Wrocław, 8 marca 2021

Sterowanie **P**rocesami **D**yskretnymi

Laboratorium 2

Opracowanie i implementacja algorytmów dla problemu $1|r_i, q_i|C_{\max}$

prowadzący: *mgr inż. Radosław Idzikowski*

1 Cel laboratorium

Celem laboratorium jest zapoznanie się z podstawami teorii szeregowania zadań oraz sposobami modelowania oraz rozwiązywania podestowych problemów na przykładzie jednomaszynowego problemu z czasami dostępności oraz opuszczenia. Wspomniany problem w praktyce jest często nazywany problemem RPQ . Obejmuje to odpowiednie zdefiniowane problemu (ograniczeń i funkcji celu), danych wejściowych oraz implementacje dedykowanych algorytmów, a także interpretację wyników.

2 Przebieg zajęć

Laboratorium obejmuje zajęcia nr 2-4 (6 godzin zajęć w tym oddanie). Praca odbywa się w ramach dwuosobowych zespołów. Każdy zespół otrzymuje do opracowania ten problem. W tym wypadku $1|r_j, q_j|C_{\max}$ oraz dedykowane dla niego algorytmy.

W trakcie zajęć nr 2 w wybranym języku `python`, `C/C++`, `Java` lub `C#` należy zaimplementować metodę generowania instancji przy użyciu załączonych generatorów dla zadanych parametrów (źródło, rozmiar, zakres). Kolejnym krokiem jest implementacja metody oceniania rozwiązania (liczenia funkcji celu). Następnie należy zacząć implementacje algorytmu Schrage w wersji z lub bez wykorzystania kolejki.

W trakcie zajęć nr 3 należy dokończyć rozpoczęte zadanie z zajęć nr 2. Następnie należy zaimplementować wersję algorytmy Schrage dla problemu $1|r_i, q_i, \texttt{pmtn}|C_{\max}$. Ostatnim etapem jest implementacja algorytmu Carliera dla problemu $1|r_i, q_i|C_{\max}$.

Podczas ostatnich zajęć z pierwszego tematu (zajęcia nr 4) będzie czas na dokończenie rozpoczętych zadań oraz na ocenę efektu końcowego. Na ocenę **3.0** należy poprawnie zaimplementować algorytm Schrage dla $1|r_i, q_i|C_{\max}$ z lub bez wykorzystania kolejki. Napisanie drugiej wersji algorytmu Schrage podnosi ocenę o 0.25. Napisanie wersji algorytmu Schrage dla problemu $1|r_i, q_i, \texttt{pmtn}|C_{\max}$ podnosi ocenę o 0.75. Napisanie algorytmu Carliera dla $1|r_i, q_i|C_{\max}$ podnosi oceną o 1.0. Oznacza to, że:

- Na ocenę 3.0 należy poprawnie zaimplementować algorytm Schrage dla problemu $1|r_i, q_i|C_{\max}$.
- Na ocenę 4.0 należy poprawnie zaimplementować algorytm Schrage w dwóch wersjach dla problemu $1|r_i, q_i|C_{\max}$ oraz jednej dla $1|r_i, q_i, \texttt{pmtn}|C_{\max}$.
- Na ocenę 5.0 należy poprawnie zaimplementować algorytm Schrage w dwóch wersjach oraz algorytm Carliera dla problemu $1|r_i, q_i|C_{\max}$ i algorytm Schrage dla problemu $1|r_i, q_i, \texttt{pmtn}|C_{\max}$.

3 Problem

W problemie $1|r_j, q_j|C_{\max}$ mamy zbiór $\mathcal{J} = \{1, 2, \ldots, n\}$ n zadań. Wszystkie zadania muszą zostać wykonanie/przetworzone na maszynie. Każde j-te zadanie składa się z trzech parametrów:

- r_j - czas przygotowania/termin dostępności,
- p_j - czas wykonania,
- q_j - czas dostarczenia/stygnięcia.

Każde zadanie musi zostać wykonywane nieprzerwanie przez p_j czasu. Jednak wcześniej musi zostać być przygotowane przez r_j czasu. Następnie musi jeszcze opuścić system przez q_j czasu. Na maszynie na raz może być wykonywane dokładnie jedno zadanie. Przez π będziemy oznaczać kolejność wykonywania zadań na maszynie. Momenty rozpoczęcia wykonywania zadań będziemy opisywać wektorem $S = (S_1, S_2, \ldots, S_n)$, gdzie czas rozpoczęcia zadania musi być większy niż czas jego przygotowania:

