ALGORYTMY WSTAWIEŃ DLA ZAGADNIENIA HARMONOGRAMOWANIA PROJEKTU ZE ZDEFINIOWANYMI KAMIENIAMI MILOWYMI

Marcin KLIMEK, Piotr ŁEBKOWSKI

Streszczenie: W artykule opisano algorytm wstawień dla problemu harmonogramowania projektu z ograniczoną dostępnością zasobów *RCPSP* (ang. *Resource-Constrained Project Scheduling Problem*) ze zdefiniowanymi terminami realizacji umownych etapów projektu tzw. kamieni milowych. Porównano efektywność proponowanych procedur przy zastosowaniu zadań testowych z biblioteki *PSPLIB* (ang. *Project Scheduling Problem LIBrary*).

Słowa kluczowe: algorytmy wstawień, kamienie milowe, harmonogramowanie projektu z ograniczonymi zasobami.

1. Wprowadzenie

W ostatnich latach powstaje wiele prac z harmonogramowania realizacji projektów (*PSP – Project Scheduling Problem*). Wiąże się to z tym, że coraz powszechniej w planowaniu produkcji stosowane są metody związane z obszarem zarządzania przedsięwzięciami i projektami [1]. W niniejszej pracy rozważany jest problem harmonogramowania projektu z ograniczoną dostępnością zasobów *RCPSP*.

W opracowaniu zaproponowany został matematyczny model problemu *RCPSP* ze zdefiniowanymi, nieprzekraczalnymi terminami realizacji etapów projektu. Definiowana została funkcja celu uwzględniająca zabezpieczenie terminowego wykonania wszystkich etapów projektu (kamieni milowych) [2]. Następnie dla tego modelu przetestowano skuteczność działania algorytmów wstawień opracowanych przez autorów, dedykowanych dla zagadnienia harmonogramowania projektu z kamieniami milowymi.

2. Sformułowanie problemu

Projekt to unikalny zbiór współzależnych czynności (zadań, operacji) realizowany dla osiągnięcia przyjętych celów w ramach określonych zasobów (pracowników, maszyn, materiałów) [1]. Zadania są niepodzielne – nie można przerywać ich realizacji i istnieje tylko jeden sposób ich wykonania (ang. *single-mode RCPSP*) tzn. nie występują różne warianty wykonania zadań przy użyciu innych typów zasobów charakterystyczne dla problemu multimodalnego [3].

Projekty przedstawiane są w reprezentacji wierzchołkowej (w sieci czynności) jako acykliczny, spójny, prosty graf skierowany *G(V, E)*, w którym *V* oznacza zbiór węzłów odpowiadający czynnościom, a *E* to zbiór łuków, które opisują relacje kolejnościowe między zadaniami. Zbiór *V* składa się z *n* zadań numerowanych od 1 do *n* w porządku topologicznym, tzn. poprzednik ma zawsze niższy numer od następnika. Do grafu *G(V, E)* dodawane są dwie pozorne czynności: czynność początkowa 0 i czynność końcowa *n*+1,

o zerowych czasach trwania i zerowym zapotrzebowaniu na zasoby, reprezentujące odpowiednio wierzchołek początkowy oraz wierzchołek końcowy tego grafu [1].

Podczas wykonywania projektu występują następujące ograniczenia [4]:

- kolejnościowe – zadania powiązane są relacjami typu koniec-początek bez zwłoki:

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

- zasobowe – do zrealizowania zadań niezbędne są zasoby odnawialne, których ilość jest ograniczona i stała w kolejnych okresach czasu:

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

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

W związku z niepewnością występującą w trakcie realizacji projektu autorzy proponują określić momenty kontroli przebiegu prac tzw. kamienie milowe (mogą to być umowne etapy projektu określone przez zleceniodawcę wykonawcy projektu), które zmniejszą ryzyko niepowodzenia lub nieterminowości realizacji przedsięwzięcia. Poszczególne zadania mają zdefiniowane nieprzekraczalne terminy ich zakończenia [2]:

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

gdzie: z_i – planowany czas zakończenia zadania *i*,
δ_i – nieprzekraczalny czas zakończenia zadania *i*, a właściwie termin realizacji tm_j najbliższego *j*-tego etapu projektu, w którym zadanie *i* musi być już wykonane.

