PROCEDURY ODPORNEJ ALOKACJI ZASOBÓW DLA PROBLEMU HARMONOGRAMOWANIA PROJEKTU Z WAŻONYMI KOSZTAMI NIESTABILNOŚCI [1]

Marcin KLIMEK, Piotr ŁEBKOWSKI

Streszczenie: Odporne harmonogramowanie projektu jest ważnym zagadnieniem często podejmowanym w ostatnich latach. Artykuł przedstawia problem odpornej alokacji zasobów dla zagadnienia harmonogramowania projektu z ważonymi kosztami niestabilności poszczególnych zadań. Zaprezentowany jest przegląd algorytmów alokacji zasobów, których zastosowanie może wpłynąć na wzrost odporności realizowanego harmonogramu.

Słowa kluczowe: odporna alokacja zasobów, harmonogramowanie projektu z ograniczoną dostępnością zasobów, ważone koszty niestabilności.

1. Wprowadzenie

Problem harmonogramowania projektu z ograniczoną dostępnością zasobów *RCPSP* (ang. *Resource Constrained Project Scheduling Problem*) jest przedmiotem wielu prac badawczych. Duże zainteresowanie zagadnieniem wynika z coraz częstszego realizowania zleceń produkcyjnych, konstrukcyjnych w ramach projektów.

Jako projekty realizuje się prace w sektorze prac publicznych, w sektorze budowlanym, w przemyśle lotniczym i obronnym itp. Możliwe jest także zastosowanie metod zarządzania projektami w sterowaniu produkcją na zamówienie *MTO* (ang. *Make-To-Order*). Produkcja na zlecenie *MTO* jest wykorzystywana przede wszystkim w produkcji małoseryjnej, przy wytwarzaniu wyrobów niestandardowych, dla odbiorcy indywidualnego. Zlecenie produkcyjne w *MTO* jest realizowane w konsultacji z klientem, którego wymagania bywają zmienne i nieprzewidywalne.

Realizacja projektu często obarczona jest niepewnością związaną z błędnymi oszacowaniami czasów trwania zadań, z czasową niedostępnością zasobów (awariami maszyn), ze zmiennością wymagań klientów, itp. Jednym ze skutków występowania nieprzewidzianych zakłóceń produkcyjnych może być nieterminowa realizacja zadań (mniejsza stabilność produkcji), całego projektu i związane z tym koszty finansowe opóźnień (np. koszty ponoszonych kar umownych, koszty przestoju maszyn, koszty przedłużonego składowania materiałów itp.).

W celu zmniejszenia negatywnego wpływu zaburzeń produkcyjnych na realizację projektu stosowane jest podejście proaktywne (ang. *proactive scheduling*), zwane też harmonogramowaniem odpornym (ang. *robust scheduling*) [3]. Harmonogram proaktywny definiuje się jako uszeregowanie zadań, które ze względu na swoje właściwości, jest niepodatne na zakłócenia pojawiające się w trakcie realizacji projektu. W szczególności uszeregowanie to powinno minimalizować skutki wzrostów czasów trwania czynności spowodowanych przez niekontrolowane czynniki. Stopień odporności uporządkowaniu zadań to jego zdolność do absorbowania zaburzeń produkcyjnych pojawiających się

[1] Praca finansowana przez Narodowe Centrum Nauki (nr projektu: N N519 645940)

w trakcie realizacji harmonogramu.
W badaniach rozpatrywane są dwa rodzaje odporności [8]:

- odporność jakości harmonogramu (ang. *quality robustness*) – podejście, w którym dąży się do zapewnienia terminowości realizacji całego projektu, proaktywne planowanie ma prowadzić do minimalizacji odchylenia planowanego terminu wykonania od rzeczywistego czasu wykonania projektu,
- odporność szczegółów harmonogramu (ang. *solution robustness*) – podejście, w którym dąży się do zapewnienia stabilności wykonywania poszczególnych zadań, do realizowania wszystkich szczegółów harmonogramu zgodnie z planem, proaktywne planowanie ma prowadzić do np. minimalizacji odchyleń czasów rozpoczęcia czynności pomiędzy planowanym a zrealizowanym harmonogramem lub do minimalizacji liczby przesuniętych zadań w zrealizowanym uszeregowaniu w porównaniu z planowanym itp.

Przy tworzeniu proaktywnego harmonogramu powinno się uwzględniać oba typy odporności. Jednak część metod koncentruje się wyłącznie na jednym z rodzajów odporności (np. w metodzie łańcucha krytycznego zabezpieczana jest terminowość realizacji całego projektu, bez uodparniania realizacji poszczególnych zadań).

Dla problemu harmonogramowania projektu z ograniczonymi zasobami uodparnianie uszeregowania może być wykonywane w dwóch etapach optymalizacyjnych: alokacji (rozdziale) zasobów i alokacji buforów [2, 3, 9, 10]. W etapie odpornego rozdziału zasobów alokowane są zasoby do realizacji poszczególnych zadań. Natomiast odporna alokacja buforów wykonywana jest przy ustalonej alokacji zasobów do zadań i polega na wstawianiu buforów czasowych i/lub zasobowych np. po zadaniach najbardziej narażonych na zakłócenia produkcyjne lub przed zadaniami, których nieterminowe rozpoczęcie jest najbardziej kosztowne.

W niniejszym artykule przedstawione są zagadnienia związane z alokacją zasobów i wpływem tej alokacji na odporność uszeregowania. Przedstawione są: model matematyczny problemu, zasady odpornej alokacji, stosowane miary odporności i algorytmy proaktywnego rozdziału zasobów dla problemu *RCPSP* z ważonymi kosztami niestabilności.

