NAUKI PODSTAWOWE
CZASOPISMO TECHNICZNE
TECHNICAL TRANSACTIONS
FUNDAMENTAL SCIENCES
WYDAWNICTWO POLITECHNIKI KRAKOWSKIEJ
1-NP/2011
ZESZYT 22
ROK 108
ISSUE 22
YEAR 108

LECH JAMROŻ, JERZY RASZKA*

SZEREGOWANIE ZADAŃ CYKLICZNYCH Z WYKORZYSTANIEM ALGORYTMÓW PRIORYTETOWYCH

ON SCHEDULING OF CYCLIC TASKS USING PRIORITY ALGORITHMS

Streszczenie

W artykule przedstawione jest zagadnienie szeregowania zadań cyklicznych w systemach czasu rzeczywistego. Oprogramowanie aplikacyjne systemów, najczęściej składa się z tego typu zadań o różnych ograniczeniach czasowych, które nie mogą być przekroczone. Stosowaną metodą szeregowania niezależnych zadań cyklicznych jest przydzielanie im priorytetów. Strategia doboru priorytetów oraz zasada posługiwania się nimi określona jest algorytmem szeregowania. Rozważane są priorytetowe algorytmy RMS, DMS oraz EDF, dla których przedstawione są warunki szeregowalności.

Słowa kluczowe: *szeregowanie, algorytmy priorytetowe, zadania cykliczne*

Abstract

This paper is devoted to the scheduling of cyclic tasks in real-time systems. The application programs of such systems have been defined as those containing cyclic tasks that have deadlines that cannot be missed. The priority algorithms: RMS, DMS and EDF for scheduling set of independent cyclic tasks are considered. There are several rules basing on which the priorities are assigned to the tasks and then the tasks are being scheduled. Schedulability constrains are considered which guarantee the deadlines of cyclic tasks.

Keywords: scheduling, priority algorithms, cyclic tasks

* Dr inż. Lech Jamroż, dr inż. Jerzy Raszka, Instytut Informatyki, Wydział Fizyki, Matematyki i Informatyki, Politechnika Krakowska.

1. Wstęp

Wiele aplikacji wykorzystywanych w systemach czasu rzeczywistego składa się ze zbioru zadań wykonywanych cyklicznie (np. odczyt danych sensorycznych, pętle sterowania, monitorowanie). Aplikacje te są zorientowane nie tylko na poprawność dostarczanych wyników, ale również na konieczność zagwarantowania, żeby każda periodyczna instancja była regularnie aktywowana z prawidłową częstotliwością i kończyła swoje obliczenia przed upływem nieprzekraczalnego terminu (*deadline*). W wielu przypadkach zapewnienie punktualności i przewidywalności systemów można uzyskać poprzez podział oprogramowania aplikacyjnego na niezależne zadania cykliczne, wykonywane z rożnymi ograniczeniami czasowymi [2, 3, 7, 10]. W celu uporządkowania zbioru tych zadań nadaje się im priorytety, które mogą zapewnić spełnienie żądanych uwarunkowań czasowych. Strategia doboru priorytetów oraz zasada posługiwania się nimi w wykonywaniu zadań określona jest algorytmem szeregowania.

Ze względu na sposób podejmowania decyzji, wyróżniamy dwie podstawowe strategie szeregowania: szeregowanie statyczne (*pre-run-time scheduling*) oraz szeregowanie dynamiczne (*run-time scheduling*).

Szeregowanie statyczne odbywa się przed uruchomieniem systemu i jego rezultatem jest Plan Aktywacji Procesów (PAP), interpretowany jako pewna funkcja określająca zadanie, które w danej chwili ma być wykonywane (PAP może być pamiętany w postaci tablicy lub listy). Warunkiem stosowalności metody jest aprioryczna znajomość zadań wraz z ich charakterystykami czasowymi. Przykładem tej strategii szeregowania są algorytmy RMS oraz DMS [1, 2, 8].

