Metody optymalizacji – laboratorium

zad. 0 Przeczytać opis pakietu `JuMP` (z języka `Julia`) w celu zapoznania się z możliwościami.

zad. 1 Załóżmy, że chcemy zebrać dane liczbowe na temat m różnych cech populacji. Ogromna ilość zebranej informacji pamiętana jest w chmurze w n różnych miejscach (serwerach). Niech T_j oznacza czas potrzebny na przeszukanie j-tego miejsca, $j = 1, \ldots, n$, przy czym zakładamy, że nie zależy on od liczby cech, których charakterystyki liczbowe zamierza się w danym momencie odczytać. Dane dotyczące niektórych cech zapisane są w więcej niż jednym miejscu, tzn. niektóre miejsca zawierają duplikaty informacji. Niech $q_{ij} = 1$ jeśli dane na temat cechy i zapisane są w miejscu j, oraz $q_{ij} = 0$ w przeciwnym przypadku. W ten sposób, np. $q_{13} = q_{18} = q_{19} = 1$ oznacza, że dane na temat cechy 1 zapisane są w miejscach 3, 8 i 9. Wyznaczyć te spośród n miejsc, które należy przeszukać, aby zminimalizować łączny czas odczytania danych dotyczących wszystkich cech.

Sformułować problem w postaci zadania programowania całkowitoliczbowego. Zapisać go korzystając z pakietu `JuMP` (z języka `Julia`) i rozwiązać jakiś egzemplarz problemu wywołując np. solver `GLPK` (lub `Cbc`). Oddzielić model od danych. Maksymalnie sparametryzować zapis modelu.

zad. 2 Niech P_{ij} będzie j-tym podprogramem obliczania funkcji i należącym do biblioteki podprogramów ($i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$). Podprogram P_{ij} zajmuje r_{ij} komórek pamięci i potrzeba na jego wykonanie t_{ij} jednostek czasu.

Należy ułożyć program P obliczający zadany zbiór funkcji I, $I \subseteq \{1, \ldots, m\}$. Zatem należy dobrać tak podprogramy P_{ij} wchodzące w skład P, obliczające wszystkie funkcje z I, aby cały program zajmował nie więcej niż M komórek pamięci, a czas jego wykonania był minimalny.

Sformułować problem w postaci zadania programowania całkowitoliczbowego. Zapisać go korzystając z pakietu `JuMP` (z języka `Julia`) i rozwiązać jakiś egzemplarz problemu wywołując np. solver `GLPK` (lub `Cbc`). Oddzielić model od danych. Maksymalnie sparametryzować zapis modelu.

zad. 3 Dany jest zbiór zadań $Z = \{1, ..., n\}$, które mają być wykonywane na trzech procesorach P_1, P_2 i P_3. Zakładamy, że:

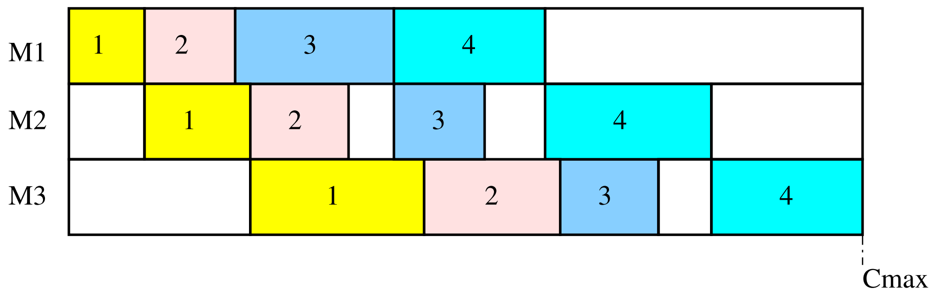
1. każdy procesor może wykonywać w danym momencie tylko jedno zadanie,
2. każde zadanie musi być wykonywana najpierw na procesorze P_1 następnie na procesorze P_2 i na końcu na procesorze P_3,
3. kolejność wykonywania zadań na wszystkich trzech procesorach jest taka sama.

Dla każdego zadania $i \in Z$ są zadane czasy trwania a_i, b_i oraz c_i odpowiednio na procesorach P_1, P_2 i P_3. Wszystkie dane są dodatnimi liczbami całkowitymi. Każdy harmonogram jest jednoznacznie określony przez pewną permutację $\pi = (\pi(1), ..., \pi(n))$ zadań należących do zbioru Z.

Niech $C_{\pi(k)}$ oznacza czas zakończenie k-go zadania na procesorze P_3 dla permutacji π. Celem jest wyznaczenie permutacji π takiej, że:

$$C_{\max} = C_{\pi(n)} \to \min .$$

Sformułować problem w postaci zadania programowania całkowitoliczbowego. Zapisać go korzystając z pakietu `JuMP` (z języka `Julia`) i rozwiązać jakiś egzemplarz problemu wywołując np. solver `GLPK` (lub `Cbc`). Oddzielić model od danych. Maksymalnie sparametryzować zapis modelu. Program powinien wizualizować rozwiązanie, np. na tekstowej konsoli, w stylu diagramu Gantt'a. Taka wizualizacja pozwala łatwo sprawdzić dopuszczalność harmonogramu.



