Instrukcja do Projektu Końcowego: "Misja Ratunkowa"

Temat: Implementacja Algorytmu Roju Cząstek (PSO) w języku C

1. Wstęp Teoretyczny: Czym jest PSO?

Particle Swarm Optimization (PSO) to algorytm optymalizacyjny inspirowany naturą – zachowaniem stad ptaków lub ławic ryb. W przeciwieństwie do algorytmów, które śledzą pojedyncze rozwiązanie, PSO zarządza populacją (rojem) rozwiązań - kandydatów, nazywanych **cząstkami** (ang. *particles*).
Każda cząstka przemieszcza się po przestrzeni poszukiwań. Jej ruch jest wypadkową trzech czynników:

1. **Bezwładności:** Cząstka chce utrzymać swój dotychczasowy kierunek ruchu.
2. **Pamięci własnej (pBest):** Cząstka pamięta najlepsze miejsce, w którym kiedykolwiek była.
3. **Wiedzy roju (gBest):** Cząstka zna najlepsze miejsce znalezione przez *którąkolwiek* cząstkę z całego roju i przemieszcza się w jego stronę.

Matematyka ruchu

W każdej iteracji aktualizujemy prędkość (v) i pozycję (x) każdej cząstki według wzorów:

$$v_i(t+1) = w * v_i(t) + c_1 r_1 * \left(pBest_i - x_i(t)\right) + c_2 r_2 * \left(gBest_i - x_i(t)\right)$$
$$x_i(t+1) = x_i(t) + v(t+1)$$

Gdzie:

- w – waga bezwładności (np. 0.5 - jak bardzo cząstka jest "ciężka").
- c_1, c_2 – współczynniki uczenia (np. 1.0 - jak bardzo cząsta ufa sobie vs. grupie).
- r_1, r_2 – losowe liczby z zakresu [0, 1] (wprowadzają element stochastyczny).
- pBest_i – najlepsza znana dotychczas pozycja danej cząstki.
- gBest – najlepsza znana dotychczas pozycja całego roju.
- i – numer i-tej cząstki
- t – dana chwila czasowa = iteracja.

---

2. Cel i Opis Zadania

Celem projektu jest napisanie symulatora "Misji Ratunkowej" z wykorzystaniem algorytmu PSO.

Wyobraźmy sobie teren, na którym zaginął turysta posiadający nadajnik radiowy. Sygnał nadajnika jest najsilniejszy w miejscu przebywania turysty, a słabnie wraz z odległością i ukształtowaniem terenu.

Program ma symulować rój dronów (cząstek), które zostają zrzucone w losowych miejscach na mapie. Ich zadaniem jest jak najszybsze zlokalizowanie źródła sygnału (maksimum funkcji na mapie).

Zasada działania:

1. Program wczytuje mapę terenu z pliku (siatka wartości).
2. Inicjalizuje rój dronów w losowych koordynatach.
3. W pętli symuluje ruch dronów zgodnie z algorytmem PSO.
4. Po zakończeniu symulacji zwraca współrzędne znalezionego celu.
5. Program zapisuje co n iteracji podanych jako argument wywołania pozycje cząstek do pliku csv
6. Opcjonalnie: stworzenie generatora mapy

---

3. Wymagania Techniczne

Projekt musi być zgodny z dobrymi praktykami programowania w języku C.

A. Modułowość

Kod musi być podzielony na pliki .c i .h. Wymagana struktura minimalna:

- main.c: Obsługa argumentów wywołania, inicjalizacja, główna pętla.
- map.c / map.h: Obsługa wczytywania mapy, zwalniania pamięci mapy, pobierania wartości z danego punktu.
- pso.c / pso.h: Logika algorytmu, struktury cząstki i roju.
- Logger.c/ logger.h: Zapis danych danych pośrednich (informacji o położeniu cząstek) do pliku
- utils.c / utils.h: (Opcjonalnie) Generatory liczb losowych, funkcje pomocnicze.
- Makefile: Skrypt do kompilacji projektu.