Szeregowanie dynamiczne odbywa się na bieżąco w trakcie pracy systemu. Podstawą działania tego szeregowania są priorytety, określające względną ważność zadań. Metoda może być stosowana, gdy charakterystyki czasowe zadań nie są z góry znane. Przykładem tej strategii szeregowania jest algorytm EDF [6, 3].

Ze względu na przyczynę podejmowania decyzji szeregujących wyróżnia się: szeregowanie wymuszane czasem (*clock driven scheduling*) oraz szeregowanie wymuszane zdarzeniami (*event driven scheduling*).

Dla szeregowania wymuszanego czasem decyzje szeregujące podejmowane są w specyficznych, często z góry ustalonych chwilach czasu. Gdy parametry zadań są znane *a priori*, plan szeregowania można ułożyć w postaci tablicy (szeregowanie cykliczne).

W przypadku szeregowania wymuszanego zdarzeniami, decyzje szeregujące podejmowane są na bieżąco, kiedy zachodzą ściśle określone zdarzenia np. zadanie staje się gotowe do wykonania, zmienia się jego priorytet albo następuje zwolnienie procesora przez zadanie bieżące. Procedura szeregująca najczęściej jest uaktywniana gdy wystąpiło przerwanie sprzętowe lub proces bieżący wykonał wywołanie systemowe. Algorytm priorytetowy może pracować w trybie wywłaszczania (*preemptive scheduling*) lub nie wywłaszczania (*non-preemptive scheduling*) [11].

Krytyczne zadania aperiodyczne (wywoływane przerwaniami) traktuje się jako cykliczne o pewnej maksymalnej częstotliwości wznawiania. Zadania sporadyczne uruchamiane są w nieregularnych odstępach czasowych, czas kolejnej aktywacji obliczany jest w trakcie aktywacji bieżącej Problem szeregowania cyklicznych zadań zależnych można rozwiązać poprzez implementowanie protokołu synchronizacji [4, 5, 11].

2. Model zadania cyklicznego

Algorytmy szeregowania zadań cyklicznych opierają się na podstawowym modelu zadnia, w którym czas dyskretny biegnie od chwili początkowej t_0 [6]. Zadanie (task) $Z_i = (p_i, T_i, d_i, , r_i)$ scharakteryzowane jest przez następujące parametry:

p_i – czas wykonania zadania (*processing time*),

T_i – okres (*period*) występowania zadania,

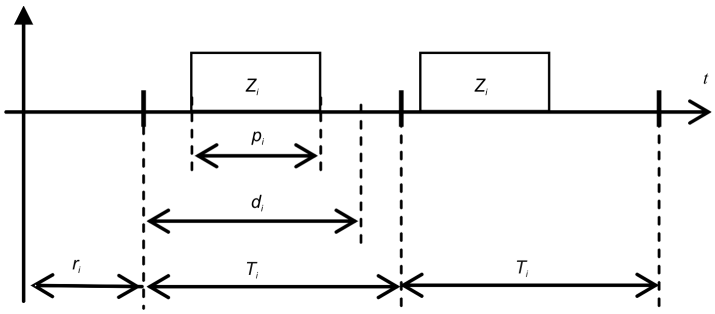
d_i – względne ograniczenie czasowe zadania (*relative deadline*), względny ostateczny termin zakończenia zadania.

r_i – faza zadania, czas pierwszego wywołania (*release time*).

Względne ograniczenie czasowe zadania oznacza limit czasu, przed upływem którego procesor powinien zakończyć jego wykonanie. Zwykle takie ograniczenie czasowe jest mniejsze lub równe okresowi występowania zadania. Okres występowania zadania T_i, czas jego wykonania p_i oraz względne ograniczenie czasowe d_i powinny spełniać warunek:

$$p_i < d_i < T_i \tag{1}$$

Parametry czasowe zadania cyklicznego przedstawiono na rysunku 1.



