Metody Optymalizacji - Lista 3

Janusz Witkowski 254663

11 czerwca 2023

1 Zadanie 1

Celem zadania było zaimplementowanie w języku **Julia** z użyciem pakietu **JuMP** 2-aproksymacyjnego algorytmu (tj. zwracającego rezultaty co najwyżej 2 razy gorsze od optimum) opartego na programowaniu liniowym dla problemu szeregowania zadań na niezależnych maszynach z kryterium minimalizacji długości uszeregowania (ang. *Scheduling on Unrelated Parallel Machines and Makespan Criterion*). Należało oprzeć się na algorytmie opisanym we wskazanej literaturze. W tej implementacji oparto się na [[1], Rozdział 17, algorytm 17.5].

1.1 Idea

Zagadnienie *SUPMMC* jest ciężkim problemem optymalizacji dyskretnej - solver ma do przejrzenia olbrzymią liczbę możliwości. Pomysł na algorytm aproksymacyjny jest taki, by dokonać relaksacji na dyskretne zmienne (pozwala to na sytuacje, w których jedno zadanie może przynależeć do maszyny w pewnym stopniu, opisanym jako liczba rzeczywista od 0 do 1). Wyznaczymy zachłannie pierwotny grafik przydzielenia zadań, aby wyznaczyć stosunkowo mały zakres długości uszeregowania $[\alpha/m, \alpha]$ (gdzie m - liczba maszyn). Określimy następnie zadanie programowania liniowego $LP(T)$, w którym bierzemy pod uwagę zadania trwające krócej niż pewne $T \in [\alpha/m, \alpha]$, co powie nam czy dana instancja jest dopuszczalna. Poprzez metodę przeszukiwania binarnego będziemy dzielić ten zakres, aż do wyznaczenia dobrego kandydata T^*. Wartość zmiennej decyzyjnej tego kandydata nazywać będziemy *extreme point solution*. Następnie znajdziemy *perfect matching* dla tej konkretnej instancji w celu stworzenia dopuszczalnego rozwiązania problemu dyskretnego.

1.2 Model LP(T)

Mamy dane czasy wykonywania zadań na maszynach $p \in \mathbb{Z}_+^{m\times n}$, gdzie n - liczba zadań, m - liczba maszyn. Definiuje się zbiór $S_T = \{(i,j) : p_{ij} \leqslant T\}$ (czyli ograniczamy się do zadań które wykonają się w interesującym nas zakresie). Niech $x \in [0,1]^{m\times n}$ będzie zmienną decyzyjną. Celem jest odpowiedź na pytanie czy podana w ten sposób instancja jest dopuszczalna, niepotrzebna jest zatem funkcja celu. Zakładamy następujące ograniczenia:

$$(\forall_{j\in[n]}) \sum_{i:(i,j)\in S_T} x_{ij} = 1$$

$$(\forall_{i\in[m]}) \sum_{j:(i,j)\in S_T} x_{ij}p_{ij} \leqslant T$$

$$(\forall_{(i,j)\in S_T}) x_{ij} \geqslant 0$$

(W ostatecznej implementacji zdefiniowano zbiór S_T jako dwie tablice S_T^{left}, S_T^{right}, aby ułatwić zapis ograniczeń w skrypcie.)

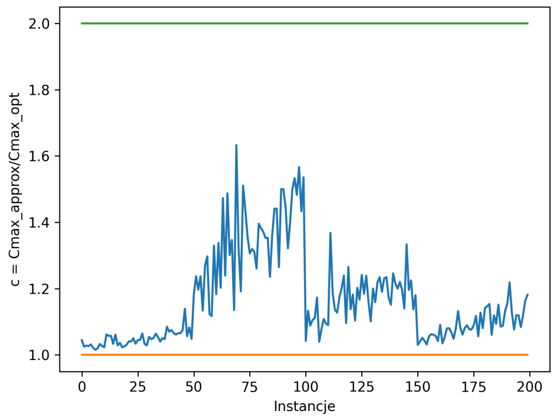
1.3 Algorytm

Mając dane czasy wykonywania zadań na maszynach p, algorytm stosuje nastepujące kroki:

