ALGORYTM PRIORYTETOWY HARMONOGRAMOWANIA PROJEKTU PRZY OGRANICZONYCH ZASOBACH

Marcin KLIMEK, Piotr ŁEBKOWSKI

Streszczenie: W artykule opisano algorytm priorytetowy dla problemu harmonogramowania projektu z ograniczoną dostępnością zasobów RCPSP (ang. *Resource-Constrained Project Scheduling Problem*) ze zdefiniowanymi umownymi etapami projektu tzw. kamieniami milowymi. Porównano efektywność znanych reguł priorytetowych, stosowanych w badaniach dotyczących RCPSP, z zasadami priorytetowania dedykowanymi dla rozpatrywanego zagadnienia.

Słowa kluczowe: algorytmy priorytetowe, kamienie milowe, projekt, harmonogramowanie.

1. Wprowadzenie

Przedsiębiorstwa produkcyjne coraz częściej stosują tzw. produkcję na zlecenie (ang. *MTO – Make-To-Order*). Zamówienia produkcyjne wykonywane są w ramach projektów (przedsięwzięć) realizowanych w konsultacji z klientem (odbiorcą). Jako projekty realizowane są m.in.: produkcja wyrobów niestandardowych, wykonywanie zleceń konstrukcyjnych (np. budowa dróg, statków) czy rozwojowych (np. tworzenie systemów informatycznych).

Harmonogramowanie projektu obarczone jest niepewnością związaną z unikalnością realizowanych zadań i ze zmiennością wymagań odbiorców. W fazie planowania mogą występować trudności z określeniem terminu realizacji poszczególnych etapów prac lub całego przedsięwzięcia. W celu zmniejszenia ryzyka niepowodzenia całego projektu, klienci często wymagają od wykonawców (zwłaszcza przy dużych projektach) określenia punktów kontroli przebiegu prac tzw. kamieni milowych. Definiowane są nieprzekraczalne terminy ich realizacji, a ewentualne opóźnienia mogą pociągać za sobą kary umowne. Mogą także doprowadzić do wycofania się klienta z zamówienia. Zadaniem planisty jest stworzenie harmonogramu, który jest maksymalnie zabezpieczony przed przekroczeniem każdego z etapów projektu.

W niniejszej pracy zaproponowano matematyczny model problemu harmonogramowania projektu z ograniczoną dostępnością zasobów RCPSP ze zdefiniowanymi, nieprzekraczalnymi terminami realizacji zadań związanych z kamieniami milowymi. Zdefiniowano funkcją celu uwzględniającą zabezpieczenie terminowego wykonania wszystkich etapów projektu. Następnie dla tego modelu przetestowano skuteczność działania algorytmu priorytetowego dla opracowanych reguł priorytetowych.

2. Sformułowanie problemu

Projekt – to unikalny zbiór współzależnych czynności (zadań, operacji) realizowany w ramach określonych zasobów (pracowników, maszyn, materiałów). Czynność – to część projektu, stanowiąca odrębną całość, do jej realizacji wymagana jest określona ilość czasu

i zasobów. Zadania są niepodzielne (ang. *nonpreemptive scheduling*) i istnieje tylko jeden sposób ich wykonania (ang. *single-mode*).

Projekty reprezentowane są jako acykliczny, spójny, prosty graf skierowany *G(V, E)*, w którym *V* oznacza zbiór węzłów odpowiadający czynnościom, a *E* to zbiór łuków opisujących relacje poprzedzań. Jest to reprezentacja tzw. sieć czynności (ang. *AON – Activity-On-Node*) [1].

Pomiędzy czynnościami występują relacje typu koniec-początek bez zwłoki (ang. *finish-start, zero-lag precedence*), w których następnik może rozpocząć się bezzwłocznie po zakończeniu zadania poprzedzającego [2]:

$$s_i + d_i \leq s_j \qquad \forall (i, j) \in E \quad , \tag{1}$$

gdzie: s_i – czas rozpoczęcia zadania *i*,
d_i – czas wykonywania zadania *i*.