Celem harmonogramowania jest znalezienie czasów rozpoczęcia poszczególnych zadań s_0, s_1, … s_{n+1}, przy spełnieniu ograniczeń zasobowych i kolejnościowych opisanych wzorami (1, 2), dla odpowiednio zdefiniowanego kryterium optymalizacyjnego, którym najczęściej jest minimalizacja czasu realizacji całego projektu. Dla modelu ze zdefiniowanymi kamieniami milowymi funkcja celu powinna uwzględniać nieprzekraczalne terminy realizacji poszczególnych etapów przedsięwzięcia. Z praktycznego punktu widzenia, w związku z niepewnością występującą przy wykonywaniu projektu, wskazane jest znalezienie takiego uszeregowania zadań, które posiada zapasy czasowe maksymalnie zabezpieczające terminową realizację wszystkich etapów projektu.

Jako funkcja celu proponowana jest maksymalizacja *F* – ważonej sumy rezerw czasowych poszczególnych kamieni milowych [2]:

$$F = \sum_{i=1}^{m} rez_i \cdot wm_i \tag{4}$$

gdzie: m – liczba umownych etapów projektu,

wm_i – waga przypisana i-temu kamieniowi milowemu,

rez_i – rezerwa czasowa i-tego kamienia milowego: różnica między terminem zakończenia i-tego etapu projektu tm_i a terminem realizacji wszystkich zadań związanych z tym etapem wyznaczona dla aktualnego harmonogramu.

Przy zastosowaniu funkcji celu F i przy odpowiednio zdefiniowanych wagach wm_i osiąga się równomierne rozłożenie zapasów czasowych dla poszczególnych etapów. Wagi są określane zgodnie z zasadą: większa waga wm_i dla mniej zabezpieczonych terminów realizacji etapu projektu. Wyliczany jest aktualny poziom zabezpieczenia pb_i poszczególnych etapów projektu według wzoru (5):

$$pb_i = \sum_{i=1}^{m} \frac{rez_i}{\sum_{\forall j \in KM_i} d_j} \tag{5}$$

gdzie: KM_i – zbiór wszystkich zadań wykonywanych podczas i-tego etapu projektu.

W tej pracy ustalanie wag wm_i wygląda następująco (dla uporządkowanych kamieni milowych malejąco według pb_i):

- dla kamienia milowego o maksymalnym pb_i: $wm_i = 1$,
- dla kamienia milowego o k-tym pb_i: $wm_i = (m - k)^2$,
- dla kamienia milowego o minimalnym poziomie pb_i: $wm_i = m^2$.

Takie zdefiniowanie wag prowadzi do rozłożenia rezerwy czasowej rez_i proporcjonalnie do wskaźników pb_i i większego zabezpieczenia etapów projektu o większej czasochłonności.

3. Algorytmy wstawień dla problemu harmonogramowania projektu z ograniczonymi zasobami ze zdefiniowanymi kamieniami milowymi

Problem harmonogramowania projektu z ograniczoną dostępnością zasobów *RCPSP* jest zadaniem silnie *NP-trudnym* [5]. Dla *NP-trudnych* problemów optymalizacyjnych czas poszukiwań rozwiązania optymalnego dla większych problemów często jest nieakceptowalny w praktyce. Z tego względu dla projektów złożonych z dużej liczby zadań uzasadnione jest stosowanie efektywnych algorytmów przybliżonych, które posiadają najczęściej wielomianową złożoność obliczeniową i ich czas wykonywania jest znacznie krótszy niż metod dokładnych o wykładniczej złożoności obliczeniowej.

Do algorytmów przybliżonych należą algorytmy konstrukcyjne oraz algorytmy lokalnych poszukiwań. Algorytmy konstrukcyjne są wykorzystywane do generowania rozwiązań początkowych dla bardziej efektywnych algorytmów lokalnych poszukiwań tj. algorytmy genetyczne, symulowanego wyżarzania, przeszukiwania z zakazami itd.

Algorytmy konstrukcyjne generują rozwiązanie w oparciu o proste mechanizmy priorytetowania (algorytmy priorytetowe) lub wstawiania zadań (algorytmy metodą wstawień). Przedmiotem analiz w tej pracy są algorytmy wstawień.