Rysunek 1: Przykładowy harmonogram dla czterech zadań wyznaczony przez permutację $\pi = (1, 2, 3, 4)$.

zad. 4 [*] Dany jest zbiór R złożony z p typów odnawialnych zasobów $R_1, R_2, \ldots, R_p$. Zasoby te są limitowane, tj. dla każdego R_i, $i = 1, \ldots, p$ podany jest limit N_i jednostek. Limity są stałe – nie zmieniają się w całym okresie planowania.

Dany jest zbiór czynności $Z = \{1, ..., n\}$. Dla każdej czynności $j \in Z$ dany jest czas jej wykonania t_j (w jednostkach czasowych) oraz wektor $\boldsymbol{r}_j = [r_1, r_2, \ldots, r_p]$ opisujący zapotrzebowanie na poszczególne zasoby $R_1, R_2, \ldots, R_p$, tzn. opisujący ilość jednostek zasobów zużywanych podczas wykonywania czynności j. Na czynności zbioru Z nałożone są ograniczenia kolejnościowe (Z jest częściowo uporządkowany). Ograniczenia kolejnościowe mogą być reprezentowane za pomocą grafu, w którym wierzchołki odpowiadają czynnością, a łuki określają poprzedzanie. Jeśli $k \to l$, to czynność l nie może być rozpoczęta przed ukończeniem czynności k.

Należy znaleźć harmonogram minimalizujący czas wykonania całego przedsięwzięcia. Harmonogram jest dopuszczalny jeśli spełnia ograniczenia kolejnościowe oraz przydział zasobów, zgodny z zapotrzebowaniem, nie przekracza podanych limitów w każdym momencie okresu planowania.

Sformułować problem w postaci zadania programowania całkowitoliczbowego. Zapisać go korzystając z pakietu `JuMP` (z języka `Julia`) i rozwiązać egzemplarz problemu (patrz poniżej) wywołując np. solver `GLPK` (lub `Cbc`). Oddzielić model od danych. Maksymalnie sparametryzować zapis modelu. Program powinien wizualizować rozwiązanie, np. na tekstowej konsoli, w stylu diagramu Gantt'a. Drukować również zapotrzebowanie na zasoby dla każdego momentu okresu planowania. Taka wizualizacja pozwala łatwo sprawdzić dopuszczalność harmonogramu.

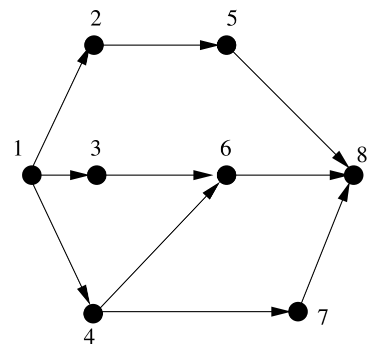
PRZYKŁAD EGZEMPLARZA PROBLEMU

Dane: liczba czynności $n = 8$, jeden typ zasobów (np. programiści) $p = 1$, limit zasobu $N_1 = 30$,

Czynność j	Czynności poprzedzające	Czasy wykonania t_j	Zapotrzeb. na zasoby $\boldsymbol{r}_j = [r_1]$
1	—	50	9
2	1	47	17
3	1	55	11
4	1	46	4
5	2	32	13
6	3,4	57	7
7	4	15	7
8	5,6,7	62	17

Graf poniżej opisuje ograniczenia kolejnościowe.

[*] Problem występuje podczas planowania i rozdziału zasobów np. w projekcie programistycznym.



Rozwiązania problemów przedstawić w sprawozdaniu, plik pdf + **wydruk**, które powinno zawierać:

1. modele
   (a) definicje zmiennych decyzyjnych (opis, jednostki),
   (b) ograniczenia (nie umieszczać źródeł modelu),
   (c) funkcje celu,
2. wyniki oraz ich interpretację.

Do sprawozdania należy dołączyć pliki w języku `julia` (`*.jl`). Pliki powinny być skomentowane: imię i nazwisko autora (**programy anonimowe nie będą sprawdzane**), komentarze zmiennych i komentarze ograniczeń. Spakowane pliki z modelami wraz ze sprawozdaniem (`*.zip`) należy przesłać e-mailem prowadzącemu. Natomiast wydruk sprawozdania należy oddać prowadzącemu na pierwszym po terminie oddania laboratorium.
Uwaga: Za zadania 1, 2, 3 (zadania obowiązkowe) można otrzymać co najwyżej ocenę dobrą. Zadania te będą punktowane następująco: zad. 1 - 8pkt, zad. 2 - 8pkt, zad. 3 - 14pkt i można otrzymać ocenę dst za 20pkt, dst+ za 25 pkt i db za 30pkt. Zad. 4 jest dodatkowe - jest na ocenę db+ lub bdb.