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:

- $J = \{1{,}2{,}3{,}...{,}n\}$ - zbiór zadań,
- $M = \{1{,}2{,}3{,}...{,}m\}$ - 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 - słyży 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 $X_{j,i} \geqslant 0$ - 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,
- Ogarniczenie $\forall_{j \in J} \sum_{i \in M} X_{j,i} = 1$ - każde zadania musi zostać wykonane w całości,
- Ogarniczenie $\forall_{i \in M} \sum_{j \in J} X_{j,i} * p_{j,i} \leqslant T$ - ogranicza koszt całkowity na jednej maszynie.

Program wykorzystuje relaksacje zmiennych, by nie był problemem całkowitoliczbowym. Dzięki temu staje się algorytmem wielomianowym, ale samo rozwiązania 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.

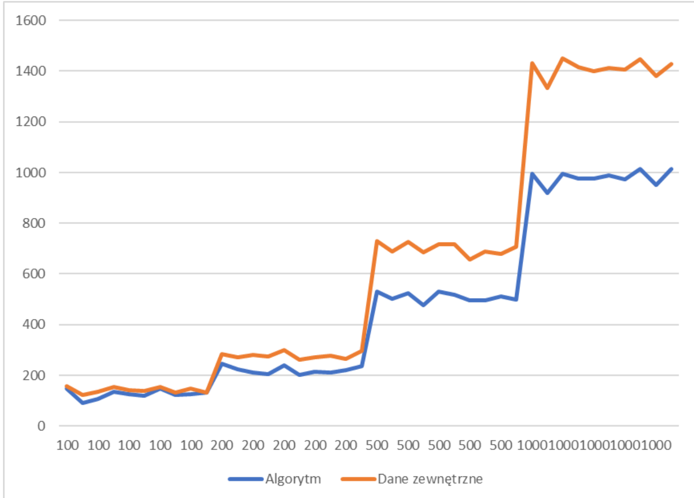
Algorytm grafowy, rozwiązuje konflikty pomiędzy pojedynczymi zadaniami rozłożonymi na różnych maszynach. Zadania częściowe posiadają conajmniej dwie maszyny, algorytm rozwiązania konfilktu polega na przypisaniu maszynie z jednym zadaniem częściowym, tego zadania i usuniciu zadania z puli częściowych.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

1.3 Wyniki i interpretacja

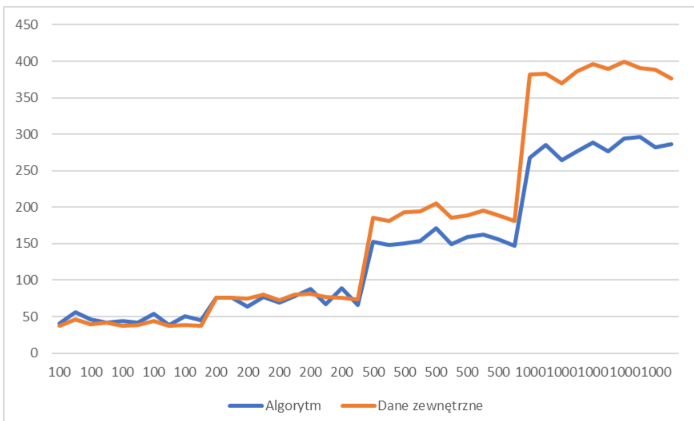
Do przetestowania rozwiązania posłużono się danymi z *http://soa.iti.es/problem-instances* do podanego problemu. Dane zostały podzielone siedem grup, po dwieście przykładów w których jest podział na ilosć zadań, maszyn i numer zestawu (np. 1025 - 10 oznacza setki zadań tzn. 10 * 100 zadań, 25 - oznacza 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,...,100],
- *instancias100a120* - czas wykonywania zadania jest z przedziału [100,101,...,120],
- *instancias100a200* - czas wykonywania zadania jest z przedziału [100,101,...,200],
- *instancias10a100* - czas wykonywania zadania jest z przedziału [10,11,...,100],
- *instancias1000a1100* - czas wykonywania zadania jest z przedziału [1000,1001,...,1100],
- *JobsCorre* - czas wykonywania jest wszczególności zmienny względem zadania,
- *MaqCorre* - czas wykonywania jest wszczególności zmienny względem maszyny,

