Jakub OLSZAK[1], Jakub PIETRUCZUK[1], Piotr FORMANOWICZ[1,2]
[1]Instytut Informatyki, Politechnika Poznańska
[2]Instytut Chemii Bioorganicznej, Polska Akademia Nauk

SZEREGOWANIE SEMI-WZNAWIALNYCH ZADAŃ NA JEDNEJ MASZYNIE Z OKRESAMI NIEDOSTĘPNOŚCI

Streszczenie. W klasycznej teorii szeregowania zadań przyjmuje się, że maszyny dostępne są w sposób ciągły. W praktyce założenie to nie zawsze jest spełnione, stąd od pewnego czasu intensywnie badane są problemy, w których zakłada się, że maszyny są niedostępne w pewnych przedziałach czasu. W przypadku tego typu problemów pojawiają się nowe typy zadań, m. in. zadania semi-wznawialne. Zadania tego rodzaju mogą zostać przerwane przez rozpoczynający się okres niedostępności i kontynuowane po jego zakończeniu, lecz jest to związane z pewnym dodatkowym kosztem. W niniejszej pracy rozważany jest problem szeregowania zadań semi-wznawialnych na jednej maszynie z wieloma okresami niedostępności i kryterium, którym jest długość uszeregowania.

SCHEDULING SEMI-RESUMABLE TASKS ON A SINGLE MACHINE WITH PERIODS OF NON-AVAILABILITY

Summary. In classical scheduling theory it is usually assumed that machines are continuously available. However, in practice this assumption is not always fulfilled. Hence, recently scheduling problems, where machines are non-available in some periods of time are intensively studied. In case of such problems new types of tasks are considered, among them semi-resumable ones. Tasks of this type can be preempted by a non-availability period and continued after it but some additional cost must be paid. In this paper a problem of scheduling semi-resumable tasks on a single machine with multiple non-availability periods in order to minimize makespan is considered.

1. Wstęp

Jednym z założeń klasycznej teorii szeregowania zadań jest ciągła dostępność maszyn. Założenie to jednak często nie jest spełnione w praktyce, stąd od pewnego czasu intensywnie badane są problemy, w których dopuszcza się ich okresową niedostępność (por. [4, 3]). Niedostępność ta może wynikać m. in. z konieczności przeprowadzenia prac konserwacyjnych, w czasie których maszyna często jest wyłączona, stosowania algorytmów planowania z przesuwanym horyzontem czasowym, bądź szeregowania zadań z różnymi priorytetami. W tym ostatnim przypadku w pierwszej kolejności szeregowane są zadania z wyższymi priorytetami, co powoduje, że w fazie, w której szeregowane są zadania z niższymi priorytetami, przedziały czasu, w których maszyna została przy-

dzielona do zadań wcześniej uszeregowanych, mogą być interpretowane jako okresy, w których nie jest ona dostępna.

W literaturze najczęściej rozważane są problemy jednomaszynowe z jednym okresem niedostępności i zadaniami niepodzielnymi lub wznawialnymi. Zadania wznawialne to takie, które mogą zostać przerwane w momencie rozpoczęcia okresu niedostępności i kontynuowane natychmiast po jego zakończeniu na tej samej maszynie bez dodatkowego kosztu. Dużo rzadziej rozważane są problemy, w których występuje dowolna liczba okresów niedostępności. W niniejszej pracy rozważany jest problem tego rodzaju, a sciślej, problem szeregowania zadań semi-wznawialnych na jednej maszynie z wieloma okresami niedostępności i kryterium, którym jest długość uszeregowania. Problem ten w rozszerzonej standardowej notacji trójpolowej zapisuje sie jako $1, h_k|srs|C_{max}$. Zadania semi-wznawialne są podobne do zadań wznawialnych z tym, że z ich wznowieniem po okresie niedostępności maszyny związany jest pewien dodatkowy koszt.

