Job-shop scheduling

Nazewnictwo

W sprawozdaniu przyjęliśmy następujące nazewnictwo:

Zadanie - zbiór prac, które muszą zostać wykonane w kolejności

Praca, kandydat - obiekt podlegający uszeregowaniu

K, k - parametr kontrolujący wielkość zbioru najlepszych kandydatów z których jeden zostanie dodany do rozwiązania.

harmonogram, uszeregowanie - kolejność wykonywania zadań na każdej maszynie.

Opis algorytmu

Do wyszukiwania rozwiązań skorzystaliśmy z algorytmu GRASP. Jest to algorytm opierający się na algorytmie zachłannym, który zmodyfikowany jest poprzez dodanie czynnika losowego w postaci parametru K. Zastosowaliśmy heurystykę priorytetyzacją takie prace których dodanie do rozwiązania powoduje najmniejszy przyrost czasu uszeregowania.

Parametr K kontroluje liczbę najlepszych kandydatów z których losowana jest praca do uszeregowania.

Główna pętla

W głównej pętli GRASPa wybierane jest najlepsze bieżące rozwiązanie. Pętla ta trwa nadaną odgórnie ilość czasu, domyślnie jest to pięć sekund. W środku pętli pobierane jest aktualne rozwiązanie wygenerowane przez procedurę GRASP wraz z czasem jego uszeregowania a następnie rozwiązanie to porównywane jest z dotychczas najlepszym rozwiązaniem. Jeśli aktualne uszeregowanie jest lepsze od dotychczas najlepszego, jest ono zapisywane jako najlepsze. Dzieje się tak aż do upłynięcia czasu działania pętli. Po wykonaniu sie pętli, funkcja zwraca najlepsze rozwiązanie. Warunek czasu działania jest sprawdzany przed każdą iteracją co wprowadza niezgodność oczekiwanego czasu działania z rzeczywistym. W rzeczywistości program będzie działał najkrócej przez zadany czas, a najdłużej przez sumę zadanego czasu, czasu jednej iteracji i czasu odczytania instancji.

Procedura GRASP

Algorytm opiera się na zachłannym wyborze następnej pracy do uszeregowania, a następnie na wpisaniu jej do rozwiązania. GRASP będzie przeprowadzał te procedury dopóki będą dostępne prace, ich brak sygnalizowany jest zwracaną wartością -1 przez funkcję wybierającą kandydatów. Procedura zwraca w takim wypadku znalezione rozwiązanie. Ostateczny czas uszeregowania jest wartością maksymalną z wektorów dostępności maszyn i prac.

Wybór kandydata

Funkcja realizuje następujące procedury:

-Znalezienie prac, które są dostępne oraz znalezienie dla każdej z nich czasu o który wydłuży bieżące uszeregowanie.
-Posortowanie znalezionych kandydatów rosnąco według czasu, o który wydłużone zostanie bieżące uszeregowanie.
-Wybór kandydata z K prac które najmniej wydłużają bieżące uszeregowanie

Jeśli liczba znalezionych prac jest mniejsza niż K to K przyjmuje wartość ich liczności.

Wybrany kandydat jest usuwany z kontenera zadań oraz zwracany, jeśli wybór nie był możliwy - czyli nie znaleziono żadnej dostępnej pracy zwracany jest sygnał zakończenia pracy.