2. Sformułowanie problemu odpornej alokacji zasobów dla harmonogramowania projektu z ważonymi kosztami niestabilności

Problem planowania projektu z ważonymi kosztami niestabilności to zmodyfikowany model konwencjonalnego, grafowego problemu *RCPSP* często rozpatrywany w badaniach dotyczących harmonogramowania projektu w warunkach niepewności.

Projekt w *RCPSP* można przedstawić jako sieć z czynnościami na węzłach *AON* (ang. *Activity On Node*). Sieć *AON* to acykliczny, spójny, prosty graf skierowany $G(V, E)$, w którym V to zbiór węzłów (zadań, czynności), a E to zbiór łuków przedstawiających relacje kolejnościowe między czynnościami. Zbiór V składa się z n+2 zadań ponumerowanych od 0 do n+1 w porządku topologicznym, tzn. poprzednik ma numer niższy od następnika. Węzeł 0 to wierzchołek początkowy a węzeł n+1 to wierzchołek końcowy grafu $G(V, E)$. Węzły 0 i n+1 nie przedstawiają rzeczywistych czynności, dodawane są jedynie w celu odpowiedniego graficznego przedstawienia projektu.

Czynności realizowane są z wykorzystaniem ograniczonych zasobów odnawialnych. Liczba zasobów jest stała w czasie i wynosi a_k (dla typów zasobu k = 1, …, K; gdzie K –

liczba typów zasobów). W każdym momencie czasu t wykorzystanie zasobów nie może przekraczać dostępnych wielkości a_k:

$$\sum_{i \in A(t)} r_{ik} \leq a_k \qquad \forall t, \forall k \tag{1}$$

gdzie:

r_{ik} – zapotrzebowanie czynności i na zasób typu k,

$A(t)$ – zbiór czynności wykonywanych w przedziale czasu $[t\text{-}1, t]$.

Między czynnościami występują relacje kolejnościowe typu *finish-start zero-lag precedence*, w których następnik może rozpocząć się bez zwłoki po zakończeniu czynności poprzedzającej:

$$s_i + d_i \leq s_j \qquad \forall (i, j) \in E \tag{2}$$

gdzie:

s_i – planowany czas rozpoczęcia zadania i,

d_i – planowany czas realizacji zadania i.

Przy tak zdefiniowanych ograniczeniach kolejnościowych i zasobowych tworzony jest harmonogram bazowy przez określenie czasów rozpoczęcia zadań $s_1,\ldots,s_{n+1}$ uwzględniający kryterium oceny, funkcję celu harmonogramowania nominalnego. Dla deterministycznego problemu *RCPSP* najczęściej stosowaną funkcją celu jest minimalizacja czasu realizacji projektu. Takie kryterium oceny jest też stosowane przy generowaniu uszeregowania bazowego w badaniach dla problemu z ważonymi kosztami niestabilności [2, 3].

W trakcie wykonania projektu planowe rozpoczynanie czynności na podstawie przyjętego harmonogramu bazowego często jest niemożliwe ze względu na zakłócenia produkcyjne (np. niekorzystne warunki atmosferyczne, awarie maszyn, czasowa niedostępność zasobów, problemy z dostawami materiałów itp.), które wpływają na zmienność czasów trwania zadań. Zagadnienie proaktywnego harmonogramowania projektu sprowadzane jest do minimalizacji ważonego kosztu niestabilności przy niepewnych czasach trwania zadań (w badaniach stosowane są np. stochastyczne czasy realizacji zadań, generowane z rozkładu β [2, 3, 9, 10]):

$$\min \left(\sum_{i=1}^{n+1} w_i \mid s_i^R - s_i \mid \right) \tag{3}$$

gdzie:

s_i^R – rzeczywisty (lub ustalony w drodze symulacji) moment rozpoczęcia zadania i,

w_i – waga zadania i, koszt niestabilności przypadający na jednostkę czasu związany z nieterminowym rozpoczęciem czynności i.

Problem *RCPSP* z funkcją celu określoną wzorem (3) można sprowadzić do terminowej realizacji całego projektu (kryterium odporności jakości harmonogramu) przy przyjęciu $w_1,\ldots, w_n = 0$ i $w_{n+1} \neq 0$. Natomiast przyjmując $w_1,\ldots, w_n = w_{n+1} \neq 0$ – problem sprowadza się do zabezpieczenia wszystkich czynności przed możliwymi zakłóceniami (kryterium stabilności harmonogramu, odporności szczegółów uszeregowania).

Koszty niestabilności (wagi) dla poszczególnych zadań mogą wynikać z dodatkowych kosztów magazynowania materiałów, z kar umownych za nieterminową realizację prac, itp. W badaniach wagi generowane są losowo. Większa waga generowana jest dla czynności końcowej n+1 (np. 10-krotnie [2, 3, 9]) ze względu na przyjęcie założenia, że największy jest koszt nieterminowości wykonania całego projektu (moment rozpoczęcia czynności końcowej s_{n+1} jest równy czasowi zakończenia projektu).