2. Zadania semi-wznawialne

W ogólności możliwe jest rozważanie różnych rodzajów zadań semi-wznawialnych, które różnią się między sobą sposobem naliczania kary za wznowienie wykonywania zadania po okresie niedostępności maszyny. Najczęściej badane są problemy z zadaniami semi-wznawialnymi, które scharakteryzowane są współczynnikiem α określającym, jaka część wznawianego zadania musi zostać ponownie wykonana. Ściślej, jeżeli p_j jest długością zadania T_j daną w instancji problemu i jeżeli przed okresem niedostępności wykonane zostało p'_j jednostek tego zadania, to po zakończeniu tego okresu dodatkowo musi zostać przeznaczonych $\alpha p'_j$ jednostek czasu na jego wykonanie. A zatem całkowita długość takiego zadania wynosi $p''_j = p_j + \alpha p'_j$. Taki rodzaj zadań semi-wznawialnych odpowiada sytuacjom, w których przerwanie wykonywanie zadania powoduje, że część pracy wykonanej przed okresem niedostępności jest tracona i trzeba ją powtórzyć po jego zakończeniu. Może też być tak, że ta część pracy nie jest tracona, ale każdorazowe rozpoczęcie wykonywania zadania (również przy jego wznawianiu) wymaga wykonania pewnych czynności wstępnych, co prowadzi do modelu zadań, w którym każde przerwanie dodaje do ich długości pewną stałą wartość (która jednak może być różna dla poszczególnych zadań). W takim przypadku rzeczywista długość wznawianego zadania wynosi $p''_j = p_j + s_j$, gdzie s_j jest wspomnianą stałą wartością charakterystyczną dla zadania T_j. W obu przypadkach, jeżeli zadanie jest przerywane przez więcej niż jeden okres niedostępności, jego początkową długość należy zwiększyć w opisany sposób odpowiednią liczbę razy.

Problemy szeregowania zadań semiwznawialnych są niełatwiejsze niż odpowiadające im problemy szeregowania zadań niepodzielnych, gdyż te ostatnie stanowią ich szczególny przepadek dla $\alpha = 1$. Złożoność podstawowych problemów szeregowania zadań niepodzielnych na jednej maszynie z jednym oraz z wieloma okresami niedostępności analizawana byla w pracach [2, 1]. Z analizy tej wynika m. in., iż badany w niniejszej pracy problem $1, h_k|srs|C_{max}$ jest silnie NP-trudny. Z faktu, iż problemy szeregowania zadań niepodzielnych są szczególnymi przypadkami problemów szeregowania zadań semi-wznawialnych wynika, iż analiza błędów popełnianych przez algorytmy przybliżone dla problemów z zadaniami niepodzielnymi dotyczy również problemów z

zadaniami semi-wznawialnymi, gdy do ich rozwiązania stosowane są te same algorytmy. Analizę taką dla podstawowych algorytmów listowych przeprowadzono m. in. w [1]. W pracy [5] analizowany był błąd popełniany przez algorytm LPT zastosowany do rozwiązania problemu $1, h_1|srs|C_{max}$ (czyli problemu z jednym okresem niedostępności), przy czym błąd ten został wyrażony jako funkcja parametru α. Wyniki przeprowadzonych analiz wskazują, że badane algorytmy mogą w najgorszym przypadku popełniać spore błędy, co stanowi motywację do podjęcia prób konstrukcji bardziej dokładnych algorytmów. W następnym rozdziale zaproponowany zostanie tego typu algorytm oparty na metaheurystyce tabu search.

3. Algorytm przybliżony

Metaheurystyka tabu search okazuje się być bardzo skuteczną metodą rozwiązywania wielu problemów szeregowania zadań. Stąd, podjęto próbę opracowania opartego na niej algorytmu przybliżonego rozwiązującego problem rozważany w niniejszej pracy. Opracowany algorytm przedstawiony jest na poniższym listingu.

