Politechnika Wrocławska

Instytut Informatyki, Automatyki i Robotyki

Marcin Winczaszek

Wybrane problemy szeregowania z optymalnym doborem przedziałów zakończenia wykonywania zadań

(rozprawa doktorska)

Promotor:

Prof. dr hab inż. Adam Janiak

Współopiekun:

Prof. Mikhail Y. Kovalyov
(*Belarus State University, Mińsk, Białoruś*)

Wrocław, 2006

Spis treści

1 Wstęp **3**

2 Podstawowe pojęcia dotyczące szeregowania zadań **7**
2.1 Formułowanie problemów szeregowania zadań 7
2.2 Metody rozwiązywania problemów szeregowania 11

3 Zastosowania praktyczne **15**

4 Stan badań **20**
4.1 Wstęp 20
4.2 Problemy z pożądanymi momentami zakończenia wykonywania zadań . . . 20
4.3 Problemy z pożądanymi przedziałami zakończenia wykonywania zadań . . 25
4.4 Podsumowanie 30

5 Problem z liniowymi identycznymi dla wszystkich zadań funkcjami kar za nieterminowość wykonania **31**
5.1 Wstęp 31
5.2 Sformułowanie problemu 31
5.3 Własności rozwiązania optymalnego 33
5.4 Algorytm optymalny 46
5.5 Podsumowanie rozdziału 48

6 Problemy z nieliniowymi identycznymi dla wszystkich zadań funkcjami kar za nieterminowość wykonania **49**
6.1 Wstęp 49
6.2 Sformułowanie problemu 50
6.3 Analiza złożoności obliczeniowej 51
6.4 Własności rozwiązania optymalnego 52
6.5 Algorytmy optymalne 63
6.6 Podsumowanie rozdziału 73

7 Minimalizacja sumy ważonych opóźnień i przyspieszeń wykonania zadań 74
7.1 Wstęp . . . 74
7.2 Sformułowanie problemu . . . 75
7.3 Własności rozwiązania optymalnego i złożoność obliczeniowa . . . 76
7.4 Algorytmy programowania dynamicznego . . . 86
7.5 W pełni wielomianowy schemat aproksymacyjny . . . 98
7.6 Wielomianowo rozwiązywalne przypadki . . . 109
7.7 Podsumowanie rozdziału . . . 120

8 Minimalizacja ważonej liczby zadań opóźnionych i przedwcześnie wykonanych 121
8.1 Wstęp . . . 121
8.2 Sformułowanie problemu . . . 121
8.3 Własności rozwiązania optymalnego . . . 123
8.4 Algorytmy optymalne . . . 133
8.5 W pełni wielomianowy schemat aproksymacyjny . . . 139
8.6 Podsumowanie rozdziału . . . 144

9 Minimalizacja ważonej liczby opóźnionych zadań oraz sumy ważonych przyspieszeń wykonania zadań 145
9.1 Wstęp . . . 145
9.2 Sformułowanie problemu . . . 145
9.3 Własności rozwiązania optymalnego . . . 146
9.4 Analiza złożoności obliczeniowej . . . 153
9.5 Algorytm optymalny . . . 156
9.6 Podsumowanie rozdziału . . . 158

10 Podsumowanie pracy 159

Rozdział 1

Wstęp

Problemy szeregowania zadań, a także inne problemy optymalizacji dyskretnej, są dziś przedmiotem dużego zainteresowania w świecie nauki i techniki. Przyczynia się do tego zwiększenie konkurencyjności na rynkach światowych, co wymusza z jednej strony obniżanie kosztów produkcji a z drugiej strony zwiększanie elastyczności systemów produkcyjnych w celu umożliwienia szybkiego dostosowywania się do potrzeb klientów. Szereg problemów, od których rozwiązania zależy efektywność działania firmy, można zamodelować przy pomocy właśnie problemów szeregowania zadań. Osiągnięcia naukowe związane z teorią szeregowania zadań stymulują również rozwój informatyki, ponieważ stosowane są m. in. do poprawy efektywności pracy schedulerów wykorzystywanych w systemach operacyjnych oraz skrócenia czasów działania algorytmów przetwarzania danych.

W ogólności problem szeregowania zadań polega na ustaleniu kolejności wykonywania zadań na maszynach (procesorach), tak aby zminimalizować (lub zmaksymalizować) wartość pewnego zadanego kryterium. Dokładna definicja problemów szeregowania zadań zostanie przedstawiona w kolejnym rozdziale niniejszej rozprawy.

Niniejsza rozprawa dotyczy wąskiej grupy problemów szeregowania zadań - *problemów z doborem pożądanego przedziału zakończenia wykonywania zadań.*

Geneza rozpatrywanych w pracy problemów jest związana z systemami produkcyjnymi Just-in-Time (JIT). JIT to filozofia zarządzania, w której dąży się do eliminowania źródeł strat poprzez wytwarzanie właściwego produktu we właściwym miejscu i właściwym czasie. Pewne aspekty filozofii JIT znalazły odzwierciedlenie w problemach szeregowania zadań z pożądanymi momentami zakończenia wykonywania zadań. W problemach takich, dla każdego zadania jest określony *pożądany moment zakończenia* jego wykonywania. Jeżeli zadanie kończy wykonywać się przed pożądanym momentem zakończenia wykonywania, to powstaje kara za *zbyt wczesne wykonanie zadania*, co związane jest m.in. z kosztami magazynowania. Z drugiej strony, jeżeli zadanie kończy wykonywać się po pożądanym momencie zakończenia wykonywania, to powstaje kara za *zbyt późne wykonanie zadania*, która odzwierciedla przykładowo ponoszenie kar określonych w umowie za niedotrzymanie przez producenta terminu dostawy. Tym samym najbardziej korzystna jest sytuacja,

w której każde zadanie kończy wykonywać się dokładnie w pożądanym momencie zakończenia jego wykonywania. Zarówno zbyt wczesne, jak i zbyt późne zakończenie jest niepożądane. Czas przedwczesnego wykonania zadania (przyspieszenie zadania) oraz czas zbyt późnego wykonania zadania (opóźnienie zadania) będą nazywane *nieterminowością wykonania zadania*.

Uogólnieniem koncepcji pożądanego momentu zakończenia wykonywania zadań jest koncepcja *pożądanego przedziału zakończenia wykonywania zadań*. W problemach z pożądanymi przedziałami zakończenia wykonywania zadań, zadanie nie wnosi żadnej kary w sytuacji, gdy zakończy wykonywać się wewnątrz pożądanego przedziału zakończenia jego wykonywania. Szczególne praktyczne znaczenia ma grupa problemów, w których pożądany przedział zakończenia wykonywania zadań nie jest z góry zadany, ale dobierany w procesie optymalizacji.

W niniejszej pracy rozważane będą właśnie problemy szeregowania z doborem pożądanego przedziału zakończenia wykonywania zadań. W problemach tych minimalizacji podlega kryterium, w którego skład wchodzą sumy kar związanych z przedwczesnym lub zbyt późnym momentem zakończenia wykonywania zadania. Na *rozwiązanie* (*harmonogram*) tego problemu składają się:

- uszeregowanie zadań na procesorach;
- wyznaczenie początku i końca pożądanego przedziału zakończenia wykonywania zadań.

Cel pracy

Cel niniejszej pracy można sformułować w następujący sposób:

Wyznaczenie *złożoności obliczeniowej*, wykazanie *szczególnych własności* oraz skonstruowanie *algorytmów rozwiązania* wybranych problemów szeregowania zadań z doborem pożądanego przedziału zakończenia ich wykonywania przy minimalizacji jednego spośród następujących kryteriów:

- ważona suma nieterminowości wykonania zadań;
- suma nieliniowych kar za nieterminowe wykonanie zadań;
- suma ważonych nieterminowości wykonania zadań;
- liczba nieterminowo wykonanych zadań;
- suma ważonych przyspieszeń wykonania zadań i liczba opóźnionych zadań.

Przedstawiony cel pracy został sformułowany na podstawie przeprowadzonego przeglądu literatury naukowej, z którego wynika, że badane problemy były rozpatrywane dotąd w niewystarczającym zakresie, lub nie były rozpatrywane wcale. Jednocześnie są one bardzo istotne z teoretycznego, a zwłaszcza z praktycznego punktu widzenia, ponieważ modelują wiele rzeczywistych procesów produkcyjnych.

Zakres pracy

Całość prezentowanego w niniejszej rozprawie materiału podzielono na 10 rozdziałów.

W Rozdziale 2 opisano sposób formułowania problemów szeregowania oraz metody rozwiązywania tych problemów.

W Rozdziale 3 przedstawiono praktyczne zastosowania analizowanych w pracy problemów.

Rozdział 4 opisuje aktualny stan badań związanych z rozpatrywaną w pracy tematyką. Najistotniejsze rezultaty naukowe zaprezentowano w nim za pomocą tabelarycznych podsumowań.

W Rozdziale 5 przedstawiono uzyskane przez autora rezultaty dla *jednoprocesorowego* problemu szeregowania z *liniowymi* oraz *identycznymi* dla wszystkich zadań funkcjami kar za nieterminowe ich wykonanie. W badanym problemie zarówno początek, jak i koniec pożądanego przedziału zakończenia wykonywania zadań są zmiennymi decyzyjnymi, z tym że szerokość tego przedziału jest ograniczona z dołu i z góry. Dla problemu tego zaprezentowano wielomianowy algorytm optymalny.

Rozdział 6 poświęcono uzyskanym przez autora rezultatom dla *wieloprocesorowego* problemu szeregowania z *nieliniowymi* oraz *identycznymi* dla wszystkich zadań funkcjami kar za nieterminowe ich wykonanie. Podobnie jak poprzednio, w badanym problemie zarówno początek, jak i koniec pożądanego przedziału zakończenia wykonywania zadań są zmiennymi decyzyjnymi a szerokość tego przedziału jest ograniczona z dołu i z góry. Dla analizowanego problemu oraz jego szczególnego przypadku jednoprocesorowego skonstruowano algorytmy optymalne oparte na metodzie programowania dynamicznego.

Rozdział 7 poświęcono uzyskanym rezultatom dla *wieloprocesorowego* problemu szeregowania z kryterium minimalizacji sumy *ważonych opóźnień i przyspieszeń* wykonania zadań. W problemie tym należy dobrać początek i koniec pożądanego przedziału zakończenia wykonywania zadań przy dolnym i górnym ograniczeniu szerokości tego przedziału. Przedstawiony został algorytm oparty na metodzie programowania dynamicznego, który rozwiązuje optymalnie ten problem. Przeanalizowano liczne przypadki szczególne tego problemu, w tym przypadek jednoprocesorowy oraz przypadki z jednostkowymi czasami wykonywania, dla których skonstruowano odpowiednio: w pełni wielomianowy schemat aproksymacyjny oraz wielomianowe algorytmy optymalne.

W Rozdziale 8 analizowano dwa *jednoprocesorowe* problemy szeregowania z kryterium

minimalizacji *ważonej liczby opóźnionych i przedwcześnie wykonanych* zadań. W pierwszym problemie początek i koniec pożądanego przedziału zakończenia wykonywania zadań są zmiennymi decyzyjnymi. Natomiast w drugim problemie, zmienną decyzyjną jest tylko początek tego przedziału, podczas gdy jego szerokość jest z góry ustalona. Dla pierwszego problemu skonstruowano wielomianowy algorytm optymalny, natomiast dla drugiego problemu opracowano pseudowielomianowy algorytm optymalny oraz w pełni wielomianowy schemat aproksymacyjny.

Rozdział 9 poświęcono uzyskanym przez autora rezultatom dla *jednoprocesorowego* problemu szeregowania, z kryterium minimalizacji *ważonej liczby opóźnionych* zadań oraz sumy *ważonych przyspieszeń* wykonania zadań. W problemie tym należy dobrać początek i koniec pożądanego przedziału zakończenia wykonywania zadań. Skonstruowano algorytm pseudowielomianowy, który rozwiązuje optymalnie ten problem.

Ostatni rozdział stanowi podsumowanie osiągniętych w pracy rezultatów.

Autor niniejszej rozprawy wyraża swoje podziękowania wszystkim osobom, które przyczyniły się do jej powstania. Należą się one przede wszystkim promotorowi Panu prof. dr. hab. inż. Adamowi Janiakowi oraz współopiekunowi Panu prof. Mikhailowi Y. Kovalyovowi za cenne uwagi i sugestie, które wpłynęły na ostateczny kształt pracy. Autor wyraża swoją wdzięczność swoim kolegom i współpracownikom m. in. Panom: dr. inż. Maciejowi Lichtensteinowi oraz dr. inż. Tomaszowi Krysiakowi za owocne dyskusje i uwagi.

Rozdział 2

Podstawowe pojęcia dotyczące szeregowania zadań

Niniejsza Praca dotyczy wybranych problemów szeregowania zadań. W związku z tym, przed przystąpieniem do ich analizy, zaprezentowane zostaną podstawowe pojęcia związane z teorią szeregowania zadań. W Podrozdziale 2.1 przedstawiono ogólne zasady formułowania problemów szeregowania zadań. Następnie, w Podrozdziale 2.2 opisano powszechnie stosowane metody rozwiązywania takich problemów.

2.1 Formułowanie problemów szeregowania zadań

Dwa podstawowe terminy w teorii szeregowania to: *zadanie* (zlecenie) oraz *procesor* (maszyna). Można zdefiniować je następująco:

Zadanie jest to zbiór operacji (czynności), które należy wykonać celem otrzymania pewnego produktu finalnego, np. obróbka półproduktu przez określone urządzenie, przetworzenie pewnej ilości danych przez procesor itp.

Procesor jest to obiekt wykonujący operacje, z których składają się zadania, np. fizyczny procesor w komputerze, obrabiarka (frezarka, tokarka, itp.). Często przez procesor będziemy także rozumieli człowieka wykonującego zadanie.

Rzeczywiste systemy, modelowane przez problemy szeregowania zadań, mają różną złożoność. Najprostszym systemem jest **system jednoprocesorowy**, w którym wszystkie zadania składają się z dokładnie jednej operacji i są przetwarzane przez jeden procesor. W przypadku, gdy mamy do dyspozycji kilka procesorów, z których każdy może przetwarzać wszystkie dostępne zadania, mówimy o **systemie wieloprocesorowym** z procesorami równoległymi. W tym przypadku również, podobnie jak w systemie jednoprocesorowym, zadania składają się z jednej operacji. W zależności od rodzaju dostępnych procesorów równoległych, wyróżniamy:

- System z równoległymi procesorami *identycznymi*, w którym czas przetwarzania danego zadania jest taki sam na każdym procesorze (ang. *identical parallel processors*).
- System z równoległymi procesorami *jednorodnymi*, w którym poszczególne procesory przetwarzają zadania z różną prędkością (ang. *uniform parallel processors*).
- System z równoległymi procesorami *niejednorodnymi*, w którym prędkość przetwarzania danego procesora zależy od tego, jakie zadanie procesor ten w danej chwili wykonuje (ang. *unrelated parallel processors*).

Wyróżnia się także systemy: **przepływowy** (ang. *flow shop*), w którym każde zadanie składa się z takiej samej liczby operacji; kolejność wykonywania poszczególnych operacji jest jednakowa dla każdego zadania; **gniazdowy** (ang. *job shop*), w którym zadania składają się z takiej samej liczby operacji a kolejność wykonywania poszczególnych operacji zadania przez kolejne procesory może być dla różnych zadań różna; **otwarty** (ang. *open shop*), w którym kolejność wykonywania poszczególnych operacji zadania przez kolejne procesory nie jest zadana i stanowi jeden z celów optymalizacji.

W niniejszej rozprawie analizowane będą jedynie systemy jednoprocesorowe, systemy z identycznymi procesorami równoległymi oraz systemy z jednorodnymi procesorami równoległymi. W związku z tym w dalszej części pracy zakładamy, że zadania składają się tylko z jednej operacji. Rozpatrywać będziemy zbiór n zadań $\mathbf{J} = \{1, \ldots, n\}$ do wykonania na m równoległych procesorach.

Każde zadanie $j \in \mathbf{J}$ opisuje pewien zbiór parametrów. Najczęściej definiowane są następujące parametry:

- p_j – *czas wykonywania* zadania j (ang. *processing time*);
- r_j – *termin dostępności* zadania j (ang. *release date*);
- d_j – *najpóźniejszy pożądany termin zakończenia* wykonywania zadania j (ang. *latest due date*);
- e_j – *najwcześniejszy pożądany termin zakończenia* wykonywania zadania j (ang. *earliest due date*);
- $\bar{d}_j$ – *nieprzekraczalny termin zakończenia* wykonywania zadania j (ang. *deadline*);

Parametry e_j oraz d_j określają odpowiednio początek i koniec *pożądanego przedziału zakończenia wykonywania zadań*. Jeżeli $e_j = d_j$, to mamy do czynienia z pożądanym *momentem* zakończenia wykonywania zadań.

Dla danego problemu szeregowania, w odniesieniu do określonych par zadań, mogą być zdefiniowane pewne *ograniczenia kolejnościowe* (ang. *precedence constraints*). Ograniczenia te wymuszają rozpoczęcie wykonywania jednego z zadań z pary, dopiero po zakończeniu wykonywania drugiego. Może również istnieć ograniczenie na podzielność zadań,

które skutkuje tym, że rozpoczęte zadanie nie może być przerwane, wówczas zadania są *niepodzielne* (ang. *non-preemptive*). W przeciwnym przypadku, zadania są *podzielne* (ang. *preemptive*). W niniejszej pracy założono, że nie ma ograniczeń kolejnościowych i że wszystkie zadania są niepodzielne.

W celu rozwiązania problemu szeregowania z procesorami równoległymi należy znaleźć:

- przyporządkowanie zadań do procesorów;
- momenty rozpoczęcia i zakończenia wykonywania zadań;
- wartości pewnych dodatkowych parametrów specyficznych dla danej grupy problemów.

Rozwiązanie określamy mianem *rozwiązania dopuszczalnego*, jeżeli spełnione są następujące warunki:

- W każdym momencie czasu procesor wykonuje co najwyżej jedno zadnie.
- W każdym momencie czasu każde zadanie jest wykonywane przez co najwyżej jeden procesor.
- Wszystkie zadania muszą zostać wykonane.

Ponadto, muszą być uwzględnione wszystkie ograniczenia wynikające ze specyfiki konkretnego problemu, np.: ograniczenia kolejnościowe, nieprzekraczalne terminy zakończenia $\bar{d}_j$, terminy dostępności r_j, itp.

Dla każdego zadania $j \in \mathbf{J}$, możemy wyznaczyć następujące wielkości:

- S_j – *termin rozpoczęcia* wykonywania zadania j (ang. *starting time*);
- C_j – *termin zakończenia* wykonywania zadania j (ang. *completion time*);
- $T_j \triangleq \max\{0, C_j - d_j\}$ – *opóźnienie* wykonania zadania j (ang. *tardiness*);
- $E_j \triangleq \max\{0, e_j - C_j\}$ – *przyspieszenie* wykonania zadania j (ang. *earliness*);
- $L_j \triangleq E_j + T_j$ – *nieterminowość* wykonania zadania j (ang. *lateness*);
- $V_j \triangleq \begin{cases} 0 & \text{jeżeli } C_j \geqslant e_j, \\ 1 & \text{jeżeli } C_j < e_j; \end{cases}$
- $U_j \triangleq \begin{cases} 0 & \text{jeżeli } C_j \leqslant d_j, \\ 1 & \text{jeżeli } C_j > d_j. \end{cases}$

Ocena rozwiązań problemów dokonywana jest przy pomocy różnych **kryteriów** (zwanych także funkcjami celu lub funkcjami kryterialnymi). Najczęściej rozpatrywane są kryteria minimalizacji następujących wielkości:

- $C_{max} = \max_{j \in \mathbf{J}}\{C_j\}$ – długość uszeregowania (ang. *makespan*);
- $L_{max} = \max_{j \in \mathbf{J}}\{L_j\}$ – maksymalna nieterminowość (ang. *maximum lateness*);
- $T_{max} = \max_{j \in \mathbf{J}}\{T_j\}$ – maksymalne opóźnienie zadania (ang. *maximum tardiness*);
- $\sum(w_j)C_j = \sum_{j \in \mathbf{J}}(w_j)C_j$ – suma (ważonych) terminów zakończenia wykonywania zadań (ang. *total (weighted) completion time*);
- $\sum(w_j)T_j = \sum_{j \in \mathbf{J}}(w_j)T_j$ – suma (ważonych) opóźnień zadań (ang. *total (weighted) tardiness*);
- $\sum(\alpha_j)V_j + \sum(\beta_j)U_j = \sum_{j \in \mathbf{J}}(\alpha_j)V_j + \sum_{j \in \mathbf{J}}(\beta_j)U_j$ – (ważona) liczba zbyt wcześnie oraz zbyt późno wykonanych zadań (ang. *total (weighted) number of early and late jobs*);
- $\sum(\alpha_j)E_j + \sum(\beta_j)T_j = \sum_{j \in \mathbf{J}}(\alpha_j)E_j + \sum_{j \in \mathbf{J}}(\beta_j)T_j$ – suma (ważonych) przyspieszeń i opóźnień zadań (ang. *total (weighted) earliness-tardiness*).

W celu zwięzłego opisu problemów szeregowania zadań stosuje się powszechnie tzw. *notację trójpolową* $A \mid B \mid C$, wprowadzoną w pracy [34].

W notacji tej pole A oznacza typ modelowanego systemu i może przyjmować następujące wartości:

- 1 – system jednoprocesorowy;
- P – system z identycznymi procesorami równoległymi;
- Q – system z jednorodnymi procesorami równoległymi;
- R – system z niejednorodnymi procesorami równoległymi;
- F – system przepływowy;
- J – system gniazdowy;
- O – system otwarty.

W przypadku systemów wieloprocesorowych, obok odpowiedniego symbolu może pojawić się dodatkowo litera m (np., Pm), określająca ustaloną liczbę procesorów. Oznacza to, że liczba procesorów nie jest parametrem wejściowym problemu.

Pole B opisuje dodatkowe wymagania lub ograniczenia, np.:

- $p_j = p$ – czasy wykonywania wszystkich zadań są identyczne, równe p;
- r_j – zadania posiadają niezerowe terminy dostępności;
- $\bar{d_j}$ – dla zadań określono nieprzekraczalne terminy zakończenia;

- *pmtn* – dopuszcza się przerywanie wykonywania zadań;
- *prec* – zdefiniowane są ograniczenia kolejnościowe dla zadań.
- *tree* – występują ograniczenia kolejnościowe typu drzewo w wykonywaniu zadań;
- *SP* – występują ograniczenia kolejnościowe wykonania zadań w postaci grafu szeregowo-równoległego;

W polu C definiuje się optymalizowane kryterium, np.: $C_{max}, \sum C_j, \sum w_j U_j$, itd.