Dany harmonogram bazowy można zrealizować przy różnych przydziałach zasobów do zadań, które mogą różnić się pod względem odporności na zaburzenia produkcyjne [9]. Ustalenie alokacji dla problemu *RCPSP* jest ważnym i aktualnym zagadnieniem w planowaniu projektów. Jest zagadnieniem silnie *NP*-trudnym, już przy jednym typie zasobów. Do opisu problemu rozdziału zasobów można zastosować sieć przepływu zasobów (ang. *resource flow networks*) [1]. Zmiennymi decyzyjnymi podczas alokacji są przepływy zasobów $f(i, j, k)$, które określają liczbę przekazywanych zasobów dla każdego typu zasobu k od kończonego zadania i do rozpoczynanego zadania j. Sieć przepływu zasobów składa się z węzłów i łuków z oryginalnej sieci $G(V, E)$ oraz ze zbioru dodatkowych łuków (tworzących zbiór E_R) łączących ze sobą wszystkie pary węzłów (zadań), między którymi występują przepływy zasobów $f(i,j,k)>0$. Zbiór E_R zawiera łuki symbolizujące przepływ zasobów, ale tylko te dodatkowe łuki, które nie występują w oryginalnej sieci $G(V, E)$ a wynikają z ustalonego przydziału zasobów.

Problem alokacji zasobów można sformułować przez dodanie ograniczeń (4, 5) do zagadnienia *RCPSP* opisanego wzorami (1-3) [2, 3, 9, 10]:

- dla każdego typu zasobu k suma wszystkich zasobów wychodzących z czynności początkowej jest równa sumie tych zasobów wchodzących do czynności końcowej i jest równa całkowitej dostępności zasobu typu k:

$$\sum_{j\in V} f(0,j,k) = \sum_{j\in V} f(j,n+1,k) = a_k \qquad \forall k \in K \tag{4}$$

- dla każdego typu zasobu k suma wszystkich zasobów danego typu wchodzących do danego węzła i (czynności i), jest równa sumie tych zasobów wychodzących z tego węzła i wynosi r_{ik} (zapotrzebowanie czynności i na zasób typu k):

$$\sum_{j\in V} f(i,j,k) = \sum_{j\in V} f(j,i,k) = r_{ik} \qquad \forall i \in V \setminus \{0,n+1\}, \forall k \in K \tag{5}$$

3. Zasady odpornej alokacji zasobów

Celem odpornej alokacji zasobów jest minimalizacja zmian w harmonogramie spowodowanych wzrostami czasów trwania zadań. Każdy dodatkowy łuk w zbiorze E_R to nowe ograniczenie kolejnościowe, które zmniejsza odporność harmonogramu. Tworzą się tzw. punkty synchronizacji, w których rozpoczęcie danej czynności uzależnione jest od zakończenia innych czynności (wydłużenie czasu realizacji poprzednika powoduje opóźnienie rozpoczęcia następnika).

W pracach badawczych dotyczących odpornego przydziału zasobów analizowany jest problem minimalizacji liczby łuków dodatkowych [3, 5, 6, 9, 10, 11], problem maksymalizacji sumy przepływów między poszczególnymi zadaniami [3]. Dla problemu z ważonymi

kosztami niestabilności minimalizowany jest wpływ możliwych zakłóceń produkcyjnych [3, 9, 10].

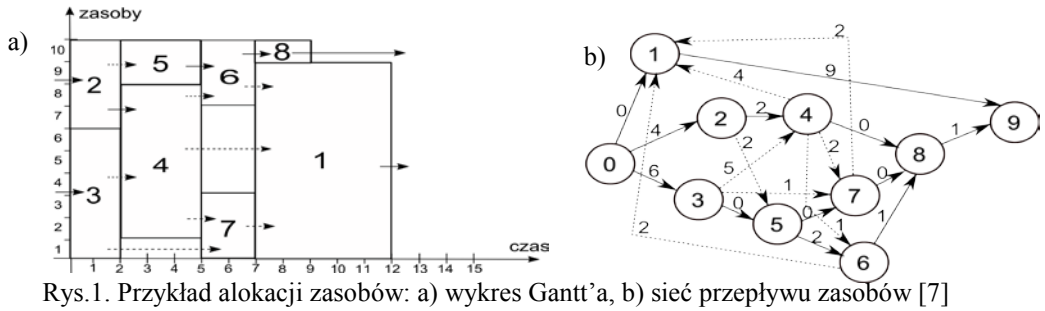
Zagadnienie odpornej alokacji zasobów dla problemu z ważonymi kosztami niestabilności najlepiej można przedstawić na przykładzie liczbowym. W tabeli 1 umieszczone są informacje o analizowanym, przykładowym projekcie [7]. Projekt ten jest wykonywany przez jeden typ zasobu o dostępności równej 10 i składa się z 10 czynności (uwzględniając węzły: początkowy i końcowy oznaczone numerami 0 i 9).

Tab. 1. Przykładowy projekt złożony z 10 zadań realizowany przy użyciu jednego typu zasobu o dostępności równej 10 [7]

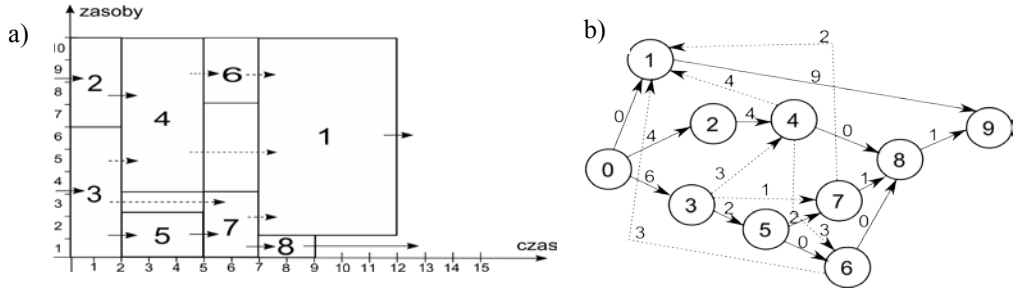
Nr zad. i	Czas trwania d_i	Bezpośrednie następniki zadania i	Zapotrzebowanie na zasób r_i	Koszt niestabilności (waga) w_i
0	0	1, 2, 3	0	0
1	5	9	9	5
2	2	4	4	3
3	2	5	6	4
4	3	8	7	4
5	3	6, 7	2	6
6	2	8	3	8
7	2	8	3	2
8	2	9	1	10
9	0	-	0	50