Dla zrealizowania czynności niezbędne są zasoby, które równocześnie stanowią ograniczenie w realizacji projektu. Zasoby są odnawialne (ang. *renewable*) tzn. ich dostępność jest stała w kolejnych okresach czasu. To ograniczenie można zapisać wzorem (2) [2]:

$$\sum_{i \in S_t} r_{ik} \leq a_k \qquad \forall t, \forall k \quad , \tag{2}$$

gdzie: a_k – liczba dostępnych zasobów typu *k*,
S_t – zbiór zadań wykonywanych w przedziale czasu *[t-1, t]*,
r_{ik} – zapotrzebowanie czynności *i* na zasób typu *k*.

Do tak opisanego problemu RCPSP autorzy proponują zdefiniować umowne punkty kontroli przebiegu prac tzw. kamienie milowe. Do każdego zadania *i* przypisany jest etap projektu z nim związany i na tej podstawie określony jest nieprzekraczalny termin zakończenia tego zadania δ_i [3]:

$$z_i \leq \delta_i \tag{3}$$

gdzie: z_i – czas zakończenia czynności *i*.

W celu zdefiniowania funkcji celu konieczne jest określenie zbiorów zadań koniecznych do realizacji umownego etapu projektu. Niech zbiór zadań bezpośrednio związanych z danym kamieniem milowym km_i będzie oznaczony symbolem M_i. M_i zawiera wszystkie czynności o identycznym nieprzekraczalnym terminie zakończenia δ_j:

$$M_i = \left\{ j : \delta_j = tm_i, j \in V \right\}, \tag{4}$$

gdzie: tm_i – termin realizacji kamienia milowego km_i.

Zbiory M_i są rozłączne. Każda z czynności należy tylko do jednego ze zbiorów M_i.

Niech KM_i to zbiór wszystkich czynności, których wykonanie jest niezbędne do realizacji danego kamienia milowego km_i (zbiór KM_i zawiera dodatkowo wszystkie poprzedniki czynności ze zbioru M_i):

$$KM_i = \{j : j \in M_i \vee j \in P_k, k \in M_i\}, \tag{5}$$

gdzie: P_k – zbiór wszystkich poprzedników (nie tylko bezpośrednich) czynności k.

Z praktycznego punktu widzenia, w związku z niepewnością występującą przy realizacji projektu, użyteczne jest znalezienie takiego harmonogramu zadań, aby maksymalnie zabezpieczyć terminową realizację wszystkich kamieni milowych [3].

Proponowaną funkcją celu F, uwzględniającą zabezpieczenie terminowego wykonania wszystkich etapów projektu, jest ważona suma poziomu zabezpieczenia kamieni milowych pb_i (obliczanych według wzoru (6)) określona wzorem (7).

$$pb_i = \frac{rez_i}{\sum_{\forall j \in KM_i} d_j} \tag{6}$$

$$F = \sum_{i=1}^{m} (pb_i \cdot wm_i), \tag{7}$$

gdzie: wm_i – waga przypisana kamieniowi milowemu km_i,
rez_i – różnica między nieprzekraczalnym terminem kontrolnym tm_j a najwcześniejszym możliwym terminem wykonania wszystkich czynności ze zbioru KM_i.

Wartość wagi wm_i zależy od aktualnego poziomu zabezpieczenia kamienia milowego km_i i ustalana jest na podstawie posortowanej rosnąco według pb_i listy kamieni milowych, którym kolejno przypisuje się malejące wm_i. W pracy przyjęto jako wm_i kolejne liczby naturalne od m do 1. Kamieniom milowym można przypisać inne wagi, przy czym powinien być spełniony warunek: większa waga wm_i dla mniej zabezpieczonych km_i.