Aktualizacja rozwiązania

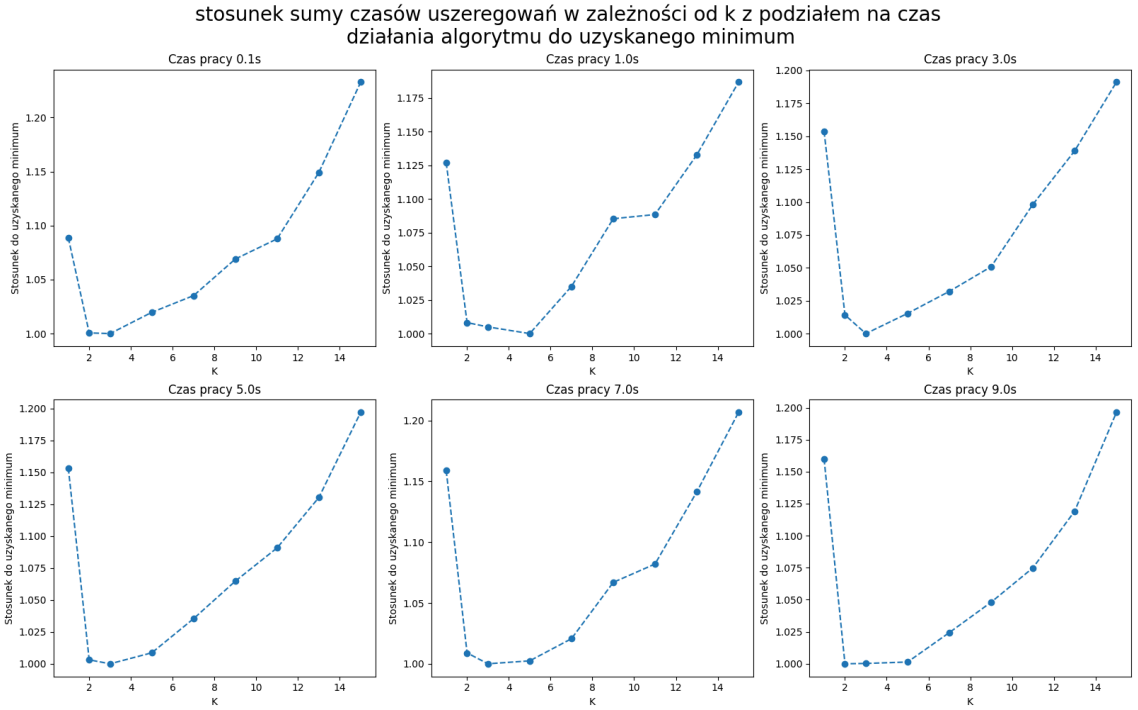
Rozwiązanie jest przechowywane jako tensor dwuwymiarowy czasów rozpoczęcia pracy. Przechowywane są dodatkowo wektory czasu dostępności maszyn i dostępności kolejnej pracy w zadaniu.

Funkcja wyznacza czas rozpoczęcia pracy poprzez wybranie większej wartości z czasu dostępności maszyny oraz czasu dostępności następnej pracy danego zadania. Następnie do bieżącego rozwiązania wpisywany jest on na odpowiednich indeksach (numer zadania, numer pracy). Ostatecznie modyfikujemy dostępność maszyny i kolejnej pracy zapisując w odpowiednich komórkach sumę czasu rozpoczęcia bieżącej pracy i jej czasu trwania.

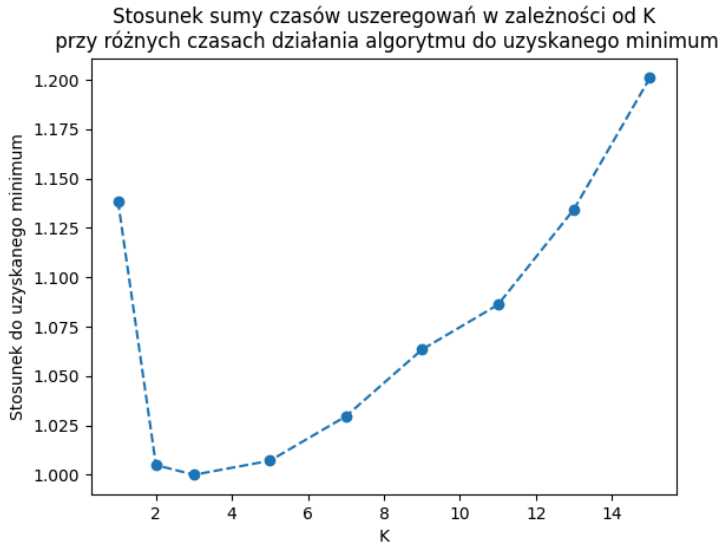
Tuner

Wystrojenie algorytmu opiera się na przetestowaniu grupy instancji dla różnych K przy zadanym czasie. Parametr ten jest przyjmowany jako 1, 2, 3, 5, 7, …, 15. Tuner następnie agreguje czasy uszeregowań według K, sumuje je oraz zwraca K dla którego suma jest minimalna.