Minimalny czas trwania analizowanego projektu wynosi 12 jcz. np. przy harmonogramie nominalnym z czasami rozpoczęcia poszczególnych zadań równymi: s_1=7, s_2=0, s_3=0, s_4=2, s_5=2, s_6=5, s_7=7, s_8=7, s_9=12. Dla tak zdefiniowanego uszeregowania bazowego możliwe jest znalezienie wielu alokacji zasobów o różnym stopniu odporności.

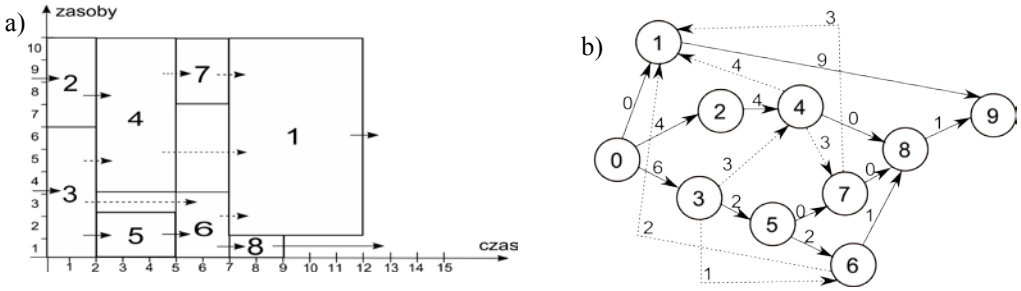
Na rysunkach 1-3 przedstawione są wykresy Gantt'a i sieci przepływu zasobów (przy strzałkach wpisane są liczby określające przepływ zasobów $f(i,j)$) dla trzech różnych przydziałów zasobów do zadań. Przerywanymi strzałkami oznaczone są dodatkowe łuki (tworzące zbiór E_R) wynikające z alokacji zasobów do zadań. Łuki, które występują w każdej z możliwych alokacji zasobów to tzw. łuki nieuniknione. Dla rozważanego harmonogramu nominalnego są to łuki: (3,4), (4,1), (6,1), (7,1). Dany łuk (i,j) jest nieunikniony, gdy liczba dostępnych zasobów (nie uwzględniając zasobów, które zrealizowały czynność i), w momencie rozpoczynania zadania j ($t = s_j$), jest mniejsza niż zapotrzebowanie na zasoby zadania j [3].



Rys.1. Przykład alokacji zasobów: a) wykres Gantt'a, b) sieć przepływu zasobów [7]



Rys.2. Alokacja zasobów z minimalną liczbą łuków dodatkowych: a) wykres Gantt'a,, b) sieć przepływu zasobów [7]



Rys.3. Alokacja zasobów z minimalną liczbą łuków dodatkowych uwzględniająca koszt niestabilności zadań: a) wykres Gantt'a, b) sieć przepływu zasobów [7]

W alokacji zasobów zaprezentowanej na rysunku 1, poza łukami nieuniknionymi, występują wygenerowane są dodatkowe zależności kolejnościowe między zadaniami (2,5), (3,7), (4,6) i (4,7), które można teoretycznie usunąć przy innej alokacji zasobów. Możliwa jest alokacja zasobów z eliminacją łuku (2,5) oraz (4,6) lub (4,7) [7]. Zatem minimalna liczba łuków dodatkowych dla analizowanego harmonogramu bazowego wynosi 6. Taka liczba elementów w zbiorze E_R występuje w rozdziałach zasobów zaprezentowanych na rysunkach 2-3. Może istnieć wiele alokacji zasobów o minimalnej liczbie łuków dodatkowych, które różnią się jakością przy uwzględnieniu wiedzy o kosztach niestabilności poszczególnych zadań.

Dla rozpatrywanego zagadnienia minimalizacji ważonego kosztu niestabilności korzystniejsza jest alokacja zasobów z rysunku 3, w której występuje łuk dodatkowy (4,7) zamiast łuku (4,6), ponieważ dla czynności 7 zdefiniowany jest niższy koszt niestabilności niż dla zadania 6 (opóźnienie rozpoczęcia czynności 7 spowodowane wydłużeniem zadania 4 jest mniej kosztowne niż opóźnienie rozpoczęcia zadania 6).

Analiza prac badawczych i powyższego przykładu liczbowego wskazuje, że odporny przydział zasobów można osiągnąć przez [1-7,9-11]:

- realizację czynności, między którymi są zależności kolejnościowe, przez te same zasoby,
- maksymalizację przepływów między poszczególnymi czynnościami uwzględniając tzw. łuki nieuniknione,
- minimalizację liczby łuków dodatkowych,
- w danym momencie czasowym w pierwszej kolejności przydział zasobów niewykorzystywanych we wcześniejszym momencie czasowym.