Rys. 1. Parametry czasowe zadania cyklicznego

Fig. 1. Timing parameters of cyclic task

Własności zadań cyklicznych:

- wszystkie zadania są cykliczne ze znanym okresem T_i pomiędzy kolejnymi wystąpieniami,
- zadania są od siebie niezależne, dopuszcza się ich wywłaszczanie,
- wszystkie instancje zadania mają niezmienny czas wykonania p_i,
- zadania wykonywane są cyklicznie z założoną częstotliwością, określoną na podstawie wymagań danej aplikacji (np. wolna, szybka pętla obliczeń).

Implementacją zadania nazywamy sekwencyjny program uruchamiany w dyskretnych chwilach – t_i takich, że $t_{i+1} = t_i + T_i$. Każde wywołanie zadania powoduje jego wykonanie w chwili – t_{ei} określonej przez algorytm szeregujący. Jeżeli t_i określa moment i-tego wywołania zadania, to wykonywanie musi rozpocząć się nie wcześniej niż w chwili – t_i a zakończyć nie później niż w chwili $t_{di} = t_i + d_i$. Początek i-tego wykonania zadania musi spełniać warunek $t_i \le t_{ei} \le t_{di} - p_i$. W algorytmach priorytetowych często zakładamy, że $T_i = d_i$, wówczas $t_{di} = t_i + T_i$.

Instancje (kolejne wystąpienia) zadania cyklicznego są regularnie aktywowane ze stałą częstotliwością i mają ten sam najdłuższy czas wykonywania WCET (*Worst Case*

Execution Time). Wszystkie instancje zadania mają tą samą wartość względnego ostatecznego terminu zakończenia.

Współczynnik wykorzystania procesora U [7]:

$$U = \sum_{i=1}^{n} \frac{p_i}{T_i} \tag{2}$$

opisuje obciążenie procesora podczas wykonywania zbioru zadań cyklicznych.

Dla różnych algorytmów szeregowania, współczynnik może przyjmować różne graniczne wartości, dla których dany zbiór zadań jest szeregowalny (tzn. może wykonać wszystkie swoje obliczenia w zadanych ograniczeniach czasowych).

Jeżeli współczynnik wykorzystania procesora jest większy od 1, to dany zbiór zadań może nie być szeregowany przez żaden algorytm priorytetowy [8].

Celem algorytmu szeregowania zadań jest takie ustalenie kolejności ich wykonywania, aby żadne zadanie nie przekroczyło swojego granicznego terminu zakończenia. Jeżeli istnieje taka kolejność to zbiór zadań jest szeregowalny. Ewentualne włączanie nowego zadania do systemu odbywa się według ściśle określonej procedury (*mode change protocol*).

W przypadku występowania zadań sporadycznych, w prowadzonych analizach zadania te są uwzględniane, lecz są one traktowane tak jak zadania cykliczne, o okresie równym minimalnemu czasowi pomiędzy kolejnymi wystąpieniami tego samego zadania sporadycznego.

3. Klasyfikacja algorytmów priorytetowych

Podstawową ideą priorytetowych algorytmów szeregowania jest przyjęcie określonej reguły przydzielania każdemu zadaniu priorytetu. W danej chwili jest uruchamiane zadanie o najwyższym priorytecie. Najczęściej implementowane algorytmy szeregowania zadań cyklicznych to:

1. RMS (*Rate Monotonic Scheduling*) – statyczny algorytm monotoniczny w częstotliwości,
2. DMS (*Deadline Monotonic Scheduling*) – statyczny algorytm monotoniczny we względnym ograniczeniu czasowym,
3. EDF (*Earliest Deadline First*) – dynamiczny algorytm „najwcześniejsza linia krytyczna najpierw”,
4. CES (*Cyclic Executive Scheduling*) - statyczny algorytm “wykonywanie cykliczne”.

3.1. Algorytm RMS

Algorytm RMS jest algorytmem statycznym. Działanie algorytmu oparte jest na systemie priorytetów przydzielanych do poszczególnych zadań w ten sposób, że im krótszy okres, tym wyższy priorytet. Stosowanie tej zasady wynika z faktu, że zadania występujące częściej zazwyczaj są ważniejsze od zadań występujących rzadziej.

Ponadto ograniczenia czasowe oraz najgorszy przypadek czasu odpowiedzi dla zadań występujących częściej są krótsze, co dodatkowo uzasadnia stosowanie tej metody.