2.2 Metody rozwiązywania problemów szeregowania

Algorytmy rozwiązania problemów szeregowania zadań można podzielić na dwie zasadnicze grupy: *algorytmy dokładne* (dostarczające rozwiązań optymalnych) i *algorytmy przybliżone* (dostarczające rozwiązań nieoptymalnych). Algorytmy przybliżone stosuje się przede wszystkim w celu rozwiązania problemów, dla których nie skonstruowano wielomianowych algorytmów dokładnych.

Metody dokładne

Do metod dokładnych zaliczają się: wielomianowe algorytmy dokładne, programowanie dynamiczne, programowanie całkowitoliczbowe, metoda podziału i ograniczeń.

Wielomianowe algorytmy dokładne

Wielomianowe algorytmy dokładne dostarczają rozwiązań optymalnych w czasie wielomianowym. Algorytmy takie można skonstruować jedynie dla problemów z klasy P.

Programowanie dynamiczne

Istnieje wiele interpretacji metody Programowania Dynamicznego. Poniżej będziemy rozważać Programowanie Dynamiczne jako proces konstrukcji zbiorów rozwiązań cząstkowych i wyboru rozwiązania *dominującego* z każdego z tych zbiorów. Wybór dokonywany jest w taki sposób, że przynajmniej jedno rozwiązanie dominujące może być rozszerzone do rozwiązania *optymalnego*.

W przypadku problemów szeregowania zadań rozwiązaniami cząstkowymi są zazwyczaj uszeregowania, które zawierają k ($k \leqslant n$) pierwszych zadań.

Z każdym rozwiązaniem cząstkowym skojarzony jest pewien zbiór $i + 1$ parametrów $V = (k, v_1, \ldots, v_i)$, zwyczajowo nazywanych zmiennymi stanu. Zbiór V zmiennych stanu jest nazywany stanem. Natomiast zbiór wszystkich stanów jest nazywany przestrzenią stanów.

Niech $X(V)$ oznacza zbiór rozwiązań cząstkowych będących w stanie V. W metodzie Programowania Dynamicznego, z każdym stanem V skojarzona jest pewna funkcja $\Phi(V)$, którą będziemy nazywać funkcją dominacji. Zazwyczaj minimalizuje lub maksymalizuje ona pewną wartość na zbiorze $X(V)$. Cząstkowe rozwiązanie w stanie V, które dostarcza wartości $\Phi(V)$ dominuje wszystkie pozostałe rozwiązania cząstkowe. Oznacza to, że może ono być rozszerzone do kompletnego rozwiązania z maksymalną wartością funkcji kryterialnej spośród wszystkich kompletnych rozwiązań, rozszerzonych z rozwiązań cząstkowych będących w stanie V. Z powyższego wynika, że w każdym zbiorze $X(V)$ wystarczy wybrać tylko rozwiązanie dominujące do dalszego rozszerzania.

W metodzie Programowania Dynamicznego, wartości funkcji dominacji $\Phi(V)$ są rekurencyjnie wyliczane dla różnych stanów a na końcu wybierany jest stan odpowiadający rozwiązaniu optymalnemu.

Optymalne rozwiązanie może zostać znalezione poprzez zastosowanie procedury iteracyjnej, zwanej *procedurą poszukiwania wstecznego* (ang. *backtracking*).

Zazwyczaj złożoność obliczeniowa algorytmów opartych na metodzie programowania dynamicznego zależy bezpośrednio od wielkości przestrzeni stanów.

Programowanie całkowitoliczbowe

Niektóre problemy można rozwiązać poprzez transformację ich do problemów programowania całkowitoliczbowego. Niestety postępowanie to dostarcza zazwyczaj zadań programowania całkowitoliczbowego o tak dużych rozmiarach, że standardowe metody ich rozwiązywania nie umożliwiają uzyskania wyniku w rozsądnym czasie.

Metoda podziału i ograniczeń

Działanie metody podziału i ograniczeń polega na bezpośrednim lub pośrednim badaniu wszystkich rozwiązań problemu. Schemat działania można przedstawić w postaci *drzewa poszukiwań*. Każdy węzeł drzewa odpowiada podzbiorowi rozwiązań dopuszczalnych danego problemu. W procesie poszukiwań, przegląda się poszczególne węzły drzewa (poczynając od korzenia) i ocenia, czy w podzbiorze rozwiązań, odpowiadającym danemu węzłowi, może znajdować się rozwiązanie optymalne. Jeśli nie, to węzły poddrzewa, poczynając od węzła analizowanego, są odrzucane z dalszego procesu poszukiwań. W przeciwnym przypadku, z danego węzła generowane są kolejne węzły. Proces przeszukiwania drzewa trwa tak długo, aż dokonamy bezpośredniego lub pośredniego przeglądu wszystkich węzłów w drzewie (tzn., dopóki wszystkie możliwe rozwiązania problemu nie zostaną sprawdzone bądź odrzucone).

Metody przybliżone

Algorytmy przybliżone warto stosować wtedy, gdy czas potrzebny na uzyskanie rozwiązania optymalnego jest zbyt długi. Najważniejszymi aspektami, na które zwraca się uwagę przy konstruowaniu algorytmów przybliżonych, jest czas ich działania oraz dokładność dostarczanych rozwiązań. Wśród metod przybliżonych wyróżnić należy: algorytmy konstrukcyjne, algorytmy typu popraw, schematy aproksymacyjne.

Algorytmy konstrukcyjne

Działanie algorytmów konstrukcyjnych opiera się najczęściej na statycznych lub dynamicznych regułach priorytetowych, zastosowaniu relaksacji do problemu prostszego lub aproksymacji rozwiązania w oparciu o inne (prostsze) problemy. W celu skonstruowania rozwiązania problemu stosowane są również bardziej wyrafinowane metody, korzystające ze specjalnych własności danego problemu. Algorytmy te charakteryzują się z reguły dużą szybkością działania.

Algorytmy typu popraw

Algorytmy popraw dzielą się na algorytmy lokalnego poszukiwania oraz algorytmy metaheurystyczne.

Zasada działania *algorytmów lokalnego poszukiwania* polega na przeglądaniu sąsiedztwa danego rozwiązania i wyborze w nim rozwiązania kolejnego. Proces ten jest powtarzany przez pewną liczbę iteracji do momentu gdy zostaną spełnione warunki stopu algorytmu. Zazwyczaj warunkiem stopu jest osiągnięcie minimum lokalnego.

Algorytmy metaheurystyczne są wynikiem zastosowania elementów sztucznej inteligencji do rozwiązywania problemów kombinatorycznych. Działają one podobnie do algorytmów poszukiwania lokalnego, jednak posiadają dużo bardziej wyrafinowane metody przeglądania sąsiedztwa oraz wyboru kolejnego rozwiązania. Podstawową cechą tych algorytmów jest to, że potrafią opuścić minimum lokalne.

Schematy aproksymacyjne

Są to rodziny algorytmów przybliżonych, które dostarczają rozwiązań o zadanej dokładności. Jeżeli czas działania tych algorytmów zależy wielomianowo od rozmiaru problemu, to mówimy wtedy o *wielomianowych* schematach aproksymacyjnych. Jeżeli natomiast czas ich działania zależy wielomianowo zarówno od rozmiaru, jak i od odwrotności zadanej dokładności, to mówimy wówczas o *w pełni wielomianowych* schematach aproksymacyjnych.

W odniesieniu do algorytmów przybliżonych celowa jest ich analiza pod kątem jakości dostarczanych rozwiązań. Najczęściej stosowane są następujące podejścia przy analizie efektywności algorytmów.

- **Analiza probabilistyczna** – Metoda ta polega na analitycznym wykazaniu statystycznej zbieżności algorytmu do rozwiązania optymalnego. W literaturze można spotkać niewiele przykładów takiej analizy, ponieważ jest ona bardzo trudna do przeprowadzenia.
- **Analiza najgorszego przypadku** – W podejściu tym wyznaczana jest różnica względna pomiędzy rozwiązaniem optymalnym a najgorszym możliwym, jakie może zostać skonstruowane przez analizowany algorytm. Podejście to jest powszechnie stosowane wśród wielu projektantów algorytmów przybliżonych
- **Analiza eksperymentalna** – Metoda ta polega na ocenie zachowania się algorytmu (błąd przybliżenia) w oparciu o wyniki przebiegu algorytmu na ograniczonej, reprezentatywnej próbce instancji. Mimo jej niedoskonałości, jest to najbardziej popularna metoda analizy algorytmów przybliżonych.

Reasumując, rozwiązanie danego problemu szeregowania zadań polega na jego precyzyjnym sformułowaniu, określeniu jego złożoności obliczeniowej i skonstruowaniu dla niego odpowiednich, efektywnych obliczeniowo, algorytmów rozwiązania.

Dla problemów wielomianowo rozwiązywalnych, właściwym algorytmem jest oczywiście optymalny algorytm wielomianowy. Dla problemów NP-trudnych są to najczęściej algorytmy pseudowielomianowe i/lub algorytmy przybliżone.

Rozdział 3

Zastosowania praktyczne

W niniejszym rozdziale przedstawione zostaną przykłady rzeczywistych procesów produkcyjnych i usługowych, które mogą być modelowane przez problemy szeregowania z doborem pożądanego przedziału zakończenia wykonywania zadań.

Organizacja pracy laboratorium przy inwentaryzacji zrzutu ścieków z zakładów przemysłowych

Przeprowadzenie inwentaryzacji zrzutu ścieków przemysłowych polega na poborze i przetransportowaniu próbek ścieków z zakładów przemysłowych do laboratorium oraz wykonaniu w nim szeregu oznaczeń (analiz) chemicznych celem określenia emisji poszczególnych zanieczyszczeń do środowiska przez zakłady przemysłowe. Inwentaryzacja zrzutu ścieków obejmuje przydzielony laboratorium obszar równy co najmniej powierzchni województwa, a tym samym dotyczy setek zakładów przemysłowych.

Standardowo laboratorium realizuje pewną ilość oznaczeń, do wykonywania których posiada wymagane odczynniki i sprzęt, a także wykwalifikowany personel. Przeprowadzenie inwentaryzacji wymaga wykonania niestandardowych oznaczeń, do których wymagany jest osobny sprzęt laboratoryjny, często także specyficzne odczynniki chemiczne (np. wzorce do oznaczeń). Z uwagi na stan wyposażenia laboratoriów (zwłaszcza państwowych jednostek ochrony środowiska) praktykowane jest wypożyczanie na czas określony sprzętu laboratoryjnego od innych laboratoriów lub wyspecjalizowanych wypożyczalni. Specjalistyczna aparatura laboratoryjna jest bardzo kosztowna, np. zakup analizatora rtęci typu AMA to wydatek rzędu 150 tys. zł, analizator spektroskopii atomowej do oznaczania metali kosztuje około 200 tys. zł. Z uwagi na to, w odniesieniu do okresowo realizowanych badań, koszt zakupu wymaganych do nich aparatów nie uległby amortyzacji. Znacząco mniejszy jest koszt wypożyczenia sprzętu, przy czym płatny jest każdy dzień korzystania ze sprzętu. W przypadku, gdy laboratorium nie jest w stanie wykonać pewnych oznaczeń z powodu braku dostępu w danej chwili do specjalistycznego aparatu, możliwe jest podzlecenie wykonania analizy innemu laboratorium. W praktyce oznacza to zwiększenie

kosztów realizacji zlecenia ze względu na marżę podwykonawcy. Może jednak zaistnieć taka sytuacja, że w pewnym momencie bardziej opłaca się podzlecić wykonanie pewnej ilości oznaczeń do innego laboratorium, niż ponosić koszty za wydłużenie okresu wynajmu specjalistycznego aparatu.

W przypadku realizacji inwentaryzacji zrzutu ścieków poboru prób dokonuje, zgodnie z wymaganiami określonymi w przepisach, kompetentny personel przygotowany do właściwego wyboru miejsca i sposobu poboru prób oraz ich utrwalenia i transportu. Czas realizacji poszczególnych zadań jest różny w zależności od odległości miejsc poboru prób. Próbka po przywiezieniu do laboratorium poddawana jest wymaganym badaniom przez odpowiednio przygotowany personel laboratorium. Jeżeli z powodu braku dostępu do aparatu nie ma możliwości wykonania pewnych oznaczeń w ciągu 24 godzin od momentu poboru próbek ścieków, konieczne jest ich dodatkowe utrwalenie i przechowywanie w chłodniach laboratoryjnych.

Poniżej opisany jest sposób zamodelowania przedstawionego powyżej problemu przy pomocy problemu szeregowania z doborem pożądanego przedziału zakończenia wykonywania zadań. Osobę pobierającą próbki wraz z przydzielonym środkiem transportu będziemy interpretować jako procesor (maszynę). Operacja pobrania i przetransportowania próbek ścieków będzie odpowiadać wykonaniu zadania (wykonanie oznaczeń chemicznych nie jest już traktowane jako część zadania). Pożądany przedział zakończenia wykonywania zadań charakteryzuje okres wynajmu specjalistycznej aparatury, a tym samym okres, w którym możliwe będzie wykonywanie oznaczeń na tym aparacie. Koszt przedwczesnego wykonania zadania to koszt dodatkowego chemicznego utrwalenia próbek oraz ich przechowywania w chłodniach do momentu, kiedy dostępny będzie wynajęty aparat do badań. Natomiast koszt zbyt późnego wykonania zadania jest związany z koniecznością zapłacenia podwykonawcy za wykonane analizy, ze względu na dostarczenie próbki do laboratorium po zakończeniu okresu wynajmu aparatu niezbędnego do wykonania tego oznaczenia.

Należy zatem wyznaczyć *harmonogram* poboru próbek ścieków oraz ustalić *okres wynajmu* specjalistycznej aparatury tak, aby zminimalizować sumę następujących kosztów:

- wynajmu aparatury;
- chemicznego utrwalenia próbek i przechowywania ich w chłodniach laboratoryjnych;
- zlecenia wykonania analiz innym laboratoriom.

W laboratoriach badawczych i laboratoriach chemicznych istnieje wiele sytuacji podobnych do opisanej powyżej. Przykładem może być prowadzenie reakcji z zakresu chemii organicznej, w których wykorzystuje się szereg nietrwałych reagentów (reagenty organiczne ulegają reakcjom utleniania lub rozkładu). Produkt jednej syntezy organicznej jest często substratem następnej. Jeżeli reagenty są przygotowane przedwcześnie, przed procesem danej syntezy, to pojawiają się koszty ich utrwalania i przechowywania w specjal-

nych warunkach (np. w chłodniach). Z drugiej strony, zbyt długie oczekiwanie produktu tej syntezy na wykorzystanie w następnej oznacza pogarszanie się jego jakości i czystości.

Harmonogramowanie produkcji w odlewni żeliwa

Odlewnia żeliwa w Toowoombie wytwarza odlewy z żeliwa dla ciężkiego transportu, części samochodowe, odlewy do pomp i inne elementy. Razem wytwarzanych jest około 1500 różnych odlewów.

Odlewnia została wybudowana około 100 lat temu. Od tego czasu wokół odlewni powstały liczne zabudowania, ograniczając możliwość składowania większej ilości produkcji.

Kiedy klient składa zamówienie na dużą partię towaru, to przeprowadzane są negocjacje, podczas których ustalony jest zarówno najpóźniejszy, jak i najwcześniejszy termin odbioru towaru. Jeżeli produkt zostanie wytworzony przed najwcześniejszym terminem odbioru, to odlewnia ponosi koszty związane z magazynowaniem. Oznacza to bowiem konieczność wynajęcia pewnej powierzchni magazynowej i uiszczenie opłaty za każdy dzień wynajmu. Jeśli natomiast produkt zostanie wytworzony później niż najpóźniejszy termin odbioru, to odlewnia musi zapłacić karę określoną w umowie za każdy dzień opóźnienia.

Odlewnia preferuje sytuację, w której terminy najwcześniejszego i najpóźniejszego odbioru są od siebie maksymalnie odległe. Z drugiej strony, klient chciałby odebrać wszystkie zamówione odlewy w możliwie najwęższym przedziale czasowym.

Z powyższego wynika, że odlewnia musi, na podstawie dostarczonego przez klienta zamówienia, stworzyć wstępny harmonogram produkcji oraz ustalić terminy odbioru produktów, minimalizujące sumę wszystkich kosztów producenta. Terminy te następnie są przedstawiane klientowi, w celu podpisania ostatecznej umowy.

Sytuacja podobna do przedstawionej powyżej może zaistnieć w innych zakładach wytwórczych, w których zbyt wczesne wytworzenie produktu powoduje powstanie kosztów związanych magazynowaniem, ubezpieczeniem oraz zamrożeniem środków finansowych, natomiast zbyt późne wytworzenie produktu związane jest z karami określonymi w umowie i kosztami przesyłek ekspresowych.

Minimalizacja kosztów magazynowania i transportu w zakładzie produkcyjnym

Rozważmy zakład produkujący na zamówienie, usytuowany w centrum miasta i przez to nie posiadający wystarczającej powierzchni magazynowej. Zakład taki zmuszony jest do wynajmowania magazynów w pobliżu, co wiąże się ze znacznymi opłatami, lub musi przewozić wytworzone produkty do magazynów położonych znacznie dalej, poza centrum miasta. Koszt wynajęcia odleglejszych magazynów jest znikomo mały w porównaniu z kosztami wynajęcia magazynów położonych w centrum, należy jednak w tym przypadku uwzględnić dodatkowe koszty transportu.

W celu zredukowania kosztów transportu gotowych produktów do odbiorcy, większość z nich jest wysyłana do klienta w dużej partii, wybranym - najbardziej ekonomicznym środkiem transportu (np. pociąg, ciężarówka). Do momentu wysłania do klienta, towar jest przechowywany w magazynach. Produkty wytworzone po momencie wysłania głównej partii produktów są dostarczane do klienta innym - droższym środkiem transportu (przesyłka kurierska).

W przypadku realizacji zamówienia na większą partię produktów, zakład musi przygotować odpowiedni harmonogram produkcji, zdecydować na jaki czas opłacalne jest wynajęcie dodatkowej, znacznie droższej, powierzchni magazynowej w pobliżu zakładu oraz ustalić moment przesłania głównej partii produktów do klienta w celu minimalizacji kosztów magazynowania i transportu.

Poniżej zostanie przedstawiony sposób transformacji przedstawionego problemu do problemu szeregowania z doborem pożądanego przedziału zakończenia wykonywania zadań. Uszeregowanie zadań będzie harmonogramem produkcji. Koniec pożądanego przedziału zakończenia wykonywania zadań to moment wysłania głównej partii produktów do klienta, natomiast szerokość tego przedziału to czas dzierżawy magazynu znajdującego się w pobliżu zakładu produkcyjnego. Koszt przedwczesnego wykonania zadania jest kosztem transportu produktu do odleglejszego magazynu. Z drugiej strony, koszt opóźnienia wykonania zadania interpretujemy jako koszt przesłania droższym środkiem transportu produktu wytworzonego już po wysłaniu głównej partii produktów.

Wyznaczanie harmonogramów obsług technicznych dla statków powietrznych

Opisany poniżej problem wyznaczania harmonogramów obsług technicznych dla statków powietrznych został zaczerpnięty z rozprawy doktorskiej Pana Marcina Marka [86]. Problem ten można zamodelować przy pomocy rozważanych w pracy problemów.

Zgodnie z obowiązującymi regulacjami prawnymi, przewoźnik lotniczy zobowiązany jest posiadać zaplanowany na rok z góry harmonogram obsług technicznych swojej floty. Są one prowadzone przez odpowiednio przygotowany personel techniczny, który wykorzystuje w tym celu specjalistyczny sprzęt w przystosowanych do tego warsztatach. Warsztat zazwyczaj posiada więcej niż jedną linię obsługową (pozwala to na jednoczesne, równoległe prowadzenie obsług technicznych kilku statków powietrznych). Obsługi techniczne danego statku powietrznego nie muszą być prowadzone w miejscu jego macierzystego lotniska. Prowadzone obsługi techniczne mają charakter cykliczny, a ich długość uzależniona jest od modelu statku powietrznego i wypadającego dla niego przyrostu resursu godzinowego. Resurs ten jest informacją o nalocie godzinowym danego statku powietrznego w trakcie jednego cyklu obsług technicznych. Poniżej przedstawiony jest taki przykładowy cykl dla hipotetycznego statku powietrznego:

- po 50 godzinach nalotu wykonuje się krótką obsługę techniczną trwającą 7 dni;
- po 100 godzinach nalotu należy wykonać obsługę techniczną średniej długości trwającą 14 dni;
- po 150 godzinach nalotu ponownie należy wykonać krótką obsługę techniczną trwającą 7 dni;
- po 200 godzinach nalotu wykonuje się długą obsługę techniczną trwającą 21 dni.

Po całym cyklu obsług technicznych resurs godzinowy jest zerowany i cały proces obsługowy zaczyna się od początku.

Czas, który upłynął od momentu rozpoczęcia eksploatacji statku powietrznego nazywany jest resursem czasowym. Każdy statek powietrzny powinien być wycofany z dalszej eksploatacji po osiągnięciu ustalonego dla danego modelu limitu resursu czasowego (np. 30 lat).

Poniżej opisany jest sposób transformacji powyższego problemu harmonogramowania obsług technicznych dla floty statków powietrznych do problemu szeregowania zadania z doborem pożądanych przedziałów zakończenia wykonywania zadań. Harmonogram obsług będziemy interpretować jako uszeregowanie zadań. Linie obsługowe w warsztatach odpowiadają procesorom. Pożądane przedziały zakończenia wykonywania zadań charakteryzują czas pobytu pilota oblatywacza na lotnisku przywarsztatowym. Ich szerokość związana jest z kosztami wynagrodzenia, zakwaterowania i wyżywienia pilota oblatywacza. Wspomniane przedziały czasowe mogą być wspólne dla poszczególnych grup zadań, co jest związane z uprawnieniami, jakie posiada pilot oblatywacz (uprawnienia do oblotu mogą dotyczyć różnych modeli statków powietrznych). Jeżeli obsługa techniczna statku powietrznego zakończy się przed przybyciem pilota oblatywacza, pojawiają się koszty związane z upływem resursu czasowego dla tego statku powietrznego przy braku jego eksploatacji oraz koszty przechowywania statku powietrznego w przywarsztatowym hangarze. Jeśli natomiast obsługa techniczna statku powietrznego zakończy się po okresie oblotów dokonywanych przez pilota oblatywacza, to zadania transportowe przewidziane dla tego statku powietrznego będą musiały być wykonane przy użyciu innego statku powietrznego (planując dla niego dodatkowe wyloty). Powoduje to powstanie kosztów związanych z utrzymaniem w dłuższej pracy urządzeń elektronawigacyjnych i radionawigacyjnych na lotniskach oraz kosztów wydłużenia czasu pracy personelu technicznego obsługującego te lotniska.

Rozdział 4

Stan badań

4.1 Wstęp

W niniejszym rozdziale dokonano przeglądu dorobku naukowego, którego tematyka stanowi genezę bądź jest związana z rozpatrywanymi w pracy problemami szeregowania.