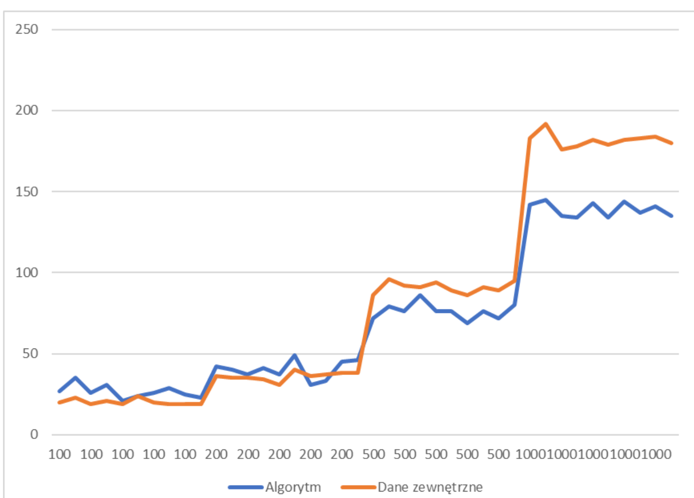
Wyniki zostały porównane z wynikami na stronie *http://soa.iti.es/problem-instances*. Wykresy z danych *instancias* są bardzo zbliżone do siebie. Jak widać algorytm lepiej działa w przypadku gdy wiele zadań przydzielone jest do jednej maszyny. Dane testowe textitJobsCorre wnioski podobne. Dla danych testowych textitMaqCorre algorytm działa bardzo zbliżenie do tego podanego w rozwiązaniu. Wyniki wszystkich testów załączony jest w załączniku.