Zbiór niezależnych, wywłaszczalnych zadań cyklicznych ze względnymi ograniczeniami czasowymi równymi ich okresom tj. $d_i = T_i$, może mieć poprawny harmonogram na jednym procesorze utworzony przez algorytm RMS, jeśli współczynnik wykorzystania procesora spełnia następujący warunek konieczny [8]:

$$\sum_{i=1}^{n} \frac{p_i}{T_i} \le 1 \tag{3}$$

Powyższy warunek na szeregowalność zadań wynika z faktu, że łączny stopień wykorzystania czasu pracy procesora przez wszystkie zadania nie może być większy niż 100%. Warunek wystarczający pozwalający na weryfikację spełnienia warunków czasu rzeczywistego (*real time*) i gwarantujący, że n zadań zostanie wykonanych przed upływem ich ograniczenia czasowego został podany przez Liu C. i Laylanda J. (1973 r.) i wyraża się wzorem [8]:

$$U_{gr} = \sum_{i=1}^{n} \frac{p_i}{T_i} \le n(2^{1/n} - 1) \tag{4}$$

Dla dużych wartości $n \to \infty$, $U_{gr} = \ln 2 \approx 0{,}69$, co oznacza, że każdy zbiór zadań cyklicznych, dla którego współczynnik wykorzystania procesora jest 0,69 zawsze będzie szeregowalny [1]. Jeśli dla danego zbioru zadań współczynnik wykorzystania procesora U mieści się w przedziale $U_{gr} \le U \le 1$, to zbiór ten może nie być szeregowalny. Wówczas dla rozstrzygnięcia należy rozważyć przypadek najgorszy (*worst case*), którego wszystkie zadania podlegające szeregowaniu wchodzą jednocześnie w stan gotowości. Jeżeli każde z rozważanych zadań zostanie zakończone przed upływem swego ograniczenia czasowego, to dany zbiór zadań jest szeregowalny.

Z kolei warunek konieczny i wystarczający na szeregowalność zbioru zadań został podany przez Sha L. (1988 r.) i wyraża się zależnością [1]:

$$\forall i : 1 \le i \le n, \quad \min_{(k,l)\in R_i} \left(\sum_{j=1}^{i-1} \frac{p_j}{T_j} \frac{T_j}{lT_k} \left\lceil \frac{lT_k}{T_j} \right\rceil + \frac{p_i}{lT_k} \right) \le 1 \tag{5}$$

gdzie

$$R_i = \left\{ (k,l) : 1 \le k \le i, \quad l = 1, \ldots \left\lfloor \frac{T_i}{T_k} \right\rfloor \right\}$$

Powyższe testy szeregowalności różnią się złożonością obliczeniową. Test (4) posiada złożoność obliczeniowa rzędu $O(n)$. Złożoność obliczeniowa testu (5) jest większa i zależna od danych instancji zadań [7].

W przypadku występowania zadań zależnych, warunek (4) staje się łagodniejszy, ponieważ w prawidłowo funkcjonującym systemie nie powinna wystąpić sytuacja, w której aktywne są wszystkie zadania. Wynika to z sekwencji następujących po sobie zadań, a więc spełnienie tego warunku jest łatwiejsze do osiągnięcia.

Algorytm RMS spośród priorytetowych algorytmów statycznych, wyznacza uszeregowanie optymalne, w tym sensie że jeżeli nie są utrzymywane ograniczenia czasowe, to żaden inny algorytm z tej klasy również ich nie dotrzymuje [2].

3.2. Algorytm DMS

Działanie statycznego algorytmu DMS oparte jest na systemie priorytetów, przydzielanych na podstawie względnego ograniczenia czasowego według zasady: mniejsza wartość ograniczenia czasowego, większy priorytet.