W Podrozdziale 4.2 przedstawione są uzyskane w literaturze rezultaty dotyczące problemów szeregowania z pożądanymi momentami zakończenia wykonywania zadań, w których powstaje koszt, jeżeli zadanie zakończy wykonywać się przed lub po wyznaczonym momencie. Natomiast rezultaty dotyczące problemów z pożądanymi przedziałami zakończenia wykonywania zadań, będące uogólnieniem problemów z pożądanymi momentami zakończenia wykonywania zadań, przedstawione są w Podrozdziale 4.3. Niniejszy rozdział zamyka krótkie podsumowanie.

4.2 Problemy z pożądanymi momentami zakończenia wykonywania zadań

Pożądane terminy zakończenia wykonywania zadań i związana z nimi minimalizacja nieterminowości wykonania zadań są od pewnego czasu przedmiotem dużego zainteresowania wśród naukowców. Powszechnie uważa się, że praca Jacksona z 1955 [43] stanowi punkt startowy, inicjujący badania nad tą właśnie problematyką. Różne aspekty rozpatrywania pożądanych terminów zakończenia wykonywania zadań są tematem następujących prac przeglądowych: Sena i Gupty z 1984 [93], Raghavarchariego z 1988 [91], Bakera i Scuddera z 1990 [11] oraz Koulamasa z 1994 [70]. Obszerny przegląd problemów szeregowania, w których dobór pożądanych terminów zakończenia wykonywania zadań odbywa się w procesie optymalizacyjnym, znajduje się w pracach Gordona, Protha i Chu z 2002 roku [30] i [31].

W literaturze rozpatrywano wiele modeli doboru pożądanego przedziału zakończenia wykonywania zadań. Poniżej przedstawione zostaną dwa z nich, które są najbliższe pro-

blematyce rozważanej w niniejszej pracy:

- CON (CONstant number): $d_j = d$,
- SLK (equal SLacKs): $d_j = d + (r_j) + p_j$,

gdzie d oznacza parametr doboru pożądanych terminów zakończenia wykonywania zadań. Oznaczenia powyższych modeli doboru pożądanego terminu zakończenia wykonywania zadań umieszcza się w drugim polu notacji trójpolowej.

Najczęściej rozpatrywany jest model CON, w którym pożądany termin zakończenia wykonywania zadań jest wspólny dla wszystkich zadań i jest dobierany w procesie optymalizacyjnym. W modelu SLK wartość pożądanego terminu zakończenia wykonywania zadania jest równa sumie jego czasu wykonywania oraz pewnej wspólnej dla wszystkich zadań zmiennej decyzyjnej.

Najistotniejsze rezultaty dotyczące problemów szeregowania z doborem pożądanych terminów zakończenia wykonywania zadań zamieszczono w kolejnych dwóch tabelach. Tabela 4.1 uwzględnia te problemy, w których występuje model CON. Tabela 4.2 zawiera rezultaty dotyczące problemów, w których występuje model SLK. W tabelach poszczególne problemy przedstawiono w omówionej w Rozdziale 2 trójpolowej konwencji zapisu.

W przedstawionych poniżej tabelach (oraz tych znajdujących się w kolejnym podrozdziale) pojawiają się dodatkowo następujące oznaczenia:

- ε – dopuszczalne odchylenie od pożądanego terminu zakończenia wykonywania zadania;
- d_0 – preferowana wartość pożądanych terminów zakończenia wykonywania zadań;
- $\alpha, \alpha_j, \alpha', \alpha'_j, \beta, \beta_j, \beta', \beta'_j, \gamma, \delta, \theta, w_j$ – wagi;
- g, h– funkcje monotonicznie niemalejące.

Tabela 4.1: Najistotniejsze rezultaty osiągnięte dla problemów szeregowania, w których występuje model CON.

Lp.	**Problem**	**Złożoność obliczeniowa**	**Dorobek badawczy**
1.	$1 \mid d_j = d \mid \sum (E_j + T_j)$	$O(n \log n)$	Kanet (1981)[67], Panwalker i in. (1982)[89], Bagchi i in. (1986)[8], Hall (1986)[88], Bagchi i in. (1987)[10]
2.	$1 \mid d_j = d \mid \sum (\alpha E_j + \beta T_j)$	$O(n \log n)$	Panwalker i in. (1982)[89], Bagchi i in. (1987)[10]
3.	$1 \mid d_j = d \mid \sum (\alpha E_j + \beta T_j + \gamma d)$	$O(n \log n)$	Panwalker i in. (1982)[89]
4.	$1 \mid d_j = d \mid \sum (\alpha E_j + \beta T_j) + \gamma \max \{0, d - d_0\}$	$O(n \log n)$	Seidmann i in. (1981)[92]
5.	$1 \mid d_j = d \mid \sum \alpha_j (E_j + T_j)$	NP-trudny	Hall i Posner (1991) [38]; *alg. prog. dyn.*: Hall i Posner (1991)[38], De i in. (1990)[22], Jurisch i in. (1997) [63]; *alg. podz. i ogr.*: Cheng (1990) [17]; *sch. aproks.*: Kovalyov i Kubiak (1999) [74]; *heuryst.*: Hao i in. (1996) [39]
6.	$1 \mid d_j = d \mid \sum (\alpha_j E_j + \beta_j T_j)$	NP-trudny	*alg. podz. i ogr.*: Dileepan (1993)[26], De i in. (1994)[25], *heuryst.*: Gupta i in. (1990) [36], Dileepan (1993) [26], De i in. (1994) [25], James(1997) [44], Lee i Kim (1995) [80]
7.	$1 \mid d_j \in \langle d - \varepsilon, d + \varepsilon \rangle \mid \sum (E_j + T_j)$	$O(n \log n)$	Cheng (1988) [15], Dickman i in. (1991)[31], Weng i Ventura (1994) [100], Wilamowsky i in. (1996)[101]
8.	$1 \mid d_j = d \mid \sum \left(E_j^2 + T_j^2\right)$	NP-trudny	Kubiak (1993) [76]; *alg. prog. dyn.*: Kahlbacher (1989) [64], De i in. (1992,1993) [22][23]; *alg. podz. i ogr.*: Bagchi i in. (1987)[10]; *heuryst.*: Gupta i in. (1993)[35], Jurisch i in. (1997)[63]

Tabela 4.1: (Kontynuacja)

Lp.	**Problem**	**Złożoność obliczeniowa**	**Dorobek badawczy**
9.	$1 \mid d_j = d \mid$ $\sum w_j U_j + \gamma \max\{0, d - d_0\}$	$O(n)$	De i in. (1991)[21]
10.	$1 \mid d_j = d \mid \sum w_j U_j + \gamma d$	$O(n)$	Kahlbacher i Cheng (1993) [66]
11.	$1 \mid d_j = d \mid \sum (U_j + g(E_j)) + h(d)$	$O(n \log n)$	Kahlbacher i Cheng (1993) [66]
12.	$1 \mid d_j = d \mid \sum (w_j U_j + \alpha E_j) + \gamma d$	$O(n^4)$	Kahlbacher i Cheng (1993) [66]
13.	$1 \mid d_j = d \mid \sum w_j U_j + h(d)$	NP-trudny	Kahlbacher i Cheng (1993)[66]
14.	$1 \mid d_j = d \mid \sum (w_j U_j + g(E_j))$	NP-trudny	Kahlbacher i Cheng (1993)[66]
15.	$1 \mid d_j = d \mid \max\{E_{\max}, T_{\max}\}$	$O(n^2)$	Cheng (1987) [14]
16.	$1 \mid d_j = d \mid \max\{w_j(E_j + T_j)\}$	NP-trudny	Li i Cheng (1994)[81]
17.	$Q \mid d_j = d \mid \sum (E_j + T_j)$	$O(n \log n)$	Emmons (1987)[27]
18.	$Q \mid d_j = d \mid \sum (\alpha E_j + \beta T_j)$	$O(n \log n)$	Emmons (1987)[27]
19.	$R \mid d_j = d \mid \sum (E_j + T_j)$	$O(n^3)$	Alidaee i Panwalker (1993)[4], Kubiak i in. (1990)[77]
20.	$P \mid d_j = d, p_j = p \mid$ $\sum (\alpha E_j + \beta T_j + \gamma d)$	$O(1)$	Cheng i Chen (1994)[18]
21.	$P2 \mid d_j = d \mid$ $\sum (\alpha E_j + \beta T_j + \gamma d)$	NP-trudny	Cheng i Chen (1994)[18]; *alg. prog. dyn.*: De i in. (1994)[24]
22.	$Pm \mid d_j = d \mid$ $\sum (\alpha E_j + \beta T_j + \gamma d)$	NP-trudny	*alg. prog. dyn.*: De i in. (1994)[24]
23.	$P \mid d_j = d \mid \sum (\alpha E_j + \beta T_j + \gamma d)$	s. NP-trudny	De i in. (1994)[24], *heuryst.*: Cheng (1989)[16]
24.	$Q \mid d_j = d \mid \sum (\alpha E_j + \beta T_j + \gamma d)$	s. NP-trudny	*heuryst.*: Adamopoulos i Papis (1998)[2]
25.	$P \mid d_j = d \mid \sum (U_j + \alpha E_j)$	$O(n \log n)$	Kahlbacher i Cheng (1993)[66]
26.	$P \mid d_j = d \mid \sum (w_j U_j + \alpha E_j)$	$O(n^4)$	Kahlbacher i Cheng (1993)[66]
27.	$P \mid d_j = d \mid \sum U_j + \gamma d$	NP-trudny	Kahlbacher i Cheng (1993)[66]
28.	$P \mid d_j = d \mid \max\{E_{\max}, T_{\max}\}$	NP-trudny	Cheng i Chen (1994)[18]
29.	$P \mid d_j = d \mid \max\{w_j(E_j + T_j)\}$	s. NP-trudny	Li i Cheng (1994)[81]

Tabela 4.2: Najistotniejsze rezultaty osiągnięte dla problemów szeregowania, w których występuje model SLK

Lp.	**Problem**	**Złożoność obliczeniowa**	**Dorobek badawczy**
1.	$1\|d_j = d + p_j\|T_{\max} + \gamma d$	$O(n \log n)$	Cheng (1989)[16], Alidaee (1991)[3]
2.	$1\|pmtn, prec, r_j, d_j = d + r_j + p_j\|T_{\max} + \gamma d$	$O(n \log n)$	Gordon (1993)[33]
3.	$1\|pmtn, tree, r_j, d_j = d + r_j + p_j\|T_{\max} + \gamma d$	$O(n \log n)$	Gordon (1993)[33]
4.	$1\|d_j = d + p_j\| \sum (E_j + T_j)$	$O(n \log n)$	Gupta i in. (1990)[36], Karacapilidis i Pappis (1993,1995)[68],[69]
5.	$1 \mid d_j = d + p_j \mid \sum (\alpha E_j + \beta T_j + \gamma d)$	$O(n \log n)$	Adamopoulos i Pappis (1996)[1]
6.	$1 \mid d_j = d + p_j \mid \sum \alpha_j (E_j + T_j)$	NP-trudny	*heuryst.*: Gupta i in. (1990)[36]
7.	$1 \mid d_j = d + p_j \mid \sum \left(E_j^2 + T_j^2\right)$	NP-trudny	Kahlbacher (1993)[65]
8.	$1 \mid d_j = d + p_j, C_j \leqslant d_j \mid \sum g(E_j)$	$O(n \log n)$	Qi i Tu (1998)[90]
9.	$1 \mid d_j = d + p_j, C_j \leqslant d_j \mid \sum \alpha_j E_j$	$O(n \log n)$	Gordon i Strusevich (1999)[32]
10.	$1 \mid d_j = d + p_j, C_j \leqslant d_j \mid \sum \gamma_j e^{\alpha E_j}$	$O(n \log n)$	Gordon i Strusevich (1999)[32]
11.	$1 \mid SP, d_j = d + p_j, C_j \leqslant d_j \mid \sum \alpha_j E_j$	$O(n^2 \log n)$	Gordon i Strusevich (1999)[32]
12.	$1 \mid SP, d_j = d + p_j, C_j \leqslant d_j \mid \sum \gamma_j e^{\alpha E_j}$	$O(n^2 \log n)$	Gordon i Strusevich (1999)[32]
13.	$1 \mid prec, d_j = d + p_j, C_j \leqslant d_j \mid \sum E_j$	s. NP-trudny	Gordon i Strusevich (1999)[32]

W Tabeli 4.3 przedstawione zostały najistotniejsze rezultaty uzyskane w literaturze dla problemów szeregowania, w których mamy do czynienia z ustalonym wspólnym dla wszystkich zadań pożądanym terminem zakończenia ich wykonywania. Oznaczenie $d_j = \hat{d}$, które będzie pojawiało się w drugim polu notacji trójpolowej, będzie oznaczało, że wszystkie zadania mają jeden wspólny i z góry zadany pożądany termin zakończenia ich wykonywania.

Tabela 4.3: Najistotniejsze rezultaty osiągnięte dla problemów szeregowania z zadanym wspólnym terminem zakończenia wykonywania zadań

Lp.	Problem	Złożoność obliczeniowa	Dorobek badawczy
1.	$1\|d_j = \hat{d}\|\sum(E_j + T_j)$	NP-trudny	Hoogeveen i van de Velde (1991) [42]; *alg. progr. dyn.*: Hall i in. (1991) [37], Ventura i Weng (1995) [98], De i in. (1993) [23]; *alg. podz. i ogr.* Bagchi i in. (1986) [8], Szwarc (1989) [97] *heurist.*: Sundararaghavan i Ahmed (1984) [96], Hoogeveen i in. (1994) [41]
2.	$1\|d_j = \hat{d}\|\sum(\alpha E_j + \beta T_j)$	NP-trudny	*alg. progr. dyn.*: De i in. (1993) [23]; *alg. podz. i ogr.*: Bagchi i in. (1987) [10]
3.	$1\|d_j = \hat{d}\|\sum \alpha_j(E_j + T_j)$	NP-trudny	*alg. progr. dyn.*: Hoogeveen i van de Velde (1991) [42]
4.	$1\|d_j = \hat{d}\|\sum(\alpha_j E_j + \beta_j T_j)$	NP-trudny	*heuryst.*: James (1997) [44]
5.	$1\|d_j = \hat{d}\|\sum(\alpha E_j^2 + \beta T_j^2)$	NP-trudny	*alg. progr. dyn.*: Kahlbacher (1989, 1993) [64, 65], De i in. (1993) [23] *alg. podz. i ogr.*: Bagchi i in. (1987) [9, 10], De i in. (1989, 1990a) [19, 20]
6.	$1\|d_j = \hat{d}\|\sum \alpha_j U_j + \gamma \max\{d - d_0, 0\}$	NP-trudny	*alg. podz. i ogr., heuryst.*: De i in. (1991) [21]

4.3 Problemy z pożądanymi przedziałami zakończenia wykonywania zadań

W niniejszym podrozdziale przedstawiony zostanie przegląd literatury poświęconej problemom szeregowania z pożądanymi przedziałami zakończenia wykonywania zadań.

W przypadku problemów z pożądanymi momentami zakończenia wykonywania zadań, zadanie nie ponosi kary jeżeli zakończy wykonać się dokładnie w pożądanym momencie zakończenia wykonania. Natomiast w przypadku problemów z pożądanymi przedziałami zakończenia wykonywania zadań, zadanie nie ponosi kary jeżeli zakończy się wykonywać wewnątrz pewnego przedziału czasowego. Wynika z tego, że koncepcja pożądanego przedziału zakończenia wykonywania zadań jest pewnym uogólnieniem koncepcji pożądanego momentu zakończenia wykonywania zadań.

W przedstawionym poniżej przeglądzie literatury zastosowane będą następujące ozna-

czenia odnoszące się do różnych *modeli pożądanych przedziałów zakończenia wykonywania zadań*:

- $\langle \hat{e}_j, \hat{d}_j \rangle$ - każde zadanie ma indywidualny pożądany przedział zakończenia jego wykonywania, który jest z góry zadany;
- $\langle \hat{e}, \hat{d} \rangle$ - wszystkie zadania mają wspólny pożądany przedział zakończenia wykonywania zadań, który jest z góry zadany;
- $\langle e, d \rangle$ - wszystkie zadania mają wspólny pożądany przedział zakończenia wykonywania zadań; zarówno początek jak i koniec tego przedziału są zmiennymi decyzyjnymi;
- $\langle e, e + D \rangle$ - wszystkie zadania mają wspólny pożądany przedział zakończenia wykonywania zadań; początek tego przedziału jest zmienną decyzyjną, natomiast jego szerokość jest z góry zadana.
- $\langle e_j = e + p_j, d_j = d + p_j \rangle$ - szerokość pożądanego przedziału zakończenia wykonywania zadań jest wspólna dla wszystkich zadań; początek tego przedziału jest równy sumie pewnej zmiennej decyzyjnej, która jest wspólna dla wszystkich zadań, oraz czasu wykonywania tego zadania.
- $\langle e_j = e, d_j = d + p_j \rangle$ - początek pożądanego przedziału zakończenia wykonywania zadań jest wspólny dla wszystkich zadań oraz jest zmienną decyzyjną; koniec tego przedziału jest równy sumie pewnej zmiennej decyzyjnej, która jest wspólna dla wszystkich zadań, oraz czasu wykonywania tego zadania.
- $\langle e_j = e + p_j, d_j = d \rangle$ - początek pożądanego przedziału zakończenia wykonywania zadań jest równy sumie pewnej zmiennej decyzyjnej, która jest wspólna dla wszystkich zadań, oraz czasu wykonywania tego zadania; koniec tego przedziału jest wspólny dla wszystkich zadań oraz jest zmienną decyzyjną.

Powyższe oznaczenia pojawiać się będą w drugim polu notacji trójpolowej.

Należy w tym miejscu podkreślić, że w niniejszej rozprawie badane będą problemy szeregowania z doborem wspólnego pożądanego przedziału zakończenia wykonywania zadań, czyli z następującymi modelami: $\langle e, d \rangle$ i $\langle e, e + D \rangle$.

Tabela 4.4: Problemy szeregowania z modelami $\langle \hat{e}_j, \hat{d}_j \rangle$ pożądanych przedziałów zakończenia wykonywania zadań

Lp.	**Problem**	**Złożoność obliczeniowa**	**Dorobek badawczy**
1.	$1\|\langle \hat{e}_j, \hat{d}_j \rangle; d_j - e_j > p_j\|$ $\sum(E_j + T_j)$	NP-trudny	*heuryst.*: Koulamas (1996)[71]
2.	$1\|\langle \hat{e}_j, \hat{d}_j \rangle\| \sum(\alpha_j E_j + \beta_j T_j)$	s. NP-trudny	*heuryst.*: Wan i in. (2002)[99]
3.	$1\|\langle \hat{e}_j, \hat{d}_j \rangle; d_j - e_j > p_j\|$ $\sum(w_j V_j + w_j U_j)$	s. NP-trudny	*heuryst.*: Koulamas (1997)[72]
4.	$1\|\langle \hat{e}_j, \hat{d}_j \rangle\|$ $\max\{\max g(E_j), \max h(T_j)\}$	$O(n \log n)$	Sidney (1997)[94]
5.	$1\|\langle \hat{e}_j, \hat{d}_j \rangle\| \sum(V_j + U_j)$	NP-trudny	*heuryst.*: Yoo i Martin-Vega (2001)[106]
6.	$1\|\langle \hat{e}_j, \hat{d}_j \rangle; r_j\| \sum(V_j + U_j)$	NP-trudny	*heuryst.*: Yoo i Martin-Vega (2001)[106]
7.	$1 \left\|\langle \hat{e}_j, \hat{d}_j \rangle\right\| \max\{g_j(E_j), h_j(T_j)\}$	NP-zupełny	*alg. aproks.*: Smutnicki (1997)[95]

Tabela 4.5: Problemy szeregowania zadań z modelami $\langle \hat{e}, \hat{d} \rangle$ pożądanych przedziałów zakończenia wykonywania zadań

Lp.	**Problem**	**Złożoność obliczeniowa**	**Dorobek badawczy**
1.	$1\|\langle \hat{e}, \hat{d} \rangle\| \sum(\alpha E_j + \beta T_j + \alpha' V_j + \beta' U_j + \delta C_j)$	NP-trudny	*alg. progr. dyn.*: Yeung i in. (2001)[103]
2.	$1\|\langle \hat{e}, \hat{d} \rangle\| \sum(\alpha_j E_j + \beta_j U_j)$	NP-trudny	*alg. progr. dyn.*: Liman i Ramaswamy (1994)[85]
3.	$1\|\langle \hat{e}, \hat{d} \rangle\| \sum(V_j + U_j)$	$O(n \log n)$	Yoo i Martin-Vega (2001)[106]
4.	$1\|\langle \hat{e}, \hat{d} \rangle; r_j\| \sum(V_j + U_j)$	$O(n \log n)$	Yoo i Martin-Vega (2001)[106]
5.	$P\|\langle \hat{e}, \hat{d} \rangle\| \sum(\alpha_j E_j + \beta_j T_j)$	s. NP-trudny	*alg. podz. i ogr.*: Chen i Lee (2002)[13]

Tabela 4.6: Problemy szeregowania zadań z modelami $\langle e; e+D\rangle$ pożądanych przedziałów zakończenia wykonywania zadań

Lp.	**Problem**	**Złożoność obliczeniowa**	**Dorobek badawczy**
1.	$1\|\langle e; e+D\rangle\|\sum(\alpha E_j+\beta T_j)+\delta e$	$O(n\log n)$	Liman i in. (1996)[82]
2.	$1\|\langle e; e+D\rangle\|$ $\sum(\alpha E_j+\beta T_j+\alpha' V_j+\beta' U_j)+\delta e$	$O(n\log n)$	Yeung i in. (2001)[104]
3.	$1\|\langle e; e+D\rangle; D \leqslant \min p_j\|$ $\sum(\alpha E_j+\beta T_j)+\gamma d+\theta C_j$	$O(n\log n)$	Wu i Wang (1999)[102]
4.	$1\|\langle e, e+D\rangle\|\sum(\alpha_j E_j+\beta_j U_j)$	NP-trudny	*alg. progr. dyn.*: Liman i Ramaswamy (1994)[85]
5.	$P2\|\langle e; e+D\rangle\|\sum(\alpha E_j+\beta T_j)$	NP-trudny	*alg. progr. dyn.*: Kramer i Lee (1994)[75]
6.	$P\|\langle e; e+D\rangle\|\alpha\sum E_j+\beta\sum T_j$	s. NP-trudny	*heuryst.* Kramer i Lee (1994)[75]
7.	$F2\|\langle e; e+D\rangle\|\sum(E_j+T_j)$	s. NP-trudny	*heuryst.*, *alg. podz. i ogr.* Yeung i in. (2004)[105]
8.	$1\|\langle e; e+D\rangle\|\sum(\alpha_j V_j+\beta_j U_j)+$ γd	NP-trudny	*alg. prog. dyn.*: Yeung i in. (2001) [104]
9.	$1\|\langle e; e+D\rangle\|\sum(\alpha_j E_j+\beta_j T_j+$ $\alpha'_j V_j+\beta'_j U_j)+\gamma\max\{d-d_0,0\}$	NP-trudny	*alg. prog. dyn.*: Yeung i in. (2001) [104]
10.	$1\|\langle e; e+D\rangle\|\sum(\alpha_j V_j+\beta_j U_j+$ $\theta_j C_j)+\gamma\max\{d-d_0,0\}$	NP-trudny	*alg. prog. dyn.*: Yeung i in. (2001)[104]
11.	$1\|\langle e, e+D\rangle\|\sum(\alpha_j E_j+\beta_j T_j)$	NP-trudny	*alg. podz. i ogr.*: Azizoglu i Webster (1997)[5]
12.	$P\,\|\langle e; e+D\rangle\|\sum(\alpha_j E_j+\beta_j T_j)$	s. NP-trudny	*alg. podz. i ogr.*: Chen i Lee (2002)[13]