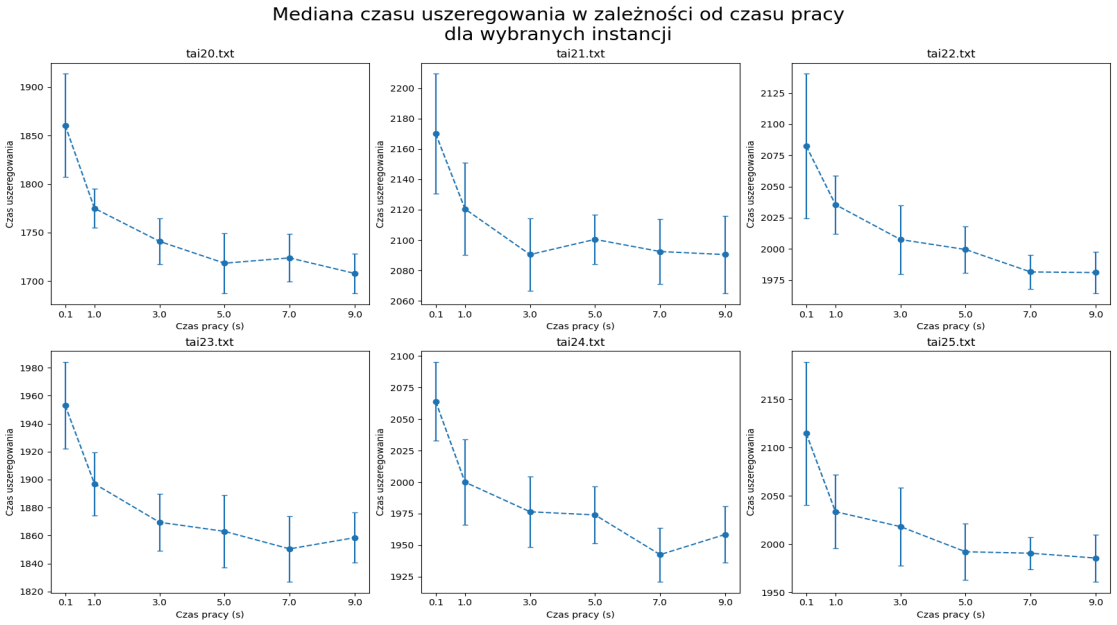
K w algorytmie dostosowaliśmy tak aby zminimalizować oczekiwany czas uszeregowania dla programu działającego dziewięć sekund bez względu na wielkość instancji. GRASP korzysta z K=3.



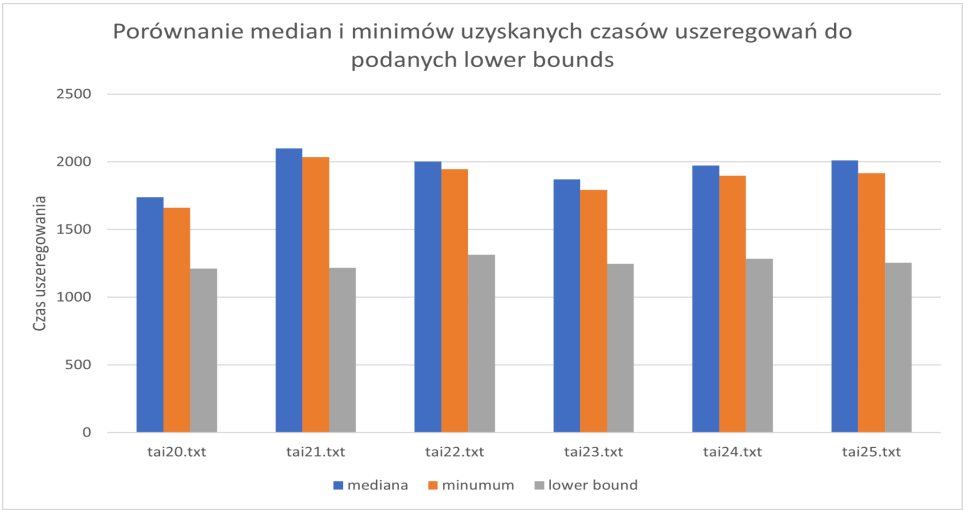
Powyższe wykresy obrazują strojenie dla różnych zadanych czasów pracy. Wyniki przedstawione są relatywnie do najkrótszego uszeregowania wygenerowanego przez nasz algorytm. Wykresy różnią się w zależności od długości pracy programu. Dla testowanych instancji problemu odpowiednie wartości k znajdują się w przedziale od dwóch do pięciu. Na potrzeby naszego zadania ustaliliśmy czas działania algorytmu na dziewięć sekund, w związku z tym domyślna wartość k jaką wybraliśmy to trzy. Warto wspomnieć, że odpowiednia wartość będzie różna w zależności od charakterystyki instancji problemów, ich wielkości oraz czasu działania algorytmu.