Przyjmowane jest założenie, że $p_i \le d_i \le T_i$. Warunek wystarczający na istnienie rozwiązania dopuszczalnego wyraża się zależnością [1]:

$$\forall i : 1 \le i \le n : \frac{p_i}{d_i} + \frac{I_i}{d_i} \le 1 \tag{6}$$

gdzie

$$I_i = \sum_{j=1}^{i-1} \left\lceil \frac{d_i}{T_j} \right\rceil p_j$$

Szeregowanie zadań według algorytmu DMS, jeżeli istnieje, to jest optymalne [1].

3.3. Algorytm EDF

Systemy, w których podczas pracy priorytet przydzielany zadaniu, może zostać zmieniony nazywamy systemami z dynamicznym priorytetowaniem.

Przykładem dynamicznego przydziału priorytetu jest algorytm EDF wykorzystujący zasadę, zgodnie z którą najwyższy priorytet otrzymuje zadanie, któremu najwcześniej kończy się ograniczenie czasowe. Algorytm EDF jest ogólnym algorytmem szeregującym zbiory zadań w oparciu o jeden procesor, przy czym żądany termin wykonania każdego z zadań jest równy jego okresowi, ponadto zadania nie są wywłaszczalne. Zbiór zadań jest szeregowalny, jeżeli spełnione są następujące warunki [6]:

$$\sum_{i=1}^{n} \frac{p_i}{T_i} \le 1 \tag{7}$$

$$\forall i : 2 \le i \le n, \forall L,\ T_1 < L < T_i : \quad L \ge p_i + \sum_{j=1}^{i-1} \left\lfloor \frac{L-1}{T_j} \right\rfloor p_j \tag{8}$$

Warunek (7) odzwierciedla wymaganie, aby średnie obciążenie procesora w systemie jednoprocesorowym nie przekraczało 100%, warunek ten dotyczy szeregowania zarówno zdań wywłaszczalnych, jak i niewywłaszczalnych dowolnym algorytmem szeregującym. Warunek (8) jest specyficzny dla algorytmu EDF zapewnia, że obciążenie procesora w przedziale czasu o długości z zakresu (T_1, T_i) rozpoczynającym się w chwili wywołania zadania Z_i nie przekracza 100%. Zbiór zadań jest szeregowalny algorytmem EDF, gdy spełnione są warunki (7) i (8) [6].

W systemach z dynamicznym priorytetowaniem przydzielenie priorytetu i rozpoczęcie wykonywania zadania jest realizowane wówczas, kiedy pojawia się nowe zadanie lub kończy się wykonywanie aktualnego zadania. Zaletą metody dynamicznego przydziału priorytetu EDF, w porównaniu do metod statycznego przydziału RMS i DMS jest lepsze wykorzystanie procesora. Natomiast jej wadą jest większe obciążenie procesora czynnościami szeregowania podczas pracy. Podstawą działania algorytmu EDF jest regularne

aktualizowanie terminów zakończenia zadań. Czynność ta wykonywana jest po wystąpieniu każdego przerwania obsługiwanego przez moduł szeregujący – przerwania zegarowego, pojawienia się nowego zadania lub zakończenia wykonywania zadania aktualnego. Pojawiające się w systemie nieokresowe zadania traktowane są przez algorytm analogicznie do zadań okresowych. Jedyną różnicą jest brak cyklicznego ich wykonywania. Reguły FIFO i LIFO oraz LST (*Least Slack Time First*) są również algorytmami dynamicznego przydziału priorytetów [12].

3.4. Algorytm CES

Podejście do szeregowania według algorytmu *CES* polega na specyficznym wywoływaniu zestawu procedur zgodnie z utworzoną tablicą wykonywania cyklicznego.

Tablica obejmuje główny cykl systemu (*major cycle*) podzielony zwykle na pośrednie cykle o stałej długości. Pośrednie cykle (*minor cycles*) decydują o minimalnym cyklu w systemie, natomiast główny cykl określa cykl maksymalny [12].

Procedura szeregująca wywoływana jest w cyklu minimalnym. Cyklem głównym jest najmniejsza wspólna wielokrotność z cykli poszczególnych zadań (T_1, T_2, ..., T_n).

