

WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI
POLITECHNIKA WROCŁAWSKA

METODY OPTYMALIZACJI

LISTA 3

KAMIL SIKORSKI
NR INDEKSU: 221481

Przedmiot prowadzony przez

Pawła Zielińskiego



Politechnika
Wrocławska

WROCŁAW 2019

Spis treści

1	Zadanie 1	1
1.1	Opis problemu	1
1.2	Opis algorytmu	1
1.3	Wyniki i interpretacja	2

Zadanie 1

1.1 Opis problemu

Zadanie polega na zaimplementowaniu algorytmu 2-aproksymacyjnego, opartym na programowaniu liniowym w języku **julia** z użyciem pakietu **JuMP**, dla problemu szeregowania zadań na niezależnych maszynach z kryterium minimalizacji długości uszeregowania (ang. Scheduling on Unrelated Parallel Machines and Makespan Criterion). Danymi są zadania J , maszyny M , oraz czas wykonywania $p_{j,i}$ zadania j na maszynie i , celem jest zminimalizowanie czasu wykonania wszystkich zadań. Dane:

- $J = \{1, 2, 3, \dots, n\}$ - zbiór zadań,
- $M = \{1, 2, 3, \dots, m\}$ - zbiór maszyn,
- $p_{j,i}$ gdzie $j \in J$, $i \in M$ - koszt wykonania zadania j na maszynie i .

1.2 Opis algorytmu

By rozwiązać problem posłużono się algorytmem podanym w książce *Approximation Algorithms-Springer-Verlag Berlin Heidelberg (2003)*. Składa się on z podprogramów:

- wyliczenia wstępnego - służy do wyliczenia przedziału w którym będzie stosować przeszukiwanie binarne.
- przeszukiwania binarnego na przedziale $[\alpha/m, \alpha]$,
- programowania liniowego z podanymi parametrami z przeszukiwania binarnego,
- operacji na grafie w celu stworzenia rozwiązania dopuszczalnego.

Wyliczenie wstępne obliczające

$$\alpha = \sum_{j \in J} \min(p_{j,i} : i \in M)$$

to znaczy suma minimalnego kosztu zadań na jakiejś maszynie.

Przeszukiwanie binarne na przedziale $[\alpha/m, \alpha]$, iteratorem jest wartość T , polega na uruchomieniu programu LP z parametrem T i znalezieniu najmniejszej możliwej wartości T z którym program LP znalazł rozwiązanie.

Programowanie liniowe z parametrami:

- T - maksymalny koszt pojedynczego zadania, maksymalny koszt na jednej maszynie,
- J - zbior zadań,
- M - zbiór maszyn,
- $p_{j,i}$ - koszt zadania j na maszynie i .

Model rozwiązujący posiada:

- Zmienną $X_{j,i} \geq 0$ - mówiącą w jakim stopniu zadanie j jest przydzielone do maszyny i ,
- Funkcje celu $\min \rightarrow \sum_{j \in J, i \in M} X_{j,i} * p_{j,i}$,
- Ograniczenie $\forall_{\substack{j \in J \\ i \in M \\ p_{j,i} > T}} X_{j,i} = 0$ - ogranicza maszynozadania które mają za duży czas wykonania,
- Ograniczenie $\forall_{j \in J} \sum_{i \in M} X_{j,i} = 1$ - każde zadanie musi zostać wykonane w całości,
- Ograniczenie $\forall_{i \in M} \sum_{j \in J} X_{j,i} * p_{j,i} \leq T$ - ogranicza koszt całkowity na jednej maszynie.

Program wykorzystuje relaksację zmiennych, by nie był problemem całkowitoliczbowym. Dzięki temu staje się algorytmem wielomianowym, ale samo rozwiązywanie zadania LP, nie musi dawać rozwiązania poprawnego. Powodem tego jest to że pojedyncze zadanie może być przydzielone częściowo do różnych maszyn, co w rzeczywistości nie daje rozwiązania dopuszczalnego.

Po znalezieniu najmniejszej wartości T oraz rozwiązania LP, trzeba naprawić rozwiązanie.

- zadania przydzielone w całości do konkretnej maszyny, przechodzą do rozwiązania,
- zadania częściowo przydzielone do różnych maszyn przechodzą do algorytmu grafowego.