B. Struktury Danych

Należy zdefiniować odpowiednie struktury danych opisujące: cząstkę, tablicę z terenem poszukiwań, rój cząstek.

C. Zarządzanie Pamięcią

- Tablica cząstek (Particle*) musi być alokowana dynamicznie w zależności od liczby cząstek podanej przez użytkownika.
- Mapa terenu musi być alokowana dynamicznie.
- Program **nie może mieć wycieków pamięci** (w tym celu należy użyć valgrind do weryfikacji).

D. Funkcja Celu (Fitness Function)

Cząstki poruszają się w przestrzeni ciągłej (double), a mapa jest dyskretna (indeksy int). Funkcja pobierająca wartość z mapy musi:

1. Rzutować pozycję (x, y) na indeksy tablicy [row][col].
2. Obsługiwać wyjście poza zakres mapy (zwracać karę, np. bardzo małą wartość, aby drony zawracały na mapę).

---

4. Argumenty Wywołania Programu (CLI)

Program nie powinien pobierać danych interaktywnie (przez scanf). Konfiguracja odbywa się przez argumenty linii poleceń (argc, argv).
Sygnatura wywołania:

./pso <plik_mapy> -p <liczba_czastek> -i <liczba_iteracji> -c <plik_konfiguracyjny z parametrami_PSO> -n <co_ktora_iteracje_zapis_postepow>

Opis argumentów:

1. plik_mapy (string): Ścieżka do pliku tekstowego z mapą.
2. liczba_czastek (int): Rozmiar roju (np. 20-100). Jeśli nie podano – domyślnie 30.
3. liczba_iteracji (int): Czas trwania symulacji. Jeśli nie podano – domyślnie 100.
4. Plik_konfiguracyjny (string) Ścieżka do pliku konfiguracyjnego z wartościami parametrów: w, c1, r1, c2, r2. Jeśli nie podano parametrów – parametry domyślne algorytmu
5. Zapis_postepow (int) domyśnie 0 (brak zapisu)

Przykład:
Bash
./pso terrain.txt -p 50 -i 200 -c config_file -n 2

---

5. Format Pliku Mapy

Przyjmujemy prosty format tekstowy, który łatwo wygenerować lub napisać ręcznie:

- Pierwsza linia: dwie liczby całkowite W H (szerokość, wysokość).
- Kolejne H linii: w każdej linii W liczb (wartości sygnału), oddzielonych spacją.

Przykład test_map.txt (mapa 3x3 z maksimum w środku):

```
7 7
−1000.00 −1000.00 −1000.00 −1000.00 24.63 25.05 25.20
−1000.00 −1000.00 −1000.00 −1000.00 7.70 26.21 26.40
−1000.00 −1000.00 −1000.00 −1000.00 7.98 8.20 27.60
−1000.00 −1000.00 −1000.00 −1000.00 −1000.00 −1000.00 −1000.00
2.05 2.16 2.27 −1000.00 −1000.00 −1000.00 −1000.00
6.81 2.15 2.25 −1000.00 −1000.00 −1000.00 −1000.00
6.72 7.06 2.22 −1000.00 −1000.00 −1000.00 −1000.00
```

---

6. Kryteria Oceny

Za projekt można uzyskać 45 punktów

W tym: 10 punktów za sprawozdanie. Sprawozdanie powinno zawierać:

- opis zadania oraz zasad działania algorytmu PSO
- opis implementacji zadania (zastosowanych struktur, podziału na moduły, podjętych decyzji implementacyjnych
- opis podziału prac w zespole
- wykonane testy działania programu (dla różnych map wejściowych, dla różnych parametrów, jak zmiany poszczególnych parametrów wpływają na zachowanie algorytmu)

- Wnioski

Oddanie projektu:

Projekt oddajemy na przedostatnich oraz ostatnich zajęciach w semestrze (lub wcześniej). Na oddanie projektu przychodzą obydwie osoby z zespołu z wyprzedzeniem dodając na isod kod programu oraz sprawozdanie. Podczas obrony projektu opowiadacie Państwo o projekcie i pokazujecie jego działanie. Możecie zostać poproszeni o modyfikację kodu.