Plan szeregowania zadań sporządzany jest dla cyklu głównego, w którym wszystkie zadania przyporządkowane do procesora powinny być wykonane tak, aby ich ograniczenia czasowe nie zostały przekroczone. Względny żądany termin zakończenia zadania jest równy jego okresowi.

Cechy charakterystyczne algorytmu to determinizm (ze względu na utworzoną tablicę) i brak wywłaszczeń. Opracowanie tablicy wykonywania cyklicznego jest ogólnie trudne (problem NP trudny), żmudna aktualizacja, powstaje problem z włączeniem zadań o długim okresie a także zadań sporadycznych.

Cechy szeregowania według algorytmu CES:

- Wyeliminowany jest zestaw procesów (zadań) z systemu. Każdy z pośrednich cyklów jest sekwencją wywołań procedur;
- Procedury współdzielą wspólną przestrzeń adresową skąd mogą wymieniać dane;
- Dane nie muszą być chronione (np. przez semafory), ponieważ nie istnieje współbieżny dostęp do zasobów;
- Wszystkie okresy zadań muszą być wielokrotnością cyklu pośredniego.

Mogą pojawić się problemy z włączeniem zadań o długim okresie. Główny cykl jest zarazem maksymalnym cyklem w systemie bez konieczności włączania dodatkowych modułów szeregujących. Podejście to praktycznie uniemożliwia zastosowanie innych, bardziej elastycznych metod szeregowania [12].

4. Wyniki badań symulacyjnych

Do badań symulacyjnych wybrano algorytmy RMS i EDF, które porównano wg wybranych wskaźników, oceniając dla nich wykonywalność zadań (możliwość uszeregowania) oraz stopień wywłaszczania zadań. Symulację przeprowadzono, przyjmując dla wszystkich zadań zerową fazę i zgodność zakończenia cyklu z przedziałem ograniczenia czasowego, czyli odpowiednio $r_i = 0$ i $T_i = d_i$ dla $i = 1, ..., n$ oraz przedziały czasu symulacji od początkowego czasu $t_0 = 0$ do czasowych ograniczeń $t_{\max} >> \max_{i=1...n}(T_i)$.

W związku z przyjętymi założeniami zadania zdefiniowane ogólnie w pkt. 2, opisano uproszczoną postacią $Z_i = (p_i, T_i)$. W celu porównania stopnia wykonywalności i wywłaszczania zadań, uwzględniono w zadanym przedziale w symulacji $[t_0, t_{max}]$:

p_i – rzeczywisty czas trwania tylko dla wykonanego zadania,

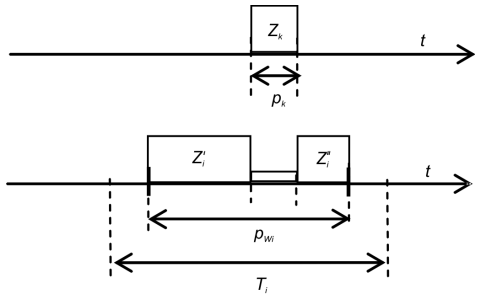
p_{Wi} – czas trwania zadania od jego początku aż do zakończenia (uwzględnia także ewentualne wywłaszczenie przez zadanie o wyższym priorytecie np p_k na rys. 2).

Do każdego przypadku zestawu n zadań obliczono na postawie uzyskanych symulacyjnie czasów p_i i p_{Wi}, odpowiednio:

U_z – wartość wskaźnika określającego względny stopień wykonywania wszystkich zadań,

U_w – wartość wskaźnika oceniającego względny stopień wszystkich wykonanych zadań z ich wywłaszczaniem.