Algorytm grafowy, rozwiązuje konflikty pomiędzy pojedynczymi zadaniami rozłożonymi na różnych maszynach. Zadania częściowe posiadają co najmniej dwie maszyny, algorytm rozwiązania konfliktu polega na przypisaniu maszynie z jednym zadaniem częściowym, tego zadania i usunięciu zadania z puli częściowych. Jeżeli powstanie cykl $m - j - m - j$ wybieramy na przemian maszyny do której zostanie przydzielony, powtarzamy całość aż wszystkie zadania częściowe staną się przydzielone.

1.3 Wyniki i interpretacja

Do przetestowania rozwiązania posłużono się danymi z <http://soa.iti.es/problem-instances> do podanego problemu. Dane zostały podzielone siedem grup, po dwieście przykładów w których jest podział na ilość zadań, maszyn i numer zestawu (np. 1025 - 10 oznacza setki zadań tzn. $10 * 100$ zadań, 25 - oznacza 20 maszyn 5 - zestaw danych; 554 - 500 zadań, 50 maszyn, 4 zestaw danych; 520 - oznacza 500 zadań, 10 maszyn, 10 zestaw danych):

- *instancias1a100* - czas wykonywania zadania jest z przedziału $[1, 2, \dots, 100]$,
- *instancias100a120* - czas wykonywania zadania jest z przedziału $[100, 101, \dots, 120]$,
- *instancias100a200* - czas wykonywania zadania jest z przedziału $[100, 101, \dots, 200]$,
- *instanciasde10a100* - czas wykonywania zadania jest z przedziału $[10, 11, \dots, 100]$,
- *instanciasde1000a1100* - czas wykonywania zadania jest z przedziału $[1000, 1001, \dots, 1100]$,
- *JobsCorre* - czas wykonywania jest wszechgólności zmienny względem zadania,
- *MaqCorre* - czas wykonywania jest wszechgólności zmienny względem maszyny,

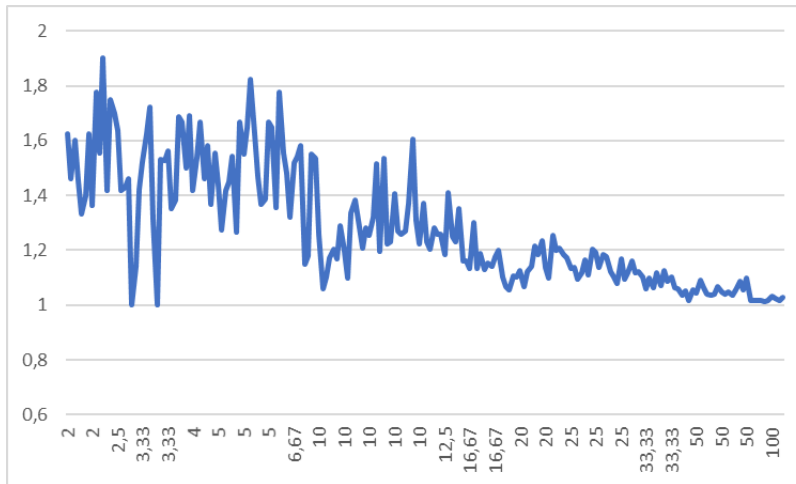
Wyniki algorytmu zostały porównane z wynikami optymalnymi. Algorytm nigdy nie zwraca wyniku o koszcie dwukrotnie większym od optymalnego.

Koszty dla danych z *JobsCorre*, *MaqCorre*, *instancias1a100*, *instanciasde10a100* przedstawiają się podobnie, Wraz ze wzrostem współczynnika $|J|/|M|$ maleje odchylenie od optymalnego rozwiązania. Czym więcej zadań musi zostać uruchomionych na jednej maszynie tym algorytm działa optymalniej.