3. Algorytmy harmonogramowania

Problem harmonogramowania projektu z ograniczoną dostępnością zasobów *RCPSP*, jako uogólnienie klasycznego problemu *job shop*, jest zadaniem silnie *NP-trudnym*. Dla *NP-trudnych* problemów optymalizacyjnych stosuje się algorytmy dokładne (dla problemów o mniejszych rozmiarach) i przybliżone (dla problemów większych). Algorytmy dokładne poprzez umiejętny przegląd wszystkich dopuszczalnych rozwiązań zawsze generują rozwiązania optymalne, ale ze względu na wykładniczą złożoność obliczeniową, mogą być stosowane jedynie dla problemów o niewielkich rozmiarach. Czas poszukiwań rozwiązania optymalnego dla większych problemów (a takie występują w rzeczywistych systemach produkcyjnych) często jest nieakceptowalny w praktyce.

Większe praktyczne znaczenie ma znalezienie efektywnych algorytmów przybliżonych. Algorytmy przybliżone posiadają najczęściej wielomianową złożoność obliczeniową. W związku z tym ich czas obliczeń jest znacznie mniejszy niż metod dokładnych

o wykładniczej złożoności obliczeniowej. Procedury przybliżone nie dają gwarancji znalezienia rozwiązania optymalnego, choć wyszukanie takiego rozwiązania jest możliwe. Jakość uzyskiwanych rozwiązań (dokładność przybliżeń) zależy od liczby analizowanych przez nie uszeregowań, czyli właściwie od czasu obliczeń.

Wśród algorytmów przybliżonych (heurystycznych) można wyróżnić dwie grupy algorytmów: algorytmy konstrukcyjne oraz algorytmy lokalnych poszukiwań (ang. *LS – Local Search*). Obie grupy mają znaczenie praktyczne. Algorytmy konstrukcyjne są przydatne do generowania rozwiązań początkowych dla bardziej efektywnych algorytmów lokalnych poszukiwań tj. algorytmy symulowanego wyżarzania (ang. *SA – Simulated Annealing*), przeszukiwanie z zakazami (ang. *TS – Tabu Search*), algorytmy genetyczne (ang. *GA - Genetic Algorithms*) itd.

Algorytmy konstrukcyjne budują rozwiązanie problemu w oparciu o proste mechanizmy priorytetowania (algorytmy priorytetowe) czy wstawiania zadań (algorytmy wstawień). Wykorzystywanie algorytmów konstrukcyjnych, jest niewystarczające, gdyż prowadzi często do harmonogramów znacznie odbiegających od optymalnych. Te uszeregowania są jednak na tyle dobre, że są bardzo przydatne jako rozwiązania startowe dla algorytmów lokalnych poszukiwań. Ich zaletą jest przede wszystkim szybki czas działania. Ich wadą jest fakt, że najczęściej są one dedykowane dla konkretnych problemów optymalizacyjnych.

W następnym rozdziale szerzej opisano algorytmy priorytetowe, które są przedmiotem analizy w tej pracy.

4. Algorytmy priorytetowe

Algorytmy priorytetowe tworzą harmonogram w oparciu o priorytety zadań określane przy pomocy różnych reguł priorytetowych, specyficznych dla danego zagadnienia. Lista priorytetowa określa ważność poszczególnych zadań. Na jej podstawie, za pomocą procedury dekodującej SGS (ang. *SGS – Schedule Generation Scheme*), tworzony jest harmonogram, czyli określane są terminy rozpoczęcia poszczególnych czynności. Stosowane są następujące schematy generowania harmonogramu SGS [4]:

- szeregowy SGS (ang. *serial SGS*) – w każdej iteracji rozpoczynana jest pierwsza nieuszeregowana czynność z listy priorytetowej, w najwcześniejszym możliwym terminie rozpoczęcia przy spełnieniu ograniczeń kolejnościowych i zasobowych,
- równoległy SGS (ang. *parallel SGS*) – iteracyjnie, w kolejnych momentach czasu *t* (w punktach decyzyjnych), rozpoczynane są wszystkie nieuszeregowane czynności, które mogą być rozpoczęte w kolejności wynikającej z listy priorytetowej przy spełnieniu ograniczeń kolejnościowych i zasobowych.