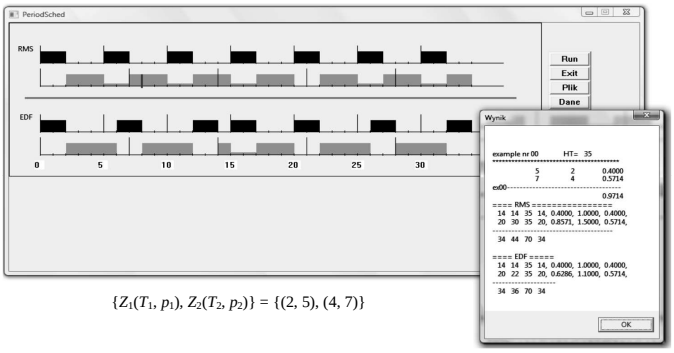
Względność czasów wykonywania odniesiono do czasu cyklu T, przez analogie do ogólnie przyjętego wskaźnika U wykorzystania procesora (2). W uzyskanych wynikach także w celach porównawczych ujęto wartość wskaźnika U.



Rys. 2. Parametry czasowe zadania w warunkach wywłaszczania

Fig. 2. Timing parameters with preemption of task

Wyniki szeregowania przykładu testującego dla dwóch zadań $\{Z_1(T_1, p_1), Z_2(T_2, p_2)\} = \{(2, 5), (4, 7)\}$ i współczynnika wykorzystania procesora $U = 0{,}97$ przedstawia rys. 3. Pogrubiona i przedłużona linia pionowa oznacza przekroczenie przez zadanie Z_2 ograniczenia wynikającego z założonego czasu cyklu T_2.



$\{Z_1(T_1, p_1), Z_2(T_2, p_2)\} = \{(2, 5), (4, 7)\}$

Rys. 3. Przykład testujący ex00

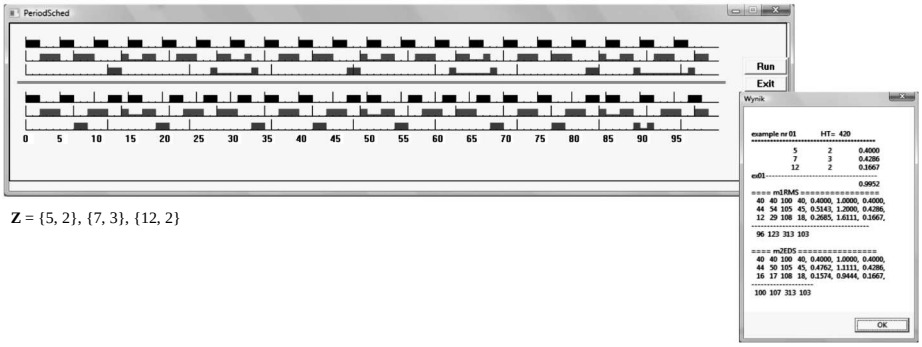
Fig. 3. Test of example ex00

Dla kolejnych czterech przykładów symulacji procesów przyjęto trzy zadania, ich dane przedstawia tabela 1, a wybrany przykład uszeregowania przestawia rys 4. Zbiorcze wyniki dla wszystkich przykładów przedstawiono na rys. 5.

Tabela 1

Dane przykładów

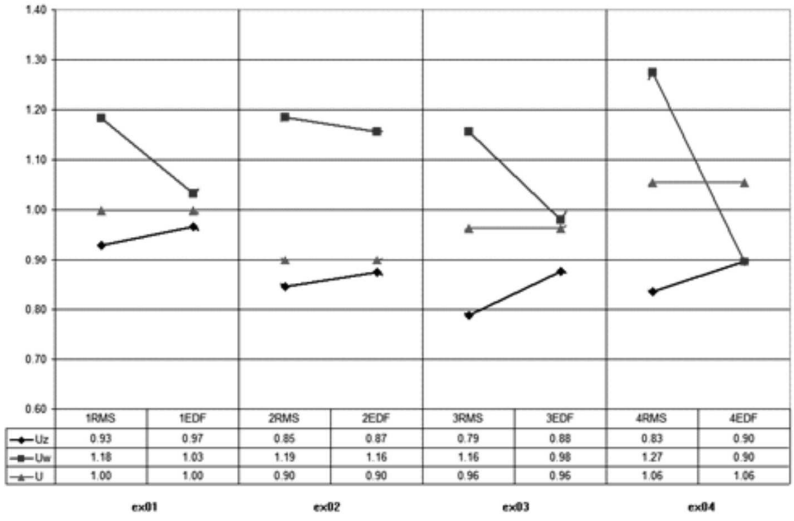
Przykład	$Z_1(T_1, p_1)$	$Z_2(T_2, p_2)$	$Z_3(T_3, p_3)$
ex01	{5, 2}	{7, 3}	{12, 2}
ex02	{7, 3}	{11, 3}	{15, 3}
ex03	{11, 3}	{14, 6}	{19, 5}
ex04	{11, 4}	{14, 6}	{19, 5}