4. Wybrane miary odporności alokacji zasobów

Przy ocenie odporności harmonogramu z alokacją zasobów stosuje się miary specyficzne dla problemu alokacji zasobów (np. uwzględniające liczbę łuków dodatkowych) lub wykorzystuje się wskaźniki stabilność uszeregowania obliczane symulacyjnie przy losowym generowaniu zakłóceń produkcyjnych np. zmian w czasach trwania zadań.

Prostą miarą odporności jest elastyczność *flex* (ang. *flexibity*) [1] liczona jako stosunek liczby par zadań w sieci przepływu zasobów, między którymi nie ma zależności kolejnościowych, do liczby wszystkich możliwych par zadań w projekcie. Wraz ze wzrostem wskaźnika *flex* maleje stopień zależności między czynnościami i rośnie odporność harmonogramu. Maksymalizacja wskaźnika *flex* prowadzi do minimalizacji liczby łuków dodatkowych.

Miernik *flex* nie bierze pod uwagę m.in. wiedzy o kosztach niestabilności poszczególnych czynności (jest identyczny dla alokacji zasobów z rysunków 2 i 3) i nie powinien być stosowany w analizowanym problemie [7]. Dla zagadnienia minimalizacji kosztów niestabilności proponowane są miary odporności dostosowane do tego problemu. Jednym z możliwych podejść jest obliczanie ważonego kosztu niestabilności w drodze eksperymentów obliczeniowych przy różnych scenariuszach przebiegu produkcji generowanych losowo na podstawie wiedzy statystycznej o czasach realizacji zadań (np. czasy trwania generowane są z rozkładu β [2, 3, 9]).

Wadą podejścia symulacyjnego jest czasochłonność. Dodatkowo nie zawsze dostępna jest wiarygodna wiedza statystyczna o czasach realizacji zadań projektowych. Występują np. problemy z oszacowaniem czasu trwania czynności nowych, niepowtarzalnych, innowacyjnych. Wskazane jest opracowanie miar odporności rozdziału zasobów, które uwzględniają wpływ wzrostu czasu trwania poszczególnych czynności na koszty niestabilności i nie są wyznaczane symulacyjnie.

Autorzy proponują zastosowanie do oceny sieci przepływu zasobów wskaźnika *stab* [6, 7]. Wskaźnik *stab* to suma ważonych kosztów niestabilności ustalonych przy założeniu, że każdorazowo jedna z czynności j (przy uwzględnieniu kolejno wszystkich czynności $j = 1 \ldots n$) jest wydłużona o 1 jednostkę czasu (pozostałe czynności mają czasy trwania równe planowanym) [7]:

$$stab = \sum_{j=1}^{n} \{ \sum_{i=1}^{n+1} [w_i \cdot (s_i^j - s_i)] \} \tag{6}$$

gdzie:

s_i^j – czas rozpoczęcia czynności i przy wzroście czasu trwania zadania j o 1 jednostkę czasu.

Przy obliczaniu wskaźnika *stab* można założyć także np. procentową zmienność czasów trwania zadań i przyjąć wzrost czasu trwania zadań nie o 1 jednostkę czasu ale o m % (czas trwania wydłużanego zadania j wynosi $d_j \cdot (1+m\%)$).

Celem alokacji zasobów jest minimalizacja wskaźnika *stab*. Przy alokacji z mniejszą wartością *stab* uszeregowanie jest bardziej odporne na zakłócenia: nieznaczne wzrosty czasów trwania zadań mają mniejszy wpływ na koszt niestabilności realizowanego harmonogramu (opóźnienia innych zadań).

Wskaźnik *stab* dla alokacji zasobów z rysunku 2 wynosi 490 a dla alokacji z rysunku 2 jest równy 478 (obliczenia w artykule autorów [7]) co potwierdza, że wykorzystanie tego wskaźnika może być użyteczne dla zagadnienia z minimalizacją ważonego kosztu niestabilności. Rozdziały zasobów zaprezentowane na rysunkach 2 i 3 mają identyczną liczbę łuków dodatkowych równą 6 i nie są rozróżnialne za pomocą miary *flex*, natomiast na podstawie miernika *stab* można stwierdzić, że preferowana jest alokacja zasobów o niższej wartości *stab* przedstawiona na rysunku 3.

5. Wybrane algorytmy alokacji zasobów

W najprostszej procedurze alokacji zasobów tworzone są osiągalne sieci przepływu zasobów, bez uwzględniania odporności uszeregowania [1]. W kolejnych momentach rozpoczynania poszczególnych zadań znajdowane są dostępne w tej chwili zasoby. Jeśli w analizowanym momencie czasowym więcej niż jedna czynność jest rozpoczynana, zadania są przydzielane w kolejności wynikającej z przypisanych im numerów (niższy numer, zadanie wcześniej przydzielane do zasobów).

Część z algorytmów alokacji zasobów stosuje koncepcje łańcuchów [3-5,11]. Schemat działania tych algorytmów przedstawiony jest na rysunku 4.

```
Posortuj wszystkie zadania rosnąco na podstawie ich czasów rozpoczęcia
Zainicjuj wszystkie łańcuchy dla zasobów jako puste
for (dla każdego typu zasobów k) do
  for (dla każdej kolejnej czynności j) do
    for 1 to r_jk do   //Przydziel czynność j do r_jk łańcuchów
      l := selectChain(j, k)
      Dodaj na koniec łańcucha l czynność j
```

Rys. 4. Schemat działania algorytmu alokacji zasobów wykorzystującej koncepcję łańcuchów

Koncepcja łańcuchów stosowana jest w procedurach *Basic Chaining, ISH* (ang. *Iterative Sampling Heuristic*), *ISH*^{2}. Poszczególne algorytmy różnią się metodą wybierania łańcucha *selectChain*(*j*, *k*). W procedurze *Basic Chaining* zadania przydzielane są do pierwszych wolnych łańcuchów związanych z kolejnymi zasobami. Wygenerowana w ten sposób alokacja zasobów jest nieodporna na zakłócenia przez tworzenie, często nadmiarowo, łuków dodatkowych zwiększających zbiór E_R.