Rysunek 1.1: Dane dla 1 - 100 z 10 maszynami

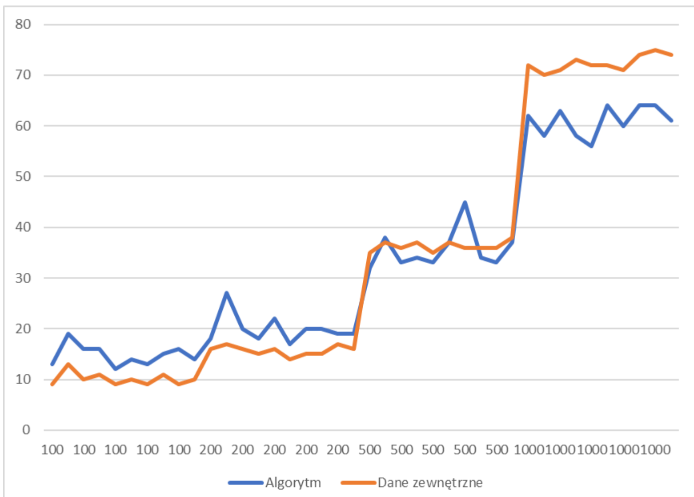


Rysunek 1.2: Dane dla 1 - 100 z 20 maszynami

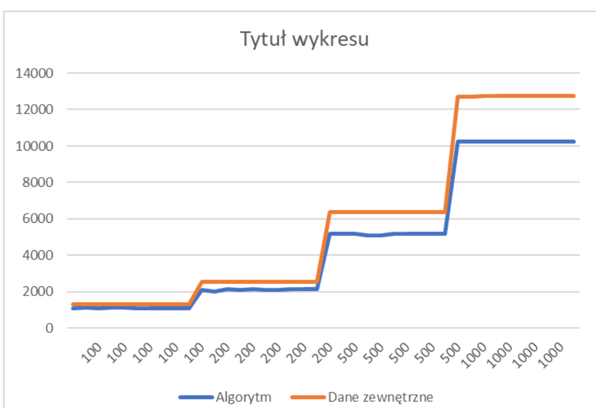


Rysunek 1.3: Dane dla 1 - 100 z 30 maszynami

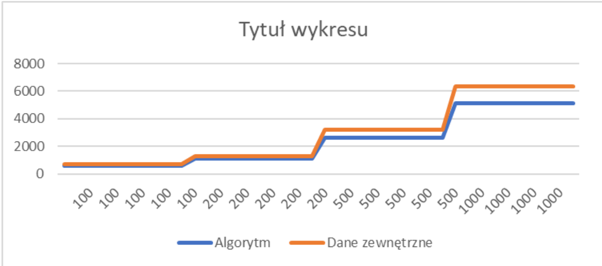
Rysunek 1.4: Dane dla 1 - 100 z 40 maszynami



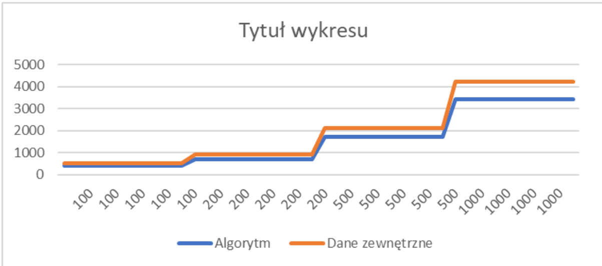
Rysunek 1.5: Dane dla 1 - 100 z 50 maszynami



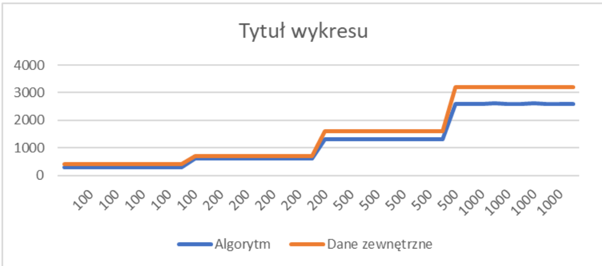
Rysunek 1.6: Dane dla 100 - 120 z 10 maszynami



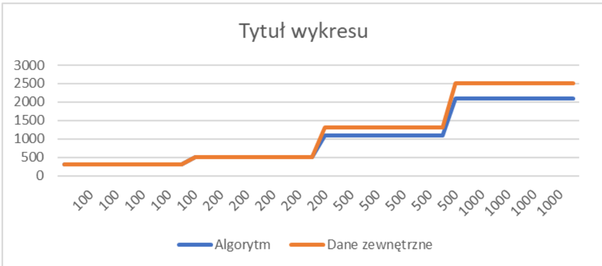
Rysunek 1.7: Dane dla 100 - 120 z 20 maszynami



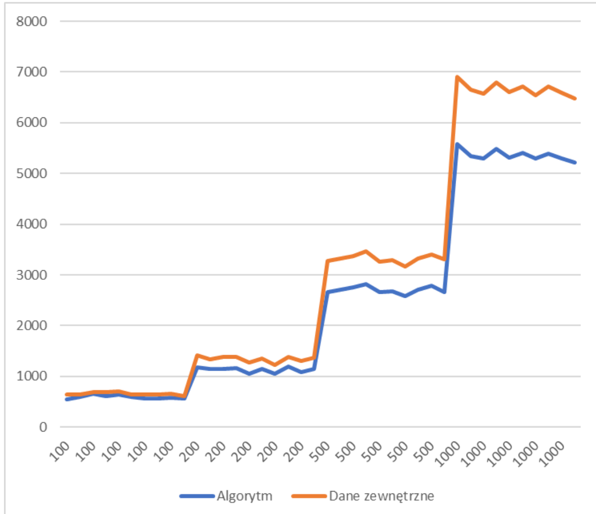
Rysunek 1.8: Dane dla 100 - 120 z 30 maszynami