W algorytmach opierających się na metodzie wstawień w fazie wstępnej ustala się początkową listę zadań przez zastosowanie wybranego algorytmu stosującego metodę priorytetową. W fazie zasadniczej tworzy się ciąg *n* permutacji częściowych, zaczynając od permutacji 1-elementowej i kończąc na permutacji *n*-elementowej. Każda kolejna permutacja częściowa jest budowana w oparciu o poprzednią permutację i kolejne zadanie z listy początkowej. Zadanie to jest wstawiane próbnie na różne pozycje w aktualnej permutacji częściowej i ostatecznie jest wstawiane na taką pozycję, aby wartość funkcji celu, była jak najlepsza. Po wstawieniu wszystkich zadań otrzymaną permutację przyjmuje się jako rozwiązanie problemu.

Początkowa lista zadań, sposób pobierania zadań z listy początkowej oraz pozycje w permutacjach częściowych, na które próbnie wstawiane są zadania z listy początkowej, są specyficzne dla konkretnego algorytmu z tej klasy.

Dla problemu *RCPSP* z kamieniami milowymi nie ma opracowanych do tej pory algorytmów konstrukcyjnych metodą wstawień. W niniejszej pracy proponowane są nowe algorytmy konstrukcyjne metodą wstawień [6]. Algorytmy te korzystają częściowo z koncepcji stosowanych przy rozwiązywaniu innych problemów harmonogramowania produkcji tj. permutacyjny system przepływowy (algorytm NEH [7], algorytm Woo i Yim [8]).

W proponowanych procedurach stosowane są specyficzne dla rozpatrywanego zagadnienia zasady budowy listy początkowej zadań oraz definiowane są funkcje celu dla permutacji częściowych. Kolejne zadania z listy początkowej wstawiane są na dowolne pozycje, ale tylko takie, które nie zaburzają ograniczeń kolejnościowych zdefiniowanych w projekcie.

Algorytmy wstawień dla problemu *RCPSP* tworzą harmonogram częściowy (terminy rozpoczęcia poszczególnych czynności) na podstawie aktualnej permutacji częściowej (listy zadań) za pomocą procedur dekodujących (ang. SGS – *Schedule Generation Scheme*). Stosowane są [9]:

- szeregowy SGS (ang. *serial SGS*) – w każdej iteracji rozpoczynana jest pierwsza nieuszeregowana czynność z aktualnej listy zadań, 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 aktualnej listy zadań przy spełnieniu ograniczeń kolejnościowych i zasobowych.

Szeregowa i równoległa procedura dekodująca tworzą harmonogram uwzględniający ograniczenia zasobowe i kolejnościowe, niezależnie od kolejności zadań w permutacji częściowej.

Jako pierwszy algorytm wstawień proponowany jest algorytm W1 [6].

Algorytm W1:

Oznaczenia:

P – lista zadań, z której przy użyciu procedury SGS, powstaje aktualny harmonogram częściowy,

L – lista aktualnie dostępnych zadań nie dodanych do listy *P*, ale takich, których poprzednikami są wyłącznie zadania już umieszczone na liście *P*.

Krok 1:

Umieszczenie na liście *L* wszystkich następników czynności pozornej początkowej 0 posortowanych według określonego priorytetu.

Krok 2:

Pobranie zadania pierwszego na liście *L* (o najwyższym priorytecie) i umieszczenie tego zadania do harmonogramu częściowego. Analizowane zadanie jest próbnie wstawiane na wszystkie możliwe pozycje (nie zaburzając ograniczeń kolejnościowych) w aktualnej liście czynności *P*, z której generowany jest dotychczasowy harmonogram częściowy. Spośród wszystkich możliwych, wybierana jest pozycja wstawienia, dla której uzyskiwane jest najlepsze uszeregowanie, biorąc pod uwagę stosowane kryterium optymalizacyjne harmonogramów częściowych. Częściowy porządek zadań powstaje z aktualnej listy czynności *P* przy zastosowaniu równoległej lub szeregowej procedury SGS.

Krok 3:

Usunięcie z listy *L* wstawionego w kroku 2 zadania i dodanie do tej listy następników tego zadania (wszystkich tych, których wszystkie zadania poprzedzające znajdują się już w harmonogramie częściowym) z zachowaniem porządku zadań wynikającego z określonego priorytetu zadań.

