Problem „Flow Shop” - przykładowe algorytmy -

Marek Sobolewski

Wprowadzenie

- Problemy szeregowania zadań:
  - Job Shop – maszyny i kolejność dowolne (określone)
  - Open Shop – kolejność dowolna (określona)
  - Flow Shop – ta sama kolejność dla wszystkich zadań („taśma produkcyjna”)
- Kryterium optymalizacyjne:
  - Tardiness – sumaryczne / ilościowe opóźnienie
  - Flowtime – śreni czas przebywania zadania w systemie
  - Makespan – czas wykonania wszystkich zadań

Złożoność obliczeniowa

- Dla więcej, niż dwóch maszyn problem Flow Shop jest NP-zupełny. Oznacza to, że nie da się znaleźć jego rozwiązania w czasie wielomianowym (tzn. w czasie proporcjonalnym do n^{α}, gdzie n to liczba danych wejściowych).

- Złożoność obliczeniowa algorytmu określa ilość zasobów lub czasu potrzebną do przeprowadzenia algorytmu w zależności od rozmiaru wejścia (wielkości n), np. $O(n)$ – zasoby/czas potrzebne do wykonania algorytmu są wprost proporcjonalne do liczby danych wejściowych.

- Najprostszy algorytm zwracający zawsze rozwiązanie optymalne to tzw. Przegląd Zupełny.

Przegląd zupełny

- Złożoność obliczeniowa O(n!)
- Załóżmy, że możemy oceniać 1 000 000 000 rozwiązań/sekundę

Liczba danych wejściowych (n)	Liczba możliwych rozwiązań	Czas potrzebny na znalezienie rozwiązania optymalnego
5	5!	0,00000012 sekundy
10	10!	0,0036288 sekundy
20	20!	~77 lat
25	25!	~500 milionów lat
50	50!	~10^{47} lat

Problem Flow Shop – oznaczenia

- Liczba zadań: N
- Liczba maszyn: M
- Dane rozwiązanie (permutacja *N*-elementowa): π
- Czas obróbki i-tego zadania na j-tej maszynie: $p_{\pi_i,j}$
- Czas zakończenia i-tego zadania na j-tej maszynie: $C(\pi_i, j)$
- Kryterium optymalizacyjne - makespan (C_{max}): $C(\pi_N, M)$
- Możliwych rozwiązań: N!

Rekurencyjna metoda obliczania C_{max}

$C(\pi_N, M)$

$$C(\pi_1, 1) = p_{\pi_1,1}$$

$$C(\pi_i, 1) = C(\pi_{i-1}, 1) + p_{\pi_i,1} \qquad i = 2, \ldots, N$$

$$C(\pi_1, j) = C(\pi_1, j-1) + p_{\pi_1,j} \qquad j = 2, \ldots, M$$

$$C(\pi_i, j) = \max\{C(\pi_i, j-1), C(\pi_{i-1}, j)\} + p_{\pi_i,j} \qquad i = 2, \ldots, N; \quad j = 2, \ldots, M$$

Algorytm CDS

- Herbert **C**ampbell, Richard **D**udek, Milton **S**mith (1970 rok)

- Uogólnienie algorytmu Johnson'a na przypadki, gdzie liczba maszyn $m > 2$

- Algorytm Johnson'a:
  - problem $F^*2||C_{max}$
  - znajduje rozwiązanie <u>optymalne</u> w czasie wielomianowym

Algorytm CDS

1. Tworzymy *p* = M-1 pomocniczych problemów z dwiema maszynami i *N* zadaniami w następujący sposób - dla *k* = 1, … , *p*:
   a) traktujemy *k* pierwszych (w cyklu technologicznym) maszyn jako jedną maszynę M_1, a (*p – k)* kolejnych maszyn (pozostałe) jako drugą maszynę M_2
   b) obliczamy dla każdego zadania czas obróbki na maszynie M_1 oraz M_2
2. Rozwiązujemy każdy z *p* problemów przy użyciu algorytmu Johnsona. Otrzymujemy *p* rozwiązań – permutacji (niekoniecznie różnych!)
3. Dla każdej z *p* permutacji obliczamy wartość parametru C_{max} przy użyciu oryginalnych danych wejściowych ($p_{i,j}$)
4. Sekwencja o najmniejszej wartości C_{max} wybierana jest jako najlepsza

Złożoność obliczeniowa: $O(N * M * \log N + N * M^2)$

Algorytm NEH

- Muhammad **N**awaz, Emory **E**nscore, Inyong **H**am (1983 rok)