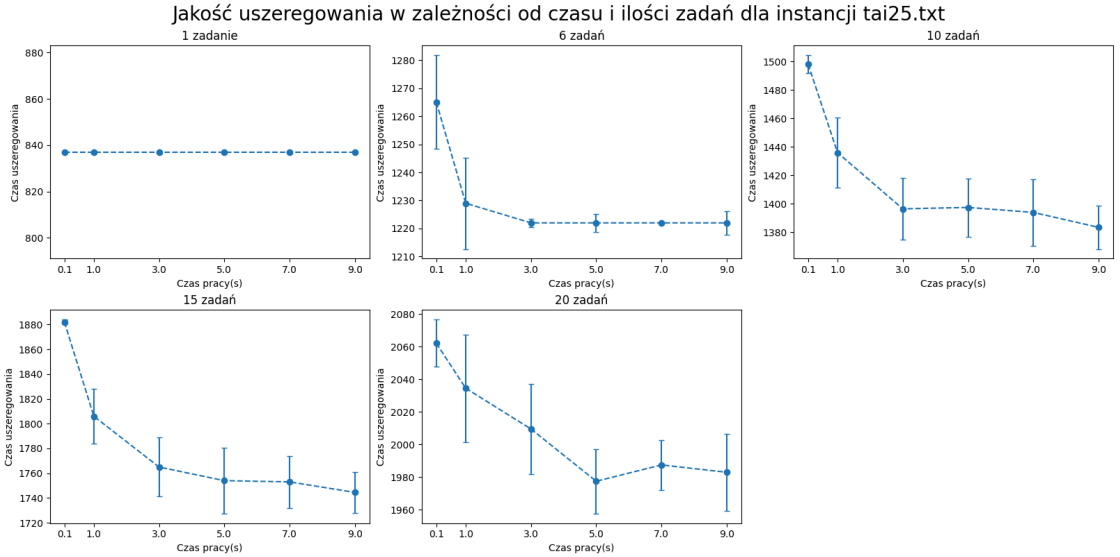
Powyższy wykres porównuje jakość uzyskiwanych rozwiązań dla różnych wartości parametru k jednak nie uwzględnia różnic w czasie działania algorytmu. Jeśli nie znalibyśmy czasu w jakim wykona się nasz algorytm, najbardziej odpowiednią wartością parametru k byłoby trzy. Należy zauważyć, że brak losowości jest kiepskim rozwiązaniem wyniki są wtedy deterministyczne i algorytm nie ma szans poprawiać rozwiązania.



Przedstawione wykresy ilustrują mediany uzyskanych czasów uszeregowań w zależności od czasu pracy dla wybranych instancji problemu, gdzie ustalony parametr k wynosi trzy. Wszystkie testowane problemy, z wyjątkiem mniejszego tai20, mają identyczny rozmiar 20x20 Zauważalne jest, że im krótszy czas pracy, tym gorsze rezultaty oraz większa mediana. To zjawisko wynika z proporcjonalnością liczby iteracji jakie wykonuje algorytm GRASP do jego czasu działania. Zwiększając czas, mediana maleje wraz z długością uszeregowania (do pewnego stopnia), co jest rezultatem większej liczby iteracji i potencjalnych uszeregowań, analizowanych przez algorytm. Algorytm ma losowy charakter i dla różnych uruchomień może zwrócić różne wyniki czego rezultatem może być gorsza jakość rozwiązania mimo wyższego czasu pracy.