Szeregowy i równoległy SGS tworzą harmonogram spełniający ograniczenia zasobowe i kolejnościowe, niezależnie od kolejności zadań na liście priorytetowej.

Dla rozważanego problemu opracowano klasyczny algorytm priorytetowania: tworzona jest lista zadań ustawionych w kolejności wynikającej z zastosowanej reguły priorytetowej, która następnie poddana jest procedurze *SGS*.

Algorytm priorytetowy do ustalenia kolejności na liście priorytetowej wykorzystuje tzw. reguły priorytetowania (ang. *priority rules*). Znalezienie odpowiednich reguł priorytetowych, jest kluczowe dla efektywności tych algorytmów. Konieczne jest opracowanie zasad ustalania kolejności zadań dla problemu *RCPSP* z kamieniami milowymi. Przy ich definiowaniu wykorzystywane będą pojęcia stosowane w analizie czasowej problemów szeregowania tj.:

– ES_i (ang. *Earliest Start*) – najwcześniejszy możliwy czas rozpoczęcia zadania *i*,
– EF_i (ang. *Earliest Finish*) – najwcześniejszy możliwy czas zakończenia zadania *i*,
– LS_i (ang. *Latest Start*) – najpóźniejszy możliwy czas rozpoczęcia zadania *i*,
– LF_i (ang. *Latest Finish*) – najpóźniejszy możliwy czas zakończenia zadania *i*.

Pojęcia te są bezpośrednio związane z szukanym harmonogramem. Zachodzą nierówności:

$$ES_i \le s_i \le LS_i, \qquad \forall i \in V \tag{8}$$

$$EF_i \le z_i \le LF_i, \qquad \forall i \in V \tag{9}$$

Najwcześniejsze możliwe czasy rozpoczęcia ES_i i zakończenia EF_i znajdowane są rekurencyjnie wykonując obliczenia kolejno dla czynności od *0* do *n+1*. Można określić następujące zależności (identyczne jak dla problemu *RCPSP* bez kamieni milowych):

$$ES_0 = 0 \tag{10}$$

$$ES_i = \max_{j \in ZP_i}(ES_j + d_j), \qquad \forall i \in V \tag{11}$$

$$EF_i = ES_i + d_i, \qquad \forall i \in V \tag{12}$$

gdzie: ZP_i - zbiór bezpośrednich technologicznych poprzedników czynności *i*.

Dla rozważanego problemu, analiza dotycząca, najpóźniejszego możliwego czasu rozpoczęcia LS_i i zakończenia LF_i, musi uwzględniać system kamieni milowych. Najpóźniejszy możliwy czas zakończenia zadania nie może przekraczać terminu zakończenia etapu projektu z nim związanego. Musi być również przeanalizowany minimalny czas realizacji wszystkich następników (nie tylko bezpośrednich) danej czynności, koniecznych do wykonania związanego z nią kamienia milowego. Analiza czasowa wykonywana jest rekurencyjnie kolejno od czynności *n+1* do *1*. Zachodzą następujące zależności:

$$LS_{n+1} = \max(ES_{n+1}, \delta_{n+1}) \tag{13}$$

$$LF_i = \min(\min_{j \in ZN_i}(LF_j - d_j), \delta_i, EF_i), \qquad \forall i \in V \tag{14}$$

$$LS_i = LF_i - d_i, \qquad \forall i \in V \tag{15}$$

gdzie: ZN_i - zbiór bezpośrednich technologicznych następników czynności *i*.

Poniżej, w tabeli 1 opisano znane reguły (reguły R1-R11), wykorzystywane w badaniach dotyczących problemu *RCPSP* [5], które będą zastosowane dla rozpatrywanego zagadnienia. W kolumnie 2 zapisano wzory do obliczania priorytetów poszczególnych

zadań. Dla celów porównawczych dodano regułę R0, w której poszczególnym zadaniom przypisywane są losowe priorytety zadań.