$$r_{\pi(j)} \leqslant S_{\pi(j)}. \tag{1}$$

Momenty zakończenia zadań opiszemy wektorem $C = (C_1, C_2, \ldots, C_n)$, gdzie czas zakończenia j zadania w kolejności π jest równy sumie momentu jego rozpoczęcia oraz czasu wykonywania:

$$C_{\pi(j)} = S_{\pi(j)} + p_{\pi(j)}. \tag{2}$$

Musimy pamiętać, że kolejne zadanie nie może zacząć wcześniej niż czas jego przygotowania (Wzór (1)) oraz czas zakończenia zadania poprzedniego ($S_{\pi(j)} \geqslant C_{\pi(j-1)}$), zatem :

$$S_{\pi(j)} = max\{r_{\pi(j)}, C_{\pi(j-1)}\} \tag{3}$$

Badanym kryterium optymalizacyjnym jest czas zakończenia wszystkich zadań oraz opuszczenia ich przez system:

$$C_{\max}(\pi) = \max_{j \in \mathcal{J}}\{C_{\pi(j)} + q_{\pi(j)}\} \tag{4}$$

Poniżej przedstawiono pseudokod metody liczenia funkcji celu:

Algorithm 1 Liczenie wartości funkcji celu

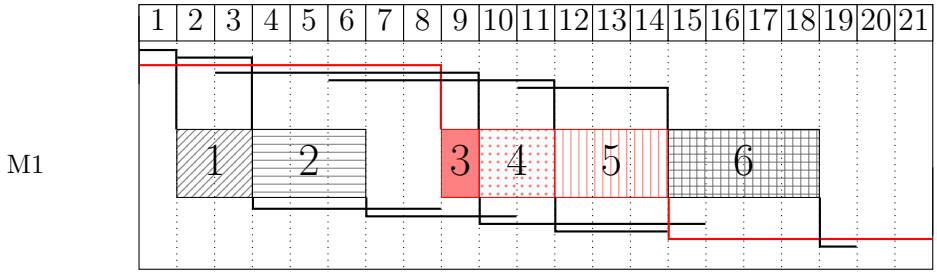
```
procedure CALCULATE(π)
    S_1 ← r_π(1)
    C_1 ← S_1 + p_π(1)
    C_max ← C_1 + q_π(1)
    for j = 2, 3, ..., n do
        S_j ← max{r_π(j), C_j−1}
        C_j ← S_j + p_π(j)
        C_max ← max{C_max, C_j + q_π(j)}
    end for
    return C_max
end procedure
```

3.1 Problem z przerwaniami

Problem $1|r_j, q_j, \texttt{pmtn}|C_{\max}$ jest zbliżony do omówionego problemu $1|r_j, q_j|C_{\max}$, ale proszę pamiętać, że jest to całkiem inny problem, ponieważ dopuszczamy możliwość przerwania zadania. Dopuszcza się wielokrotne przerywanie jednego zadania. Zadanie na maszynie możemy przerwać w momencie $t = r_j$, kiedy pojawi się nam dostępne zadanie o większym czasie q_j niż aktualne wykonywane. Zadanie przerwane uważa się za częściowo wykonane należy odłożyć do zbioru $\mathcal{G}$ z pomniejszym czasem wykonywania. Zmodyfikowany algorytm Schrage rozwiązuje problem $1|r_j, q_j, \text{pmtn}|C_{\max}$ optymalnie.

4 Przykład

Na Rysunku 1 mamy graficzne rozwiązanie problemu dla danych z Tabeli 1. Każde zadanie dla ułatwienia oznaczono innym kolorem. Czas wykonywania zadań na maszynie jest zaprezentowany w formie bloczku. Czas przygotowania jest oznaczony "*wąsem*". Na przedstawionym schemacie gantta dobrze są widoczne przestoje (*przerwy*) na maszynie.



Rysunek 1: Problem RPQ: $n = 3$ dla $\pi = (1, 2, 3)$

Tabela 1: Instancja $n = 6$

j	1	2	3	4	5	6
r_j	1	2	8	7	6	4
p_j	2	3	1	2	3	4
q_j	5	4	6	3	7	1

Rozwiązanie możemy również zaprezentować w formie harmonogramu (czasy S_j oraz C_j), co przedstawiono w Tabeli 2. Czas zakończenia oraz dostarczenia wszystkich zdań pogrubiono.

Tabela 2: Tworzenie harmonogramu dla $\pi = (1, 2, 3)$

j	1	2	3	4	5	6
S_j	1	3	8	9	11	14
C_j	3	6	9	11	14	18
$C_j + q_j$	8	10	15	14	**21**	19

5 Sposób generowania instancji

Dla parametru n oraz ziarna Z:

1. init(Z).
2. Dla każdego $j \in \mathcal{J}$: $p_j \leftarrow \text{nextInt}(1, 29)$.
3. $A \leftarrow \sum_{j=1}^{n} p_i$.
4. Dla każdego $j \in \mathcal{J}$:$r_j \leftarrow \text{nextInt}(1, A)$.
5. Dla każdego $j \in \mathcal{J}$:$q_j \leftarrow \text{nextInt}(1, X)$.