Tabela 4.7: Problemy szeregowania zadań z modelami $\langle e; d\rangle$ pożądanych przedziałów zakończenia wykonywania zadań

Lp.	**Problem**	**Złożoność obliczeniowa**	**Dorobek badawczy**
1.	$1\|\langle e; d\rangle\|\sum(\alpha E_j+\beta T_j)+\delta e+$ $\gamma(d-e)$	$O(n\log n)$	Liman i in. (1998)[84]
2.	$1\|\langle e; d\rangle; p_j=p'_j-x_j\|\sum(\alpha E_j+$ $\beta T_j+G_j x_j)+\delta e+\gamma(d-e)$	$O(n^3)$	Liman i in. (1997)[83]

Tabela 4.7: (Kontynuacja)

Lp.	**Problem**	**Złożoność obliczeniowa**	**Dorobek badawczy**
3.	$P2\|\langle e;d\rangle\|\sum(E_j+T_j)+(d-e)$	NP-trudny	*sch. aproks.*, *alg. podz. i ogr.*: Janiak i Marek (2002,2004) [51, 54]
4.	$P2\|\langle e;d\rangle\|\max\{\alpha\max E_j,$ $\beta\max T_j,\gamma(d-e)\}$	NP-trudny	*sch. aproks.*, *alg. podz. i ogr.*: Janiak i Marek (2003,2004) [52, 54]
5.	$P\|\langle e;d\rangle\|\max\{\alpha\max E_j,$ $\beta\max T_j,\gamma(d-e)\}$	s. NP-trudny	*heuryst.*: Janiak i Marek (2004)[54]
6.	$P\|\langle e;d\rangle\|\max\{\alpha\max E_j,$ $\beta\max T_j,\gamma(d-e),\theta e\}$	s NP-trudny	*heuryst.*: Mosheiov (2001)[87]

Tabela 4.8: Problemy szeregowania zadań z innymi modelami pożądanych przedziałów zakończenia wykonywania zadań

Lp.	**Problem**	**Złożoność obliczeniowa**	**Dorobek badawczy**
1.	$1\|\langle e_j=e+p_j;d_j=d+p_j\rangle\|$ $\max\{\alpha\max E_j,\beta\max T_j,$ $\gamma(d-e)\}$	$O(n)$	Janiak i Marek (2004)[55]
2.	$1\|\langle e_j=e;d_j=d+p_j\rangle\|$ $\max\{\alpha\max E_j,\beta\max T_j,$ $\gamma(d-e)\}$	$O(n)$	Janiak i Marek (2004)[86]
3.	$1\|\langle e_j=e+p_j;d_j=d\rangle\|$ $\max\{\alpha\max E_j,\beta\max T_j,$ $\gamma(d-e)\}$	$O(n)$	Janiak i Marek (2004)[86]
4.	$1\|\langle e_j=e+p_j;d_j=d+p_j\rangle\|$ $\sum(\alpha E_j+\beta T_j)+\gamma(d-e)$	$O(n\log n)$	Janiak i Marek (2004)[55]
5.	$P2\|\langle e_j=e+p_j;d_j=d+p_j\rangle\|$ $\sum(E_j+T_j)+(d-e)$	NP-trudny	*sch. aproks.*, *alg. podz. i ogr.*: Janiak i Marek (2004)[54]
6.	$P2\|\langle e_j=e+p_j;d_j=d+p_j\rangle\|$ $\max\{\alpha\max E_j,\beta\max T_j,$ $\gamma(d-e)\}$	NP-trudny	*sch. aproks.*, *alg. podz. i ogr.*: Janiak i Marek (2004)[54]

4.4 Podsumowanie

Jak wynika z przedstawionego powyżej przeglądu literatury, problemy z doborem wspólnego pożądanego przedziału zakończenia wykonywania zadań są stosunkowo od niedawna przedmiotem zainteresowania naukowców i jak dotąd rozpatrywane były w bardzo wąskim zakresie. Ponadto, wiele typów kryteriów optymalizacji nie było w ogóle rozpatrywanych. Przykładowo, należy zauważyć, że spośród problemów, w których zarówno początek, jak i koniec pożądanego przedziału zakończenia wykonywania zadań są zmiennymi decyzyjnymi, nie rozpatrywano wcześniej problemów z różnymi dla poszczególnych zadań funkcjami kar za nieterminowe wykonanie, mimo że modelują one wiele procesów rzeczywistych.

W związku z tym, w niniejszej pracy podjęto badania nad wybranymi problemami szeregowania z doborem wspólnego pożądanego przedziału zakończenia wykonywania zadań. Wyniki tych badań przedstawiono w kolejnych rozdziałach.

Rozdział 5

Problem z liniowymi identycznymi dla wszystkich zadań funkcjami kar za nieterminowość wykonania

5.1 Wstęp

W niniejszym rozdziale przedstawione zostaną nowe wyniki, uzyskane przez autora niniejszej pracy, dotyczące jednoprocesorowego problemu szeregowania z optymalnym doborem pożądanego przedziału zakończenia wykonywania zadań, przy zadanym dolnym i górnym ograniczeniu na szerokość tego przedziału. Minimalizacji podlega ważona suma przyspieszeń i opóźnień wykonania zadań oraz nieliniowa kara związana z szerokością tego przedziału. Problem ten jest uogólnieniem dwóch problemów rozważanych w pracach [82] i [84].

Część przedstawionych w tym rozdziale rezultatów zostało wcześniej opublikowanych w pracy [56].

Niniejszy rozdział zawiera precyzyjne sformułowanie badanego problemu (Podrozdział 5.2). Następnie wprowadzone zostają niezbędne oznaczenia oraz wykazane własności rozwiązania optymalnego (Podrozdział 5.3). W Podrozdziale 5.4 opisany jest algorytm, który rozwiązuje optymalnie analizowany problem w czasie wielomianowym. Niniejszy rozdział zamyka krótkie podsumowanie uzyskanych wyników.

5.2 Sformułowanie problemu

Poniżej zostanie przedstawiona definicja badanego problemu.

Zadany jest zbiór n niezależnych i niepodzielnych zadań $\mathbf{J} = \{1, \ldots, n\}$ do wykonania na pojedynczym procesorze. Procesor ten może wykonywać jednocześnie tylko jedno zadanie. Każde zadanie $j \in \mathbf{J}$, o czasie wykonywania p_j, jest dostępne w momencie 0.

Wszystkie zadania mają wspólny pożądany przedział zakończenia wykonywania. Początek i koniec tego przedziału oznaczmy przez e i d.

Rozwiązanie (harmonogram) określa momenty zakończenia wykonywania zadań oraz początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

Przypomnijmy następujące oznaczenia:

- S_j - moment rozpoczęcia wykonywania zadania j;
- C_j - moment zakończenia wykonywania zadania j;
- $E_j \triangleq \{e - C_j, 0\}$ - czas przedwczesnego wykonania zadania j (przyspieszenie wykonania zadania j);
- $T_j \triangleq \{C_j - d, 0\}$ - czas zbyt późnego wykonania zadania j (opóźnienie wykonania zadania j).

W celu podkreślenia, że konkretne oznaczenie, np. S_j, odnosi się do pewnego, określonego harmonogramu σ, stosowany będzie następujący zapis: $S_j(\sigma)$. Ta sama zasada będzie obowiązywać również dla oznaczeń wprowadzonych w dalszej części niniejszej pracy.

Niech D_{min} ($D_{min} \geqslant 0$) oraz D_{max} ($D_{max} \leqslant \sum p_j$) oznaczają odpowiednio dolne i górne ograniczenie na szerokość pożądanego przedziału zakończenia wykonywania zadań, tzn. każde rozwiązanie dopuszczalne musi spełniać następujące nierówności $D_{min} \leqslant d - e \leqslant D_{max}$.

Celem jest znalezienie takiego harmonogramu σ (spełniającego ograniczenie $D_{min} \leqslant d - e \leqslant D_{max}$), który minimalizuje wartość następującego kryterium:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha E_j + \beta T_j) + \theta e + f_W(d - e),$$

gdzie $\alpha \geqslant 0$, $\beta \geqslant 0$ oraz $\theta \geqslant 0$ są zadanymi wagami, które przyjmują tylko wartości całkowite, natomiast f_W jest dowolną wypukłą i niemalejącą funkcją określoną na przedziale $[D_{min}, D_{max}]$, taką że $f_W(D_{min}) = 0$.

Zakładamy również, że czasy wykonywania zadań oraz początek i koniec przedziału $[e, d]$ przyjmują tylko wartości całkowite.

W trójpolowej notacji [34], problem może być przedstawiony następująco:

$$1|\langle e, d\rangle; D_{min} \leqslant d - e \leqslant D_{max}| \sum (\alpha E_j + \beta T_j) + \theta e + f_W(d - e).$$

W celu skrócenia notacji, badany problem będziemy oznaczać również jako **P1**.

5.3 Własności rozwiązania optymalnego

W niniejszym podrozdziale przedstawione zostaną własności rozwiązania optymalnego problemu **P1**.

Własność 5.1. *Istnieje rozwiązanie optymalne problemu* **P1***, w którym nie ma okresów bezczynności procesora pomiędzy wykonywaniem poszczególnych zadań.*

Dowód. Załóżmy, że znane jest rozwiązanie optymalne σ. Niech π oznacza kolejność wykonywania zadań (permutację). Przyjmijmy, że rozwiązanie σ nie spełnia dowodzonej własności, tzn. istnieją przynajmniej dwa takie zadania $\pi(j)$ oraz $\pi(j+1)$, że $S_{\pi(j+1)} \geqslant C_{\pi(j)}$. W celu uproszczenia notacji wprowadźmy następujące oznaczenia: $t_1 = C_{\pi(j)}$ oraz $t_2 = S_{\pi(j+1)}$.

Pokażemy, że przedział bezczynności procesora $[t_1, t_2]$ można zlikwidować nie zwiększając wartości funkcji celu. W tym celu należy rozpatrzeć trzy możliwe przypadki:

Przypadek 1. Załóżmy, że $t_2 \leqslant d$.

Utwórzmy harmonogram σ' z harmonogramu σ taki, że momenty rozpoczęcia wykonywania zadań $\pi(k)$, gdzie $k = 1, \ldots, j$, są zwiększone o wartość $t_2 - t_1$, tzn. $S_{\pi(k)}(\sigma') = S_{\pi(k)}(\sigma) + (t_2 - t_1)$. Oczywiste jest, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się. Łatwo również zauważyć, że $E_{\pi(k)}(\sigma') \leqslant E_{\pi(k)}(\sigma)$ dla $k = 1, \ldots, j$. Z powyższego wynika, że $Z(\sigma') \leqslant Z(\sigma)$.

Przypadek 2. Załóżmy, że $t_1 \geqslant d$.

Utwórzmy harmonogram σ'' z harmonogramu σ taki, że momenty rozpoczęcia wykonywania zadań $\pi(k)$, gdzie $k = j+1, \ldots, n$, są zmniejszone o wartość $t_2 - t_1$, tzn. $S_{\pi(k)}(\sigma'') = S_{\pi(k)}(\sigma) - (t_2 - t_1)$. Oczywiste jest, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się. Łatwo również zauważyć, że $T_{\pi(k)}(\sigma'') \leqslant T_{\pi(k)}(\sigma)$ dla $k = j+1, \ldots, n$. Z powyższego wynika, że $Z(\sigma'') \leqslant Z(\sigma)$.

Przypadek 3. Załóżmy, że $t_1 < d < t_2$.

Wystarczy zauważyć, że przypadek ten możemy rozbić na dwa przypadki rozpatrywane powyżej dzieląc przedział bezczynności $[t_1, t_2]$ na dwa przedziały $[t_1, d]$ (Przypadek 1) oraz $[d, t_2]$ (Przypadek 2).

Ostatecznie możemy stwierdzić, że wyeliminowanie okresu bezczynności $[C_{\pi(j)}, S_{\pi(j+1)}]$ nie zwiększy wartości funkcji celu. Możemy zatem, powtarzając powyższe postępowanie dla każdego okresu bezczynności, uzyskać harmonogram spełniający dowodzoną własność bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P1**, które spełniają Własność 5.1.

Własność 5.2. *Istnieje rozwiązanie optymalne problemu* **P1**, *w którym pierwsze zadanie rozpoczyna wykonywać się w momencie* 0.

Dowód. Załóżmy, że znane jest rozwiązanie optymalne σ, które nie spełnia dowodzonej własności. Oznaczmy moment rozpoczęcia pierwszego zadania w tym harmonogramie przez λ. Wartość kryterium dla harmonogramu σ wynosi:

$$Z(\sigma) = \sum_{j\in\mathbf{J}}(\alpha E_j(\sigma) + \beta T_j(\sigma)) + f_W(d-e) + \theta e = Z_0 + \theta e,$$

gdzie $Z_0 = \sum_{j\in\mathbf{J}}(\alpha E_j(\sigma) + \beta T_j(\sigma)) + f_W(d-e)$.

Utwórzmy harmonogram σ' z harmonogramu σ poprzez przesunięcie całego harmonogramu o wartość λ w lewo na osi czasu, tzn. $S_j(\sigma') = S_j(\sigma) - \lambda$ dla $j \in \mathbf{J}$ oraz $e' = e - \lambda$ i $d' = d - \lambda$, gdzie e' i d' oznaczają odpowiednio początek i koniec pożądanego przedziału zakończenia wykonywania w harmonogramie σ'. Wartość kryterium dla harmonogramu σ' wynosi:

$$Z(\sigma') = \sum_{j\in\mathbf{J}}(\alpha E_j(\sigma') + \beta T_j(\sigma')) + f_W(d'-e') + \theta e' =$$
$$\sum_{j\in\mathbf{J}}(\alpha \max\{e' - C_j(\sigma'), 0\} + \beta \max\{C_j(\sigma') - d', 0\}) + f_W(d'-e') + \theta e' =$$
$$\sum_{j\in\mathbf{J}}(\alpha \max\{e - \lambda - (C_j(\sigma) - \lambda), 0\} + \beta \max\{C_j(\sigma) - \lambda - (d-\lambda), 0\}) +$$
$$f_W(d - \lambda - (e-\lambda)) + \theta(e-\lambda) =$$
$$\sum_{j\in\mathbf{J}}(\alpha \max\{e - C_j(\sigma), 0\} + \beta \max\{C_j(\sigma) - d, 0\}) + f_W(d-e) + \theta(e-\lambda) =$$
$$\sum_{j\in\mathbf{J}}(\alpha E_j(\sigma) + \beta T_j(\sigma)) + f_W(d-e) + \theta(e-\lambda) = Z_0 + \theta(e-\lambda).$$

Skoro $\theta \geqslant 0$, to mamy:

$$Z(\sigma) - Z(\sigma') = \theta e - \theta(e-\lambda) = \theta\lambda \geqslant 0,$$

a stąd:

$$Z(\sigma) \geqslant Z(\sigma').$$

Z powyższego wynika, że przesunięcie całego harmonogramu o wartość λ w lewo na osi czasu nie zwiększy wartości funkcji celu. Możemy zatem uzyskać harmonogram spełniający dowodzoną własność bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P2.1**, które spełniają Własność 5.2.

Niech π oznacza kolejność wykonywania zadań. Z Własności 5.1 i 5.2 wynika, że mo-

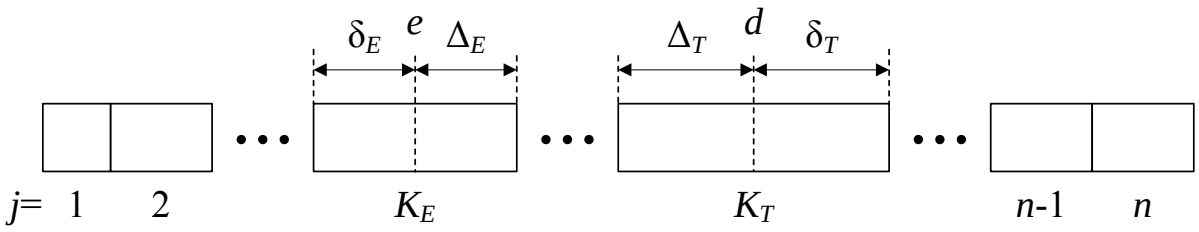
ment zakończenia wykonywania zadania $\pi(j)$ możemy wyrazić w sposób następujący:

$$C_{\pi(j)} = \sum_{l=1}^{j} p_{\pi(l)}.$$

Zapis $\sigma = (\pi, e, d)$ będzie oznaczał harmonogram σ, w którym π określa kolejność wykonywania zadań na procesorze, natomiast e i d to odpowiednio początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

Dla zadanego harmonogramu zdefiniujmy:

- $K_E \triangleq \min\{j : C_{\pi(j)} \geqslant e\}$ - pozycja pierwszego zadania, które kończy wykonywać się w lub po momencie e;
- $K_T \triangleq \max\{j : S_{\pi(j)} \leqslant d\}$ - pozycja ostatniego zadania, które rozpoczyna wykonywać się w lub przed momentem d;
- $\Delta_E \triangleq C_{\pi(K_E)} - e$ - część zadania $\pi(K_E)$, która wykonuje się wewnątrz przedziału $[e, d]$;
- $\delta_E \triangleq e - S_{\pi(K_E)}$ - część zadania $\pi(K_E)$, która wykonuje się przed przedziałem $[e, d]$;
- $\Delta_T \triangleq d - S_{\pi(K_T)}$ - część zadania $\pi(K_T)$, która wykonuje się wewnątrz przedziału $[e, d]$;
- $\delta_T \triangleq C_{\pi(K_T)} - d$ - część zadania $\pi(K_T)$, która wykonuje się za przedziałem $[e, d]$.



Rysunek 5.1: Ilustracja zdefiniowanych oznaczeń.

Zdefiniowane oznaczenia zostały zilustrowane na Rysunku 5.1.

Zauważmy, że wartości parametrów e i d można wyrazić w sposób następujący:

$$e = C_{\pi(K_E)} - \Delta_E;$$

$$d = S_{\pi(K_T)} + \Delta_T = C_{\pi(K_T-1)} + \Delta_T.$$

Łatwo zauważyć, że:

- $E_{\pi(j)} > 0$ tylko i wyłącznie dla $j = 1, \ldots, K_E - 1$;
- $T_{\pi(j)} > 0$ tylko i wyłącznie dla $j = K_T, \ldots, n$.

Wartość przyspieszenia wykonania zadania $\pi(j)$ dla $j = 1, \ldots, K_E - 1$ wynosi:

$$E_{\pi(j)} = \max\{e - C_{\pi(j)}, 0\} = e - C_{\pi(j)} = C_{\pi(K_E)} - \Delta_E - C_{\pi(j)} = \sum_{l=1}^{K_E} p_{\pi(l)} - \sum_{l=1}^{j} p_{\pi(l)} - \Delta_E = \sum_{l=j+1}^{K_E} p_{\pi(l)} - \Delta_E,$$

a stąd:

$$\alpha \sum_{j=1}^{n} E_{\pi(j)} = \alpha \sum_{j=1}^{K_E-1} E_{\pi(j)} = \alpha \sum_{j=1}^{K_E-1} \left(\sum_{l=j+1}^{K_E} p_{\pi(l)} - \Delta_E \right) = \alpha \sum_{j=1}^{K_E-1} \sum_{l=j+1}^{K_E} p_{\pi(l)} - \alpha \sum_{j=1}^{K_E-1} \Delta_E =$$

$$\alpha \left(\sum_{l=2}^{K_E} p_{\pi(l)} + \sum_{l=3}^{K_E} p_{\pi(l)} + \ldots + \sum_{l=K_E}^{K_E} p_{\pi(l)} \right) - \alpha(K_E - 1)\Delta_E =$$

$$\alpha \left(p_{\pi(2)} + 2p_{\pi(3)} + \ldots + (K_E - 1)p_{\pi(K_E)} \right) - \alpha(K_E - 1)\Delta_E =$$

$$\sum_{j=1}^{K_E} \left(\alpha(j-1)p_{\pi(j)} \right) - \alpha(K_E - 1)\Delta_E.$$

Wartość opóźnienia wykonania zadania $\pi(j)$ dla $j = K_T, \ldots, n$ wynosi:

$$T_{\pi(j)} = \max\{C_{\pi(j)} - d, 0\} = C_{\pi(j)} - d = C_{\pi(j)} - C_{\pi(K_T-1)} - \Delta_T = \sum_{l=1}^{j} p_{\pi(l)} - \sum_{l=1}^{K_T-1} p_{\pi(l)} - \Delta_T = \sum_{l=K_T}^{j} p_{\pi(l)} - \Delta_T,$$

a stąd:

$$\beta\sum_{j=1}^{n}T_{\pi(j)}=\beta\sum_{j=K_T}^{n}T_{\pi(j)}=\beta\sum_{j=K_T}^{n}\left(\sum_{l=K_T}^{j}p_{\pi(l)}-\Delta_T\right)=$$
$$\beta\sum_{j=K_T}^{n}\sum_{l=K_T}^{j}p_{\pi(l)}-\beta\sum_{j=K_T}^{n}\Delta_T=$$
$$\beta\left(\sum_{l=K_T}^{K_T}p_{\pi(l)}+\ldots+\sum_{l=K_T}^{n-1}p_{\pi(l)}+\sum_{l=K_T}^{n}p_{\pi(l)}\right)-\beta\sum_{j=K_T}^{n}\Delta_T=$$
$$\beta\left((n-K_T+1)p_{\pi(K_T)}+\ldots+2p_{\pi(n-1)}+p_{\pi(n)}\right)-\beta\sum_{j=K_T}^{n}\Delta_T=$$
$$\sum_{j=K_T}^{n}\left(\beta(n-j+1)p_{\pi(j)}\right)-\beta(n-K_T+1)\Delta_T.$$

Koszt związany z początkiem przedziału $[e,d]$ możemy wyrazić w sposób następujący:

$$\theta e=\theta(C_{\pi(K_E)}-\Delta_E)=\sum_{j=1}^{K_E}\theta p_{\pi(j)}-\theta\Delta_E.$$

Z powyższych rozważań wynika, że wartość funkcji celu dla harmonogramu $\sigma=(\pi,e,d)$ możemy wyrazić w sposób następujący:

$$Z(\sigma)=\sum_{j\in\mathbf{J}}(\alpha E_j+\beta T_j)+\theta e+f_W(d-e)=$$
$$\sum_{j=1}^{n}\alpha E_{\pi(j)}+\sum_{j=1}^{n}\beta T_{\pi(j)}+\theta e+f_W(d-e)=$$
$$\sum_{j=1}^{K_E}\left(\alpha(j-1)p_{\pi(j)}\right)-\alpha(K_E-1)\Delta_E+\sum_{j=K_T}^{n}\left(\beta(n-j+1)p_{\pi(j)}\right)-\beta(n-K_T+1)\Delta_T+$$
$$\sum_{j=1}^{K_E}\theta p_{\pi(j)}-\theta\Delta_E+f_W(d-e)=$$
$$\sum_{j=1}^{K_E}\left[\left(\alpha(j-1)+\theta\right)p_{\pi(j)}\right]-\left[\alpha(K_E-1)+\theta\right]\Delta_E+$$
$$\sum_{j=K_T}^{n}\left(\beta(n-j+1)p_{\pi(j)}\right)-\beta(n-K_T+1)\Delta_T+f_W(d-e).$$

Niech:

$$w_j\triangleq\begin{cases}\alpha(j-1)+\theta, & \text{jeżeli } j\leqslant K_E;\\ \beta(n-j+1), & \text{jeżeli } j\geqslant K_T\end{cases}$$

oznacza wagę pozycji j. Wartości wag pozycji $j=K_E+1,\ldots,K_T-1$ nie są zdefiniowane, ponieważ nie będą wykorzystywane w dalszych rozważaniach.