Tab. 1. Znane reguły priorytetowe

Reguła priorytetowa	Wzór na priorytet zadania *i*	Opis reguły
R0	$r_0(i) = random$	Losowe priorytety zadań
R1	$r_1(i) = -ES_i$	Minimalny najwcześniejszy czas rozpoczęcia zadania
R2	$r_2(i) = -LS_i$	Minimalny najpóźniejszy czas rozpoczęcia zadania
R3	$r_3(i) = -LF_i$	Minimalny najpóźniejszy czas zakończenia zadania
R4	$r_4(i) = -EF_i$	Minimalny najpóźniejszy czas zakończenia zadania
R5	$r_5(i) = -(LS_i - ES_i)$	Minimalna różnica między najpóźniejszym a najwcześniejszym możliwym czasem rozpoczęcia czynności (ang. *MINSLK - Minimum Slack*)
R6	$r_6(i) = -(LF_i - EF_i)$	Minimalna różnica między najpóźniejszym a najwcześniejszym możliwym czasem zakończenia czynności
R7	$r_7(i) = \#N_i$	Maksymalna liczba wszystkich następników czynności (ang. *MTS – Most Total Successors*)
R8	$r_8(i) = \#ZN_i$	Maksymalna liczba wszystkich bezpośrednich następników czynności (ang. *MITS – Most Immediate Total Successors*)
R9	$r_9(i) = -d_i$	Najkrótszy czas trwania czynności (ang. *SPT – Shortest Processing Time*)
R10	$r_{10}(i) = d_i + \sum_{j \in N_i} d_j$	Maksymalna suma czasów trwania danej czynności i jej wszystkich następników (ang. *MTSPT – Most Total Successors Processing Time*)
R11	$r_{11}(i) = d_i \cdot \sum_{k=1}^{K} r_{ik} + \sum_{j \in N_i} d_j \cdot \sum_{k=1}^{K} r_{jk}$	Maksymalna suma iloczynu czasu trwania i zasobochłonności danej czynności i jej wszystkich następników

Poza znanymi regułami priorytetowymi opracowano zasady priorytetowania wykorzystujące informacje o zbiorach kamieni milowych, dedykowane dla analizowanego problemu (reguły R12-R21) przedstawione w tabeli 2. Powstały one przy wykorzystaniu reguł opisanych w tabeli 1, przy uwzględnieniu dodatkowo wiedzy o umownych etapach projektu: dla każdego zadania uwzględniono nieprzekraczalny termin jego realizacji δ_i.

Jeśli stosując daną regułę priorytetowania zadania mają takie same priorytety, na wcześniejszych pozycjach listy priorytetowej umieszczane są zadania oznaczone niższym numerem.

Tab. 2. Reguły priorytetowe dedykowane dla problemu *RCPSP* z kamieniami milowymi

Reguła priorytetowa	Wzór na priorytet zadania *i*
R12	$r_{12}(i) = -\delta_i$
R13	$r_{13}(i) = -\delta_i \cdot LS_i$
R14	$r_{14}(i) = -\delta_i \cdot ES_i$
R15	$r_{15}(i) = -\delta_i \cdot LF_i$
R16	$r_{16}(i) = -\delta_i \cdot EF_i$
R17	$r_{17}(i) = -\delta_i \cdot (LS_i - ES_i)$
R18	$r_{18}(i) = -\delta_i \cdot (LF_i - EF_i)$
R19	$r_{19}(i) = \frac{\#N_i}{\delta_i}$
R20	$r_{20}(i) = \frac{\#ZN_i}{\delta_i}$
R21	$r_{21}(i) = \frac{d_i + \sum_{j \in N_i} d_j}{\delta_i}$

5. Wyniki badań eksperymentalnych

Badania przeprowadzono przy użyciu 120 instancji testowych z biblioteki *PSPLIB* zaproponowanych w pracy [6] (po 60 dla problemów złożonych z 30 i 120 zadań). Dla każdego problemu testowego zdefiniowano dodatkowo cztery kamienie milowe, określając losowo czynności należące do poszczególnych zbiorów M_i. Następnie określono tm_i – wspólne, nieprzekraczalne terminy realizacji zadań należących do tego samego zbioru M_i. Aby zapewnić kontrole przebiegu prac podczas trwania całego projektu założono, że występują pewne odstępy czasowe między terminami kontrolnymi tm_i. Przyjęto równomierne rozłożenie punktów kontrolnych podczas realizacji projektu:

$$tm_{i+1} - tm_i \approx \frac{tm_{m-1}}{m}, \qquad \forall i \in <0, m), \tag{16}$$

gdzie: tm_{m-1} – termin realizacji całego projektu ($tm_{m-1} = \delta_{n+1}$),
m – liczba kamieni milowych.

