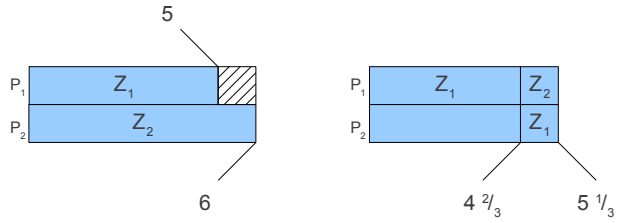
Algorytm Horvatha Lama i Sethiego

minimalizacja długości uszeregowania dla procesorów jednorodnych zadania zależne, podzielne

Dla tego problemu rozwiązanie zostało podane w postaci algorytmu o złożoności $O(n^2)$, wykorzystującego kolejno dwa algorytmy stosujące tzw. podział procesora.

W celu ich opisania, zakłada się, że standardowe czasy wykonywania zadań i współczynniki procesorów są uporządkowane, tzn $\tau_1 \geq \tau_2 \geq ... \geq \tau_n$ i $b_1 \geq b_2 \geq ... \geq b_m$. Oznacza się przez $B_k = \sum_{i=1}^{k} b_i$ łączna moc obliczeniową zbioru k procesorów, a przez $X_k = \sum_{j=1}^{k} \tau_j$ łączne zapotrzebowanie na obsługę zbioru *k* zadań. Ponieważ każde zadane musi być w danej chwili wykonywane przez co najwyżej jeden procesor, dolna granica długości uszeregowania wynosi $\frac{\tau_i}{b_1}$, jeśli drugi procesor jest dużo wolniejszy od pierwszego, może się zdarzyć, że $\frac{\tau_2}{b_2} > \frac{\tau_1}{b_1}$. W takim przypadku należy zwiększyć prędkość wykonywania zadania drugiego, zmniejszając prędkość wykonywania zadania pierwszego. Obrazuje to poniższy rysunek:



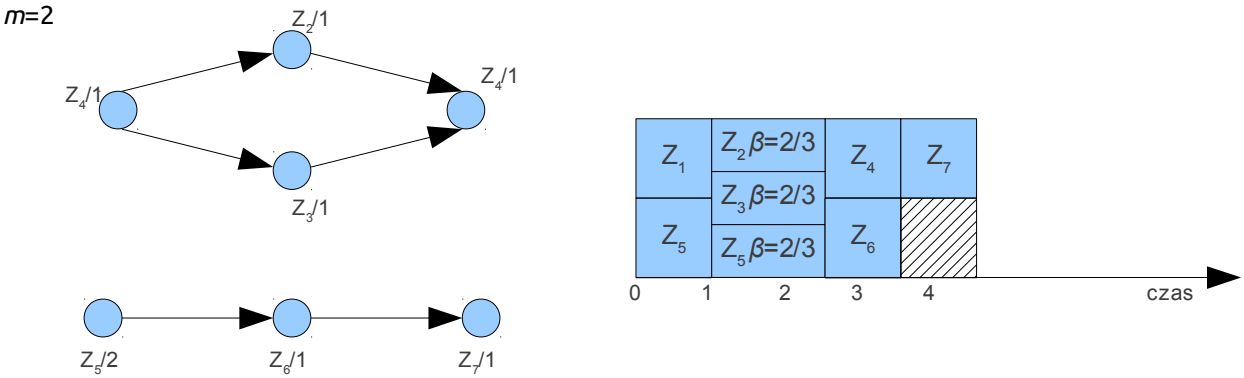
Ponieważ łączna moc obliczeniowa w tym przypadku wynosi $B_2 = b_1 + b_2$, a łączne zapotrzebowanie na obsługę wynosi $X_2 = \tau_1 + \tau_2$, więc czas wykonywania zbioru zadań $C^*_{max} = \frac{X_2}{B_2}$. Kontynuując te rozważania można zauważyć, że minimalny czas wykonywania zbioru C^*_{max} spełnia relację: $C^*_{max} = \{ \max_{1 \leq k \leq m} \{ \frac{X_k}{B_k} \}, \frac{X_n}{B_m} \}$.