Zauważmy, że:

- $w_j \leqslant w_{j+1}$ dla każdej pozycji $j = 1, \ldots, K_E - 1$;
- $w_j \geqslant w_{j+1}$ dla każdej pozycji $j = K_T, \ldots, n - 1$.

Możemy wykorzystać wprowadzone oznaczenie do przedstawienia wartości funkcji celu w sposób następujący:

$$Z(\sigma) = \sum_{j=1}^{K_E} w_j p_{\pi(j)} - w_{K_E}\Delta_E + \sum_{j=K_T}^{n} w_j p_{\pi(j)} - w_{K_T}\Delta_T + f_W(d-e).$$

Własność 5.3. *Istnieje rozwiązanie optymalne problemu* **P1**, *w którym:*

- *jeżeli waga pozycji K_E jest mniejsza od wagi pozycji K_T, to pewne zadanie kończy wykonywać się w momencie e;*
- *jeżeli waga pozycji K_T jest nie większa niż waga pozycji K_T, to pewne zadanie kończy wykonywać się w momencie d;*

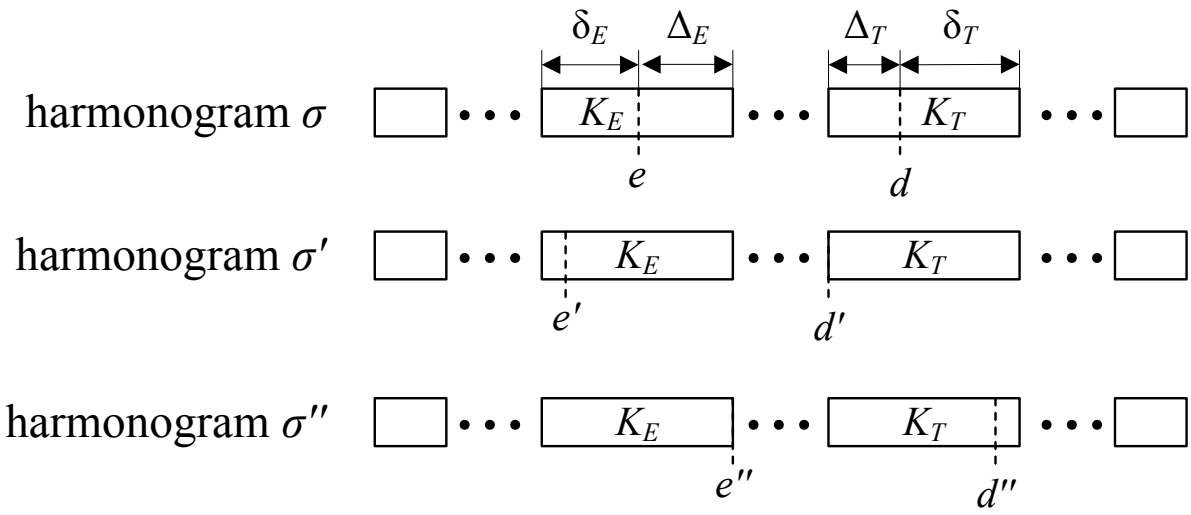
Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Przyjmijmy również, że rozwiązanie to nie spełnia dowodzonej własności. W niniejszym dowodzie zakładamy, że oznaczenia K_E i K_T będą odnosiły się do harmonogramu σ. Wartość kryterium dla rozwiązania σ wynosi:

$$Z(\sigma) = \sum_{j\in\mathbf{J}} (\alpha E_j + \beta T_j) + f_W(d-e) + \theta e =$$
$$\alpha \sum_{j=1}^{n} \max\{e - C_{\pi(j)}, 0\} + \beta \sum_{j=1}^{n} \max\{C_{\pi(j)} - d, 0\} + f_W(d-e) + \theta e =$$
$$\alpha \sum_{j=1}^{K_E-1} (e - C_{\pi(j)}) + \beta \sum_{j=K_T}^{n} (C_{\pi(j)} - d) + f_W(d-e) + \theta e.$$

W celu wykazania prawdziwości niniejszej własności musimy rozpatrzeć dwa możliwe przypadki.

Przypadek 1. Załóżmy, że waga pozycji K_E jest mniejsza od wagi pozycji K_T w harmonogramie σ, czyli $\alpha(K_E - 1) + \theta < \beta(n - K_T + 1)$.

Niech $\lambda_1 = \min\{\Delta_T(\sigma), \delta_E(\sigma)\}$. Utwórzmy harmonogram $\sigma' = (\pi, e', d')$ z harmonogramu σ taki, że $e' = e - \lambda_1$ i $d' = d - \lambda_1$. Wartość kryterium dla harmonogramu σ' wynosi:



Rysunek 5.2: Przykładowe harmonogramy σ, σ' i σ''.

$$Z(\sigma') = \sum_{j\in\mathbf{J}}(\alpha E_j + \beta T_j) + f_W(d' - e') + \theta e' =$$

$$\alpha\sum_{j=1}^{n}\max\{e' - C_{\pi(j)}, 0\} + \beta\sum_{j=1}^{n}\max\{C_{\pi(j)} - d', 0\} + f_W(d' - e') + \theta e' =$$

$$\alpha\sum_{j=1}^{K_E-1}(e' - C_{\pi(j)}) + \beta\sum_{j=K_T}^{n}(C_{\pi(j)} - d') + f_W(d' - e') + \theta e' =$$

$$\alpha\sum_{j=1}^{K_E-1}(e - \lambda_1 - C_{\pi(j)}) + \beta\sum_{j=K_T}^{n}(C_{\pi(j)} - (d - \lambda_1)) + f_W(d - e) + \theta(e - \lambda_1) =$$

$$\alpha\sum_{j=1}^{K_E-1}(e - C_{\pi(j)}) + \beta\sum_{j=K_T}^{n}(C_{\pi(j)} - d) + f_W(d - e) + \theta e +$$

$$\lambda_1(\beta(n - K_T + 1) - \alpha(K_E - 1) - \theta).$$

Uwzględniając założenie $\alpha(K_E - 1) + \theta < \beta(n - K_T + 1)$ otrzymujemy:

$$Z(\sigma) - Z(\sigma') = \lambda_1(\beta(n - K_T + 1) - \alpha(K_E - 1) - \theta) > 0.$$

Powyższa nierówność zaprzecza optymalności harmonogramu σ.

Przypadek 2. Załóżmy, że waga pozycji K_E jest nie mniejsza niż waga pozycji K_T w harmonogramie σ, czyli $\alpha(K_E - 1) + \theta \geqslant \beta(n - K_T + 1)$.

Niech $\lambda_2 = \min\{\delta_T(\sigma), \Delta_E(\sigma)\}$. Utwórzmy teraz harmonogram $\sigma' = (\pi, e'', d'')$ z harmonogramu σ taki, że $e'' = e + \lambda_2$ i $d'' = d + \lambda_2$. Wartość kryterium dla harmonogramu

σ'' wynosi:

$$Z(\sigma'') = \sum_{j\in\mathbf{J}}(\alpha E_j + \beta T_j) + f_W(d-e) + \theta e =$$
$$\alpha\sum_{j=1}^{n}\max\{e'' - C_{\pi(j)}, 0\} + \beta\sum_{j=1}^{n}\max\{C_{\pi(j)} - d'', 0\} + f_W(d''-e'') + \theta e'' =$$
$$\alpha\sum_{j=1}^{K_E-1}(e'' - C_{\pi(j)}) + \beta\sum_{j=K_T}^{n}(C_{\pi(j)} - d'') + f_W(d''-e'') + \theta e'' =$$
$$\alpha\sum_{j=1}^{K_E-1}(e + \lambda_2 - C_{\pi(j)}) + \beta\sum_{j=K_T}^{n}(C_{\pi(j)} - (d+\lambda_2)) + f_W(d-e) + \theta(e+\lambda_2) =$$
$$\alpha\sum_{j=1}^{K_E-1}(e - C_{\pi(j)}) + \beta\sum_{j=K_T}^{n}(C_{\pi(j)} - d) + f_W(d-e) + \theta e +$$
$$\lambda_2(\alpha(K_E-1) + \theta - \beta(n - K_T + 1)).$$

Uwzględniając założenie $\alpha(K_E-1) + \theta \geqslant \beta(n-K_T+1)$ otrzymujemy:

$$Z(\sigma) - Z(\sigma'') = \lambda_2(\alpha(K_E-1) + \theta - \beta(n-K_T+1)) \geqslant 0.$$

Rozważmy dwa przypadki:

- Jeżeli $\lambda_2 = \Delta_T(\sigma)$, to $\Delta_T(\sigma'') = 0$ (patrz Rysunek 5.2) oraz $K_T(\sigma'') = K_T(\sigma) + 1$ i $K_E(\sigma'') = K_E(\sigma) + 1$. Łatwo zauważyć, że wówczas harmonogram σ'' spełnia dowodzoną własność.

- Jeżeli $\lambda_2 = \Delta_E(\sigma)$, to $\Delta_E(\sigma'') = 0$ (patrz Rysunek 5.2) a $K_E(\sigma'') = K_E(\sigma) - 1$. Jeżeli $\alpha(K_E(\sigma'') - 1) + \theta \geqslant \beta(n - K_T(\sigma''))$, to harmonogram σ'' spełnia dowodzoną własność. W przeciwnym przypadku powtarzamy przedstawione powyżej postępowanie w celu skonstruowania harmonogramu spełniającego dowodzoną własność.

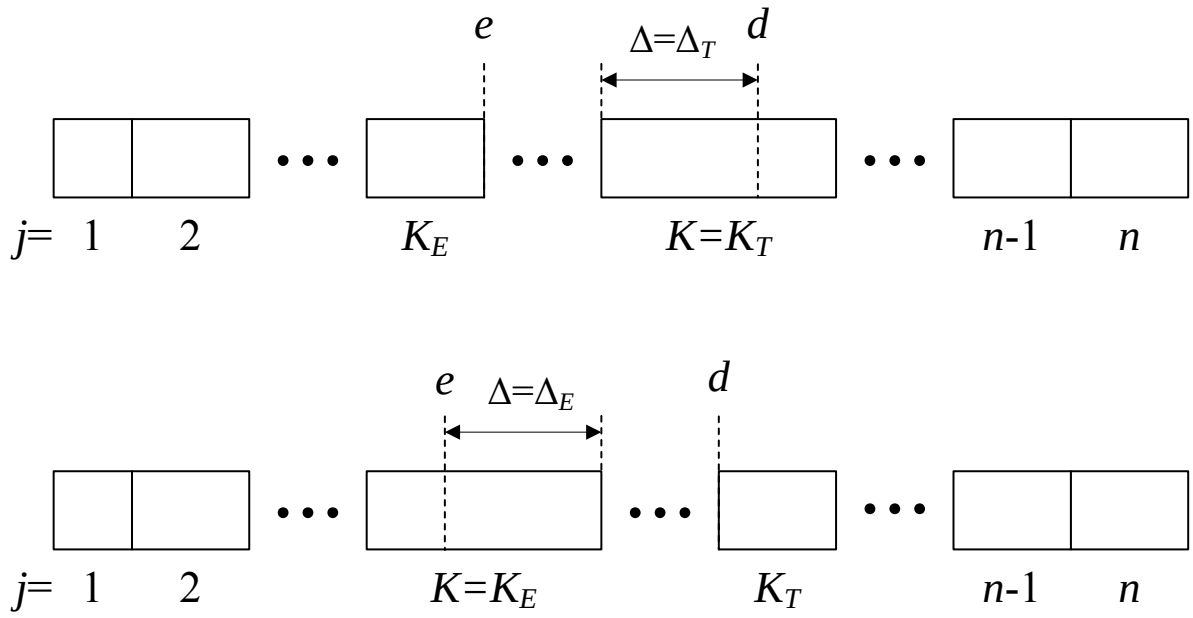
Z powyższego wynika, że możemy zawsze uzyskać harmonogram, dla którego rozważana własność jest prawdziwa, bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P1**, które spełniają Własność 5.3.

Z Własności 5.3 wynika, że jeżeli $w_{K_E} < w_{K_T}$, to $\Delta_E = 0$, a jeżeli $w_{K_E} \geqslant w_{K_T}$, to $\Delta_T = 0$. Tak więc w celu uproszczenia dalszych rozważań zdefiniujmy:

- $\Delta \triangleq \max\{\Delta_E, \Delta_T\}$;

- $K \triangleq \begin{cases} K_E, & \text{jeżeli } \Delta_E > 0; \\ K_T, & \text{w przeciwnym przypadku.} \end{cases}$



Rysunek 5.3: Dwa przykładowe harmonogramy z zaznaczonymi oznaczeniami Δ i K.

Zdefiniowane oznaczenia zostały zilustrowane na Rysunku 5.3.

Uwzględniając wprowadzone wcześniej oznaczenia, wartość funkcji celu dla harmonogramu $\sigma = (\pi, e, d)$ może być wyrażona w sposób następujący:

$$Z(\sigma) = \sum_{j=1}^{K_E} w_j p_{\pi(j)} + \sum_{j=K_T}^{n} w_j p_{\pi(j)} - w_K \Delta + f_W(d - e). \tag{5.1}$$

Dla zadanego harmonogramu $\sigma = (\pi, e, d)$ zdefiniujmy następujące zbiory zadań:

- $\mathbf{E} \triangleq \{j \in \mathbf{J} : S_j < e\}$ - zbiór zadań, które rozpoczynają wykonywać się przed momentem e;
- $\mathbf{WI} \triangleq \{j \in \mathbf{J} : S_j \geqslant e \wedge C_j \leqslant d\}$ - zbiór zadań wykonanych całkowicie wewnątrz przedziału $[e, d]$;
- $\mathbf{T} \triangleq \{j \in \mathbf{J} : C_j > d\}$ - zbiór zadań, które kończą wykonywać się po momencie d.

Zauważmy, że:

- $\mathbf{E} = \{\pi(1), \pi(2), \ldots, \pi(K_E)\}$;
- $\mathbf{WI} = \{\pi(K_E + 1), \pi(K_E + 2), \ldots \pi(K_T - 1)\}$;
- $\mathbf{T} = \{\pi(K_T), \pi(K_T + 1), \ldots, \pi(n)\}$;
- $\mathbf{E} \cup \mathbf{WI} \cup \mathbf{T} = \mathbf{J}$.

Przedstawiony poniżej lemat zostanie wykorzystany do wykazania prawdziwości kolejnej własności.

Lemat 1. [40] *Niech* $\bar{a} = (a_1, a_2, \ldots, a_n)$ *oraz* $\bar{b} = (b_1, b_2, \ldots, b_n)$ *oznaczają dwa wektory o dodatnich współczynnikach. Iloczyn skalarny wektorów a i b jest minimalny, gdy współczynniki tych wektorów są odwrotnie uporządkowane, czyli* $a_1 \leqslant a_2 \leqslant \ldots \leqslant a_n$*;* $b_1 \geqslant b_2 \geqslant \ldots \geqslant b_n$ *lub* $a_1 \geqslant a_2 \geqslant \ldots \geqslant a_n$*;* $b_1 \leqslant b_2 \leqslant \ldots \leqslant b_n$*.*

Własność 5.4. *W optymalnym rozwiązaniu problemu* **P1***, zadania ze zbioru* $\mathbf{E} \cup \mathbf{T}$ *są uszeregowane tak, że zadanie o najdłuższym czasie wykonywania jest uszeregowane na pozycji o najmniejszej wadze, zadanie o następnym najdłuższym czasie wykonywania jest uszeregowane na pozycji o następnej najmniejszej wadze, itd.*

Dowód. Prawdziwość Własności 5.4 wynika bezpośrednio z Lematu 2 oraz wyrażenia (5.1). □

Własność 5.5. *W rozwiązaniu optymalnym problemu* **P1** *zadania ze zbioru* **WI** *są wykonywane w dowolnej kolejności.*

Dowód. Prawdziwość tej własności wynika bezpośrednio z faktu, że kolejność wykonywania zadań ze zbioru **WI** nie ma wpływu na wartość funkcji celu. □

Własność 5.6. *Istnieje rozwiązanie optymalne problemu* **P1***, w którym każde zadanie ze zbioru* **WI** *ma nie dłuższy czas wykonywania niż każde zadanie ze zbioru* $\mathbf{E} \cup \mathbf{T}$*.*

Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Przyjmijmy ponadto, że rozwiązanie to nie spełnia dowodzonej własności, czyli istnieją co najmniej dwa takie zadania $k \in \mathbf{WI}$ i $l \in \mathbf{E} \cup \mathbf{T}$, że $p_k > p_l$. Bez straty ogólności możemy założyć, że zadanie l jest najkrótszym zadaniem ze zbioru $\mathbf{E} \cup \mathbf{T}$. Zgodnie z Własnością 5.3, najkrótsze zadanie ze zbioru $\mathbf{E} \cup \mathbf{T}$ wykonuje się na pozycji K_E lub K_T. Rozważmy zatem następujące dwa przypadki.

Przypadek 1. Załóżmy, że zadanie l wykonuje się na pozycji K_E, czyli $l = \pi(K_E)$.

Skoro kolejność wykonywania zadań ze zbioru **WI** nie wpływa na wartość kryterium, to możemy założyć, bez straty ogólności, że zadanie k wykonuje się na pozycji $K_E + 1$, tzn. $k = \pi(K_E + 1)$. Łatwo zauważyć, że $e < C_l(\sigma) < C_k(\sigma) < d$. Wynika z tego, iż:

$$E_l(\sigma) = T_l(\sigma) = E_k(\sigma) = T_k(\sigma) = 0.$$

Utwórzmy harmonogram $\sigma' = (\pi', e, d)$ z harmonogramu σ taki, że zadania $\pi(K_E)$ i $\pi(K_E + 1)$ są wykonywane w odwrotnej kolejności, tzn. $\pi'(K_E) = \pi(K_E + 1)$ oraz $\pi'(K_E + 1) = \pi(K_E)$. Zauważmy, że po takiej zamianie, terminy rozpoczęcia i zakończenia

wykonywania pozostałych zadań nie zmienią się. Oczywiste jest, że $C_{\pi'(K_E+1)} = C_{\pi(K_E+1)}$ i $S_{\pi'(K_E)} = S_{\pi(K_E)}$. Ponadto uwzględniając założenie $p_k > p_l$, otrzymujemy:

$$C_{\pi'(K_E)} = S_{\pi'(K_E)} + p_{\pi'(K_E)} = S_{\pi(K_E)} + p_k >$$
$$S_{\pi(K_E)} + p_l = S_{\pi(K_E)} + p_{\pi(K_E)} = C_{\pi(K_E)} > e.$$

Wynika z tego, iż:

$$E_l(\sigma') = T_l(\sigma') = E_k(\sigma') = T_k(\sigma') = 0.$$

Oznacza to, że:

$$Z(\sigma) = Z(\sigma').$$

Przypadek 2. Załóżmy, że zadanie l wykonuje się na pozycji K_T, czyli $l = \pi(K_T)$.

Skoro kolejność wykonywania zadań ze zbioru **WI** nie wpływa na wartość kryterium, to możemy założyć, bez straty ogólności, że zadanie k wykonuje się na pozycji $K_T - 1$, tzn. $k = \pi(K_T - 1)$. Łatwo zauważyć, że $e < C_k(\sigma) \leqslant d < C_l(\sigma)$. Wynika z tego, iż

$$E_k(\sigma) = T_k(\sigma) = E_l(\sigma) = 0.$$

Utwórzmy harmonogram $\sigma'' = (\pi'', e, d)$ z harmonogramu σ taki, że zadania $\pi(K_T)$ i $\pi(K_T - 1)$ są wykonywane w odwrotnej kolejności, tzn. $\pi''(K_T) = \pi(K_T - 1)$ oraz $\pi''(K_T - 1) = \pi(K_T)$. Zauważmy, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się. Oczywiste jest, że $C_{\pi''(K_T)} = C_{\pi(K_T)}$. Wynika z tego, że:

$$T_l(\sigma'') = T_k(\sigma).$$

Uwzględniając założenie $p_k > p_l$, otrzymujemy:

$$C_{\pi''(K_T-1)} = S_{\pi''(K_T-1)} + p_{\pi''(K_T-1)} = S_{\pi(K_T-1)} + p_l <$$
$$S_{\pi(K_T-1)} + p_k = S_{\pi(K_T-1)} + p_{\pi(K_T-1)} = C_{\pi(K_T-1)} \leqslant d.$$

Wynika z tego, iż:

$$E_l(\sigma'') = E_k(\sigma'') = T_k(\sigma'') = 0.$$

Oznacza to, że:

$$Z(\sigma) = Z(\sigma'').$$

Z powyższego wynika, że zamiana zadań k i l nie zwiększy wartości funkcji celu. Możemy zatem uzyskać harmonogram spełniający dowodzoną własność, bez wzrostu wartości kryterium. □

Podsumujmy teraz własności rozwiązania optymalnego problemu **P1**:

- Nie ma przestojów procesora pomiędzy wykonywaniem poszczególnych zadań;
- Pierwsze zadanie rozpoczyna wykonywać się w momencie 0;
- Pewne zadanie kończy wykonywać się w momencie e, jeżeli waga pozycji K_E jest mniejsza od wagi pozycji K_T;
- Pewne zadanie kończy wykonywać się w momencie d, jeżeli waga pozycji K_E jest nie mniejsza od wagi pozycji K_T;
- Zadania ze zbioru $\mathbf{E} \cup \mathbf{T}$ są uszeregowane tak, że zadanie o najdłuższym czasie wykonywania jest uszeregowane na pozycji o najmniejszej wadze, zadanie o kolejnym najdłuższym czasie wykonywania jest wykonywane na pozycji o kolejnej najmniejszej wadze, itd.
- Kolejność wykonywania zadań ze zbioru **WI** jest dowolna;
- Czas wykonywania każdego zadania ze zbioru **WI** jest nie dłuższy od czasu wykonywania każdego zadania ze zbioru $\mathbf{E} \cup \mathbf{T}$;

Załóżmy, że znana jest szerokość pożądanego przedziału zakończenia wykonywania zadań oraz, że zadania ponumerowane są według niemalejących czasów wykonywania, tzn. $p_1 \leqslant p_2 \leqslant \ldots \leqslant p_n$. Z Własności 5.5 i 5.6 wynika wówczas, że zbiór **WI** zawiera k najkrótszych zadań, gdzie $k = \max\{l : \sum_{j=1}^{l} p_j < d - e\}$, które wykonywane są w dowolnej kolejności. Optymalna kolejność wykonywania zadań ze zbioru $\mathbf{E} \cup \mathbf{T}$ wynika z Własności 5.4. Na bazie Własności 5.3 możemy ustalić optymalne wartości parametrów Δ_E i Δ_T. Dzięki Własnościom 5.1 i 5.2 możemy wyznaczyć momenty zakończenia wykonywania zadań, jeżeli znana jest kolejność ich wykonywania. W konsekwencji, dla zadanego rozmiaru przedziału $[e, d]$ rozwiązanie, które spełnia Własności 5.1-5.6 jest optymalne.