Zmniejszenie liczby elementów w zbiorze E_R to cel heurystyki *ISH*, która sprowadza się do przydzielania zadań o zapotrzebowaniu na dany typ zasobu większym niż jeden, w taki sposób, aby zmaksymalizować liczbę wspólnych łańcuchów, z ostatnimi czynnościami aktualnie znajdującymi się w dostępnych łańcuchach. Dla danej czynności *j* i typu zasobu *k* losowany jest jeden z dostępnych łańcuchów *l*. W przypadku gdy $r_{jk} > 1$, zadanie *j* jest przydzielane w pierwszej kolejności do łańcuchów, w których ostatnim elementem jest czynność ostatnia w wylosowanym łańcuchu *l*.

Procedura *ISH* nie uwzględnia relacji kolejnościowych występujących w oryginalnej sieci projektu, co może prowadzić do powstawania nadmiarowych zależności

kolejnościowych. Relacje poprzedzań uwzględnia heurystyka ISH^2. W procedurze tej przy ustalaniu łańcucha dla danej czynności *j* w pierwszej kolejności przydzielane są łańcuchy, w których ostatnia czynność jest bezpośrednim poprzednikiem czynności *j*. Gdy zapotrzebowanie na zasoby analizowanej czynności *j* jest większe niż poprzedników tej czynności pozostałe łańcuchy są wybierane zgodnie z procedurą *ISH*.

Kolejne modyfikacje heurystyk *ISH*, które mają na celu zwiększenie odporności uzyskanego rozmieszczenia zasobów, są opracowane przez autorów [4, 5]. W pierwszej proponowanej wersji algorytmu *ISH*, określonej jako *ISH-UA*, procedura działa jak algorytm ISH^2, przy czym na początku uruchamiana jest procedura znajdowania łuków nieuniknionych. W pierwszej kolejności zadanie *j* jest przydzielane do łańcuchów, w których ostatnia czynność jest bezpośrednim poprzednikiem zadania *j* lub jest połączona z nim łukiem nieuniknionym. W drugiej proponowanej wersji heurystyki *ISH*, określonej jako *ISH-R*, na początku zliczane są liczby wystąpień poszczególnych czynności jako ostatnich zadań na liście dostępnych łańcuchów. Czynność *j* jest przydzielana do wspólnych łańcuchów z czynnością o liczbie wystąpień równej zapotrzebowaniu na zasoby czynności *j* (w dalszej kolejności liczbie wystąpień większej o jeden od tego zapotrzebowania itd.). Wybierane są takie łańcuchy, aby liczba punktów synchronizacji była jak najmniejsza przez przydzielanie wspólnych zasobów do zadań powiązanych relacjami kolejnościowymi lub łukami nieuniknionymi. Proponowane algorytmy *ISH-UA* oraz *ISH-R* są skuteczniejsze od innych prostych heurystyk tj. *Basic Chaining, ISH*, ISH^2dla problemu minimalizacji liczby łuków dodatkowych [5].

Zagadnienie odpornej alokacji zasobów można zapisać także jako problem programowania całkowitoliczbowego ze zmiennymi decyzyjnymi zapisanymi w postaci wartości całkowitych dodatnich (często zerojedynkowych). Dla problemu alokacji zasobów *RCPSP* znane są następujące procedury optymalizacji całkowitoliczbowej tj. [3]:

- procedura *MinEA* (ang. *Minimize Extra Arcs*), w której minimalizowana jest liczba łuków dodatkowych;
- procedura *MaxPF* (ang. *Maximize sum of Pairwise Floats*), w której maksymalizowana jest suma przepływów zasobów między czynnościami;
- procedura *MinED* (ang. *Minimize Estimated Disruptions*), w której minimalizowany jest ważony koszt niestabilności.

Czas działania procedur programowania całkowitoliczbowego jest znacznie większy niż heurystyk takich jak *ISH*. Algorytmy programowania całkowitoliczbowego nie generują rozwiązań w akceptowalnym czasie dla problemów większych, np. złożonych z 90 zadań.

Dla problemu harmonogramowania projektu z ważonymi kosztami niestabilności stosowany jest także algorytm alokacji zasobów *MABO* [3] (ang. *Myopic Activity-Based Optimization*). Jest to procedura z „krótkowzroczną" optymalizacją oparta na analizie wpływu alokacji kolejno rozważanych zadań na wskaźnik stabilności (łączny ważony koszt niestabilności). Wpływ ten jest analizowany dla uszeregowania z częściowym rozdziałem zasobów wykonanym do momentu rozpoczęcia czynności, której aktualnie przydzielane są zasoby. Algorytm *MABO* składa się z trzech kroków przeprowadzanych dla wszystkich zadań. Czynności rozpatrywane są w kolejności ich czasów rozpoczęcia w harmonogramie nominalnym (dla zadań o identycznym czasie rozpoczęcia kryterium rozstrzygającym jest szacowany wpływ na koszt niestabilności danego zadania). W kroku 1 sprawdzany jest warunek czy dla aktualnie rozważanej czynności *i* całe zapotrzebowanie na zasoby może być zaspokojone przez zasoby, w których ostatnia czynność jest bezpośrednim poprzednikiem czynności *i*. Jeśli zapotrzebowanie na zasoby zadania *i* jest większe niż jej poprzedników, w kroku 2 dodawany jest nowy łuk (lub łuki) E_R. Analizowane są wszystkie możliwe łuki do dodania (wszystkie przepływy zasobów między zadaniami bez zależności kolejnościowych) i wybierany jest łuk (lub łuki) z minimalnym wpływem na ważony koszt