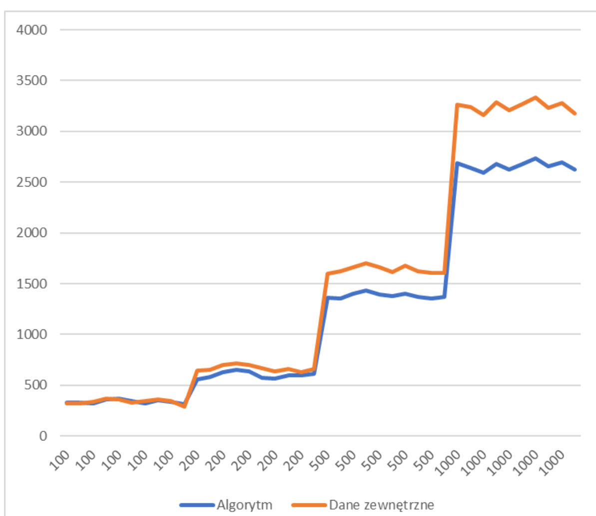
Rysunek 1.9: Dane dla 100 - 120 z 40 maszynami



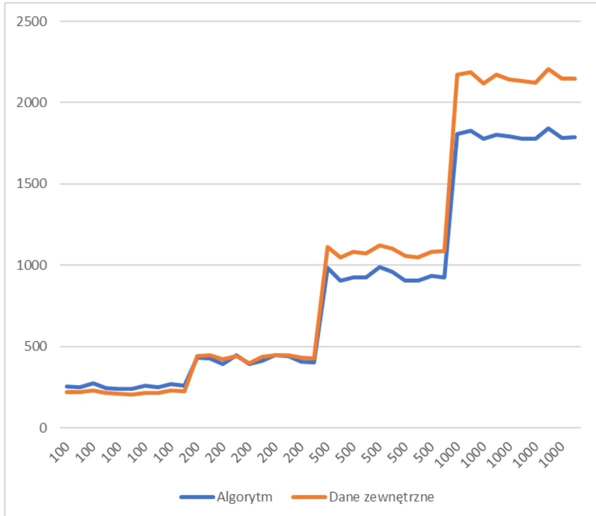
Rysunek 1.10: Dane dla 100 - 120 z 50 maszynami



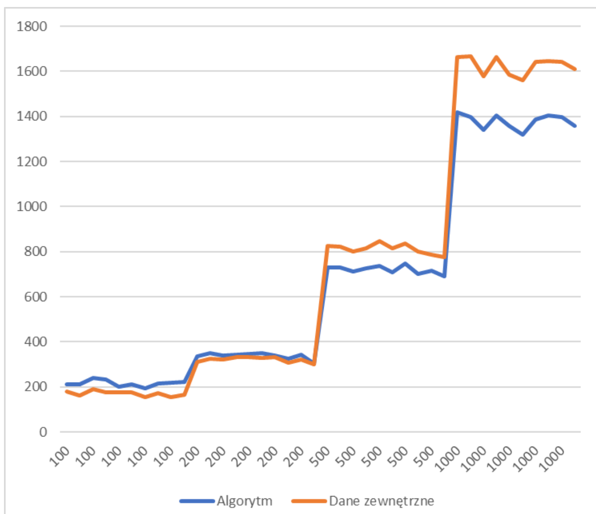
Rysunek 1.11: Dane dla JobsCorre z 10 maszynami



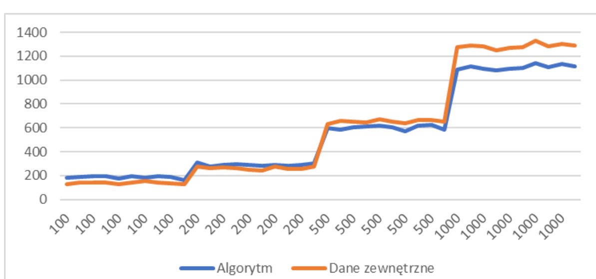
Rysunek 1.12: Dane dla JobsCorre z 20 maszynami