Zdefiniujmy problem **P1**' jako problem **P1**, w którym szerokość przedziału $[e, d]$ nie jest w żaden sposób ograniczona a minimalizacji podlega następujące kryterium:

$$\sum(\alpha E_j + \beta T_j) + \theta e + \tilde{f}_W(d - e).$$

Funkcję $\tilde{f}_W$ definiujemy na bazie funkcji f_W w następujący sposób:

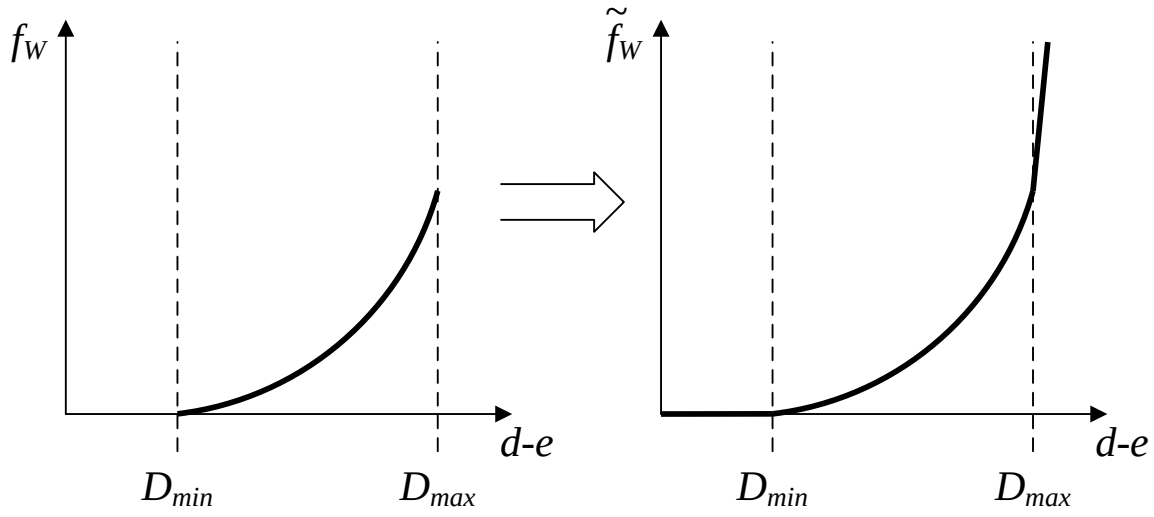
- $\tilde{f}_W(x) \triangleq \begin{cases} 0, & \text{jeżeli } x < D_{min}; \\ f_W(x), & \text{jeżeli } D_{min} \leqslant x \leqslant D_{max}; \\ \max\{\nu, \psi\}(x - D_{max}) + f_W(D_{max}), & \text{jeżeli } x \geqslant D_{min}, \end{cases}$

  gdzie:

  - $\nu = f_W(D_{max}) - f_W(D_{max} - 1)$,

$\diamond$ $\psi = \theta + n \max_j\{\alpha_j, \beta_j\}$.

Łatwo zauważyć, że funkcja $\tilde{f}_W$ jest również funkcją wypukłą i niemalejącą. Na Rysunku 5.4 przedstawiono przykładową transformację funkcji f_W do funkcji $\tilde{f}_W$.



Rysunek 5.4: Przykładowa transformacja funkcji f_W do funkcji $\tilde{f}_W$.

Łatwo zauważyć, że istnieje harmonogram optymalny problemu **P1**', w którym szerokość przedziału $[e, d]$ jest nie mniejsza niż D_{min}, ponieważ $\tilde{f}_W(x) = 0$ dla $x \leqslant D_{min}$. Łatwo również zauważyć, że w optymalnym harmonogramie problemu **P1**' szerokość przedziału $[e, d]$ jest nie większa niż D_{max}, ze względu na zbyt duży koszt związany z dalszym rozszerzaniem tego przedziału. Z powyższych rozważań wynika, że w celu rozwiązania problemu **P1** wystarczy rozwiązać problem **P1**'.

Łatwo zauważyć, że Własności 5.1-5.6 są również spełnione dla problemu **P1**'.

W celu uproszczenia notacji zdefiniujmy dodatkowo następującą funkcję:

- $\tilde{f}_W^{\Delta}(x) \triangleq \tilde{f}_W(x+1) - \tilde{f}_W(x)$.

Ponieważ $\tilde{f}_W$ jest funkcją wypukłą i niemalejącą, to $\tilde{f}_W^{\Delta}$ jest funkcją niemalejącą.

Skoro potrafimy ustalić harmonogram optymalny dla zadanej szerokości przedziału $[e, d]$, to oznaczmy przez $\sigma^*(D)$ optymalny harmonogram przy założeniu, że szerokość przedziału $[e, d]$ wynosi D. Oznaczmy ponadto przez $w_K(D)$ wartość parametru w_K dla harmonogramu $\sigma^*(D)$. Łatwo zauważyć, że $Z(\sigma^*(D+1)) - Z(\sigma^*(D)) = \tilde{f}_W^{\Delta}(D) - w_K(D)$.

Własność 5.7. *Istnieje rozwiązanie optymalne problemu* **P1**', *w którym szerokość przedziału* $[e, d]$ *wynosi* D^*, *gdzie* $\tilde{f}_W^{\Delta}(D^*) \geqslant w_K(D^*)$ *oraz* $\tilde{f}_W^{\Delta}(D^* - 1) \leqslant w_K(D^* - 1)$.

Dowód. W celu wykazania prawdziwości niniejszej własności wystarczy pokazać, że $Z(\sigma^*(D^*)) = \min\{Z(\sigma^*(D)) : D = 0, 1, \ldots, \sum p_j\}$, czyli $Z(\sigma^*(D^*)) \leqslant Z(\sigma^*(D))$ dla każdego $D = 0, 1, \ldots, \sum p_j$.

Skoro funkcja $\tilde{f}_W^{\Delta}$ jest niemalejąca, to $f_W^{\Delta}(x+1) \geqslant f_W^{\Delta}(x)$ dla każdego $x = 0, \ldots, \sum p_j - 1$. Bezpośrednio z definicji wag pozycji wynika, że $w_K(x+1) \leqslant w_K(x)$ dla każdego $x = 0, \ldots, \sum p_j - 1$. Ponieważ $\tilde{f}_W^{\Delta}(D^*) \geqslant w_K(D^*)$ oraz $\tilde{f}_W^{\Delta}(D^* - 1) \leqslant w_K(D^* - 1)$, to :

- $\tilde{f}_W^{\Delta}(x) \geqslant w_K(x)$ dla każdego $x = D^*, \ldots, \sum p_j$ oraz
- $\tilde{f}_W^{\Delta}(x) \leqslant w_K(x)$ dla każdego $x = 0, \ldots, D^* - 1$.

Rozważmy dwa możliwe przypadki.

Przypadek 1. Załóżmy, że $D < D^*$.

Różnica pomiędzy wartościami kryterium dla harmonogramów $\sigma^*(D^*)$ i $\sigma^*(D)$ wynosi:

$$Z(\sigma^*(D^*)) - Z(\sigma^*(D)) = \sum_{x=D}^{D^*-1} \left(\tilde{f}_W^{\Delta}(x) - w_K(x) \right).$$

Skoro $\tilde{f}_W^{\Delta}(x) \leqslant w_K(x)$ dla każdego $x = 0, \ldots, D^* - 1$, to:

$$Z(\sigma^*(D^*)) \leqslant Z(\sigma^*(D)).$$

Przypadek 2. Załóżmy, że $D > D^*$.

Różnica pomiędzy wartościami kryterium dla harmonogramów $\sigma^*(D^*)$ i $\sigma^*(D)$ wynosi:

$$Z(\sigma^*(D^*)) - Z(\sigma^*(D)) = \sum_{x=D^*}^{D-1} \left(\tilde{f}_W^{\Delta}(x) - w_K(x) \right).$$

Skoro $\tilde{f}_W^{\Delta}(x) \geqslant w_K(x)$ dla każdego $x = D^*, \ldots, \sum p_j$, to:

$$Z(\sigma^*(D^*)) \leqslant Z(\sigma^*(D)).$$

Z powyższego wynika, że $Z(\sigma^*(D^*)) = \min\{Z(\sigma^*(D) : D = 0, \ldots, \sum p_j\}$. □

5.4 Algorytm optymalny

Na bazie przedstawionych powyżej własności można skonstruować algorytm optymalny dla problemu **P1**'. Algorytm ten działa w dwóch etapach. W pierwszym, konstruowana jest optymalna kolejność wykonywania zadań i wyliczana jest pozycja przedziału $[e, d]$ przy założeniu, że szerokość tego przedziału wynosi 0. W drugim etapie, przedział $[e, d]$ jest sukcesywnie poszerzany poprzez zmniejszanie wartości e lub zwiększanie wartości d tak długo, dopóki rozszerzanie to powoduje zmniejszenie wartości funkcji celu.

Szczegółowy opis algorytmu jest podany poniżej. Kroki 1-2 odpowiadają pierwszemu etapowi a Kroki 3-7 odpowiadają drugiemu etapowi. W algorytmie tym wykorzystana będzie procedura *PoszukiwanieBinarne*, która zostanie przedstawiona w drugiej kolejności.

Algorytm A1'

Krok 1. Ponumeruj zadania w porządku SPT, tzn. tak, że $p_1 \leqslant \ldots \leqslant p_n$. Podstaw $x := 1$, $y := n$, $j := 1$.

Krok 2. Jeżeli $\alpha(x-1)+\theta < \beta(n-y+1)$, to umieść zadanie j na pozycji x ($\pi(x) := j$) i podstaw $x := x+1$. W przeciwnym przypadku umieść zadanie j na pozycji y ($\pi(y) := j$) i podstaw $y := y-1$. Następnie podstaw $j := j+1$. Jeżeli $j \leqslant n$, to powtórz Krok 2. W przeciwnym przypadku podstaw $e := d := C_{\pi(x-1)}$.

Krok 3. Podstaw $D := d - e$. Jeżeli $w_{K_E} > w_{K_T}$, to idź do Kroku 6.

Krok 4. Jeżeli $\tilde{f}_W^{\Delta}\left(D + p_{\pi(K_T)} - 1\right) < w_{K_T}$, to podstaw $d := d + p_{\pi(K_T)}$ i idź do Kroku 3.

Krok 5. W przeciwnym przypadku znajdź taką wartość $l = 0, \ldots, p_{\pi(K_T)} - 1$, że $\tilde{f}_W^{\Delta}(D+l+1) \geqslant w_{K_T}$ a $\tilde{f}_W^{\Delta}(D+l) \leqslant w_{K_T}$, korzystając z procedury *PoszukiwanieBinarne*$(p_{[K_T]}, w_{K_T}, D)$. Następnie podstaw $d := d + l$ i STOP

Krok 6. Jeżeli $\tilde{f}_W^{\Delta}\left(D + p_{\pi(K_E)} - 1\right) < w_{K_E}$, to podstaw $e := e - p_{\pi(K_E)}$ i idź do Kroku 3.

Krok 7. W przeciwnym przypadku znajdź taką wartość $l = 0, \ldots, p_{\pi(K_T)} - 1$, że $\tilde{f}_W^{\Delta}(D+l+1) \geqslant w_{K_T}$ a $\tilde{f}_W^{\Delta}(D+l) \leqslant w_{K_T}$, korzystając z procedury *PoszukiwanieBinarne*$(p_{[K_E]}, w_{K_E}, D)$. Następnie podstaw $e := e - l$ i STOP

PoszukiwanieBinarne$(\hat{p}, \hat{w}, D)$

Krok 1. Podstaw $a := 0$ i $b = \hat{p} - 1$. Jeżeli $\tilde{f}_W^{\Delta}(D+a) > \hat{w}$, to podstaw $l := 0$ i STOP.

Krok 2. Jeżeli $b = a + 1$, to podstaw $l := b$ i STOP.

Krok 3. Podstaw $c := a + \left\lceil \frac{b-a}{2} \right\rceil$. Jeżeli $\tilde{f}_W^{\Delta}(D+c) < \hat{w}$, to podstaw $a := c$. W przeciwnym przypadku podstaw $b := c$. Idź do Kroku 2.

Złożoność obliczeniowa procedury *PoszukiwanieBinarne*$(\hat{p}, \hat{w}, D)$ wynosi $O(\log \hat{p})$. Wypukłość funkcji $\tilde{f}_W$ gwarantuje poprawność działania tej procedury.

Niech $p_{max} \triangleq \max_j p_j$.

Lemat 2. *Algorytm* **A1'** *rozwiązuje optymalnie problem* **P1'** *w czasie* $O(n \log n + \log p_{max})$.

Dowód. Łatwo zauważyć, że o złożoności obliczeniowej całego algorytmu decyduje złożoność obliczeniowa Kroków 1, 5, i 7. Krok 1 potrzebuje czasu $O(n \log n)$. Natomiast Kroki 5 i 7 wykonują się w czasie $O(\log p_{max})$. Wynika z tego, że całkowita złożoność obliczeniowa algorytmu **A1**' wynosi $O(n \log n + \log p_{max})$.

Optymalność algorytmu wynika bezpośrednio z Własności 5.1 - 5.7. □

Twierdzenie 1. *Problem* **P1** *można rozwiązać optymalnie w czasie* $O(n \log n + \log p_{max})$.

Prawdziwość Twierdzenia 1 wynika bezpośrednio z Lematu 2 i wcześniejszych rozważań.

5.5 Podsumowanie rozdziału

W niniejszym rozdziale analizowano jednoprocesorowy problem szeregowania z doborem pożądanego przedziału zakończenia wykonywania zadań przy kryterium minimalizacji sumy kar za nieterminowe wykonanie zadań oraz kar związanych z początkiem i szerokością tego przedziału. Funkcja kary związanej z szerokością przedziału jest dowolną wypukłą funkcją niemalejącą a pozostałe funkcje kar są funkcjami liniowymi. Wykazano liczne własności rozwiązania optymalnego tego problemu. Na ich podstawie skonstruowano algorytm o złożoności $O(n \log n + \log p_{max})$, który rozwiązuje optymalnie analizowany problem.

Rozdział 6

Problemy z nieliniowymi identycznymi dla wszystkich zadań funkcjami kar za nieterminowość wykonania

6.1 Wstęp

W niniejszym rozdziale przedstawione zostaną nowe wyniki, uzyskane przez autora niniejszej pracy, dotyczące wieloprocesorowego problemu szeregowania z optymalnym doborem pożądanego przedziału zakończenia wykonywania zadań przy zadanym górnym i dolnym ograniczeniu na szerokość tego przedziału. Minimalizacji podlega suma kar za nieterminowe wykonania zadań oraz kar związanych z początkiem i szerokością pożądanego przedziału zakończenia wykonywania zadań. Wszystkie kary opisane są funkcjami nieliniowymi identycznymi dla wszystkich zadań. Analizowany problem jest uogólnieniem problemu rozważanego w Rozdziale 5.

Część przedstawionych w tym rozdziale rezultatów zostało wcześniej opublikowanych w pracach [58] i [62].

W niniejszym rozdziale przedstawione zostaną precyzyjne sformułowanie analizowanego problemu oraz definicje pewnych niezbędnych oznaczeń (Podrozdział 6.2). Następnie ustalona zostanie złożoność obliczeniowa badanego problemu oraz jego szczególnego przypadku z pojedynczym procesorem (Podrozdział 6.3) W Podrozdziale 6.4 wykazane będą własności rozwiązania optymalnego, które posłużą do konstrukcji algorytmów optymalnych opartych na metodzie programowania dynamicznego dla analizowanego problemu oraz jego szczególnego przypadku z pojedynczym procesorem (Podrozdział 6.5). Niniejszy rozdział zamyka krótkie podsumowanie uzyskanych wyników.

6.2 Sformułowanie problemu

Poniżej zostanie przedstawiona definicja badanego problemu.

Zadany jest zbiór n niezależnych i niepodzielnych zadań $\mathbf{J} = \{1, \ldots, n\}$ do wykonania na m równoległych i identycznych procesorach. Każdy procesor może wykonywać jednocześnie tylko jedną operację. Każde zadanie $j \in \mathbf{J}$ o czasie wykonywania $p_j > 0$ jest dostępne w momencie zerowym. Wszystkie zadania mają wspólny pożądany przedział zakończenia wykonywania. Początek i koniec tego przedziału oznaczmy przez e i d.

Harmonogram zadań określa przydział zadań do procesorów, momenty zakończenia wykonywania zadań oraz początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

Przypomnijmy definicje następujących oznaczeń:

- S_j - moment rozpoczęcia wykonywania zadania j;
- C_j - moment zakończenia wykonywania zadania j;
- $E_j \triangleq \{e - C_j, 0\}$ - czas przedwczesnego wykonania zadania j (przyspieszenie wykonania zadania j);
- $T_j \triangleq \{C_j - d, 0\}$ - czas zbyt późnego wykonania zadania j (opóźnienie wykonania zadania j).

Przypomnijmy również, że w celu podkreślenia, iż konkretne oznaczenie (np. S_j) odnosi się do pewnego, określonego harmonogramu σ, stosowany będzie następujący zapis: $S_j(\sigma)$.

Szerokość przedziału $[e, d]$ jest ograniczona z dołu przez D_{min} oraz z góry przez D_{max}, tzn. $D_{min} \leqslant d - e \leqslant D_{max}$.

Celem jest znalezienie takiego harmonogramu σ (spełniającego ograniczenie $D_{min} \leqslant d - e \leqslant D_{max}$), który minimalizuje wartość następującego kryterium:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\, f_E(E_j) + \, f_T(T_j)) + f_W(d - e) + f_D(e),$$

gdzie f_E, f_T, f_W i f_D są dowolnymi niemalejącymi funkcjami takimi, że $f_E(0) = f_T(0) = f_W(0) = f_D(0) = 0$.

Zakładamy, że czasy wykonywania zadań oraz początek i koniec pożądanego przedziału zakończenia wykonywania zadań są wartościami całkowitymi.

W trójpolowej notacji [34], problem ten może być przedstawiony następująco:

$$P|\langle e, d\rangle; D_{min} \leqslant d - e \leqslant D_{max}| \sum (\, f_E(E_j) + \, f_T(T_j)) + f_W(d - e) + f_D(e).$$

Badany problem będzie oznaczany również jako **P2**.

Problem **P2** jest problemem bardzo złożonym. W związku z tym będzie analizowany również jego szczególny jednoprocesorowy przypadek, który w notacji trójpolowej można zdefiniować w sposób następujący:

$$1|\langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max}|\sum(\ f_E(E_j) + \ f_T(T_j)) + f_W(d-e) + f_D(e).$$

Problem ten będziemy oznaczać również jako **P2.1**.

Dla zadanego harmonogramu zdefiniujmy następujące zbiory zadań:

- $\mathbf{J}^i$ - zbiór zadań wykonywanych na procesorze i;
- $\mathbf{E} \triangleq \{j \in \mathbf{J} : S_j < e\}$ - zbiór zadań, które rozpoczynają wykonywać się przed momentem e;
- $\mathbf{T} \triangleq \{j \in \mathbf{J} : C_j > d\}$ - zbiór zadań, które kończą wykonywać się po momencie d;
- $\mathbf{WI} \triangleq \{j \in \mathbf{J} : S_j \geqslant e \wedge C_j \leqslant d\}$ - zbiór zadań wykonanych całkowicie wewnątrz przedziału $[e,d]$.

Jeżeli $\mathbf{X}$ jest pewnym zbiorem zadań, to zbiór $\mathbf{X} \cap \mathbf{J}^i$ będzie oznaczany jako $\mathbf{X}^i$.
Zdefiniujmy ponadto następujące zadanie:

- h^i - zadanie ze zbioru $\mathbf{J}^i$, które rozpoczyna wykonywać się przed momentem e, a kończy wykonywać się po momencie d, tzn. $S_{h^i} < e$ oraz $C_{h^i} > d$ (takie zadanie może w harmongramie nie istnieć);

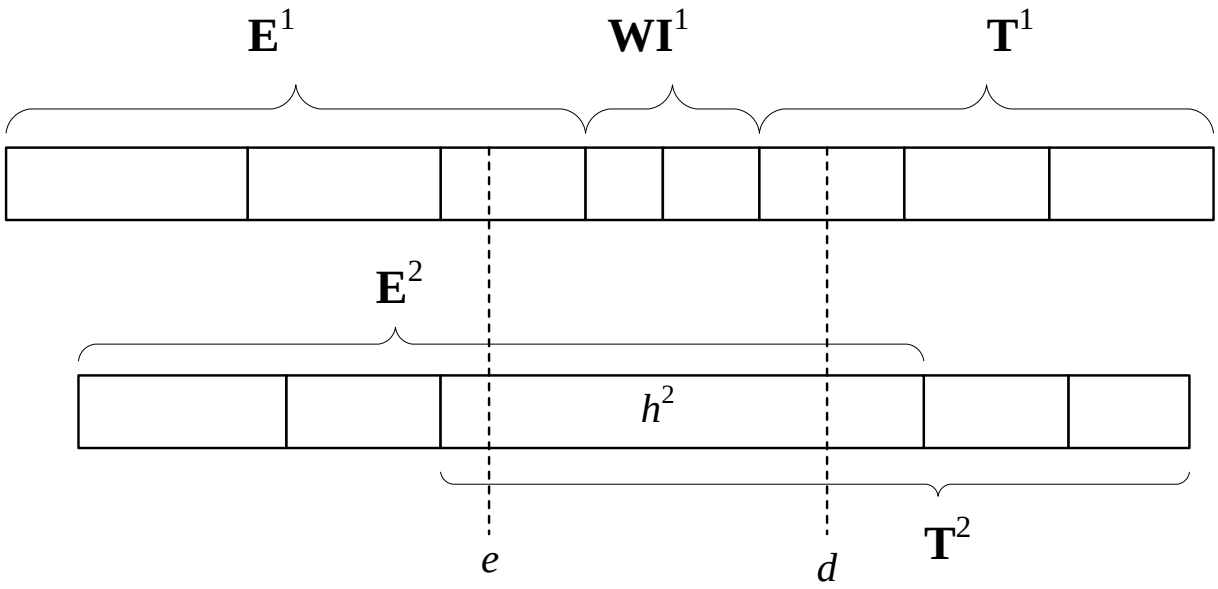
Dla problemu **P2.1**, czyli problemu jednoprocesorowego, w celu uproszczenia notacji, górny indeks w oznaczeniu h^i będzie pomijany.