- Bazuje na założeniu, że zadania o większym sumarycznym czasie wykonywania powinny mieć nadany wyższy priorytet

- Najczęściej cytowany / porównywany algorytm w naukowych opracowaniach problemy Flow Shop

Algorytm NEH

1. Sortujemy zadania zgodnie z malejącym (nierosnącym) sumarycznym czasem obróbki na wszystkich maszynach
2. Ustawiamy dwa pierwsze zadania w kolejności umożliwiającej uzyskanie krótszego czasu zakończenia C_{max} (dwie możliwości)
3. Dla *k* = 3, … , *N* wykonujemy krok 4
4. Wstawiamy *k*-te zadanie do sekwencji w miejsce, gwarantujące najmniejszy przyrost czasu C_{max} (*k* możliwości)
5. Uzyskana sekwencja traktowana jest jako wynik działania algorytmu

Złożoność obliczeniowa: $O(M * N^2)$

NEH - przykład

- Problem dla pięciu maszyn i czterech zadań:

Czasy obróbki ($p_{i,j}$)		Maszyny				
		M1	M2	M3	M4	M5
Zadania	J1	5	9	8	10	1
	J2	9	3	10	1	8
	J3	9	4	5	8	6
	J4	4	8	8	7	2

NEH – przykład (krok 1)

- Obliczamy sumaryczny czas wykonywania każdego zadania:

  $t_1 = 5 + 9 + 8 + 10 + 1 = 33$

  $t_2 = 9 + 3 + 10 + 1 + 8 = 31$

  $t_3 = 9 + 4 + 5 + 8 + 6 = 32$

  $t_4 = 4 + 8 + 8 + 7 + 2 = 29$

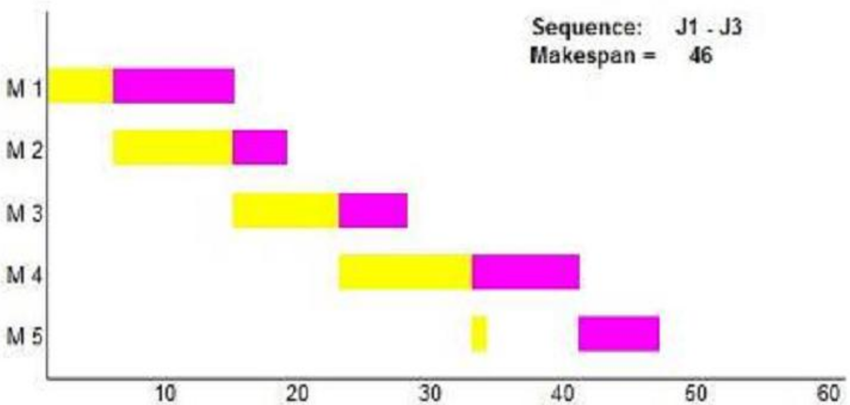
- Sortujemy zadania zgodnie z nierosnącą kolejnością obliczonych czasów:

  J1, J3, J2, J4

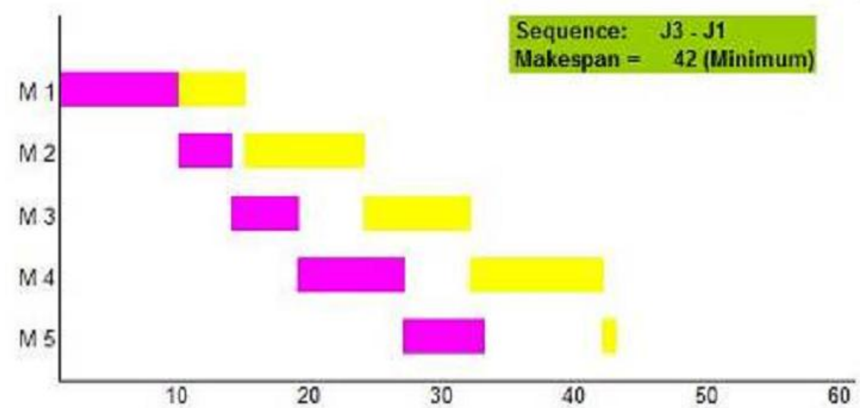
NEH – przykład (krok 2)

- Wybieramy najlepszą kolejność dla pierwszych dwóch zadań (czyli J1 i J3)