W ramach testu należy przyjąć najpierw $X = 29$, a potem $X = A$. W ramach oceny za implementacje algorytmów, należy sprawdzić czy zakres parametrów q_j ma wpływ na działanie algorytmów.

6 Metody rozwiązania

6.1 Algorytm Schrage

Każdy problem optymalizacji dyskretnej można rozwiązać metodą zachłanną. W każdym etapie algorytmu jest wybierane najlepsza w danym momencie decyzja. Z powodu konstrukcyjnego charakteru metody, przerwanie algorytmu podczas działa pozbawi nas szansy otrzymania kompletnego rozwiązania. Dla badanego problemu znanym algorytmem zachłannym jest algorytm Schrage. W tym celu potrzebujemy wprowadzić dodatkowe elementy:

- $\mathcal{N}$ – zbiór zadań nieuszeregowanych,
- $\mathcal{G}$ – zbiór zadań gotowych do realizacji,
- t – zmienna pomocnicza symbolizującą chwilę czasową.

Sposób działa algorytmu mamy przedstawiony za pomocą pseudokodu 2. W liniach $2-5$ mamy inicjalizacje zmiennych początkowych, gdzie k – numer zadania w permutacji π, które aktualnie chcemy rozpatrzyć, zbiór zadań $\mathcal{G}$ początkowo zostawiamy pusty, ponieważ nie mamy żadnych zadań gotowych do realizacji na maszynie, do zbioru $\mathcal{N}$ przypisujemy wszystkie zdania, początkowa wartość zmiennej czasu t jest równa najmniejszemu terminowi dostępności zadania r_j ze zbioru zadań nieuszeregowanych $\mathcal{N}$.

Algorithm 2 Schrage

```
procedure SCHRAGE(𝒥)
    k ← 1
    𝒢 ← ∅
    𝒩 ← 𝒥
    t ← min_{j∈𝒩} r_j
    while 𝒢 ≠ ∅ ∨ 𝒩 ≠ ∅ do
        while 𝒩 ≠ ∅ ∧ min_{j∈𝒩} r_j ⩽ t do
            j* ← arg min_{j∈𝒩} r_j
            𝒢 ← 𝒢 ∪ {j*}
            𝒩 ← 𝒩 \ {j*}
        end while
        if 𝒢 ≠ ∅ then
            j* ← arg max_{j∈𝒢} q_j
            𝒢 ← 𝒢 \ {j*}
            π(k) ← j*
            t ← t + p_{j*}
            k ← k + 1
        else
            t ← min_{j∈𝒩} r_j
        end if
    end while
    return π
end procedure
```

Główny schemat algorytmu jest zawarty w pętli `while` w liniach $6-21$, która wykonuję się dopóki mamy jeszcze zadania nieuszeregowane w zbiorze $\mathcal{G}$ lub $\mathcal{N}$. Pętla `while` w liniach $7-11$ odpowiada,

że przesunięcie przygotowanych zadań w chwili t ze zbioru $\mathcal{N}$ do zbioru $\mathcal{G}$. W instrukcji warunkowej `if` jeśli zbiór zadań gotowych $\mathcal{G}$ nie jest pusty, to należy przyporządkować zadanie o najdłuższym czasie czasie dostarczenia q_j do wykonania na maszynie, pamiętając o tym, że zadanie nie może się zacząć wykonywać wcześniej niż jego termin dostępności i czas zakończenia ostatniego zadania:

$$S_{\pi(j)} = max\{r_{\pi(j)}, C_{\pi(j-1)}\}, \tag{5}$$

jeśli nie mam zadań gotowych do realizacji ($\mathcal{G} = \emptyset$), to należy zmienić chwilę czasową t na termin dostępności najbliższego zadania ze zbioru zadań nieuszeregowanych (o najmniejszym r_j).

Analizując opis oraz pseudokod algorytmu można łatwo zauważyć, że jesteśmy często zmuszeni do przeszukiwania zbioru $\mathcal{N}$ i $\mathcal{G}$, dlatego jednym z celów laboratorium jest napisanie wersji programu z wyszukiwaniem minimum/maximum oraz drugiej z wykorzystaniem struktur samo-organizujących się (np.: kolejka priorytetowa).

