Wrocław, 16-10-2025 r.

Sofiya Ignatovich 276704
Ivan Hancharyk 264511
Grupa 3, Czw 10:45

Opis założeń projektu

1. Temat i cel projektu

Temat**:** Problemy szeregowania zadań na jednym procesorze $(1||\Sigma Ci\ /\ 1||C_{max}$ **).**

Cel projektu: Zaprojektowanie i implementacja programu porównującego trzy metody porządkowania zadań (bez preempcji) pod kątem minimalizacji sumy czasów zakończenia ΣCi oraz zbadanie wpływu uruchomień współbieżnych (wątki) na czas obliczeń i metryki wydajności [3].

2. Sformułowanie problemu

Dane wejściowe: zbiór **n** niezależnych zadań o znanych czasach przetwarzania $p_i > 0$. Dostępny jest jeden procesor; w danej chwili wykonywane jest co najwyżej jedno zadanie; brak preempcji [1].

- Wariant decyzyjny: Czy istnieje kolejność, dla której $\Sigma Ci \leq T$ (lub $C_{max} \leq T$)?
- Wariant optymalizacyjny: Wyznaczyć permutację minimalizującą ΣCi lub C_{max}.

  Miary: $\Sigma C_i = \Sigma^n_{i=1} C_i$; $C_{max} = max_i C_i$.

3. Analiza złożoności obliczeniowej problemu

- Obliczenie ΣCi dla danej permutacji: $O(n)$.
- Algorytm SPT: $O(nlogn)$ (sortowanie) [2].
- Pozostałe metody: koszt zależny od konstrukcji/iteracji.

4. Metoda i algorytmy rozwiązywania problemu

SPT (Shortest Processing Time first) [1]

- Opis: sortowanie zadań rosnąco po p_i; rozwiązanie optymalne dla $1||\Sigma Ci$.
- Złożoność: $O(nlogn)$ [2].
- Rola: linia bazowa (benchmark jakości i czasu).

Cheapest Insertion (wstawianie w najlepsze miejsce) [1]

- Opis: konstrukcja harmonogramu przez sukcesywne wstawianie kolejnych zadań w pozycję minimalizującą przyrost ΣCi.
- Złożoność: $O(n^2)$.
- Rola: alternatywna heurystyka konstrukcyjna do porównania z SPT.

Lokalne ulepszanie „2-swap / first-improvement" (multi-start) [1],[2]

- Opis: start z losowej permutacji; iteracyjnie rozpatrujemy zamiany par (i, j) w kolejności pseudo-losowej; wykonujemy pierwszą zamianę zmniejszającą ΣCi; kończymy po spełnieniu warunku stopu: brak poprawy w 1000·n próbach lub budżet czasu T = 2 s na restart.
- Złożoność: ocena pojedynczej zamiany $O(1)$ (inkrementalnie); łączny koszt zależy od budżetu/iteracji.
  Rola: pokazanie kompromisu jakość–czas względem SPT i konstruktora.

Współbieżność:

- Równoległy multi-start metody (3): **R = 100** niezależnych restartów rozdzielonych na **p ∈ {1,2,4,8}** wątków (std::thread).
- Synchronizacja: aktualizacja wspólnego „global best" w sekcji krytycznej (mutex/atomic).
- Dodatkowo batch eksperymentów (różne instancje) dzielony między wątki [3].

5. Metoda, technologie i narzędzia implementacji

Implementacja w **C++ 20/23** z wykorzystaniem **IDE CLion** oraz standardowych bibliotek:

- **<thread> / <future>** – uruchomienia współbieżne,
- **<chrono>** – pomiar czasu,
- **STL** – struktury danych i sortowanie,
- kompilacja: **CMake** oraz **g++,**
- Analiza wyników w arkuszu excel (wykresy/tabele).

6. Sposób testowania i oceny jakości rozwiązań

- Poprawność: przykłady obliczeniowe na małych instancjach (walidacja obliczeń funkcji celu, zgodność SPT jako rozwiązania referencyjnego).
- Jakość i wydajność: dla **n ∈ {50, 200, 1000}**, dwa rozkłady p_i (jednostajny 1..100 oraz bimodalny 80/20), liczba wątków **p ∈ {1,2,4,8} [3]**.

- Raportowane metryki: ΣCi, czas [ms], przyspieszenie $S(p) = T_1/T_p$, efektywność $E(p) = S(p)/p$.
- Prezentacja: tabele i wykresy (czas vs p, S(p) vs p; rozkład ΣCi dla metod); wnioski o kompromisie jakość–czas i efektach zrównoleglania [3].

7. Literatura

[1] Błażewicz J., *Problemy optymalizacji kombinatorycznej*, PWN.
[2] Cormen T. H., Leiserson C. E., Rivest R. L., Stein C., *Wprowadzenie do algorytmów* / *Introduction to Algorithms*.
[3] Karbowski A., Niewiadomska-Szynkiewicz E. (red.), *Programowanie równoległe i rozproszone*, OW PW.