niestabilności aż do „zaspokojenia" pełnego zapotrzebowania czynności *i*. Ten wpływ ustalany jest w drodze symulacji obliczeniowych dla częściowego harmonogramu. W ostatnim kroku określane są przepływy zasobów *f*(*j, i, k*), od czynności *j* do *i.* Czynności *j* to zadania poprzedzające czynność *i* w oryginalnej sieci projektu oraz zadania ustalone w kroku 2.

W oparciu o koncepcje zastosowane w procedurze *MABO* i w heurystykach *ISH* autorzy proponują algorytm *RA*1, którego schemat działania przedstawiony jest na rysunku 5 [4].

```
Znajdź łuki nieuniknione
Stwórz listę zadań LRA uporządkowanych rosnąco wg ich czasów
rozpoczęcia, przy równych czasach rozpoczęcia zastosuj
kryterium dodatkowe sortowania Ks
for (dla każdej kolejnej czynności i z listy LRA) do
  for(dla każdego typu zasobu k, dla którego r_ik > 0) do
    Wyczyść zbiór PA
    Dodaj do PA pośrednie i bezpośrednie poprzedniki zad. i,
    które są ost. elementem w dostępnych łańcuchach zasobów
    w momencie s_i //PA = {j: (j,i)€E∪E_R}
    while (suma dostępnych zasobów z PA < r_ik)
      Z czynności spoza PA, które są ostatnim elementem
      w dostępnych łańcuchach zasobów w momencie s_i wybierz
      losowo czynność l i dodaj tą czynność l do zbioru PA

      Przydziel zasoby do zad.i ustalając przepływy f(j,i,k)
      przy zastosowaniu kryterium KsPA dla zadań z PA
```

Rys. 5. Schemat działania algorytmu alokacji zasobów *RA*1 [4]

W pierwszym kroku procedury *RA*1 szukane są łuki nieuniknione, które dodawane są do zbioru E_R. Następnie tworzona jest lista *LRA* zadań posortowanych rosnąco na podstawie ich czasów rozpoczęcia w harmonogramie bazowym. Przy identycznych czasach rozpoczęcia stosowane są dodatkowe kryteria porządkowania *Ks*1-*Ks*4:

- *Ks*1: sortowanie malejące wg zapotrzebowania czynności na zasób danego typu,
- *Ks*2: sortowanie rosnące wg zapotrzebowania czynności na zasób danego typu,
- *Ks*3: porządkowanie malejące według sumy zapotrzebowań zadań na zasoby wszystkich typów,
- *Ks*4: losowa kolejność zadań.

Kryterium rozstrzygającym o kolejności przy równych wartościach kryterium *Ks* jest minimalny numer zadania.

W następnym kroku *RA*1 alokowane są zasoby do wszystkich kolejnych zadań rozważanych w kolejności ich występowania na liście *LRA*. Przydział zasobów dla zadania *i* polega na wyznaczeniu w kolejnych momentach czasu $t = s_i$ przepływu zasobów od czynności, które zwolniły dostępne zasoby w chwili *t*, do czynności *i*. W tym celu do pustego zbioru *PA* dodawane są pośrednie i bezpośrednie poprzedniki czynności *i*, które są ostatnim elementem w dostępnych łańcuchach w chwili *t*. Jeśli suma zasobów zwolnionych

przez czynności ze zbioru *PA* jest mniejsza niż zapotrzebowanie na zasoby zadania *i*, szukane są brakujące zasoby. Spośród zadań spoza zbioru *PA*, które są ostatnim elementem w dostępnych łańcuchach zasobów w chwili *t*, wybierane są losowo czynności, które dodawane są do zbioru *PA*, aż do momentu, gdy zapotrzebowaniu na dany zasób zadania *i* jest równe lub mniejsze od liczby zwalnianych łańcuchów przez czynności ze zbioru *PA*.

Następnym krokiem jest wyznaczenie dla czynności *i* przepływów zasobów *f*(*j, i, k*). Rozpatrywane są wszystkie czynności *j* należące do *PA*. Każdy łuk (*j, i*), który nie należy do zbioru $E_U E_R$ jest dodawany do zbioru łuków dodatkowych E_R i uwzględniany przy alokacji dla innych typów zasobów.

Kolejność rozpatrywania zadań (i alokowanych zwalnianych przez nie zasobów) ze zbioru *PA* przy alokacji zasobów do aktualnej czynności *i* zależy od przyjętego kryterium *KsPA*. Przykładowe kryteria porządkowania zadań w zbiorze *PA*:

- *KsPA*1: czynności w *PA* uporządkowane są malejąco na podstawie zapotrzebowania na zasoby danego typu, przy czym na początku znajdują się zadania będące bezpośrednimi lub pośrednimi poprzednikami zadania *i*,
- *KsPA*2: czynności w *PA* uporządkowane są rosnąco na podstawie zapotrzebowania na zasoby danego typu, przy czym na początku znajdują się zadania będące bezpośrednimi lub pośrednimi poprzednikami zadania *i*,
- *KsPA*3: czynności wybierane są w sposób losowy: na początku spośród zadań będących bezpośrednimi lub pośrednimi poprzednikami czynności *i*, w dalszej kolejności losowo z pozostałych zadań.

