr Rudego. Do badania użyto wartości z następujących plików:

* m14.astp
* m15.astp
* ulysses16.tsp
* gr17.tsp
* gr21.tsp
* ulysses22.tsp
* gr24.tsp

http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/files/pea/instances.zip

1. Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu od wielkości instancji. W przypadku algorytmu realizującego przegląd zupełny przestrzeni rozwiązań dopuszczalnych nie występowały parametry programu, które mogły mieć wpływ na czas i jakość uzyskanego wyniku. W związku z tym procedura badawcza polegała na uruchomieniu programu i wklejeniu do niego danych z plików wybranych do badania.

Każda z instancji została wykonana do 10 razy, aby uśrednić czasy, limitem okazało się n=22, mimo usuwania poprzednich wyników zajęta pamięć (według Visual Studio 2019) przekroczyła 19GB, test został przerwany. Wyniki były zapisywane w Excelu, następnie na ich podstawie została przeprowadzona analiza.

Pomiar czasu został wykonany przy użyciu bilbioteki chrono. Po otrzymaniu wyniku należy go podzielić przez liczbę powtórzeń wywołań algorytmu.

Text

Description automatically generated



1. Wyniki

Wyniki opracowano w programie Excel:

Rysunek 3: Czasy działania algorytmu dla *n* wierzchołków - graf



Rysunek 4: Czasy działania algorytmu dla *n* wierzchołków - tabela

Rysunek 5: Zużycie pamięci algorytmu dla *n* wierzchołków - graf

Rysunek 6: Porównanie czasu działania dotychczas opracowanych algorytmów dla *n* wierzchołków - graf

\*dane testowe nie są kolejnymi wartościami n stąd połączenie znanych wyników linią

1. Analiza

Krzywa wzrostu czasu względem wielkości instancji ma charakter losowy (Rysunek 3). Wynika to z faktu, że algorytm B&B ma złożoność czasową od O(*n*) do O(*n!*), tak niska złożoność wynika z faktu, że jeśli jedna ścieżka będzie tańsza od kosztu wejścia do każdego innego pierwszego wierzchołka niż wybrany to zrealizujemy tylko ją i będziemy mieć pewność, że otrzymaliśmy najlepszy wynik. Widać to na wykresach, gdzie czas wykonania *n*=17 jest dużo wyższy niż dla *n*=21. Warto zwrócić uwagę, że nawet wykorzystując dynamiczne struktury danych użyteczność algorytmu jest ograniczona, ponieważ złożoność pamięciowa wynosi od O(*n\*n*2) do O(*n!\*n*2).

Porównanie dotychczas zrealizowanych algorytmów pokazuje jak dużą przewagę ma podejście dynamiczne nad siłowym. Czas wykonania Helda-Karpa zauważalnie niższy jednak jest to okupione wysokim zużyciem pamięci. W przypadku B&B mamy szansę na ogromną poprawę czasu, ale również większą szansę na przekroczenie pamięci, co w moim odczuciu stawia to podejście poniżej algorytmu Helda-Karpa ze względu na jego zawodność.