Możliwość I



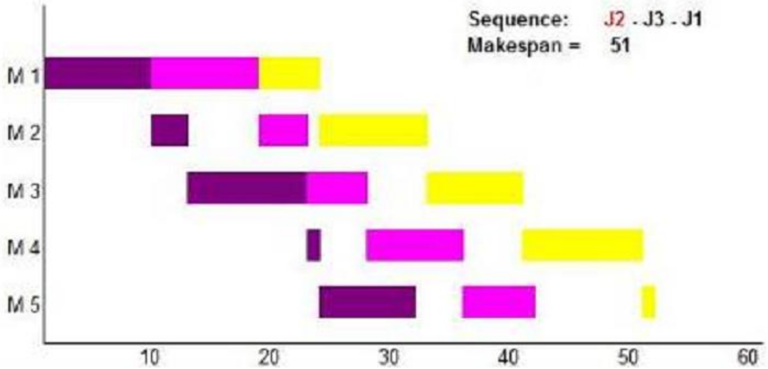
Możliwość II

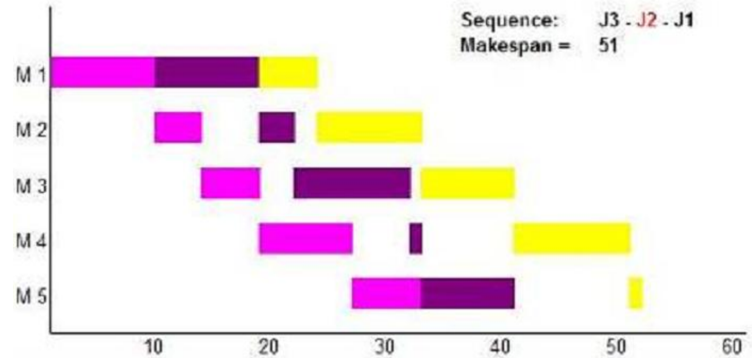


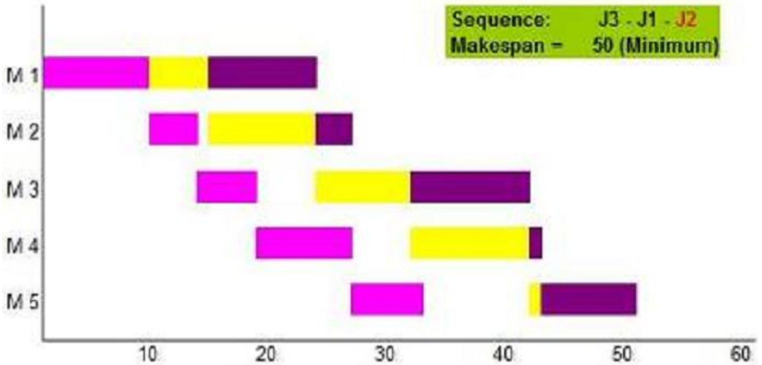
- W kolejnych krokach zadanie J3 będzie zawsze przed zadaniem J1

NEH – przykład (krok 3)

- Znajdujemy najlepsze miejsce dla zadania trzeciego (czyli J2)



