Karta główna

# Wstęp

# Problem szeregowania zadań

W przypadku planowania wykonania pewnych czynności (zadań) w momencie korzystania z ograniczonej liczby zasobów (procesorów) można stwierdzić, że kolejność wykonanych zadań, ich uszeregowanie, pozwala na osiągnięcie lepszych lub gorszych czasów realizacji całego przedsięwzięcia. Przykładowymi sytuacjami w których takie szeregowanie ma istotne znaczenie są:

* określenie kolejności zadań, obliczeń w wykonywanym algorytmie programu komputerowego. W tym wypadku zadaniem jest krok algorytmu, a zasobem czas procesora
* stworzenie harmonogramu prac inwestycji budowlanej. Zasobami są ludzie, sprzęt budowlany, a szeregowaniu podlegają prace budowlane takie jak np. położenie powierzchni asfaltowej
* zdefiniowanie modelu produkcyjnego pewnych dóbr złożonych z wielu podzespołów. Zasobem może być tu praca ludzka, dostęp do urządzeń koniecznych do wykonania podzespołów, uszeregowaniu podlega tworzenie kolejnych podzespołów

W zależności od warunków istnieje wiele algorytmów, które pozwalają na znalezienie optymalnego rozdzielenia zadań na zasoby [1]. Pierwszego podziału można dokonać na dwa typy szeregowania: dynamiczne i statyczne. Z szeregowaniem dynamicznym najczęściej można się spotkać przy rozdzielaniu czasu procesora pomiędzy procesy, gdy nie są znane czasy zakończenia zadania przed jego uruchomieniem. Najczęściej stosowanymi algorytmami przy tym szeregowaniu są: FIFO. Z kolei szeregowanie statyczne pojawia się, gdy znany jest czas wykonania wszystkich zadań, znane są także zasoby i koszty przesyłania zadań. Poniższa praca skupia się na problemie statycznego szeregowania.

Następnego podziału można dokonać ze względu na liczbę zasobów potrzebnych do wykonania czynności. Ich liczba może być ograniczona lub nieskończona. Drugi przypadek jest rzadko spotykany w rzeczywistej sytuacji, jednak najistotniejsze w tym przypadku byłoby znalezienie najkrótszego czasu szeregowania biorąc pod uwagę wyłącznie zależności pomiędzy czynnościami oraz koszt transmisji danych pomiędzy zasobami. Praca rozważa problem ograniczonych zasobów, w którym wielokrotnie wiele zadań mogłoby być wykonanych równocześnie, jednak ze względu na brak wolnych zasobów nie jest to możliwe.

Stworzono algorytmy, które można wykorzystać w sytuacji, dla której czasy wykonania wszystkich procesów są stałe. Nie jest to często spotykane w rzeczywistych problemach, jednak wartym podkreślenia faktem jest możliwość rozwiązania takiego problemu przy dodatkowych warunkach jak problemu klasy P-trudnego. W poniższej pracy czas wykonywania poszczególnych zadań nie jest obłożony tym ograniczeniem, istotne jest, że nie może być ujemny lub zerowy.

Kosztem przesyłania danych pomiędzy zasobami określa się czas jaki jest konieczny na transmisję danych, produktów wykonanego zadania do zasobu, na którym zostanie wykonane kolejne zadanie. Koszt musi zostać uwzględniony wyłącznie jeśli zadania są zależne. Częstym założeniem, istotnym w tej pracy, jest zerowy koszt w przypadku, gdy zadania były wykonywane na tym samym procesorze, zasobie. Z praktycznego punktu widzenia jeśli informacja może zostać zachowana w danym zasobie bez kosztu jej składowania nie ma potrzeby wykonywania dodatkowych czynności.

Rodzaje algorytmów szeregowania możemy podzielić również ze względu na uwzględnianie kosztu przy szeregowaniu (np. w systemach z pamięcią …) lub jego ignorowanie (np. w systemach z pamięcią współdzieloną). Także istotne jest czy przesyłanie może odbywać się równolegle czy musi być ono kolejkowane.

Zadania, które podlegają szeregowania, można podzielić na przerywalne i nieprzerywalne. Pierwsze z wymienionych dotyczy przypadku, gdy wykonywaną czynność można przerwać w dowolnym momencie, następnie można ją wznowić na innym procesorze. Z teoretycznego punktu widzenia powinno to pozwolić osiągnąć lepsze czasy szeregowania, jednak należy uwzględnić dodatkowe koszty związane z przesłaniem podzielonego zadania, które mogą być niewspółmierne do zysku związanego ze wcześniejszym zakończeniem innych zadań. Koszty nie mogą być również obliczone na starcie szeregowania, a dopiero w momencie przenoszenia podzielonego zadania na inny zasób. Pamiętać należy także o tym, iż nie w każdych warunkach jest możliwe podzielenie zadania na części. Jest to także bardzo trudny przypadek do implementacji.

Udowodniono, że problem szeregowania jest problemem klasy NP-trudnych, z wyjątkiem trzech sytuacji:

* przy drzewiastej strukturze grafu zależności zadań z arbitralną wartością procesorów
* graf procesów z jednakowym czasem wykonania zadań na dwóch procesorach
* graf typu interval-order (*interval-ordered*) z jednakowym czasem wykonania zadań i stałą liczbą procesorów.

Oznacza to, że nie jest możliwe szybkie znalezienie najlepszego rozwiązania i głównie używa się algorytmów heurystycznych, które próbują znaleźć optimum we względnie krótkim czasie, nie gwarantując jednak jego odnalezienia.

## Struktura prezentacji grafu procesów

Problem podziału zadań i zależności można przedstawić w postaci grafu jak na rys.. Przedstawia on

* Opisać podział na procesory, że procesory u mnie są takie same –czas wykonywania zadań jest taki sam na wszystkich, napisać o kosztach komunikacji – jak je się liczy, co to jest
* Różne sposoby wyboru najlepszej ścieżki – u mnie przy obliczaniu czasu szeregowania There are various ways to determine the priorities of nodes, such as HLF (Highest Level First) [Coffman 1976]; LP (Longest Path) [Coffman 1976]; LPT (Longest Processing Time) [Friesen 1987; Gonzalez, Jr. 1977]; and CP (Critical Path) [Graham et al. 1979]. Opisać wybrany szczegółowo
* Cele algorytmu/program – znalezienie najlepszego czasu wykonania, czasy wykoania poszczególnych zadań znane na początku, koszty zmiany procesowa i koszt wykonania na tym samym procesorze. Szczegółowo i wspomnieć, że przesyłanie można zacząć po zakończniu zadania i tak liczymy, równoległe przesyłanie
* Opisać graf programu równoległego, acykliczny skeirowany, co to węzły co krawędzie
* opis grafu procesorów

# Bibliografia

1. **Kwok Y. i Ahmad I.** Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors. *ACM Computing Surveys.* Grudzień 1999, Tom 31, 4, strony 407-471.