Następny zestaw wykresów przedstawia jakość rozwiązań generowanych przez algorytm GRASP przy czasie pracy ustalonym na dziewięć sekund. Jak widać, uzyskane wyniki znacząco odbiegają od dolnego ograniczenia, może to być spowodowane nieoptymalnym dobraniem funkcji celu oraz nieuwzględnieniem powstających “dziur” w uszeregowaniu podczas dodawania pracy. Warto zauważyć stosunkowo niewielką różnicę między minimalnym czasem uszeregowania a medianą, co może świadczyć o regularności jakości generowanych rozwiązań.



Przedostatni zestaw wykresów prezentuje zależność czasu uszeregowania od czasu pracy w kontekście liczby zadań branych pod uwagę w instancji tai25.

Gdy rozważamy tylko jedno zadanie, algorytm osiąga globalne minimum czasu uszeregowania (wielkość przestrzeni rozwiązań to 1).

Przy uwzględnieniu sześciu zadań, algorytm w praktyce nie poprawia się przy czasie działania większym od trzech sekund. Algorytm w tym czasie sprawdza większość możliwych rozwiązań przy zadanym k, jednak nie mamy gwarancji że wszystkie zostaną sprawdzone. Nie mamy również gwarancji że znaleźliśmy minimum globalne, ponieważ k < liczba zadań przez co algorytm nie sprawdzi wszystkich możliwych solucji oraz ze względu na naszą implementacje w razie powstania dziury w harmonogramie nie próbujemy jej wypełnić.

Analizując ostatnie trzy wykresy w tym zestawie, możemy zauważyć, że czas uszeregowania maleje wraz z wydłużeniem czasu pracy. Zjawisko to wynika z tego, że zwiększenie liczby iteracji wykonania algorytmu GRASP prowadzi do przeglądania większej liczby potencjalnych rozwiązań. w Przestrzeni poszukiwań pozostaje wiele niesprawdzonych rozwiązań, których nie rozważyliśmy w zadanym czasie pracy.

Testowanie efektywności

Testy instancji tai20-tai25 zostały przeprowadzone dziesięciokrotnie. Po wykonaniu testów wyliczyliśmy mediany i odchylenia standardowe oraz sporządziliśmy ich wykresy. Czas pracy mierzony był za pomocą biblioteki chrono i funkcji high resolution clock, a wykresy przygotowane zostały za pomocą biblioteki matplotlib i excel’a.

Wnioski

Wybór algorytmu GRASP do rozwiązania problemu Job Shop Scheduling cechuje się stosunkowo dobrymi wynikami, może być atrakcyjny ze względu na krótki czas wykonania w porównaniu do bardziej zaawansowanych metod, takich jak algorytmy genetyczne czy tabu search.

W porównaniu do prostych strategii, takich jak zwykły algorytm zachłanny czy kompletnie losowy, GRASP oferuje lepsze wyniki, co stanowi jego zaletę, zwłaszcza gdy zależy nam na efektywności czasowej. Jednakże, jeśli dąży się do uzyskania rozwiązań zbliżonych do optymalnych, bardziej zaawansowane algorytmy mogą być preferowane, nawet kosztem wydłużenia czasu działania.

Warto zauważyć, że wybór algorytmu GRASP wiąże się z koniecznością dostosowania parametrów, takich jak wartość k (liczba kandydatów) dla uzyskania optymalnych wyników. Nieodpowiedni dobór wartości tego parametru może wpłynąć negatywnie na jakość generowanych rozwiązań, co podkreśla konieczność ostrożnego dostrajania algorytmu.

Proces dostrajania parametrów może być czasochłonny i wymaga dostatecznej liczby podobnych instancji problemów do tych, które chcemy rozwiązać. Jeśli jednak problemy są znacząco różne, dostrajanie może być niewłaściwe i prowadzić do wyników oddalonych od optymalnych. Warto więc odpowiednio dostosować podejście w zależności od specyfiki problemu Job Shop Scheduling i oczekiwanych kryteriów wydajności algorytmu.

Ważnymi cechami algorytmu GRASP są wybrana heurystyka i sposób aktualizacji rozwiązania. W przypadku gdy nie rozważamy dziur powstałych w harmonogramie, ograniczamy możliwość dalszej optymalizacji rozwiązania. Odpowiednia heurystyka może znacząco poprawić generowany wynik.