6.2 Algorytm Carlier

Algorytm Carlier'a jest algorytmem dokładnym dla problemu $1|r_j, q_j|C_{\max}$ bazującym na metodzie podziału i ograniczeń (*Branch and Bound*). Dla poprawnego działania będą potrzebne dwie wersje algorytmu Schrage:

- dla problemu $1|r_j, q_j|C_{\max}$ (SCHRAGE),
- dla problemu $1|r_j, q_j, \texttt{pmtn}|C_{\max}$ (SCHRAGEPMTN).

Algorytm SCHRAGE w wersji podstawowej będzie wykorzystywany do ustalania kolejności, a algorytm SCHRAGEPMTN będzie użyty do wyznaczenia dolnego ograniczenia, ponieważ rozwiązanie optymalne problemu $1|r_j, q_j|C_{\max}$ nie może być lepsze niż rozwiązanie optymalne problemu $1|r_j, q_j, \text{pmtn}|C_{\max}$.

Algorytm będzie działał na zmodyfikowanych danych wejściowych dlatego do algorytmu należy przekazać zbiór zadań. Pierwszym krokiem algorytmu jest wyznaczenie kolejności algorytmem (SCHRAGE), za pierwszym razem będzie to nasze nowe górne oszacowanie (*Upper Bound*). Kolejnym krokiem (linie 7-9) jest wyznaczenie bloku krytycznego, który jest dobrze widoczny na Rys. 1, gdzie kolorem czerwonym jest zaznaczona cała ścieżka krytyczna. Zadanie a jest to pierwsze zadanie na ścieżce krytycznej, a zadanie b jest ostatnim. Następnie w bloku (a, b) należy wyszukać zadanie o jak najwyższej pozycji, ale z mniejszym $q_{\pi(j)} < q_{\pi(b)}$. Dla ułatwienia należy utworzyć zbiór zadań $\mathcal{K}$ z bloku zadań $(c+1, b)$. Następnie szukamy minimalnego $r_{\pi(j)}$ i $q_{\pi(j)}$ oraz sumy czasów wykonywania zadań w zbiorze $\mathcal{K}$. Teraz należy zmodyfikować zadanie c zwiększając jego termin dostępności $r_{\pi(c)}$, co wymusi jego późniejsza realizację. Teraz algorytm (SCHRAGEPMTN) wyznaczamy dolne ograniczenie (*Lower bound*), jeśli będzie obiecujące (mniejsze niż nasze górne oszacowanie) należy wywołać algorytm (CARLIER) jeszcze raz. Wycofując się z rekurencji odtwarzamy termin dostępności zadania c. Następnie analogicznie postępujemy z czasem stygnięcia w tym wypadku wymuszając jego szybszą realizację.

opracował: *Radosław Idzikowski*

Algorithm 3 Carlier

procedure CARLIER($\mathcal{J}$)
  $U \leftarrow$ SCHRAGE($\mathcal{J}$)
  if $U < UB$ **then**
    $UB \leftarrow U$
    $\pi^* \leftarrow \pi$
  end if
  $b \leftarrow \max\{j \in \mathcal{J} : C_{\max} = C_{\pi(j)} + q_{\pi(j)}\}$
  $a \leftarrow \min\{j \in \mathcal{J} : C_{\max} = r_{\pi(j)} + \sum_{k=j}^{b} p_{\pi(k)} + q_{\pi(b)}\}$
  $c \leftarrow \max\{j \in \mathcal{J} : a \leqslant j < b \wedge q_{\pi(j)} < q_{\pi(b)}\}$
  if $c = \emptyset$ **then**
    return π^*
  end if
  $\mathcal{K} \leftarrow \{c+1, c+2, \ldots, b\}$
  $\hat{r} \leftarrow \min_{j \in \mathcal{K}} r_{\pi(j)}$
  $\hat{q} \leftarrow \min_{j \in \mathcal{K}} q_{\pi(j)}$
  $\hat{p} \leftarrow \sum_{j \in \mathcal{K}} p_{\pi(j)}$
  $r_{\pi(c)} \leftarrow \max\{r_{\pi(c)}, \hat{r} + \hat{p}\}$
  $LB \leftarrow$ SCHRAGEPMTN($\mathcal{J}$)
  if $LB < UB$ **then**
    CARLIER($\mathcal{J}$)
  end if
  restore $r_{\pi(c)}$
  $q_{\pi(c)} \leftarrow \max\{q_{\pi(c)}, \hat{q} + \hat{p}\}$
  $LB \leftarrow$ SCHRAGEPMTN($\mathcal{J}$)
  if $LB < UB$ **then**
    CARLIER($\mathcal{J}$)
  end if
  restore $q_{\pi(c)}$
end procedure