Rozproszone rozwiązanie problemu komiwojażera (travelling salesman problem) - kolonia mrówek (ant colony)

Maksym Kazhaiev, Volodymyr Tymkiv

Problem

Problem komiwojażera (TSP) to klasyczne zadanie optymalizacyjne, w którym należy znaleźć najkrótszą trasę, która odwiedza wszystkie miasta dokładnie raz i wraca do miasta początkowego. Jest to problem znany z kombinatorycznej złożoności obliczeniowej, co oznacza, że liczba możliwych rozwiązań rośnie wykładniczo wraz z liczbą miast.

Algorytm mrówkowy (ACO) jest metaheurystyką inspirowaną zachowaniami mrówek w naturze. Mrówki poruszają się losowo, pozostawiając ślady feromonowe, które wzmacniają drogi, które zostały wybrane przez większą liczbę mrówek. Algorytm ACO stosuje tę ideę do rozwiązywania problemów optymalizacyjnych, takich jak TSP.

W kontekście problemu TSP, algorytm ACO działa w następujący sposób:

1. Inicjalizacja Feromonów: Początkowo, feromony na wszystkich trasach są ustawiane na równą wartość. Jest to etap początkowy, gdzie mrówki nie mają preferencji dla żadnej trasy.
2. Ruch Mrówek: Mrówki są umieszczane losowo w różnych miastach i poruszają się po trasach w sposób losowy, wybierając kolejne miasta zgodnie z pewnym prawdopodobieństwem, które zależy od ilości feromonów i odległości do miasta. Mrówki pozostawiają feromony na trasach, którymi przemieszczają się.
3. Aktualizacja Feromonów: Po zakończeniu ruchu wszystkich mrówek, feromony na trasach są aktualizowane. Krótsze trasy otrzymują większą ilość feromonów, co z czasem prowadzi do wzmocnienia tych tras.
4. Parowanie Feromonów: Aby zapobiec zbyt szybkiemu skupieniu się feromonów na jednej trasie, feromony parują z czasem, zmniejszając swoją wartość.
5. Warunek Stopu: Algorytm kontynuuje iteracje przez określoną liczbę razy lub do momentu, gdy zostanie spełniony pewien warunek stopu.
6. Wybór Najlepszej Trasy: Po zakończeniu wszystkich iteracji, algorytm wybiera trasę z największą ilością feromonów, co jest równoznaczne z najkrótszą trasą w przypadku problemu TSP.

Algorytm ACO jest często wykorzystywany do rozwiązywania problemu TSP ze względu na swoją zdolność do znajdowania zbliżonych do optymalnych rozwiązań w stosunkowo krótkim czasie, nawet dla dużych instancji problemu. Dodatkowo, algorytm ten jest łatwo skalowalny i można go łatwo dostosować do równoległego przetwarzania, co czyni go atrakcyjnym dla zastosowań w rozproszonych systemach obliczeniowych.

Implementacja

Projekt jest zaimplementowany w języku C z wykorzystaniem biblioteki MPI. Kod źródłowy znajduje się w pliku `main.c`. Projekt wykorzystuje pomocnicze skrypty napisane w języku Python dla generacji danych i tworzenia wykresu.

Struktury i Zmienne Globalne

`City` : Struktura przechowująca współrzędne miasta.

`Ant` : Struktura reprezentująca mrówkę, przechowująca trasę mrówki i długość trasy.

`pheromones` : Dwuwymiarowa tablica przechowująca ścieżki feromonowe między miastami.

`cities` : Tablica przechowująca współrzędne wszystkich miast. ants: Tablica przechowująca wszystkie mrówki.

Stałe Parametry Algorytmu ACO

`ALPHA`: Waga feromonów.

`BETA`: Waga informacji heurystycznej.

`Q`: Stała depozytu feromonu.

`RHO`: Współczynnik parowania feromonu.

Funkcje

`initialize()`: Inicjalizuje miasta, feromony i mrówki.

`ant_movement()`: Symuluje ruch mrówek.

`update_pheromones()`: Aktualizuje feromony w zależności od śladów pozostawionych przez mrówki.

`select_next_city()`: Wybiera kolejne miasto, które mrówka odwiedzi.

`distance()`: Oblicza odległość euklidesową między dwoma miastami.

`main()`: Główna funkcja programu, zarządza algorytmem ACO.

Pętle Główne

Główna pętla algorytmu ACO przetwarza kolejne iteracje do osiągnięcia maksymalnej liczby iteracji. W każdej iteracji, mrówki poruszają się po trasach, aktualizują feromony, a najlepsza znaleziona trasa jest aktualizowana.

Komunikacja MPI

Proces 0 pełni rolę głównego procesu. Po inicjalizacji, pomiędzy procesami następuje rozproszenie tablicy feromonów i zbieranie zaktualizowanych wartości feromonów.

Argumenty Wiersza Poleceń

Liczba mrówek `num_ants` i liczba iteracji `max_iterations` są pobierane z argumentów wiersza poleceń.

Uruchomienie

Projekt uruchamia się za pomocą pliku Makefile. Plik zawiera następne argumenty dla polecenia:

`all`: Buduje główny plik wykonywalny i generuje dane dla problemu TSP, uruchamia algorytm z użyciem mpiexec, a następnie wywołuje skrypt Pythona `plot.py` do wygenerowania wykresu.

`main`: Buduje plik wykonywalny dla głównego programu.

`generate_data`: Generuje dane dla problemu TSP przy użyciu skryptu Pythona.

`clean`: Czyści pliki wykonywalne i pliki wygenerowane przez Makefile.

`prepare_data`: Podobnie jak generate_data, generuje dane dla problemu TSP, ale jest to specjalny cel przygotowujący dane.

`generate_nodes`: Generuje listę węzłów na podstawie skryptu stacji.

W pliku można zainicjalizować zmienne globalne `NUM_CITIES` dla ilości węzłow w grafie oraz `NUM_ANTS`.

`1MAX_ITERAEIONS` : ustawinie ilości iteracji w programie

Schemat blokowy