Algorithm 1 Algorytm przybliżony

```
1:  s_0 ← losowe uszeregowanie
2:  s_Best ← s_0
3:  s_LocalBest ← s_0
4:  TabuList ← [ ]
5:  while TabuList.size < maxTabuSize do {
6:      s_BestPair ← s_LocalBest
7:      while True do {
8:          for all T_j ∈ T do {
9:              for all T'_j ∈ T do {
10:                 s_Temp ← s_LocalBest po zamianie miejscami zadań: T_j i T'_j
11:                 if C_max(s_Temp) < C_max(s_BestPair) then
12:                     s_BestPair ← s_Temp
13:             }
14:         }
15:         if C_max(s_BestPair) < C_max(s_LocalBest) then
16:             s_LocalBest ← s_BestPair
17:         else Break
18:     }
19:     if C_max(s_LocalBest) < C_max(s_Best) then
20:         s_Best ← s_LocalBest
21:     H ← podzbiór kilku kolejnych zadań o największej długości
22:     TabuList.add(H)
23:     s_LocalBest ← s_LocalBest po przesunięciu zadań ze zbioru H na koniec uszeregowania
24: }
```

Początkowo jako rozwiązania startowe zastosowane zostały sekwencje losowe oraz wygenerowane za pomocą reguł SPT i LPT. Nie miały one jednak większego wpływu na wynik końcowy, dlatego na początku algorytmu zadania zostały uszeregowane

losowo. Funkcja przeszukiwania lokalnego została opisana w algorytmie w wierszach od 7 do 18. Polega ona na sprawdzeniu dla każdej pary zadań, czy ich zamiana miejscami będzie skutkować najlepszym lokalnym uszeregowaniem. Funkcja ta jest powtarzana do momentu znalezienia ostatniej pary zadań wpływającej na wynik uszeregowania. Jeżeli istnieje uszeregowanie lepsze od wcześniej znalezionego optimum lokalnego $C_{max}(s_{LocalBest})$, to w wierszu 16 jest ono ustawiane jako nowe uszeregowanie $s_{LocalBest}$. W przeciwnym wypadku algorytm w wierszu 17 wychodzi z pętli poszukującej lokalne optimum.

Pierwsze wyniki eksperymentów obliczeniowych pozwoliły zauważyć, że największy wpływ na koszt związany ze wznowieniem zadania po okresie niedostępności mają najdłuższe zadania. Dlatego w badanym algorytmie lista tabu jest budowana z najdłuższych zadań do uszeregowania. W każdej iteracji kolejne zadania z listy są przenoszone na koniec uszeregowania, a następnie ponownie uruchamiana jest funkcja przeszukiwania lokalnego. Po wyczerpaniu możliwości zamiany par zadań w każdej iteracji zwracane jest najlepsze znalezione uszeregowanie $s_{LocalBest}$. Jeżeli jest ono lepsze od poprzedniego, to następuje nadpisanie uszeregowania s_{Best} nowym lokalnym optimum. Parametrem determinującym liczbę wykonywanych powtórzeń opisanej procedury jest $maxTabuSize$. Jest on wykorzystany w wierszu 5, w którym następuje sprawdzenie, czy budowana lista tabu osiągnęła zadeklarowaną wielkość. Jeżeli tak, działanie algorytmu kończy się.