Algorytm 1. pozwala znaleźć taki przydział procesorów do zadań, dla którego czas wykonywania zbioru zadań jest dany powyższym wzorem, i który wykorzystuje pojęcie podziału procesora. Terminem tym określa się możliwość przydzielania zadaniom części β, $0 < \beta \leq \max_i \{ b_i \}$ mocy obliczeniowej procesorów. Oczywiście w takim przypadku czas t_i wykonania zadania Z_i wyniesie: $t_i = \frac{\tau_i}{\beta}$

Algorytm 1. Konstrukcja przydziału procesorów do zadań, wykorzystującego podział procesora i minimalizującego czas wykonania zbioru zadań

1. Niech j będzie liczbą wolnych procesorów a *k* liczbą zadań o najwyższym poziomie, przy czym poziom zadania jest określony przez wartość standardowego czasu wykonywania. Jeśli $k \leq j$, to przydziel k zadań w celu wykonania ich z jednakową prędkością (z równym współczynnikiem β) *k* najszybszym procesorom. W przeciwnym przypadku przydziel każdemu zadaniu $\beta = \sum_{i=1}^{j} \frac{b_i}{k}$ mocy obliczeniowej procesorów. Jeśli są jeszcze wolne procesory rozważ zadania o niższym poziomie
2. Przerwij wykonywanie zadań i powtórz krok 1, jeśli:
   a) którekolwiek z zadań zostało zakończone, albo
   b) została osiągnięta chwila w której zadanie o niższym poziomie byłoby przy obecnym przydziale, wykonywane szybciej (tzn. przy większym współczynniku β) niż zadanie o poziomie wyższym
3. Jeśli wszystkie zadania zostały uszeregowane zakończ procedurę

Przykład szeregowania wykorzystującego podział procesora



Aby z otrzymanego przydziału procesorów zadań uzyskać uszeregowanie optymalne, należy posłużyć się algorytmem drugim, dla każdej z części tego przydziału zawartej między dwoma kolejnymi zdarzeniami określonymi w 2 kroku algorytmu pierwszego. Oznacza się czas trwania tej części przez *y*.

Algorytm 2. Konstrukcja uszeregowania optymalnego na podstawie znajomości przydziału procesorów do zadań, otrzymanego w wyniku zastosowania algorytmu pierwszego.

1. Jeśli *k* zadań przydzielono do *k* procesorów, to każde z nich wykonuj na każdym procesorze przez czas $\frac{y}{k}$
2. W przeciwnym przypadku, tzn. gdy liczba zadań była większa od liczby procesorów (*k*>*j*), oblicz rzeczywisty (w odniesieniu do czasu standardowego) czas wykonywania każdego z k zadań $\tau = y\beta$. Jeśli dla każdego zadania zachodzi $\frac{\tau}{b} < y$, gdzie b jest współczynnikiem prędkości najwolniejszego procesora, to przydziel zadania zgodnie z algorytmem McNaughtona, mimo różnych współczynników prędkości procesorów. Jeśli $\frac{\tau}{b} \geq y$ dla któregokolwiek zadania, to podziel *y* na *k* równych odcinków. Przydziel *k* zadań w ten sposób by każde zadanie było wykonywane przez *j* odcinków, w każdym na innym procesorze.

Złożoność obliczeniowa powyższego podejścia $O(n^2)$ wynika z faktu, ze w pierwszym z algorytmów należy w najgorszym przypadku rozpatrzeć w czasie każdej przerwy $O(n)$ zadań, a liczba tych przerw w najgorszym przypadku może być również $O(n)$.

Przykład zastosowania szeregowania przy użyciu obu algorytmów

n=4, m=2 $\tau_1=35, \tau_2=26, \tau_3=14, \tau_4=10, b_1=3, b_2=1$