Krok 4:

Powtórzenie kroków 2-3 aż do stworzenia harmonogramu pełnego, złożonego z wszystkich czynności.

Algorytmy W2 i W3 działają podobnie jak algorytm W1. Różnice występują w kroku 2. Poza tym w W2 i W3 kolejność zadań na liście *L* jest dowolna, nie jest stosowane sortowanie zadań zgodnie z określoną regułą priorytetową. W kroku 2 algorytmu W2 testowane jest wstawienie na ostatnią pozycję listy P każdego z zadań znajdujących się aktualnie na liście *L*. Wybierane jest to z zadań, dla którego uzyskuje się najlepszy harmonogram częściowy.

Z kolei algorytm W3 próbnie wstawia każde z zadań z listy *L* na wszystkie możliwe pozycje listy *P* do każdego z zadań znajdujących się aktualnie na liście *L*. Wybierane są zadanie i ta pozycja wstawienia, dla których wygenerowany jest najlepszy harmonogram częściowy. Algorytm W3 ma większą złożoność obliczeniową niż algorytmy W1 i W2.

Ważne dla efektywności algorytmów W1, W2 i W3 jest zdefiniowanie funkcji celu harmonogramów częściowych. Musi ona zapewnić dobrą jakość uszeregowania złożonego z wszystkich zadań mierzoną funkcją celu harmonogramowania nominalnego *F* określoną wzorem (8). Proponowane są następujące kryteria oceny uszeregowań częściowych:

- kryterium K1: maksymalizacja funkcji celu harmonogramowania nominalnego *F*, przy czym procedura SGS tworzy harmonogram dla listy zadań złożonej z wszystkich zadań: kolejno z zadań z aktualnej listy *P* i w dalszej kolejności z pozostałych zadań uporządkowanych według stosowanej reguły priorytetowej,
- kryterium K2: minimalizacja czasu zakończenia wszystkich zadań aktualnie znajdujących się na liście *P*;
- kryterium K3: maksymalizacja funkcji celu harmonogramowania nominalnego *F*, wyliczona dla harmonogramu częściowego złożonego jedynie z zadań aktualnie znajdujących się na liście *P*.

Do zastosowania kryterium K1 oceny harmonogramów częściowych konieczne jest zdefiniowanie zasad porządkowania zadań nie umieszczonych na liście *P*. Z kolei algorytm W1 wykorzystuje reguły priorytetowe do ustalenia kolejności rozpatrywania zadań z listy *L*. Do powyższych celów używane są reguły priorytetowania R0-R21.

W tabeli 1 zaprezentowane są znane reguły (reguły R1-R11), wykorzystywane w badaniach dotyczących problemu *RCPSP* [10], które są stosowane przez autorów dla rozpatrywanego zagadnienia. Dla celów porównawczych dodana jest reguła 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
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
R8	$r_8(i) = \#ZN_i$	Maksymalna liczba wszystkich bezpośrednich następników czynności
R9	$r_9(i) = -d_i$	Najkrótszy czas trwania czynności
R10	$r_{10}(i) = d_i + \sum_{j \in N_i} d_j$	Maksymalna suma czasów trwania danej czynności i jej wszystkich następników
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

Dla rozważanego problemu, analiza, która dotyczy najpóźniejszego możliwego czasu rozpoczęcia LS_i i zakończenia LF_i uwzględnia nieprzekraczalne terminy zakończenia kamieni milowych [11]. Najpóźniejszy możliwy czas zakończenia zadania nie może przekraczać terminu zakończenia etapu projektu z nim związanego.

Poza znanymi regułami priorytetowymi opracowane są zasady priorytetowania wykorzystujące informacje o zbiorach kamieni milowych, dedykowane dla analizowanego problemu (reguły R12-R21) przedstawione w tabeli 2. Reguły te bazują na regułach R1-R10, dodatkowo uwzględniają dla każdego zadania nieprzekraczalny termin jego realizacji δ_i [11].