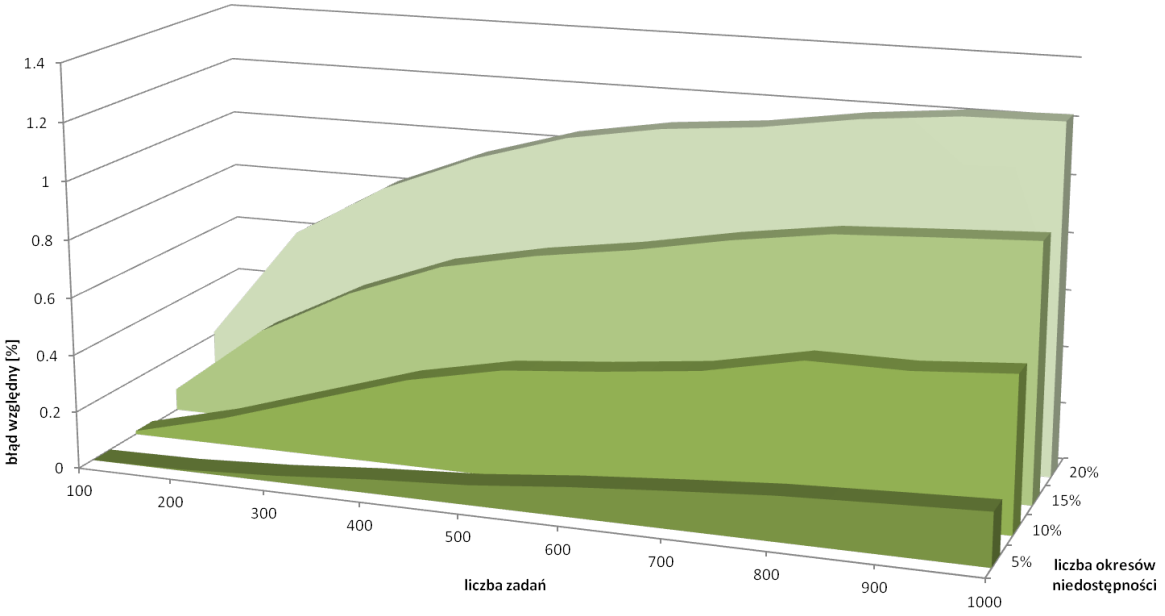
4. Eksperymenty obliczeniowe

W niniejszym rozdziale przedstawione zostaną wyniki eksperymentów obliczeniowych, które zostały przeprowadzone w celu zbadania jakości opisanego algorytmu. Na potrzeby eksperymentów wygenerowane zostały następujące instancje:
- czasy wykonywania zadań p_j zostały wylosowane z przedziału $[10, 100]$,
- liczby zadań są równe $100, 200, ..., 1000$,
- liczba okresów niedostępności wynosi: 5%, 10%, 15% i 20% liczby zadań,
- przedstawione wyniki są wartością średnią z wykonania algorytmu dla 100 losowo wygenerowanych instancji problemu z każdego przedziału.

Eksperymenty przeprowadzono dla trzech różnych wartości $\alpha = 0.25, 0.5, 0.75$.

Ze względu na złożoność przedstawionego problemu, sprawdzenie jakości wyników uzyskanych przez opisany algorytm jest mocno problematyczne. Spowodowane jest to tym, że osiągnięcie dokładnych wyników dla większej liczby zadań jest bardzo czasochłonne. Dlatego, na potrzeby eksperymentów obliczeniowych, wyniki otrzymane za pomocą zaproponowanego algorytmu przybliżonego zostały porównane z dolnym ograniczeniem. Było nim uszeregowanie początkowe, przy założeniu, że zadania są wznawialne. W przypadku zadań wznawialnych dowolne uszeregowanie jest optymalne, a zatem jego długość jest nie większa niż długość najlepszego uszeregowania zadań semi-wznawialnych.

Przyjmując dla pierwszego eksperymentu wartość parametru $\alpha = 0.25$ zakładamy, że niewielki procent wykonanej części zadania musi zostać powtórzony. Otrzymane wyniki przedstawiono na wykresie zamieszonym na rysunku 1, który pokazuje błąd procentowy w zależności od liczby zadań oraz liczby okresów niedostępności. Nawet w najgorszym przypadku, przy największej liczbie okresów niedostępności wynik nie