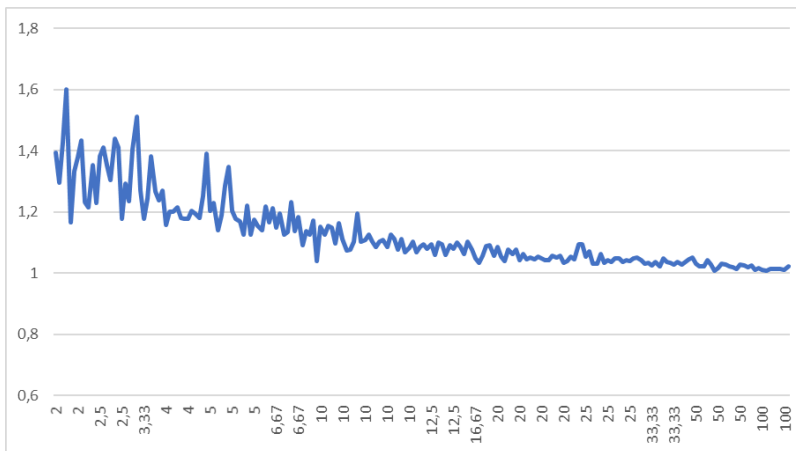
Dla danych z *instancias100a120*, *instancias100a200*, *Instanciasde1000a1100* można zaobserwować, że dla współczynnika $|J|/|M|$ będącego liczbą niecałkowitą, następują mniejsze odchylenia od wartości optymalnej. Spowodowane jest to przez przedział kosztu pojedynczego zadania (np. *Instanciasde1000a1100* posiada stałą 1000 i przedział $[0..100]$), jeżeli iloraz stałej i górnego ograniczenia przedziału jest odpowiednio duży oraz na

co najmniej jedną maszynę musi przypisać $|J|/|M|+1$ zadań. To minimalny koszt zbiega do $(|J|/|M|+1)*stala$, dzięki temu maszyny z mniejszą ilością zadań mają pewną downolność względem wyboru zadań.

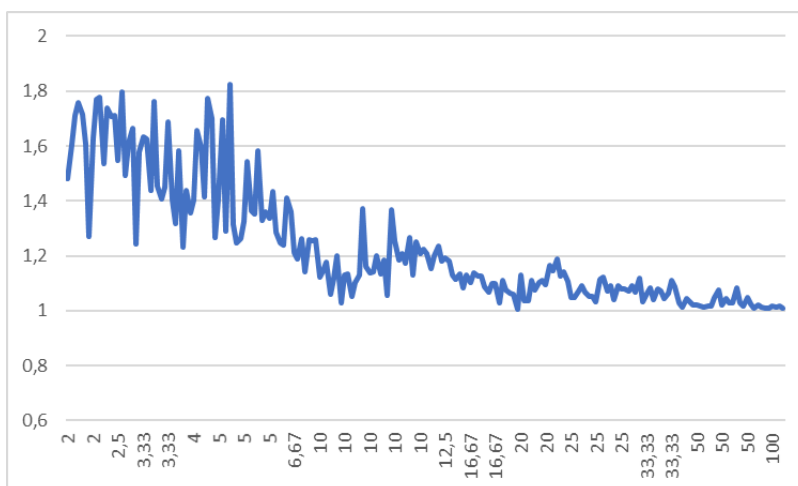
Inną rzeczą którą można zauważyć jest mniejsze odchylenie od optimum względem przedziału pojedynczego zadania. Dla *JobsCorre*, *MaqCorre*, *instancias1a100* iloraz kosztu do optymalnego sięga 1.9, gdzie dla danych z *instancias100a120*, *instancias100a200*, *Instanciasde1000a1100*, *instanciasde10a100* nie jest większe niż 1.6.