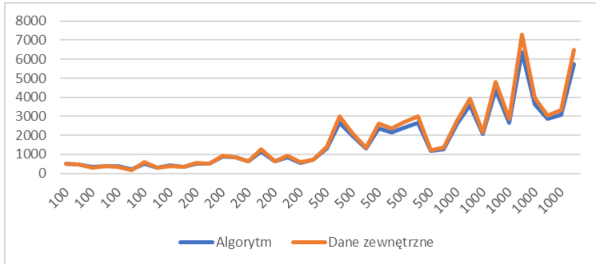
Rysunek 1.13: Dane dla JobsCorre z 30 maszynami



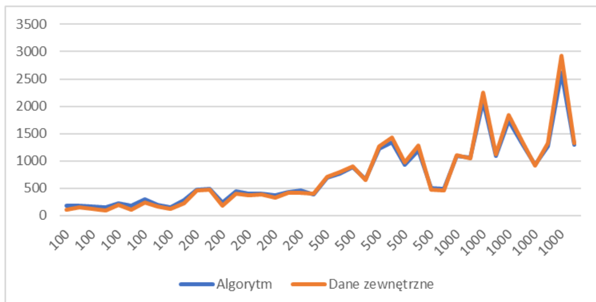
Rysunek 1.14: Dane dla JobsCorre z 40 maszynami



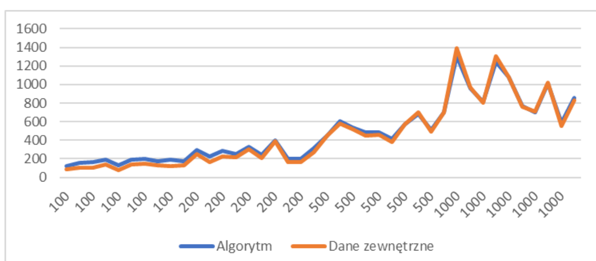
Rysunek 1.15: Dane dla JobsCorre z 50 maszynami



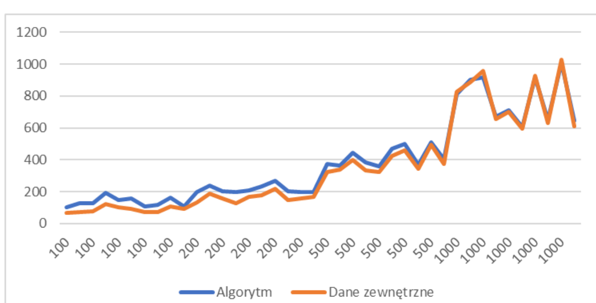
Rysunek 1.16: Dane dla MaqCorre z 10 maszynami



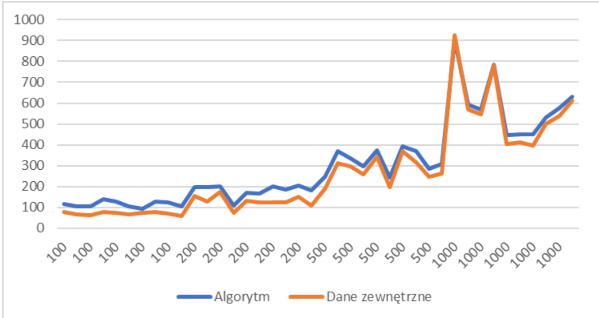
Rysunek 1.17: Dane dla MaqCorre z 20 maszynami



Rysunek 1.18: Dane dla MaqCorre z 30 maszynami



Rysunek 1.19: Dane dla MaqCorre z 40 maszynami



Rysunek 1.20: Dane dla MaqCorre z 50 maszynami