Eksperymenty prowadzono na komputerze klasy Pentium z procesorem 1,7 GHz przy użyciu programu zaimplementowanego w języku C# w środowisku Visual Studio.NET. Wyniki obliczeń zestawiono w tabeli 3.

Tab. 3.Wyniki eksperymentów obliczeniowych

Reguła priorytetowa	30 zadań						120 zadań					
	szeregowy SGS			równoległy SGS			szeregowy SGS			równoległy SGS		
	a	b	c	a	b	c	a	B	c	a	b	C
R0	11	25	33,4%	15	27	29,2%	0	11	84,5%	0	17	64,0%
R1	14	27	24,8%	14	30	22,0%	0	7	67,0%	0	13	60,2%
R2	33	56	4,7%	20	49	10,1%	24	60	8,5%	0	51	24,8%
R3	31	56	3,3%	21	50	9,4%	19	60	8,5%	1	51	22,7%
R4	11	24	25,6%	12	27	25,6%	0	8	66,2%	0	14	55,5%
R5	26	54	6,2%	22	49	10,3%	10	58	15,4%	0	49	27,1%
R6	26	54	4,9%	21	50	10,0%	9	58	15,5%	0	51	25,8%
R7	19	39	17,2%	16	37	17,0%	3	26	59,4%	0	26	45,5%
R8	15	33	20,9%	13	35	19,3%	2	21	49,3%	0	30	45,2%
R9	12	24	26,8%	12	29	21,8%	0	9	83,3%	0	15	62,6%
R10	19	34	21,1%	17	37	19,7%	5	24	68,9%	0	30	48,3%
R11	17	33	24,2%	16	35	21,0%	5	24	67,8%	1	28	49,0%
R12	28	55	2,6%	22	50	9,5%	13	60	3,0%	0	55	17,2%
R13	39	57	**1,5%**	23	50	9,13%	36	60	**1,2%**	0	54	20,5%
R14	20	46	12,6%	16	38	17,3%	0	56	20,1%	0	50	25,5%
R15	33	56	**1,6%**	23	50	9,0%	33	60	**0,8%**	1	55	19,4%
R16	18	50	11,0%	16	40	15,9%	1	58	13,5%	0	48	23,6%
R17	28	56	2,5%	20	50	9,5%	13	60	5,2%	0	55	18,8%
R18	29	57	2,3%	24	51	9,0%	12	60	3,5%	0	55	19,0%
R19	15	37	19,4%	13	39	16,8%	3	27	55,8%	0	28	46,8%
R20	15	33	21,0%	13	34	19,4%	3	26	57,4%	0	26	48,0%
R21	15	33	21,1%	13	34	19,6%	3	26	57,4%	0	26	48,0%

a – liczba rozwiązań najlepszych (spośród 60 eksperymentów)
b – liczba rozwiązań lepszych od średniej (spośród 60 eksperymentów)
c – średnie procentowe odchylenie względne od najlepszego znalezionego uszeregowania

Eksperymenty wykazały lepszą przeciętną skuteczność równoległej procedury dekodującej SGS niż szeregowego SGS (o 1,9% mniejsze średnie procentowe odchylenie względne od najlepszego uszeregowania zadania dla problemów 30 zadaniowych i o 0,3% mniejsze odchylenie dla problemów 120 zadaniowych). Równoległy SGS jest jednak bardziej czasochłonny obliczeniowo niż szeregowy SGS: czas generowania harmonogramu dla algorytmu priorytetowego stosującego równoległy SGS był średnio 3,9 krotnie dłuższy niż algorytmu wykorzystującego szeregowy SGS. Poza tym można zauważyć, że szeregowy SGS jest bardziej efektywny niż równoległy SGS dla reguł priorytetowych, które są wydajne w rozważanym zagadnieniu (reguły R2, R3, R5, R6, R12, R13, R15, R17, R18). Natomiast dla reguł nieefektywnych równoległy SGS generuje lepsze harmonogramy, ale i tak dość słabej jakości.