Zdefiniowane oznaczenia są przedstawione na Rysunku 6.1. Należy zwrócić uwagę na fakt, iż jeżeli istnieje zadanie h^i, to zbiór $\mathbf{WI}^i$ jest pusty.

6.3 Analiza złożoności obliczeniowej

Poniżej zostanie wykazana złożoność obliczeniowa problemów **P2.1** i **P2**.

Łatwo zauważyć, że szczególny przypadek problemu **P2.1**, w którym $D_{min} = D_{max} = 0$, $f_E(x) \equiv x^2$, $f_T(x) \equiv x^2$ i $f_D(x) \equiv 0$ jest równoważny jednoprocesorowemu problemowi szeregowania z doborem pożądanego momentu zakończenia wykonywania zadań przy kryterium minimalizacji $\sum(E_j^2 + T_j^2)$ (w notacji trójpolowej problem ten definiujemy następująco: $1|d_j = d|\sum\left(E_j^2 + T_j^2\right)$). Kubiak [76] wykazał, że problem $1|d_j = d|\sum(E_j^2 + T_j^2)$ jest NP-trudny. Z powyższego wynika prawdziwość Wniosku 1.



Rysunek 6.1: Przykładowy harmonogram dla problemu dwuprocesorowego.

Wniosek 1. *Problem* **P2.1** *jest NP-trudny.*

Jeżeli $f_D(x) \equiv 0$, $f_E(x) \equiv x$, $f_T(x) \equiv x$, $f_W(x) \equiv x$, $D_{min} = 0$ oraz $D_{max} \geqslant \sum_{j \in \mathbf{J}} p_j$, to problem $P|\langle e, d\rangle; D_{min} \leqslant d - e \leqslant D_{max}|\sum(f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e)$ sprowadza się do problemu $P|\langle e, d\rangle|\sum(E_j + T_j) + (d - e)$. Janiak i Marek [54] wykazali, że problem $P|\langle e, d\rangle|\sum(E_j + T_j) + (d - e)$ jest silnie NP-trudny w ogólnym przypadku oraz NP-trudny w zwykłym sensie dla ustalonej liczby procesorów większej lub równej 2. Z powyższego wynika prawdziwość Wniosku 2.

Wniosek 2. *Problem* **P2** *jest NP-trudny w silnym sensie. Jest on NP-trudny w zwykłym sensie dla dowolnej ustalonej ilości procesorów* $m \geqslant 2$.

6.4 Własności rozwiązania optymalnego

W niniejszym podrozdziale zostaną wykazane własności rozwiązania optymalnego problemów **P2** i **P2.1**. Własności te zostaną wykorzystane później przy konstrukcji algorytmów optymalnych ich rozwiązania.

Własności rozwiązania optymalnego problemu P2.1

W pierwszej kolejności zostaną przedstawione własności problemu **P2.1**, czyli problemu jednoprocesorowego.

Własność 6.1. *Istnieje optymalne rozwiązanie problemu* **P2.1**, *w którym nie występują żadne czasy bezczynności pomiędzy wykonywaniem poszczególnych zadań.*

Dowód. Załóżmy, że znane jest rozwiązanie optymalne σ. Niech π oznacza kolejność wykonywania zadań (permutację). Przyjmijmy, że rozwiązanie σ nie spełnia dowodzonej własności, tzn. istnieją przynajmniej dwa takie zadania $\pi(j)$ oraz $\pi(j+1)$, że $S_{\pi(j+1)} \geqslant C_{\pi(j)}$. W celu uproszczenia notacji wprowadźmy następujące oznaczenia: $t_1 = C_{\pi(j)}$ oraz $t_2 = S_{\pi(j+1)}$.

Pokażemy, że przedział bezczynności procesora $[t_1, t_2]$ można zlikwidować nie zwiększając wartości funkcji celu. W tym celu należy rozpatrzeć trzy możliwe przypadki.

Przypadek 1. Załóżmy, że $t_2 \leqslant d$.

Utwórzmy harmonogram σ' z harmonogramu σ taki, że momenty rozpoczęcia wykonywania zadań $\pi(k)$, gdzie $k = 1, \ldots, j$, są zwiększone o wartość $t_2 - t_1$, tzn. $S_{\pi(k)}(\sigma') = S_{\pi(k)}(\sigma) + (t_2 - t_1)$. Oczywiste jest, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się. Łatwo również zauważyć, że $E_{\pi(k)}(\sigma') \leqslant E_{\pi(k)}(\sigma)$ dla $k = 1, \ldots, j$. Skoro zatem funkcja kary za przedwczesne wykonanie zadań, f_E, jest niemalejąca, to $f_E(E_{\pi(k)}(\sigma')) \leqslant f_E(E_{\pi(k)}(\sigma))$ dla $k = 1, \ldots, j$. Wynika z tego, że $Z(\sigma') \leqslant Z(\sigma)$.

Przypadek 2. Załóżmy, że $t_1 \geqslant d$.

Utwórzmy harmonogram σ'' z harmonogramu σ taki, że momenty rozpoczęcia wykonywania zadań $\pi(k)$, gdzie $k = j+1, \ldots, n$, są zmniejszone o wartość $t_2 - t_1$, tzn. $S_{\pi(k)}(\sigma'') = S_{\pi(k)}(\sigma) - (t_2 - t_1)$. Oczywiste jest, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się. Łatwo również zauważyć, że $T_{\pi(k)}(\sigma'') \leqslant T_{\pi(k)}(\sigma)$ dla $k = j+1, \ldots, n$. Skoro zatem funkcja kary za opóźnione wykonanie zadań, f_T, jest niemalejąca, to $f_T(T_{\pi(k)}(\sigma'')) \leqslant f_T(T_{\pi(k)}(\sigma))$ dla $k = j+1, \ldots, n$. Wynika z tego, że $Z(\sigma'') \leqslant Z(\sigma)$.

Przypadek 3. Załóżmy, że $t_1 < d < t_2$.

Wystarczy zauważyć, że przypadek ten możemy rozbić na dwa przypadki rozpatrywane powyżej dzieląc przedział bezczynności $[t_1, t_2]$ na dwa przedziały $[t_1, d]$ (Przypadek 1) oraz $[d, t_2]$ (Przypadek 2).

Ostatecznie możemy stwierdzić, że wyeliminowanie okresu bezczynności $[C_{\pi(j)}, S_{\pi(j+1)}]$ nie zwiększy wartości funkcji celu. Możemy zatem, powtarzając powyższe postępowanie dla każdego okresu bezczynności, uzyskać harmonogram spełniający dowodzoną własność bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P2.1**, które spełniają Własność 6.1.

Własność 6.2. *Istnieje optymalne rozwiązanie problemu* **P2.1**, *w którym pierwsze zadanie zaczyna wykonywać się w momencie* 0.

Dowód. Załóżmy, że znane jest rozwiązanie optymalne σ, które nie spełnia dowodzonej własności. Oznaczmy moment rozpoczęcia wykonywania pierwszego zadania w tym harmonogramie przez λ. Wartość kryterium dla harmonogramu σ wynosi:

$$Z(\sigma) = \sum_{j\in\mathbf{J}}(f_E(E_j(\sigma)) + f_T(T_j(\sigma))) + f_W(d-e) + f_D(e) = Z_0 + f_D(e),$$

gdzie $Z_0 = \sum_{j\in\mathbf{J}}(f_E(E_j(\sigma)) + f_T(T_j(\sigma))) + f_W(d-e)$.

Utwórzmy harmonogram σ' z harmonogramu σ poprzez przesunięcie całego harmonogramu o wartość λ w lewo na osi czasu, tzn. $S_j(\sigma') = S_j(\sigma) - \lambda$ dla $j \in \mathbf{J}$ oraz $e' = e - \lambda$ i $d' = d - \lambda$, gdzie e' i d' oznaczają odpowiednio początek i koniec pożądanego przedziału zakończenia wykonywania w harmonogramie σ'. Wartość kryterium dla harmonogramu σ' wynosi:

$$Z(\sigma') = \sum_{j\in\mathbf{J}}(f_E(E_j(\sigma')) + f_T(T_j(\sigma'))) + f_W(d'-e') + f_D(e') =$$
$$\sum_{j\in\mathbf{J}}(f_E(\max\{e' - C_j(\sigma'), 0\}) + f_T(\max\{C_j(\sigma') - d', 0\})) + f_W(d'-e') + f_D(e') =$$
$$\sum_{j\in\mathbf{J}}(f_E(\max\{e - \lambda - (C_j(\sigma) - \lambda), 0\}) + f_T(\max\{C_j(\sigma) - \lambda - (d-\lambda), 0\})) +$$
$$f_W(d - \lambda - (e-\lambda)) + f_D(e-\lambda) =$$
$$\sum_{j\in\mathbf{J}}(f_E(\max\{e - C_j(\sigma), 0\}) + f_T(\max\{C_j(\sigma) - d, 0\})) + f_W(d-e) + f_D(e-\lambda) =$$
$$\sum_{j\in\mathbf{J}}(f_E(E_j(\sigma)) + f_T(T_j(\sigma))) + f_W(d-e) + f_D(e-\lambda) = Z_0 + f_D(e-\lambda).$$

Uwzględniając fakt, że f_D jest funkcją niemalejącą otrzymujemy:

$$Z(\sigma) - Z(\sigma') = f_D(e) - f_D(e-\lambda) \geqslant 0,$$

a stąd:

$$Z(\sigma) \geqslant Z(\sigma').$$

Z powyższego wynika, że przesunięcie całego harmonogramu o wartość λ w lewo na osi czasu nie zwiększy wartości funkcji celu. Możemy zatem uzyskać harmonogram spełniający dowodzoną własność bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P2.1**, które spełniają Własność 6.2.

Zauważmy, że dla rozwiązania spełniającego Własności 6.1 oraz 6.2, kolejność wykonania zadań implikuje momenty ich rozpoczęcia i zakończenia. Jeżeli poprzez π oznaczymy kolejność wykonywania zadań, to momenty zakończenia wykonywania zadań można wy-

liczyć z następującego wzoru:

$$C_{\pi(j)} = \sum_{l=1}^{j} p_{\pi(l)}.$$

Zapis $\sigma = (\pi, e, d)$ będzie oznaczał harmonogram σ, w którym π określa kolejność wykonywania zadań na procesorze, natomiast e i d to odpowiednio początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

Własność 6.3. *Istnieje optymalne rozwiązanie problemu* **P2.1***, w którym zadania ze zbioru* $\mathbf{E} \setminus \{h\}$ *są uszeregowane w nierosnącym porządku czasów ich wykonywania.*

Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Załóżmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. istnieją co najmniej dwa zadania $\pi(j)$ i $\pi(j+1)$ należące do zbioru $\mathbf{E} \setminus \{h\}$ takie, że $p_{\pi(j)} < p_{\pi(j+1)}$. Wartość kryterium dla rozwiązania σ wynosi:

$$\begin{gathered}
Z(\sigma) = \sum_{j \in \mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e) = \\
Z_0 + f_E(E_{\pi(j)}) + f_E(E_{\pi(j+1)}) = \\
Z_0 + f_E(\max\{e - C_{\pi(j)}, 0\}) + f_E(\max\{e - C_{\pi(j+1)}, 0\}) = \\
Z_0 + f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j)}), 0\}) + f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)}), 0\}),
\end{gathered}$$

gdzie $Z_0 = \sum_{j \in \mathbf{J} \setminus \{\pi(j), \pi(j+1)\}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e)$.

Utwórzmy harmonogram $\sigma' = (\pi', e, d)$ z harmonogramu σ taki, że zadania $\pi(j)$ i $\pi(j+1)$ wykonywane są w odwrotnej kolejności, tzn. $\pi'(j) = \pi(j+1)$ oraz $\pi'(j+1) = \pi(j)$. Zauważmy, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się. W związku z tym, wartość kryterium wynosi:

$$\begin{gathered}
Z(\sigma') = \sum_{j \in \mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e) = \\
Z_0 + f_E(E_{\pi'(j)}) + f_E(E_{\pi'(j+1)}) = \\
Z_0 + f_E(\max\{e - C_{\pi'(j)}, 0\}) + f_E(\max\{e - C_{\pi'(j+1)}, 0\}) = \\
Z_0 + f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j+1)}), 0\}) + f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)}), 0\}).
\end{gathered}$$

Uwzględniając nierówność $p_{\pi(j)} < p_{\pi(j+1)}$, mamy:

$$\begin{gathered}
Z(\sigma) - Z(\sigma') = \\
f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j)}), 0\}) + f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)}), 0\}) - \\
\Big[f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j+1)}), 0\}) + f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)}), 0\})\Big] = \\
f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j)}), 0\}) - f_E(\max\{e - (S_{\pi(j)} + p_{\pi(j+1)}), 0\}) \geqslant 0,
\end{gathered}$$

a stąd:

$$Z(\sigma) \geqslant Z(\sigma').$$

Z powyższego wynika, że zamiana kolejności wykonywania zadań $\pi(j)$ i $\pi(j+1)$ nie zwiększy wartości funkcji celu. Możemy zatem uzyskać harmonogram, w którym zadania ze zbioru $\mathbf{E} \setminus \{h\}$ są uszeregowane w nierosnącym porządku czasów ich wykonywania, bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P2.1**, które spełniają Własność 6.3.

Własność 6.4. *Istnieje rozwiązanie optymalne problemu* **P2.1***, w którym zadania ze zbioru* $\mathbf{T} \setminus \{h\}$ *są uszeregowane w niemalejącym porządku czasów ich wykonywania.*

Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Załóżmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. istnieją co najmniej dwa zadania $\pi(j)$ i $\pi(j+1)$ należące do zbioru $\mathbf{T} \setminus \{h\}$ takie, że $p_{\pi(j)} > p_{\pi(j+1)}$. Wartość kryterium dla rozwiązania σ wynosi:

$$\begin{gathered}
Z(\sigma) = \sum_{j \in \mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e) = \\
Z_0 + f_T(T_{\pi(j)}) + f_T(T_{\pi(j+1)}) = \\
Z_0 + f_T(\max\{C_{\pi(j)} - d, 0\}) + f_T(\max\{C_{\pi(j+1)} - d, 0\}) = \\
Z_0 + f_T(\max\{S_{\pi(j)} + p_{\pi(j)} - d, 0\}) + f_T(\max\{S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)} - d, 0\}),
\end{gathered}$$

gdzie $Z_0 = \sum_{j \in \mathbf{J} \setminus \{\pi(j), \pi(j+1)\}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e)$.

Utwórzmy harmonogram $\sigma' = (\pi', e, d)$ z harmonogramu σ taki, że zadania $\pi(j)$ i $\pi(j+1)$ wykonywane są w odwrotnej kolejności, tzn. $\pi'(j) = \pi(j+1)$ oraz $\pi'(j+1) = \pi(j)$. Zauważmy, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się. W związku z tym, wartość kryterium wynosi:

$$\begin{gathered}
Z(\sigma') = \sum_{j \in \mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e) = \\
Z_0 + f_T(T_{\pi'(j)}) + f_T(T_{\pi'(j+1)}) = \\
Z_0 + f_T(\max\{C_{\pi'(j)} - d, 0\}) + f_T(\max\{C_{\pi'(j+1)} - d, 0\}) = \\
Z_0 + f_T(\max\{S_{\pi(j)} + p_{\pi(j+1)} - d, 0\}) + f_T(\max\{S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)} - d, 0\}).
\end{gathered}$$

Uwzględniając nierówność $p_{\pi(j)} > p_{\pi(j+1)}$, otrzymujemy:

$$Z(\sigma) - Z(\sigma') =$$
$$f_T(\max\{S_{\pi(j)} + p_{\pi(j)} - d, 0\}) + f_T(\max\{S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)} - d, 0\}) -$$
$$\Big[f_T(\max\{S_{\pi(j)} + p_{\pi(j+1)} - d, 0\}) + f_T(\max\{S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)} - d, 0\})\Big] =$$
$$f_T(\max\{S_{\pi(j)} + p_{\pi(j)} - d, 0\}) - f_T(\max\{S_{\pi(j)} + p_{\pi(j+1)}, 0\}) \geqslant 0,$$

a stąd

$$Z(\sigma) \geqslant Z(\sigma').$$

Z powyższego wynika, że zamiana kolejności wykonywania zadań $\pi(j)$ i $\pi(j+1)$ nie zwiększy wartości funkcji celu. Możemy zatem uzyskać harmonogram, w którym zadania ze zbioru $\mathbf{T} \setminus \{h\}$ są uszeregowane w niemalejącym porządku czasów ich wykonywania, bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P2.1**, które spełniają Własność 6.4.

Własność 6.5. *Istnieje rozwiązanie optymalne problemu* **P2.1***, w którym, jeżeli zadanie h istnieje, to:*

- *ma najkrótszy czas wykonywania spośród zadań ze zbioru* $\mathbf{E}$ *lub*
- *ma najkrótszy czas wykonywania spośród zadań ze zbioru* $\mathbf{T}$*.*

Dowód. Załóżmy, że znany jest harmonogram optymalny $\sigma = (\pi, e, d)$, w którym zadanie h wykonuje się na pozycji j. Zgodnie z Własnościami 6.3 i 6.4 zadanie o najkrótszym czasie wykonywania ze zbioru $\mathbf{E} \setminus \{h\}$ znajduje się na pozycji $j-1$, natomiast zadanie o najkrótszym czasie wykonywania ze zbioru $\mathbf{T} \setminus \{h\}$ znajduje się na pozycji $j+1$. Załóżmy, że harmonogram σ nie spełnia dowodzonej własności, tzn. $p_{\pi(j-1)} < p_{\pi(j)}$ oraz $p_{\pi(j+1)} < p_{\pi(j)}$.

Wartość kryterium dla harmonogramu σ zostanie poniżej wyrażona na dwa różne sposoby:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d - e) + f_D(e) =$$
$$Z_1 + f_E(E_{\pi(j-1)}) + f_T(T_{\pi(j)}) =$$
$$Z_1 + f_E(\max\{e - C_{\pi(j-1)}, 0\}) + f_T(\max\{C_{\pi(j)} - d, 0\}) =$$
$$Z_1 + f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j-1)}), 0\}) + f_T(\max\{S_{\pi(j-1)} + p_{\pi(j-1)} + p_{\pi(j)} - d, 0\}),$$

gdzie $Z_1 = \sum_{j \in \mathbf{J} \setminus \{\pi(j-1), \pi(j)\}} (f_E(E_j) + f_T(T_j)) + f_W(d - e) + f_D(e)$;

$$Z(\sigma) = \sum_{j\in\mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e) =$$
$$Z_2 + f_T(T_{\pi(j)}) + f_T(T_{\pi(j+1)}) =$$
$$Z_2 + f_T(\max\{C_{\pi(j)} - d, 0\}) + f_T(\max\{C_{\pi(j+1)} - d, 0\}) =$$
$$Z_2 + f_T(\max\{C_{\pi(j+1)} - p_{\pi(j+1)} - d, 0\}) + f_T(\max\{C_{\pi(j+1)} - d, 0\}),$$

gdzie $Z_2 = \sum_{j\in\mathbf{J}\setminus\{\pi(j),\pi(j+1)\}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e)$.

Utwórzmy harmonogram $\sigma' = (\pi', e, d)$ z harmonogramu σ taki, że zadania $\pi(j-1)$ i $\pi(j)$ wykonywane są w odwrotnej kolejności, tzn. $\pi'(j) = \pi(j-1)$ oraz $\pi'(j-1) = \pi(j)$. Zauważmy, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się.

Utwórzmy również harmonogram $\sigma'' = (\pi'', e, d)$ z harmonogramu σ taki, że zadania $\pi(j)$ i $\pi(j+1)$ wykonywane są w odwrotnej kolejności, tzn. $\pi''(j) = \pi(j+1)$ oraz $\pi''(j+1) = \pi(j)$. Zauważmy, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się.

Rozważmy teraz trzy możliwe przypadki:

Przypadek 1. Załóżmy, że $C_{\pi'(j)} \leqslant d$.