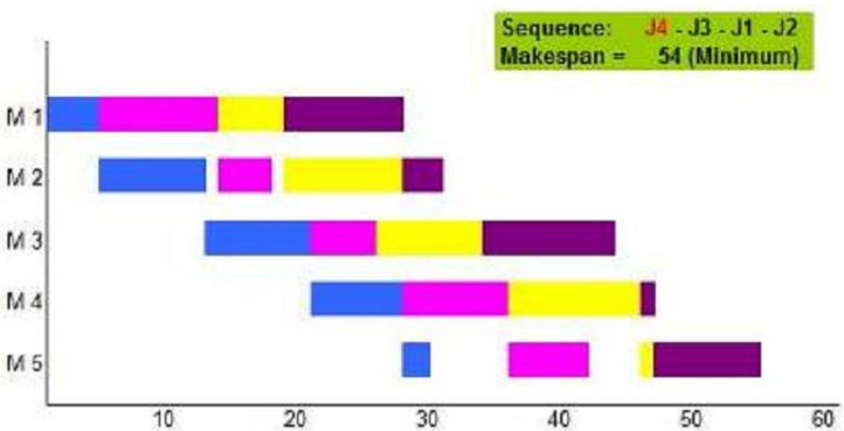


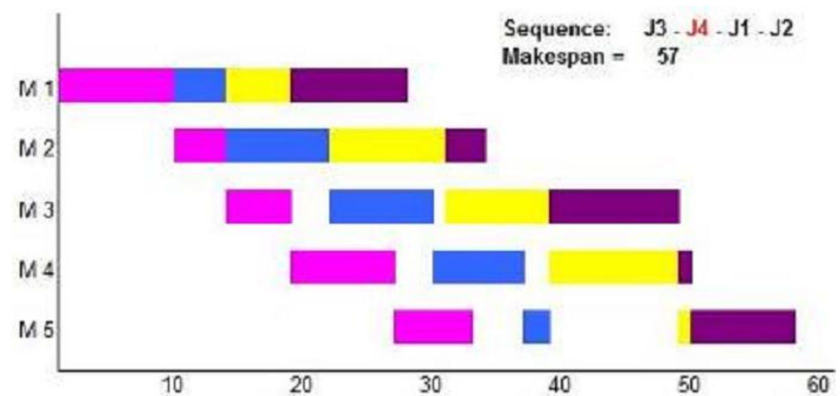


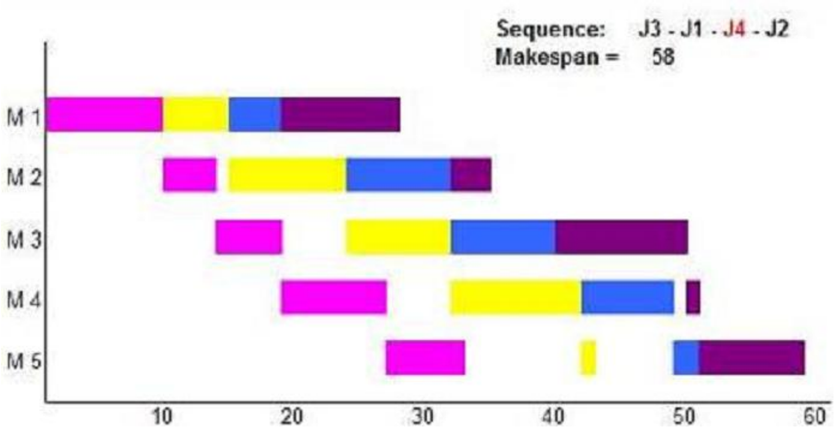
Najlepsze ustawienie:
J3, J1, J2 (makespan = 50)

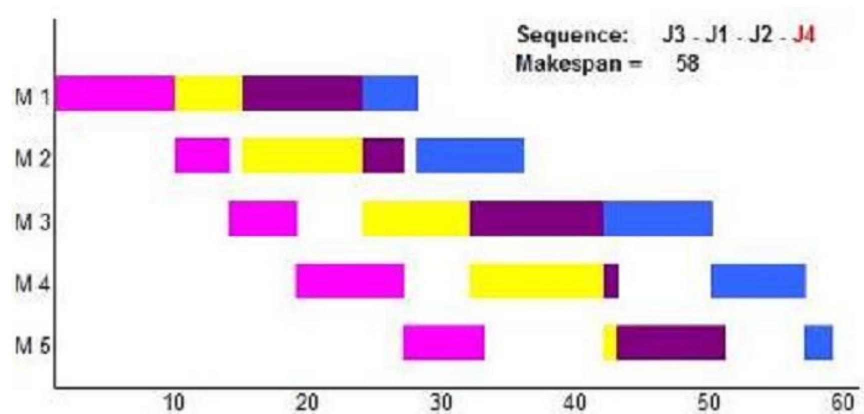
NEH – przykład (krok 4)