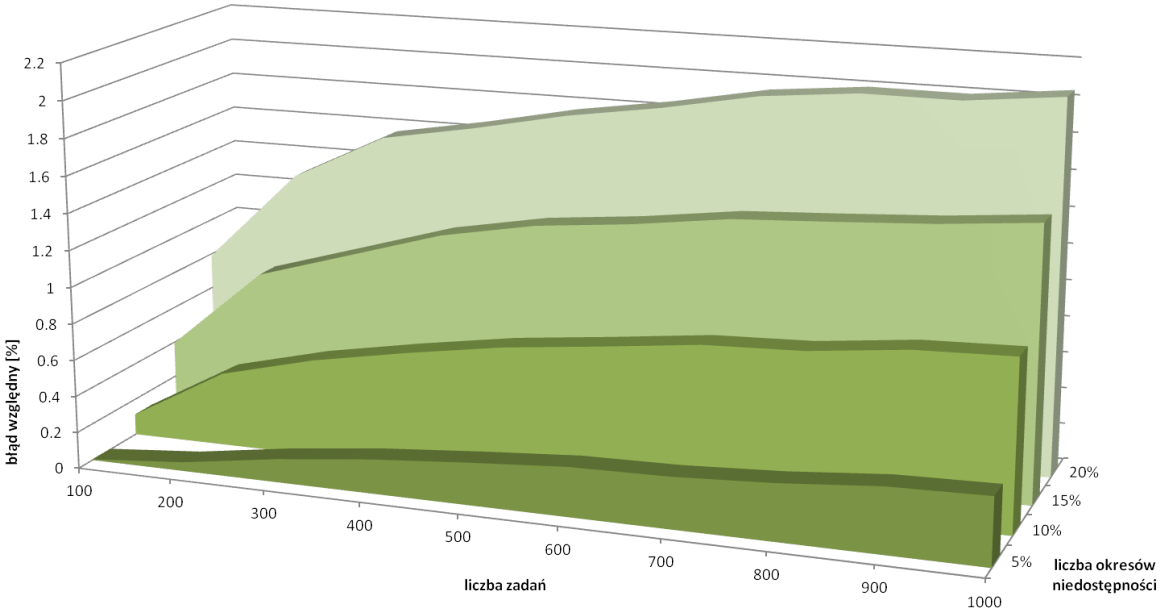
Rys. 1. Średni błąd procentowy algorytmu przybliżonego w stosunku do dolnego ograniczenia dla $\alpha = 0.25$.

odbiega od dolnego ograniczenia o więcej niż 1.2%. Błąd rośnie wraz ze wzrostem liczby zadań, co ma bezpośredni związek z większą przestrzenią poszukiwań. Algorytm tabu niezależnie od wielkości instancji powtarza swoje kroki określoną w parametrze $maxTabuSize$ liczbę razy. W związku z tym dla większej liczby zadań mniejsza liczba istniejących optimów lokalnych dla danej instancji zostanie znaleziona przez algorytm.

W celu pokazania wpływu parametru α na jakość wygenerowanych rozwiązań na rysunkach 2 oraz 3 przedstawiono wyniki dla zwiększonego $\alpha = 0.5$ oraz $\alpha = 0.75$. W porównaniu do pierwszego wykresu na rysunku 1 wyniki pogorszyły się dla niektórych przedziałów prawie dwukrotnie. W najgorszym przypadku jednak są one gorsze od dolnego ograniczenia o ponad 2%. Zebrane wyniki pozwalają stwierdzić, że dobrze zostało wybrane dolne ograniczenie, które jest punktem odniesienia dla badania jakości naszego algorytmu. Dodatkowo pomimo dość dobrych wyników nawet w przypadku większej liczby zadań i okresów niedostępności można rozszerzyć badany algorytm o kolejne kroki, które w dalszym stopniu zredukują uzyskany poziom błędu.

