Karta główna

# Wstęp

Jednym z najważniejszych aspektów w dobie wszechobecnego kryzysu jest optymalizacja procesów produkcyjnych. Odpowiedni harmonogram prac to często być albo nie być dla wielu przedsiębiorstw. Tworząc taki harmonogram należy zwrócić uwagę na możliwość optymalnego wykorzystanie zasobów, tak aby specjalistyczne maszyny i urządzenia nie stały bezczynnie, a wykorzystywały swój potencjał w jak największym czasie. Tego typu problemy pozwalają rozwiązać algorytmy szeregowania zadań.

# 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-zupełnego. 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 NP-zupełnym, 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.

## Graf programu równoległego

W przypadku szeregowania statycznego z uwzględnieniem kosztów przesłania danych najefektywniejszym i najczytelniejszym sposobem prezentacji zależności pomiędzy zadaniami jest użycie acyklicznego grafu skierowanego. Przykład takiego grafu jest widoczny na Rys. 2.1. Węzeł grafu symbolizuje proces, zaś krawędź zależność pomiędzy zadaniami. W węźle widoczne są dwie wartości: numer procesu oraz czas wykonania (wyróżniany w komórce z szarym tłem). Zadanie numer 0 wykona się w czasie 3 jednostek, z kolei 2-gie zadanie w czasie 4 jednostek. Krawędzie są skierowane i krawędź prowadząca od węzła nr 0 do węzła nr 2 oznacza, że możliwe jest wykonanie zadania nr 2 dopiero, gdy zakończy się zadanie 0. Wartość przy krawędzi oznacza koszt przesłania danych pomiędzy modułami. Należy zwrócić uwagę, że koszt przesłania liczymy według wzoru:

|  |  |  |
| --- | --- | --- |
|  |  |  |

gdzie: tkl – czas komunikacji między modułami k i l

dkl – minimalna odległość pomiędzy zasobami wykorzystywanymi przez moduły k i l, odległość definiuje się według grafu procesów opisanego w rozdziale X

akl – waga krawędzi pomiędzy węzłami k i l

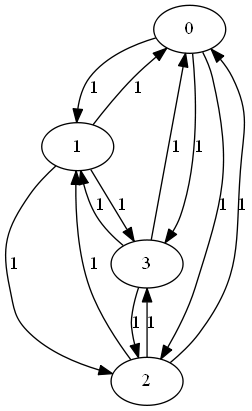
Jeśli procesy są alokowane na tym samym procesorze, odległość dkl wynosi 0 i koszt także jest równy 0. Z kolei, gdy procesory nie są połączone bezpośrednio należy zsumować odległość pomiędzy nimi uwzględniając koszt komunikacji (może być różny pomiędzy procesorami). W pracy założono, że czas wykonywania poszczególnych modułów na wszystkich procesorach jest identyczny.



Rys. . Przykładowy graf procesów rozpatrywany w ramach problemu szeregowania

## Graf procesorów

Zasoby można podobnie jak zadania przedstawić w formie grafu. Rys. 2.2 pokazuje przykładowy graf cykliczny dla 4 procesorów. Wszystkie procesory są ze sobą połączone co oznacza, że można wymieniać dane pomiędzy każdym z nich. Koszt w każdym wypadku wynosi 1 (wartość przy krawędzie).



Rys. . Graf przedstawiający powiązanie procesorów wraz z kosztem zmiany procesora

Liczba procesorów jest nieograniczona, przy czym minimalną logiczną wartością są 2 procesory. Także koszty komunikacji mogą być różne i zależne od przypadku. Na potrzebny poniższej pracy skorzystano z 2, 4 i 8 procesorów z założeniem, że komunikacji może odbywać się na zasadzie każdy z każdym, a koszt transmisji danych wynosi 1.

# Listowe algorytmy szeregowania

Algorytmy listowe są najprostszymi sposobami uszeregowania zadań. Ich ideę można zapisać w postaci prostego algorytmu:

1. pobierz listę możliwych do uruchomienia zadań
2. określ priorytety zadań i wybierz moduł z najwyższym priorytetem
3. wybierz wolny zasób i przydziel dla niego zadanie
   1. jeśli brak zasobów przejdź do kroku 2 ignorując zadania dla których brakuje zasobów
   2. jeśli zasób dostępny przejdź do kroku 1

## Highest Level First

## Longest Path

## Longest Processing Time

## Algorytm dynamicznej ścieżki krytycznej (Dynamic Critical Path)

# Algorytm uogólnionej ekstremalnej optymalizacji

W celu rozwiązania problemów NP-zupełnych często stosuje się algorytmy heurystyczne i metaheurystyczne. Jednym z algorytmów metaheurystycznych jest algorytm uogólnione ekstremalnej optymalizacji (*generalized extremal optimization*), który wykorzystano w poniższej pracy. Ma on swoje zastosowania w problemach optymalizacji, pozwalając na dobranie najlepszych parametrów optymalizowanej funkcji.

* Różne sposoby wyboru najlepszej ścieżki – u mnie przy obliczaniu czasu szeregowania – polityka 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

# 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.