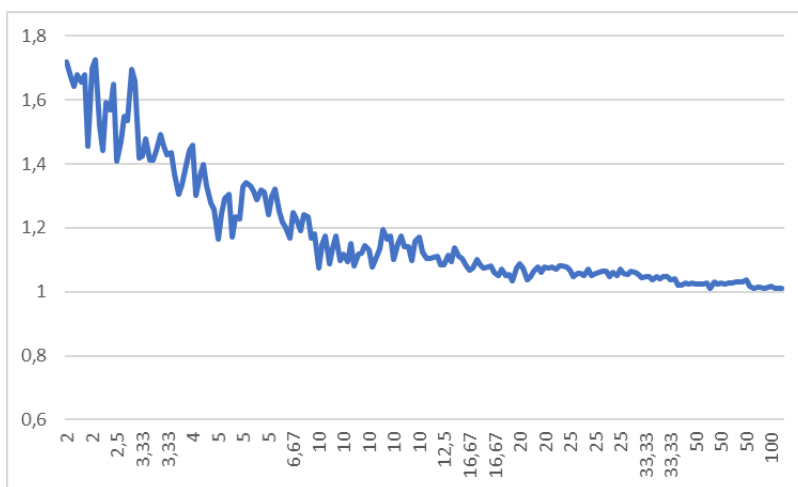
Rysunek 1.1: Dane dla 1 - 100



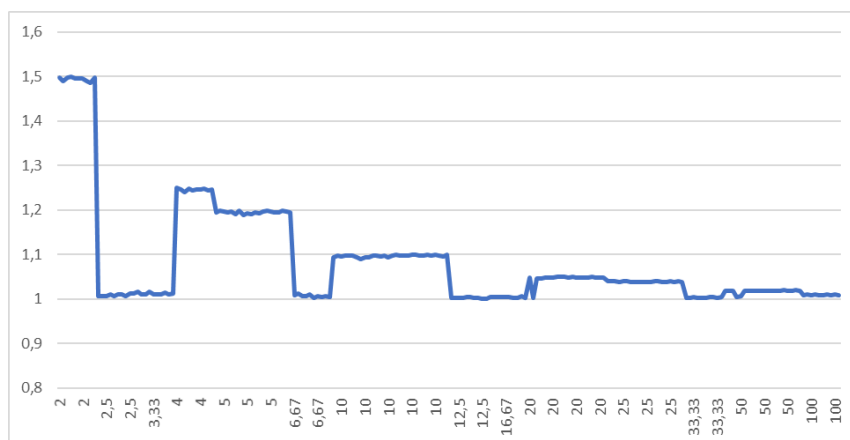
Rysunek 1.2: Dane dla 10 - 100



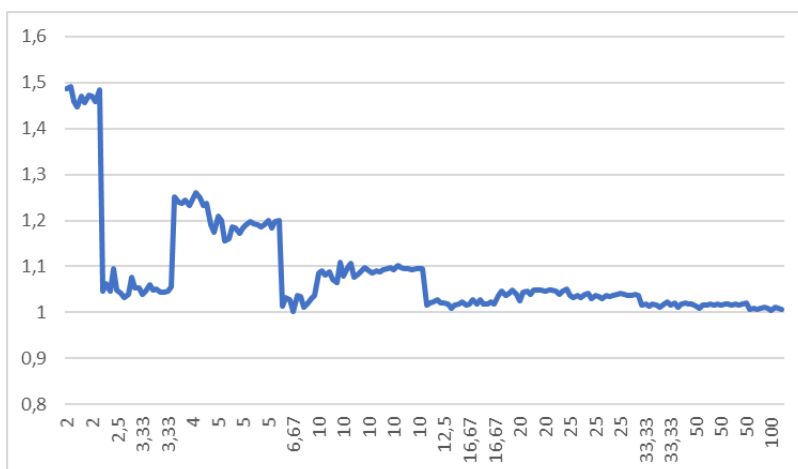
Rysunek 1.3: Dane dla MaqCorre



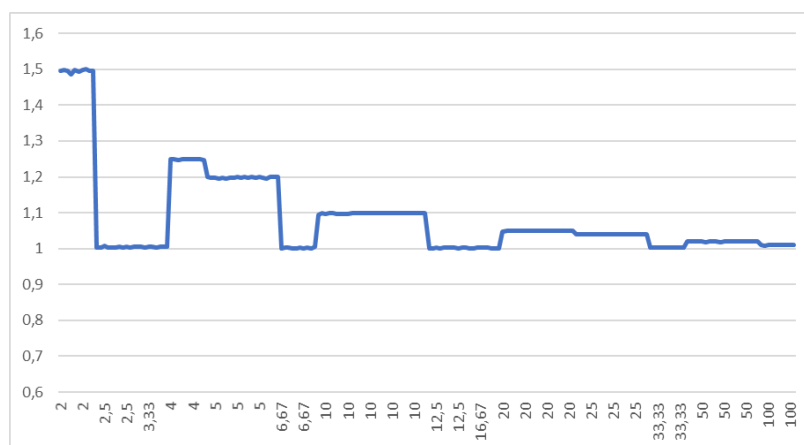
Rysunek 1.4: Dane dla JobsCorre



Rysunek 1.5: Dane dla 100 - 120



Rysunek 1.6: Dane dla 100 - 200



Rysunek 1.7: Dane dla 1000 - 1100