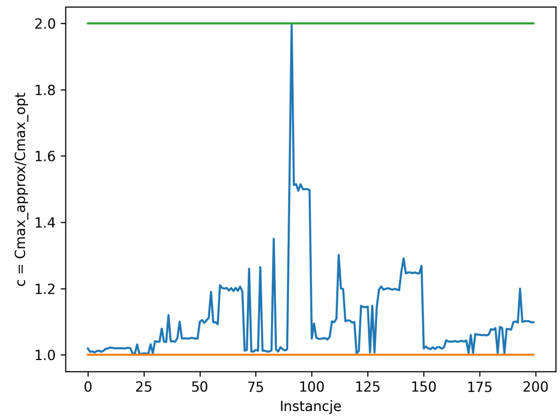
1. Wylicz zachłannie górne ograniczenie α,
2. Używając przeszukiwania binarnego na przedziale $[\alpha/m, \alpha]$, znajdź najmniejsze $T \in \mathbb{Z}_+$ które generuje dopuszczalną instancję $LP(T)$ i oznacz je przez T^*,
3. Wyłuskaj *extreme point solution* x^* dla $LP(T^*)$,
4. Przypisz do maszyn w ostatecznym rozwiązaniu te zadania, które są w całości przypisane do pojedynczych maszyn w x^*,
5. Znajdź *perfect matching* dla pozostałych zadań i przypisz je wedle niego do maszyn.

1.4 Rezultaty

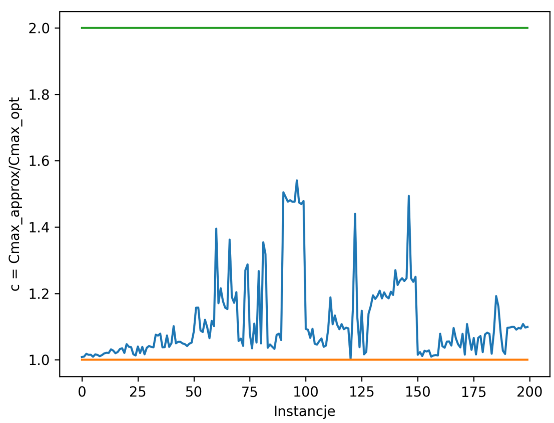
Do przetestowania algorytmu użyto bazy instancji podanej na liście zadań, by porównać zarówno wartości funkcji celu (1), jak i czasy wyznaczenia (2), z rozwiązaniami optymalnymi.