Reguły dedykowane dla rozważanego problemu (uwzględniające informacje o terminach realizacji kamieni milowych) są lepsze niż znane reguły priorytetowe. Spośród znanych reguł priorytetowych skuteczne są te, które przy wyliczaniu korzystają z informacji o kamieniach milowych (reguły R2 z LS_i oraz R3 z LF_i). Najlepsze zasady ustalania kolejności zadań to minimalny najpóźniejszy czas rozpoczęcia (R13, R2) oraz minimalny najpóźniejszy czas zakończenia zadań (R15, R3) przy szeregowej procedurze dekodującej. Analizując wyniki można wyróżnić dwie grupy reguł:

- efektywne (reguły R2, R3, R5, R6, R12, R13, R15, R17, R18) o skuteczności

znacznie przewyższającej regułę R0, generującą losowe priorytety,
- nieefektywne, o skuteczności bliskiej regule R0.

Duże zróżnicowanie wyników wskazuje na korzyści wynikające ze stosowania efektywnych zasad priorytetowania zadań.

6. Zakończenie

W artykule przedstawiono algorytm priorytetowy dla problemu harmonogramowania projektu z kamieniami milowymi. Wyniki testów obliczeniowych pozwoliły znaleźć skuteczne reguły priorytetowe dla rozważanego zagadnienia. Najlepsze rezultaty są osiągane dla reguł dedykowanych dla problemu RCPSP z kamieniami milowymi uwzględniających termin realizacji poszczególnych etapów projektu. Najskuteczniejsze zasady priorytetowania to minimalny najpóźniejszy czas rozpoczęcia oraz minimalny najpóźniejszy czas zakończenia zadań przy stosowaniu szeregowej procedury dekodującej.

Procedury tworzenia harmonogramu w oparciu o listę priorytetową czynności generują uszeregowania najczęściej dalekie od optymalnych, ale mogą być użyteczne jako rozwiązania startowe (bazowe) w innych, bardziej zaawansowanych algorytmach. Wiedza o skuteczności danej reguły priorytetowej może być także wykorzystana w operatorach przeszukiwania przestrzeni rozwiązań. Przedmiotem dalszych badań będzie wykorzystanie najlepszych zasad priorytetowania w algorytmach harmonogramowania metodą wstawień i metaheurystykach.

Literatura

1. Kostrubiec A.: Harmonogramowanie projektów - przegląd modeli. Inżynieria Zarządzania Przedsięwzięciami, Wydawnictwo Politechniki Gdańskiej, Gdańsk 2003, s. 33-52.
2. Van De Vonder S, Demeulemeester E., Herroelen W., Leus R.: The trade-off between stability and makespan in resource-constrained project scheduling, International Journal of Production Research, 44(2), 2006, s. 215-236.
3. Klimek M., Łebkowski P.: Miary odporności harmonogramów. Komputerowo Zintegrowane Zarządzanie (red.) R. Knosala, Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, t. I, Opole 2008, s. 569-577.
4. Kolisch R.: Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research, 90, 1996, s. 320-333.
5. Kolisch, R. Efficient priority rules for the resource-constrained project scheduling problem. Journal of Operations Management 14, 1996, s. 179–192.
6. Kolisch R., Sprecher A.: PSPLIB – a project scheduling library. European Journal of Operational Research 96, 1997, s. 205–216.

Mgr inż. Marcin KLIMEK
Instytut Informatyki
Państwowa Wyższa Szkoła Zawodowa
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