Tab. 2. Reguły priorytetowe dla problemu *RCPSP* z kamieniami milowymi [11]

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	$r_{21}(i) = \frac{d_i + \sum_{j \in N_i} d_j}{\delta_i}$

4. Wyniki badań eksperymentalnych

Celem eksperymentów jest znalezienie najlepszych algorytmów wstawień: najskuteczniejszych reguł priorytetowych oraz najefektywniejszych kryteriów oceny harmonogramów częściowych. Ustalane są najlepsze reguły priorytetowe stosowane do ustalania kolejności zadań na liście *L* oraz najlepsze zasady ustalania kolejności zadań nie umieszczonych na liście *P* przy kryterium K1 oceny uszeregowań częściowych.

Badania przeprowadzane są przy użyciu 960 instancji testowych z biblioteki *PSPLIB* [12] (po 480 dla problemów złożonych z 30 i 90 zadań). Do każdego z projektów z biblioteki *PSPLIB* dodawane są losowo cztery kamienie milowe (i związane z nimi zadania) z określonymi terminami realizacji tm_i, rozłożonymi równomiernie w trakcie realizacji całego projektu:

$$tm_{i+1} - tm_i = \frac{tm_m}{m}, \qquad \forall i \in <1, m) \tag{6}$$

gdzie: tm_m – termin realizacji całego projektu ($tm_m = \delta_{n+1}$).

Wszystkie eksperymenty prowadzone są na komputerze HP Compaq z procesorem Intel Pentium 4 CPU 3,0 GHz przy użyciu programu zaimplementowanego w języku C# w środowisku Visual Studio.NET. W tabeli 3 zestawione są średnie czasy trwania

i najlepsze średnie wartości funkcji celu harmonogramowania nominalnego poszczególnych algorytmów wstawień.

Tab. 3. Średnie czasy trwania $\bar{t}$ i najlepsze średnie wartości funkcji celu $\overline{F}$

	Algorytm	szeregowy SGS		równoległy SGS	
		$\bar{t}$	max $\overline{F}$	$\bar{t}$	max $\overline{F}$
30 zadań	**W1, K1**	0.011	**250.8 (R13)***	0.042	229.1 (R13)
	W1, K2	0.006	216.7 (R13)	0.014	214.8 (R13)
	W1, K3	0.006	246.2 (R15)	0.014	231.1 (R2)
	W2, K1	0.010	221.2 (R1)	0.039	219.7 (R0)
	W2, K2	0.004	149.0	0.009	44.3
	W2, K3	0.004	160.7	0.009	47.0
	W3, K1	0.015	198.3 (R13)	0.089	193.9 (R0)
	W3, K2	0.005	129.4	0.021	94.9
	W3, K3	0.005	132.1	0.023	59.0
90 zadań	**W1, K1**	0.340	331.7 (R13)	2.234	296.6 (R13)
	W1, K2	0.161	297.9 (R13)	0.778	290.2 (R2)
	W1, K3	0.166	**344.6 (R2)**	0.785	309.7 (R2)
	W2, K1	0.458	211.3 (R1)	2.449	202.4 (R1)
	W2, K2	0.067	98.2	0.317	–107.1
	W2, K3	0.061	110.9	0.259	–10.4
	W3, K1	0.653	190.9 (R11)	5.878	187.2 (R1)
	W3, K2	0.161	87.1	0.723	–35.3
	W3, K3	0.191	97.6	0.848	–77.2

* - w nawiasie zastosowana reguła priorytetowania w algorytmie o największej wartości *F*

Najlepsze harmonogramy są znajdowane przez algorytm W1. Najmniej skuteczny jest najbardziej czasochłonny obliczeniowo algorytm W3.

Na efektywność W1 znaczny wpływ ma użyta reguła priorytetowa stosowana do ustalania kolejności rozpatrywanych zadań (porządku na liście *L*). W zastosowaniu do procedury W1 najefektywniejsze są reguły, które są wydajne także dla algorytmów priorytetowych [11] (reguły R2, R3, R13, R15), niezależnie od przyjętego kryterium oceny harmonogramów częściowych.

Reguły priorytetowe stosowane przy ustalaniu kolejności zadań nie umieszczonych na liście zadań *P* przy stosowaniu kryterium K2 i K3 oceny harmonogramów częściowych nie mają dużego wpływu na efektywność uszeregowania złożonego z wszystkich zadań. Kryterium oceny harmonogramów częściowych K2 jest najmniej efektywne dla każdego z proponowanych algorytmów wstawień.