Z = {5, 2}, {7, 3}, {12, 2}

Rys. 4. Przykład ex01

Fig. 4. Example ex01



Rys. 5. Wyniki przykładów

Fig. 5. Examples of results

5. Wnioski

W artykule przedstawiono przegląd podstawowych algorytmów priorytetowych do szeregowania zadań cyklicznych w systemach czasu rzeczywistego.

Algorytmy ze stałymi priorytetami są łatwiejsze w implementacji. W przypadku przeciążenia zachowanie algorytmów RMS, DMS jest bardziej przewidywalne, nie wykonają się zadania o niższym priorytecie. Algorytmy ze stałymi priorytetami umożliwiają poprawne szeregowanie, pomimo przekroczonego teoretycznego współczynnika wykorzystania procesora. System z wykonywaniem cyklicznym wg algorytmu CES jest w pełni deterministyczny, mogą pojawiać się jednak problemy z aktualizacją tablicy oraz z włączeniem zadań o długim okresie wykonywania lub zadań sporadycznych.

Większość systemów czasu rzeczywistego o twardych ograniczeniach czasowych wykorzystuje szeregowanie statyczne. Dynamiczne przydzielanie priorytetów według algorytmu EDF jest mniej przewidywalne, może występować inwersja priorytetów. Szeregowanie dynamiczne daje zazwyczaj lepsze średnie wykorzystanie procesora. Potwierdzają to uzyskane wyniki, dla których dynamiczny algorytm EDF w stosunku do statycznego algorytmu RMS dla tych samych danych zadań, uzyskał mniejszy stopień wywłaszczania U_w przy równocześnie większym stopniu wykonywalności U_z.

Literatura

[1] Audsley N.C., *Deadline Monotonic Scheduling*, YCS 146, Department of Computer Science, Uniwersity of York, 1990.

[2] Buhr R., Bailey D., *An Introduction to Real-Time Systems*, Prentice Hall, 1999.

[3] Gaj P., Kwiecień A [red]., *Systemy informatyczne z ograniczeniami czasowymi*, Wydawnictwo Komunikacji i Łączności, Warszawa 2006, 115-126.

[4] Jamroż L., Raszka J., *Performance model of cyclic processes*, Czasopismo Techniczne, 1-NP/2010, Wydawnictwo PK, Kraków 2010.

[5] Jamroż L., Raszka J., *The use of a max-plus algebra in a scheduling of cyclic processe*, Zeszyty Naukowe Politechniki Śląskiej, Automatyka, No. 151, 2008.

[6] Jestratjew A., Kwiecień A., [w:] *Systemy czasu rzeczywistego. Projektowanie i aplikacji*, Gaj P. (red.), Wydawnictwo Komunikacji i Łączności, Warszawa 2005.

[7] Liu J., *Real Time Systems*, Prentice Hall, 2000.

[8] Liu C., Layland J., *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time environment*, Journal of the ACM, No. 1, 1973.

[9] Smutnicki Cz., *Algorytmy Szeregowania*, Akademicka Oficyna Wydawnicza Exit, Warszawa 2002.

[10] Szmuc T., *Specyfikacja i projektowanie oprogramowania systemów czasu rzeczywistego*, Uczelniane Wydawnictwo Naukowo-Dydaktyczne, AGH, Kraków 2000.

[11] Tanenbaum A., *Rozproszone systemy operacyjne*, PWN, Warszawa 1997.

[12] Ułasiewicz J., *Systemy czasu rzeczywistego*, Wydawnictwo BTC, Warszawa 2007.