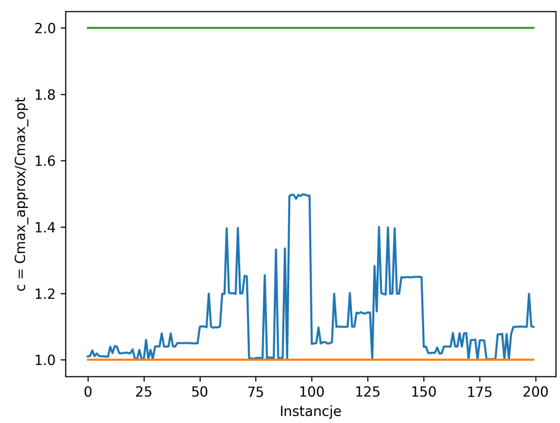
(a) 10a100



(b) 100a120

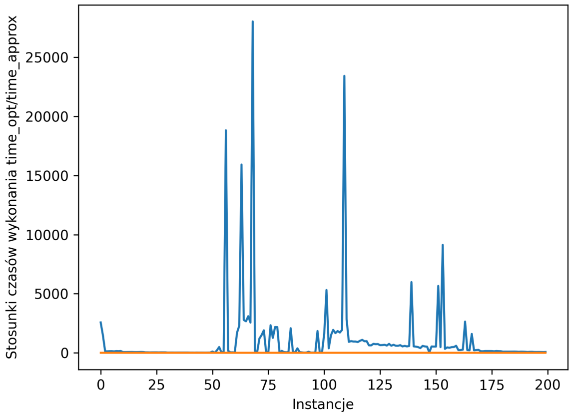


(c) 100a200

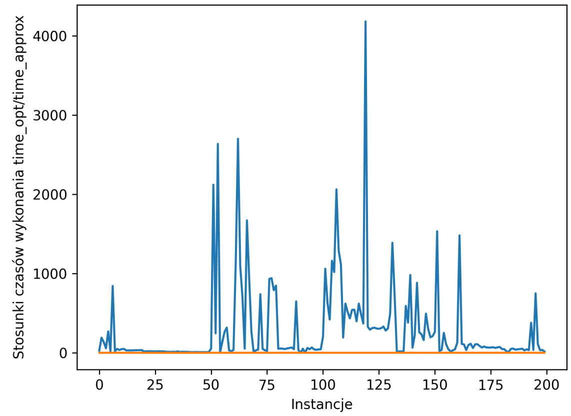


(d) 1000a1100

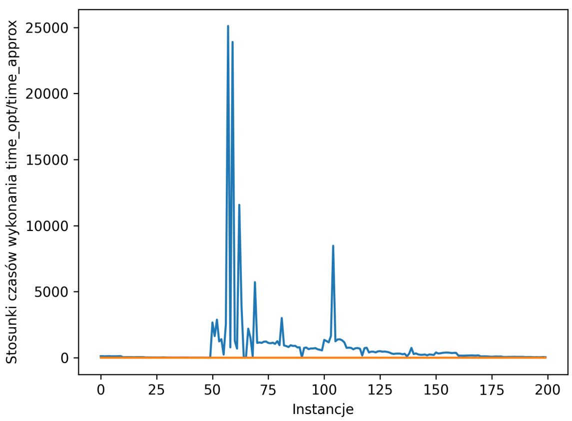
Rysunek 1: Stosunki wartości makespanu metody aproksymacyjnej do wartości funkcji celu najlepszych znanych rozwiązań, ograniczone z góry i z dołu.



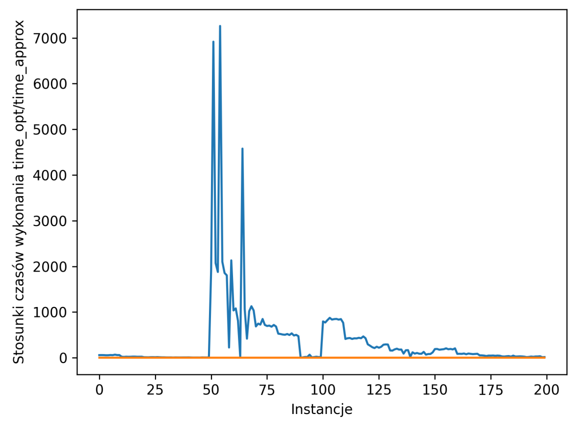
(a) 10a100



(b) 100a120



(c) 100a200



(d) 1000a1100

Rysunek 2: Stosunki czasów wykonania CPLEXa do metody aproksymacyjnej, ograniczone z dołu.

Można zauważyć, że błąd metody aproksymacyjnej rzeczywiście da się wyrazić w postaci stosunku rezultatu metody aproksymacyjnej do wartości funkcji celu CPLEXa w zakresie $[1, 2]$. Ponadto, metoda aproksymacyjna zapewnia nawet do kilkunastu-set razy szybsze wykonanie niż czas obliczeń CPLEXa. Oba te empirycznie sprawdzone fakty sprawiają, że metoda aproksymacyjna stanowi atrakcyjne podejście do zagadnienia optymalizacyjnego.

Literatura

[1] V. V. Vazirani. *Approximation algorithms.* 2003.