Eksperymenty wykazują lepszą przeciętną skuteczność szeregowej procedury dekodującej niż równoległej SGS. Korzyści ze stosowania szeregowej SGS można zaobserwować zwłaszcza dla wydajnych algorytmów wstawień. Dla każdego typu algorytmu wstawień najlepsze wartości są znajdowane za pomocą szeregowej procedury dekodującej.

Dla problemów 30-zadaniowych przeciętnie najlepsze rozwiązania osiągane są przy zastosowaniu W1 z kryterium oceny K1 z regułą priorytetową R13 (250,8 to średnia

wartość *F*). Dla problemów 90-zadaniowych najlepsze harmonogramy uzyskiwane są przy wykorzystaniu W1 z kryterium oceny K3 z regułą priorytetową R2 (344,6 to przeciętna wartość *F*).

Średnie wartości funkcji celu *F* są wyższe niż dla najlepszych algorytmów priorytetowych [11] i nieznacznie odbiegają od najlepszych wartości funkcji celu uzyskanych przy zastosowania algorytmów metaheurystycznych [2].

5. Zakończenie

W artykule zaprezentowane są nowe algorytmy wstawień dla problemu harmonogramowania projektu ze zdefiniowanymi kamieniami milowymi. Algorytmy te są zbliżone do algorytmów metodą wstawień stosowanych dla innych zagadnień szeregowania zadań tj. problem permutacyjny. Działanie algorytmów jest testowane przy użyciu problemów testowych z biblioteki *PSPLIB*, zmodyfikowanych przez zdefiniowanie umownych etapów projektu.

Wyniki testów potwierdzają przydatność algorytmów wstawień do generowania harmonogramów dla rozważanego problemu. Harmonogramy uzyskane przy wykorzystaniu algorytmu W1, najlepszego i najmniej czasochłonnego z algorytmów, są nieznacznie przeciętnie gorsze od rozwiązań wygenerowanych przez najefektywniejsze algorytmy metaheurystyczne [2].

Literatura

1. Kostrubiec A.: Harmonogramowanie projektów - przegląd modeli. Inżynieria Zarządzania Przedsięwzięciami, Wydawnictwo Politechniki Gdańskiej, Gdańsk 2003, s. 33-52.
2. Klimek M., Łebkowski P.: Algorytmy metaheurystyczne dla problemu harmonogramowania projektu z kamieniami milowymi. Zeszyty Naukowe Politechniki Śląskiej, Seria: Automatyka, z. 150, 2008, s. 63–72.
3. Węglarz J. (Red.): Project Scheduling: Recent Models, Algorithms and Applications, Kluwer Academic Publishers, 1998.
4. Herroelen W., De Reyck B., Demeulemeester E.: Resource constrained scheduling: a survey of recent developments. Computers and Operations Research 25, 1998, s. 279-302.
5. Błażewicz J., Lenstra J., Kan A. R.: Scheduling subject to resource constraints - classification and complexity. Discrete Applied Mathematics 5, 1983, s. 11-24.
6. Klimek M., Łebkowski P.: Algorytmy wstawień dla problemu harmonogramowania projektu z ograniczoną dostępnością zasobów, Wybrane zagadnienia logistyki stosowanej (red.) Lech Bukowski. Wydawnictwa AGH, 2009, s.120-127.
7. Nawaz M., Enscore E., Ham I.: A heuristic algorithm for the m machine, n-job flow-shop sequencing problem. OMEGA 11, 1983, s. 91-95.
8. Woo D.S., Yim H.S.: A heuristic algorithm for mean flowtime objective in flowshop scheduling, Computers and Operations Research 25, 1998, s.175-182.
9. Kolisch R.: Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research, 90, 1996, s. 320-333.
10. Kolisch, R. Efficient priority rules for the resource-constrained project scheduling problem. Journal of Operations Management 14, 1996, s. 179-192.

11. Klimek M., Łebkowski P.: Algorytm priorytetowy harmonogramowania projektu przy ograniczonych zasobach. Komputerowo Zintegrowane Zarządzanie (red.) R. Knosala, Opole: Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, T.I, 2009, s. 532-540.
12. 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 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