5. Podsumowanie

W ramach niniejszej pracy poruszono temat szeregowania zadań semi-wznawialnych na jednej maszynie z wieloma okresami niedostępności. Problem ten jest

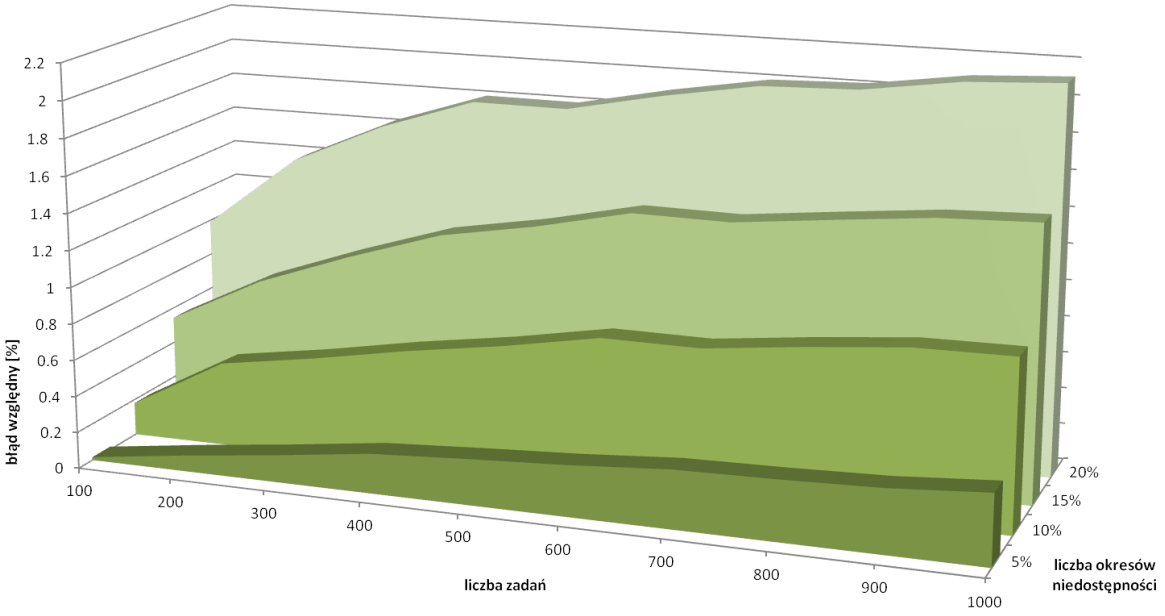


Rys. 2. Średni błąd procentowy algorytmu przybliżonego w stosunku do dolnego ograniczenia dla $\alpha = 0.5$.

obliczeniowo trudny, stąd zaproponowny został algorytm przybliżony oparty o metodę tabu search. Uzyskane wyniki porównano z wynikami dowolnego uszeregowania dla przypadku zadań wznawialnych, które posłużyło jako dolne ograniczenie. Na tej podstawie wyliczono błąd względny, który w najgorszym z badanych przypadków nieznacznie przekraczał 2%. W ramach dalszych prac nad badanym problemem można rozważyć rozbudowę zaproponowanego algorytmu, zbadanie kolejnych kryteriów optymalności, jak również zbadanie problemów szeregowania zadań z innymi rodzajami semiwznawialności.

LITERATURA

1. Formanowicz P. Szeregowanie zadań w systemach z ograniczoną dostępnością procesorów. Rozprawa doktorska. Politechnika Poznańska, Wydział Elektryczny, Poznań, 2000.
2. Chung-Yee Lee Machine scheduling with an availibility constraint. Journal of Global Optimization, 1996, 9, p. 395–416
3. Maa Y., Chu Ch., Zuo Ch. A survey of scheduling with deterministic machine availability constraints. Computers & Industrial Engineering, 2010, 58, p. 199–211.



Rys. 3. Średni błąd procentowy algorytmu przybliżonego w stosunku do dolnego ograniczenia dla $\alpha = 0.75$.

4. Schmidt G. Scheduling with limited machine availability. European Journal of Operational Research, 2000, 121, p. 1–15.

5. Ying M., Shan-lin Y., Cheng-bin Ch. Minimizing Makespan in Semiresumable Case of Single-Machine Scheduling with an Availability Constraint. Systems Engineering – Theory & Practice, 2009, 29, p. 128–134.