- Znajdujemy najlepsze miejsce dla zadania J4





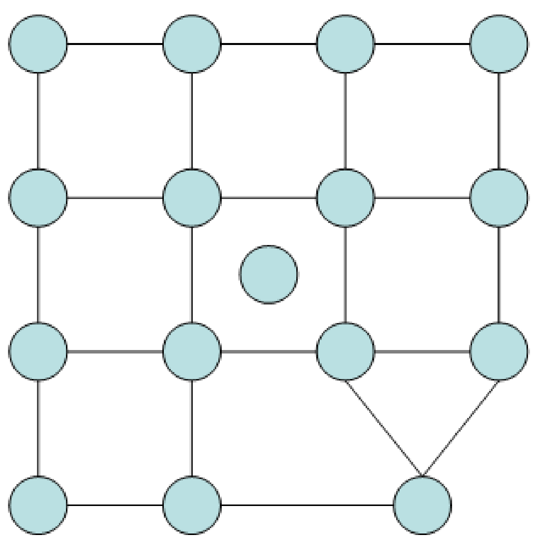




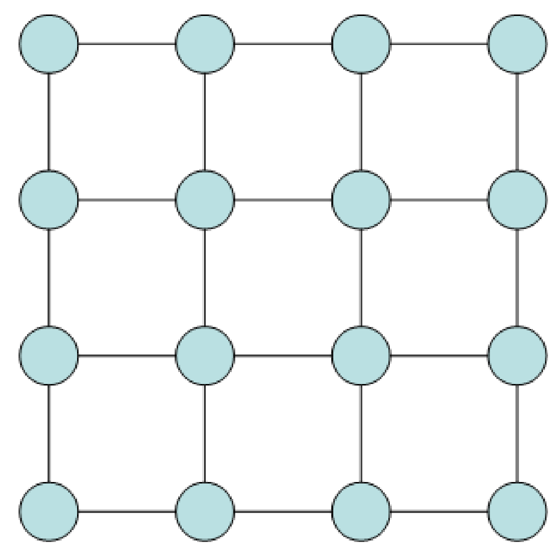
Symulowane wyżarzanie (Simulated Annealing)

- Analogia – powolne ochładzanie ciała stałego

Kryształ z defektami
(podwyższona energia stanu)



Kryształ idealny
(minimalna energia stanu)



Symulowane wyżarzanie a Flow Shop

Cecha	**Natura**	**Flow Shop**
Rozwiązanie	Struktura krystaliczna	Sekwencja zadań
Funkcja celu	Energia stanu	Makespan
Sąsiedztwo	Perturbacje struktury pod wpływem temperatury	Niewielka modyfikacja permutacji (swap, insert)
Studzenie	Niemonotoniczna funkcja energii stanu	Możliwość akceptowania rozwiązań gorszych niż aktualne

Symulowane wyżarzanie - algorytm

- Określone:
  t_{max} – temp. początkowa, t_{min} – temp. końcowa, α - współczynnik chłodzenia
- Funkcja akceptacji Boltzmanna: $B(\beta,\pi,t) = e^{-\frac{C(\beta)-C(\pi)}{t}}$

$\pi^* := \pi$ ← bieżące rozwiązanie traktuj jako najlepsze (jako lidera)
$t := t_{max}$ ← temperatura początkowa
while $t > t_{min}$ ← powtarzaj dopóki cieplej niż t_{min}
    $\beta := \mathit{sasiad}(\pi)$ ← wylosuj permutację z otoczenia bieżącej (i oznacz jako β)
    if $C(\beta) < C(\pi^*)$ **then**
        $\pi^* := \beta$
    } jeśli makespan β rozwiązania jest mniejszy niż makespan lidera, to β jest nowym liderem
    if ($B(\beta, \pi, t) > \mathit{random}(0\text{-}1)$) **then**
        $\pi := \beta$;
    } przyjmij β jako rozwiązanie bieżące z prawdopodobieństwem $e^{-\frac{C(\beta)-C(\pi)}{t}}$
    $t := t * \alpha$ ← studzenie
return π^* ← jako wynik zwróć permutację z najmniejszym makespan (sposród zbadanych)

Podsumowanie

	CDS	**NEH**	**Simulated Annealing**
Czas wykonywania	niewielki	niewielki	duży (zależny od schematu chlodzenia)
Jakość rozwiązania	Niska (błąd 10-20%)	Wysoka (błąd 5-10%)	Wysoka (zależna od czasu wykonywania)
Kryterium	Makespan	Makespan, Tardiness, Flowtime	Dowolne (algorytm uniwersalny)
Zalecenia	Dobry dla małej liczby maszyn i wielu zadań	Dobry dla niewielkich instancji problemu	Dobry, gdy nie jest istotna szybkość znajdowania rozwiązania

Bibliografia

- Nawaz M., Enscore E., Ham I., *A heuristic algorithm for the m-machine n-job flow-shop sequencing problem*, Omega. The International Journal of Management Science. v11. 91-95, 1983
- Stawowy A., Mazur Z., *Heurystyczne algorytmy szeregowania zadań produkcyjnych i grupowania wyrobów, Nowoczesne metody zarządzania produkcją*, pod red. Z.Martyniaka, Wydział Zarządzania AGH, Kraków, 1996
- Campbell H., Dudek R., Smith M., *A Heuristic Algorithm For The N Job, M Machine Sequencing Problem*, 1970
- Jaszkiewicz A., *Wprowadzenie do metaheurystyk (slajdy do wykładu)*, Instytut Informatyki, Politechnika Poznańska
- http://www.free.of.pl/s/szeregowaniezadan/teoria.html
- Symulator: http://www.organizaja.yoyo.pl/fb/index.html

Dziękuję za uwagę!