Autorzy proponują również procedurę *RALS* z lokalnym przeszukiwaniem alokacji zasobów [4]. Początkowe rozwiązania ustalane jest przy użyciu algorytmu *ISH-R*. Przebieg procedury *RALS* jest zbliżony do algorytmu *RA*1. Lokalne poszukiwania dotyczą kolejności rozpatrywania zadań ze zbioru *PA*, w momentach czasowych, w których rozpoczyna jest większa liczba zadań niż 1. W *RALS* kolejność ta jest zmieniana w wyniku wykonywania ruchów ograniczonych do zadań znajdujących się w zbiorze *PA* (mogą być wykonywane ruchy stosowane przy reprezentacji permutacyjnej np.: zamień, wstaw, zamień sąsiednie itp.). Wykonanie ruchu zmienia uporządkowanie czynności na liście *PA* i zgodnie z tym porządkiem tworzona jest alokacja zasobów dla aktualnego zadania *i* (w pierwszej kolejności przydzielane są zasoby zwalniane przez zadania będące bezpośrednimi lub pośrednimi poprzednikami zadania *i*). Ta alokacja zasobów jest porównywana z aktualnie najlepszą. Kryterium oceny jest stosowana miary odporności (może to być wskaźnik *flex* lub *stab*). Lepszy przydział zasobów jest zapamiętywany i modyfikowany w następnej iteracji.

Algorytmy *RA*1 i *RALS* są skuteczne dla problemu harmonogramowania projektu ze zdefiniowanymi kamieniami milowymi [4]. Nie była do tej pory testowana ich efektywność dla zagadnienia harmonogramowania projektu z ważonymi kosztami niestabilności.

6. Podsumowanie

W artykule przedstawiono zagadnienia odpornej alokacji zasobów dla problemu harmonogramowania projektu z ograniczonymi zasobami. Opisano często analizowany w ostatnich latach problem harmonogramowania z ważonymi kosztami niestabilności w warunkach niepewności. W piśmiennictwie polskim liczba opracowań dotyczących tego aktualnego zagadnienia nie jest duża a planowanie projektów z uwzględnieniem niepewności oraz minimalizacją kosztów stanowi aktualny nurt badań w zakresie zarządzania projektami.

Liczba prac dotyczących odpornej alokacji zasobów, zwłaszcza w języku polskim, jest niewielka, a zagadnienie jest istotne z praktycznego punktu widzenia. Brak jest również

prac przeglądowych dotyczących algorytmów odpornej alokacji zasobów dla problemu *RCPSP* z ważonymi kosztami niestabilności. Poza analizą znanych rozwiązań w artykule zaprezentowano także procedury opracowane przez autorów. Dalsze prace autorów skoncentrują się na porównaniu zaproponowanych i znanych rozwiązań przy wykorzystaniu bibliotek testowych dla problemu *RCPSP*.

Literatura

1. Artigues C., Michelon P., Reusser S.: Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research 149(2), 2003, 249-267.
2. Braeckmans K., Demeulemeester E., Herroelen W., Leus R.: Proactive resource allocation heuristics for robust project scheduling, Raport badawczy KBI_0567, K.U.Leuven, 2005.
3. Deblaere F., Demeulemeester E.L., Herroelen W.S., Van De Vonder S.: Proactive resource allocation heuristics for robust project scheduling. Raport badawczy KBI_0608, K.U.Leuven, 2006.
4. Klimek M.: Predyktywno-reaktywne harmonogramowanie produkcji z ograniczoną dostępnością zasobów, Praca doktorska, AGH Kraków, 2010.
5. Klimek M., Łebkowski P.: Algorytmy odpornej alokacji zasobów dla problemu harmonogramowania projektu z ograniczoną dostępnością zasobów. Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, t. 13 z. 2, 2009, 371-379.
6. Klimek M., Łebkowski P.: Resource allocation for robust project scheduling. Bulletin of the Polish Academy of Sciences Technical Sciences, Vol. 59, No. 1, 2011, 51-55.
7. Klimek M., Łebkowski P.: Alokacja zasobów dla problemu harmonogramowania projektu z ważonymi kosztami niestabilności. Automatyka: półrocznik AGH w Krakowie, t. 15, z. 2, 2011, s. 237-245.
8. Kobylański P., Kuchta D.: A note on the paper by M. A. Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling, International Journal of Production Economics, 107, s.496-501, 2007.
9. Leus R.: The generation of stable project plans. Praca doktorska, Katolicki Uniwersytet Lowański, Belgia, Leuven, Katholieke Universiteit Leuven, Belgia, 2003.
10. Leus R, Herroelen W.: Stability and resource allocation in project planning. IIE Transactions, 36(7), 2004, 667–682.
11. Policella N., Oddi A., Smith S., Cesta A.: Generating robust partial order schedules. In Proceedings of CP2004, Toronto, Canada, 2004.

Dr inż. Marcin KLIMEK
Instytut Informatyki
Państwowa Szkoła Wyższa
21-500 Biała Podlaska, ul. Sidorska 95/97
e-mail: marcin_kli@interia.pl

Dr hab. inż. Piotr ŁEBKOWSKI, prof. AGH
Katedra Badań Operacyjnych i Technologii Informacyjnych
Wydział Zarządzania
Akademia Górniczo-Hutnicza
30-059 Kraków, al. Mickiewicza 30
e-mail: plebkows@zarz.agh.edu.pl