Wartość kryterium dla harmonogramu σ' wynosi:

$$Z(\sigma') = \sum_{j\in\mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e) =$$
$$Z_1 + f_E(E_{\pi'(j-1)}) + f_T(T_{\pi'(j)}) =$$
$$Z_1 + f_E(\max\{e - C_{\pi'(j-1)}, 0\}) + f_T(\max\{C_{\pi'(j)} - d, 0\}) =$$
$$Z_1 + f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j)}), 0\}) + f_T(\max\{S_{\pi(j-1)} + p_{\pi(j-1)} + p_{\pi(j)} - d, 0\}).$$

Uwzględniając nierówność $p_{\pi(j-1)} < p_{\pi(j)}$, otrzymujemy:

$$Z(\sigma) - Z(\sigma') =$$
$$f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j-1)}), 0\}) + f_T(\max\{S_{\pi(j-1)} + p_{\pi(j-1)} + p_{\pi(j)} - d, 0\}) -$$
$$\Big[f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j)}), 0\}) + f_T(\max\{S_{\pi(j-1)} + p_{\pi(j-1)} + p_{\pi(j)} - d, 0\}\Big] =$$
$$f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j-1)}), 0\}) - f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j)}), 0\}) \geqslant 0,$$

a stąd:

$$Z(\sigma) \geqslant Z(\sigma').$$

Przypadek 2. Załóżmy, że $S_{\pi''(j)} \geqslant e$.
Wartość kryterium dla harmonogramu σ'' wynosi:

$$Z(\sigma'') = \sum_{j \in \mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e) =$$
$$Z_2 + f_T(T_{\pi''(j)}) + f_T(T_{\pi''(j+1)}) =$$
$$Z_2 + f_T(\max\{C_{\pi''(j)-d}, 0\}) + f_T(\max\{C_{\pi''(j+1)} - d, 0\}) =$$
$$Z_2 + f_T(\max\{C_{\pi(j+1)} - p_{\pi(j)} - d, 0\}) + f_T(\max\{C_{\pi(j+1)} - d, 0\}).$$

Uwzględniając nierówność $p_{\pi(j+1)} < p_{\pi(j)}$, otrzymujemy:

$$Z(\sigma) - Z(\sigma'') =$$
$$f_T(\max\{C_{\pi(j+1)} - p_{\pi(j+1)} - d, 0\}) + f_T(\max\{C_{\pi(j+1)} - d, 0\}) -$$
$$\Big[f_T(\max\{C_{\pi(j+1)} - p_{\pi(j)} - d, 0\}) + f_T(\max\{C_{\pi(j+1)} - d, 0\})\Big] =$$
$$f_T(\max\{C_{\pi(j+1)} - p_{\pi(j+1)} - d, 0\}) - f_T(\max\{C_{\pi(j+1)} - p_{\pi(j)} - d, 0\}) \geqslant 0,$$

a stąd:

$$Z(\sigma) \geqslant Z(\sigma'').$$

Przypadek 3. Załóżmy, że $C_{\pi'(j)} > d$ oraz $S_{\pi''(j)} < e$.
Wartość kryterium dla harmonogramu σ' wynosi:

$$Z(\sigma') = \sum_{j \in \mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e) =$$
$$Z_1 + f_T(T_{\pi'(j-1)}) + f_T(T_{\pi'(j)}) =$$
$$Z_1 + f_T(\max\{C_{\pi'(j-1)} - d, 0\}) + f_T(\max\{C_{\pi'(j)} - d, 0\}) =$$
$$Z_1 + f_T(\max\{S_{\pi(j-1)} + p_{\pi(j)} - d, 0\}) + f_T(\max\{S_{\pi(j-1)} + p_{\pi(j-1)} + p_{\pi(j)} - d, 0\}).$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma') =$$
$$f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j-1)}), 0\}) + f_T(\max\{S_{\pi(j-1)} + p_{\pi(j-1)} + p_{\pi(j)} - d, 0\}) -$$
$$\Big[f_T(\max\{S_{\pi(j-1)} + p_{\pi(j)} - d, 0\}) + f_T(\max\{S_{\pi(j-1)} + p_{\pi(j-1)} + p_{\pi(j)} - d, 0\})\Big] =$$
$$f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j-1)}), 0\}) - f_T(\max\{S_{\pi(j-1)} + p_{\pi(j)} - d, 0\}). \qquad (6.1)$$

Dla harmonogramu σ'' wartość kryterium wynosi:

$$Z(\sigma'') = \sum_{j\in\mathbf{J}} (f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e) =$$
$$Z_2 + f_E(E_{\pi''(j)}) + f_T(T_{\pi''(j+1)}) =$$
$$Z_2 + f_E(\max\{e - C_{\pi''(j)}, 0\}) + f_T(\max\{C_{\pi''(j+1)} - d, 0\}) =$$
$$Z_2 + f_E(\max\{e - (C_{\pi(j+1)} - p_{\pi(j)}), 0\}) + f_T(\max\{C_{\pi(j+1)} - d, 0\}).$$

W efekcie otrzymujemy:

$$Z(\sigma) - Z(\sigma'') =$$
$$f_T(\max\{C_{\pi(j+1)} - p_{\pi(j+1)} - d, 0\}) + f_T(\max\{C_{\pi(j+1)} - d, 0\}) -$$
$$\Big[f_E(\max\{e - (C_{\pi(j+1)} - p_{\pi(j)}), 0\}) + f_T(\max\{C_{\pi(j+1)} - d, 0\})\Big] =$$
$$f_T(\max\{C_{\pi(j+1)} - p_{\pi(j+1)} - d, 0\}) - f_E(\max\{e - (C_{\pi(j+1)} - p_{\pi(j)}), 0\}). \qquad (6.2)$$

Wykażemy, że:

- jeżeli wyrażenie (6.1) przyjmuje wartość ujemną to wyrażenie (6.2) przyjmuje wartość dodatnią;
- jeżeli wyrażenie (6.2) przyjmuje wartość ujemną to wyrażenie (6.1) przyjmuje wartość dodatnią.

W tym celu wystarczy rozważyć dwa następujące przypadki:

Przypadek 3a. Załóżmy, że $Z(\sigma) - Z(\sigma') < 0$.

Z powyższego założenia wynika, że:

$$f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j-1)}), 0\}) < f_T(\max\{S_{\pi(j-1)} + p_{\pi(j)} - d, 0\}).$$

Łatwo zauważyć, że prawdziwe są następujące zależności:

$$S_{\pi(j-1)} + p_{\pi(j-1)} = C_{\pi(j-1)} \leqslant C_{\pi(j+1)} - p_{\pi(j)};$$

$$S_{\pi(j-1)} + p_{\pi(j)} \leqslant S_{\pi(j+1)} = C_{\pi(j+1)} - p_{\pi(j+1)}.$$

Uwzględniając fakt, że funkcje f_E oraz f_T są niemalejące, otrzymujemy:

$$f_E(\max\{e - (C_{\pi(j+1)} - p_{\pi(j)}), 0\}) \leqslant f_E(e - (S_{\pi(j-1)} + p_{\pi(j-1)})) <$$
$$f_T(\max\{S_{\pi(j-1)} + p_{\pi(j)} - d, 0\}) \leqslant f_T(\max\{C_{\pi(j+1)} - p_{\pi(j+1)} - d, 0\}),$$

a stąd:

$$Z(\sigma) - Z(\sigma'') =$$
$$f_T(\max\{C_{\pi(j+1)} - p_{\pi(j+1)} - d, 0\}) - f_E(\max\{e - (C_{\pi(j+1)} - p_{\pi(j)}), 0\}) > 0.$$

Przypadek 3b. Załóżmy, że $Z(\sigma) - Z(\sigma'') < 0$.

Z powyższego założenia wynika, że:

$$f_T(\max\{C_{\pi(j+1)} - p_{\pi(j+1)} - d, 0\}) < f_E(\max\{e - (C_{\pi(j+1)} - p_{\pi(j)}), 0\}).$$

Przypomnijmy, że prawdziwe są następujące zależności:

$$S_{\pi(j-1)} + p_{\pi(j-1)} \leqslant C_{\pi(j+1)} - p_{\pi(j)};$$

$$S_{\pi(j-1)} + p_{\pi(j)} \leqslant C_{\pi(j+1)} - p_{\pi(j+1)}.$$

Uwzględniając fakt, że funkcje f_E oraz f_T są niemalejące, otrzymujemy:

$$f_T(\max\{S_{\pi(j-1)} + p_{\pi(j)} - d, 0\}) \leqslant f_T(\max\{C_{\pi(j+1)} - p_{\pi(j+1)} - d, 0\}) <$$
$$f_E(\max\{e - (C_{\pi(j+1)} - p_{\pi(j)}), 0\}) \leqslant f_E(e - (S_{\pi(j-1)} + p_{\pi(j-1)})),$$

a stąd:

$$Z(\sigma) - Z(\sigma') =$$
$$f_E(\max\{e - (S_{\pi(j-1)} + p_{\pi(j-1)}), 0\}) - f_T(\max\{S_{\pi(j-1)} + p_{\pi(j)} - d, 0\}) > 0.$$

Z powyższego wynika, że zamiana kolejności wykonywania zadań $\pi(j)$ i $\pi(j+1)$ bądź zamiana kolejności wykonywania zadań $\pi(j-1)$ i $\pi(j)$ nie zwiększy wartości funkcji celu. Możemy zatem uzyskać harmonogram spełniający dowodzoną własność bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P2.1**, które spełniają Własność 6.5.

Własność 6.6. *W optymalnym rozwiązaniu problemu* **P2.1**, *zadania ze zbioru* **WI** *są uszeregowane w dowolnej kolejności.*

Dowód. Wystarczy zauważyć, że wartość kryterium nie zależy od kolejności wykonywania zadań ze zbioru **WI**. □

Podsumujmy teraz własności rozwiązania optymalnego problemu **P2.1**:

- pierwsze zadanie rozpoczyna wykonywać się w momencie 0;
- nie ma okresów bezczynności pomiędzy wykonywaniem poszczególnych zadań;
- zadania ze zbioru $\mathbf{E} \setminus \{h\}$ są uszeregowane w nierosnącym porządku czasów ich wykonywania;
- zadania ze zbioru $\mathbf{T} \setminus \{h\}$ są uszeregowane w niemalejącym porządku czasów ich wykonywania;
- zadanie h ma najkrótszy czas wykonywania spośród zadań ze zbioru $\mathbf{E}$ lub ma najkrótszy czas wykonywania spośród zdań ze zbioru $\mathbf{T}$;
- zadania ze zbioru **WI** wykonywane są w dowolnej kolejności.

Własności rozwiązania optymalnego problemu P2

Poniżej przedstawione zostaną własności problemu **P2**, czyli wieloprocesorowej wersji problemu **P2.1**.

Własność 6.7. *Istnieje optymalne rozwiązanie problemu* **P2***, w którym procesory wykonują zadania bez przestojów.*

Dowód Własności 6.7 zostanie pominięty, ponieważ byłby on niemal identyczny jak dowód analogicznej Własności 6.1.

Własność 6.8. *Istnieje rozwiązanie optymalne problemu* **P2***, w którym przynajmniej jedno zadanie zaczyna wykonywać się w momencie* 0*.*

Dowód Własności 6.8 zostanie pominięty, ponieważ byłby on niemal identyczny jak dowód analogicznej Własności 6.2.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P2**, które spełniają Własności 6.7 i 6.8.

Zauważmy, że kiedy pozycja i szerokość przedziału $[e, d]$ są ustalone oraz zadania są przydzielone do poszczególnych procesorów, to możemy rozważać oddzielnie uszeregowanie zadań na poszczególnych procesorach. W związku z tym, prawdziwość Własności 6.9-6.12 wynika bezpośrednio z prawdziwości Własności 6.3-6.6.

Własność 6.9. *Istnieje optymalne rozwiązanie problemu* **P2***, w którym zadania ze zbioru* $\mathbf{E}^i \setminus \{h^i\}$ *(*$i = 1, \ldots, m$*) są uszeregowane według nierosnących czasów wykonywania.*

Własność 6.10. *Istnieje optymalne rozwiązanie problemu* **P2***, w którym zadania ze zbioru* $\mathbf{T}^i \setminus \{h^i\}$ *(*$i = 1, \ldots, m$*) są uszeregowane według niemalejących czasów wykonywania.*

Własność 6.11. *Istnieje optymalne rozwiązanie problemu* **P2***, w którym, jeżeli zadanie* h^i *istnieje* ($i = 1, \ldots, m$)*, to ma najkrótszy czas wykonywania spośród zadań ze zbioru* $\mathbf{E}^i$ *lub ma najkrótszy czas wykonywania spośród zadań ze zbioru* $\mathbf{T}^i$*.*

Własność 6.12. *W optymalnym harmonogramie* **P2***, zadania ze zbioru* $\mathbf{WI}^i$ ($i = 1, \ldots, m$) *są uszeregowane w dowolnej kolejności.*

6.5 Algorytmy optymalne

Teraz przedstawione zostaną algorytmy programowania dynamicznego **A2** i **A2.1**, które rozwiązują optymalnie odpowiednio problemy **P2** i **P2.1**.

Zakładamy, że zadania są ponumerowane zgodnie z niemalejącym porządkiem czasów wykonywania (tzn. $p_1 \leqslant p_2 \leqslant \ldots \leqslant p_n$).

W celu uproszczenia dalszych rozważań, dla zadanego harmonogramu σ, zdefiniujmy:

- $\tau_k \triangleq \sum_{j=1}^{k} p_j$;
- $\tilde{\mathbf{E}}^i \triangleq \begin{cases} \mathbf{E}^i, & \text{jeżeli } p_{h^i} < \min_{j \in \mathbf{E}^i} p_j, \\ \mathbf{E}^i \setminus \{h^i\}, & \text{w przeciwnym przypadku;} \end{cases}$
- $\tilde{\mathbf{T}}^i \triangleq \begin{cases} \mathbf{T}^i, & \text{jeżeli } p_{h^i} < \min_{j \in \mathbf{T}^i} p_j, \\ \mathbf{T}^i \setminus \{h^i\}, & \text{w przeciwnym przypadku.} \end{cases}$

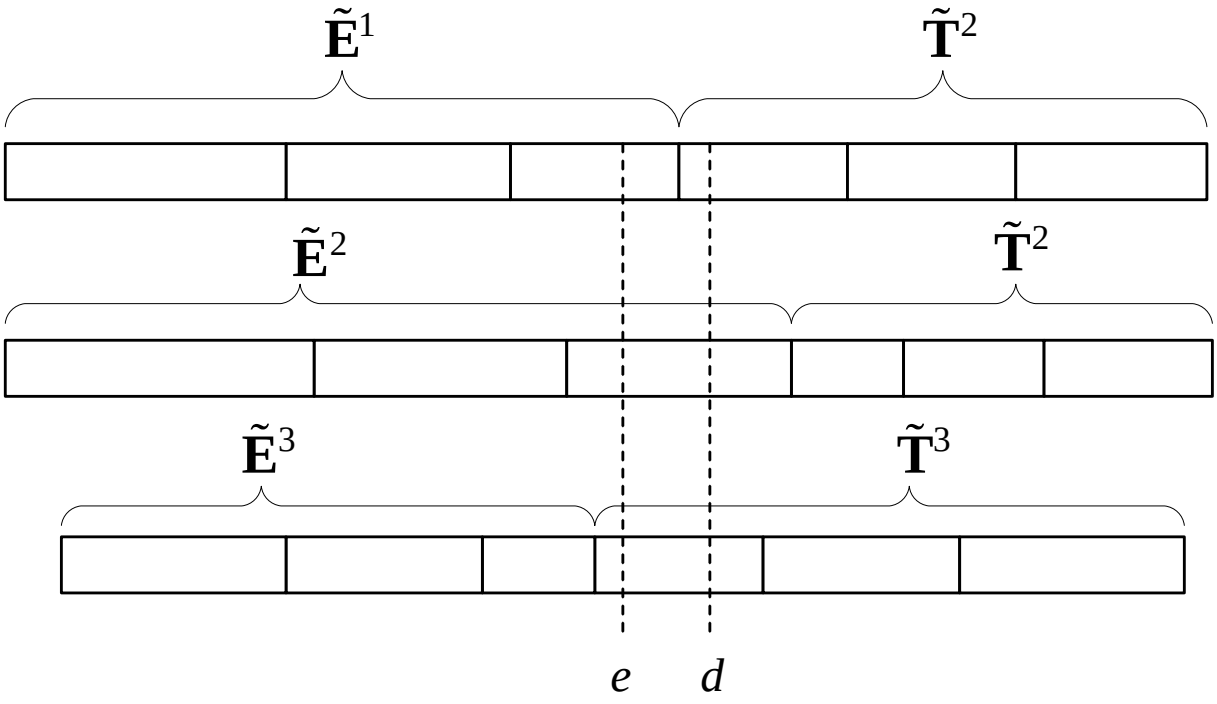
Zauważmy, że z Własności 6.9, 6.10, i 6.11 wynika, że w optymalnym harmonogramie problemu **P2**:

- $h^i \in \tilde{\mathbf{E}}^i$ lub $h^i \in \tilde{\mathbf{T}}^i$ dla każdego $i = 1, \ldots, m$;
- zadania ze zbioru $\tilde{\mathbf{E}}^i$ ($i = 1, \ldots, m$) są uszeregowane w nierosnącym porządku czasów ich wykonywania;
- zadania ze zbioru $\tilde{\mathbf{T}}^i$ ($i = 1, \ldots, m$) są uszeregowane w niemalejącym porządku czasów ich wykonywania.

W przypadku problemu **P2.1** (czyli problemu jednoprocesorowego), w oznaczeniach podanych powyżej, będzie pomijany górny indeks, który oznacza numer procesora.

Algorytm dla problemu P2.1

Na podstawie wykazanych własności oraz powyższych rozważań możemy stwierdzić, że aby uzyskać harmonogram optymalny wystarczy określić, które zadania należą do zbioru $\tilde{\mathbf{E}}$, które do zbioru $\tilde{\mathbf{T}}$ a które do zbioru **WI**. Tak więc w celu znalezienia rozwiązania



Rysunek 6.2: Przykładowy harmonogram z zaznaczonymi zbiorami $\tilde{\mathbf{E}}^i$ oraz $\tilde{\mathbf{T}}^i$.

optymalnego należy dla każdego zadania podjąć decyzję, czy należy ono do zbioru $\tilde{\mathbf{E}}$, czy do zbioru **WI**, czy do zbioru $\tilde{\mathbf{T}}$.

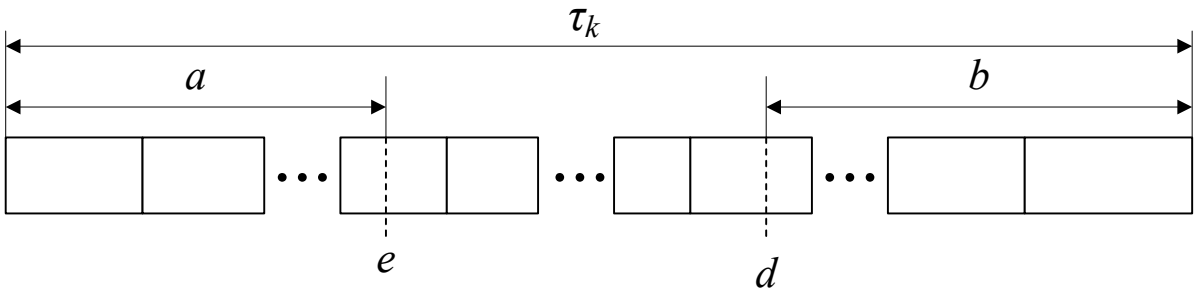
Poniżej opisane jest działanie Algorytmu **A2.1**, który rozwiązuje optymalnie problem **P2.1**. Rozpoczynamy od harmonogramu zerowego σ_0, czyli takiego, który nie zawiera żadnego zadania. Następnie generujemy kolejne harmonogramy cząstkowe σ_k (harmonogramy zawierające k pierwszych zadań) z harmonogramów σ_{k-1} poprzez przyporządkowanie zadania k do zbioru $\tilde{\mathbf{E}}$ lub do zbioru **WI**, lub do zbioru $\tilde{\mathbf{T}}$.

Będziemy mówić, że harmonogram jest w stanie (k, a, b) jeżeli:

- zawiera k pierwszych zadań;
- różnica pomiędzy początkiem przedziału $[e, d]$, a momentem rozpoczęcia wykonywania pierwszego zadania jest równa a oraz
- różnica pomiędzy zakończeniem wykonywania ostatniego zadania, a końcem przedziału $[e, d]$ jest równa b.

Parametry k, a i b będziemy nazywać zmiennymi stanu. Przykładowy harmonogram w stanie (k, a, b) został przedstawiony na Rysunku 6.3.

Niech $F_k(a, b)$ oznacza minimalną wartość sumy kar za nieterminowe wykonanie zadań dla harmonogramów w stanie (k, a, b). Zauważmy, że kara za szerokość i początek pożądanego przedziału zakończenia wykonywania zadań jest jednakowa dla wszystkich harmonogramów w stanie (k, a, b) i wynosi odpowiednio $f_E(\tau_k - a - b)$ oraz $f_D(a)$.



Rysunek 6.3: Przykładowy harmonogram w stanie (k, a, b).

Łatwo zauważyć, że optymalna wartość funkcji celu wynosi:

$$\min\{F_n(a,b) + f_W(\tau_n - a - b) + f_D(a) :$$
$$a = 0, \ldots, \tau_n;\ b = 0, \ldots, \tau_n; D_{min} \leqslant \tau_n - a - b \leqslant D_{max}\}.$$

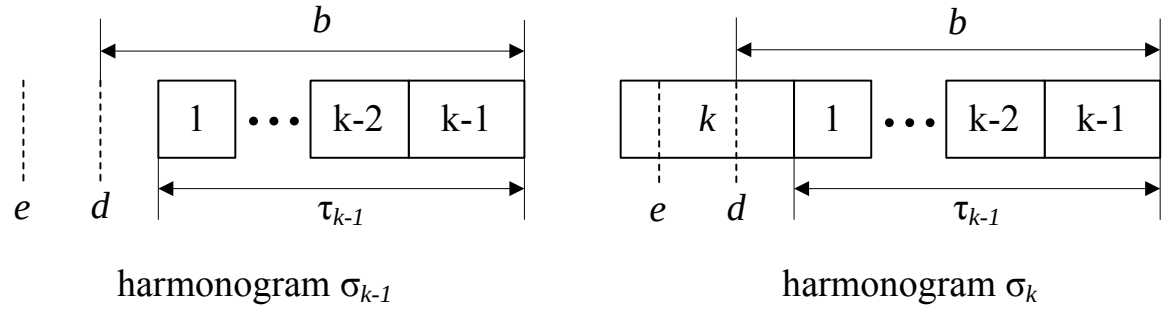
Załóżmy, że mamy harmonogram cząstkowy znajdujący się w stanie $(k-1, a, b)$.

- Jeżeli zadanie k przyporządkujemy do zbioru $\tilde{\mathbf{E}}$, to uzyskamy harmonogram w stanie $(k, a + p_k, b)$. Z Własności 6.3 wynika, że w otrzymanym harmonogramie zadanie k kończy wykonywać się w momencie $e - a$ (patrz Rysunek 6.3).
  - Jeżeli zadanie k kończy wykonywać się przed momentem d, czyli $k \neq h$, to mamy $F_k(a + p_k, b) = F_{k-1}(a, b) + f_E(\min\{a, 0\})$.
  - Jeżeli natomiast zadanie k kończy wykonywać się za momentem d, czyli $k = h$, to $F_k(a + p_k, b) = F_{k-1}(a, b) + f_T(b - \tau_{k-1})$ (patrz Rysunek 6.4).

  Łatwo zauważyć, że wartość wyrażenia $b - \tau_{k-1}$ jest dodatnia tylko i wyłącznie gdy $k = h$. Tak więc, uwzględniając obydwa powyższe przypadki, otrzymujemy: $F_k(a + p_k, b) = F_{k-1}(a, b) + f_E(\max\{a, 0\}) + f_T(\max\{b + a, 0\})$.
- Jeżeli zadanie k przyporządkujemy do zbioru **WI**, to uzyskamy harmonogram w stanie (k, a, b), natomiast $F_k(a, b) = F_{k-1}(a, b)$.
- Jeżeli zadanie k przyporządkujemy do zbioru $\tilde{\mathbf{T}}$, to uzyskamy harmonogram w stanie $(k, a, b + p_k)$. Z Własności 6.4 wynika, że w otrzymanym harmonogramie zadanie k kończy wykonywać się w momencie $d + b$ (patrz Rysunek 6.3). Mamy wówczas $F_k(a, b + p_k) = F_{k-1}(a, b) + f_T(b + p_k)$, niezależnie od tego czy $k = h$, czy nie.

Z powyższych rozważań wynika, że harmonogram w stanie (k, a, b) możemy uzyskać:

- z harmonogramu w stanie $(k-1, a-p_k, b)$ poprzez przyporządkowanie zadania k do zbioru $\tilde{\mathbf{E}}$. W tym przypadku mamy $F_k(a, b) = F_{k-1}(a - p_k, b) + f_E(\max\{a - p_k, 0\}) + f_T(\max\{b - \tau_{k-1}, 0\})$.



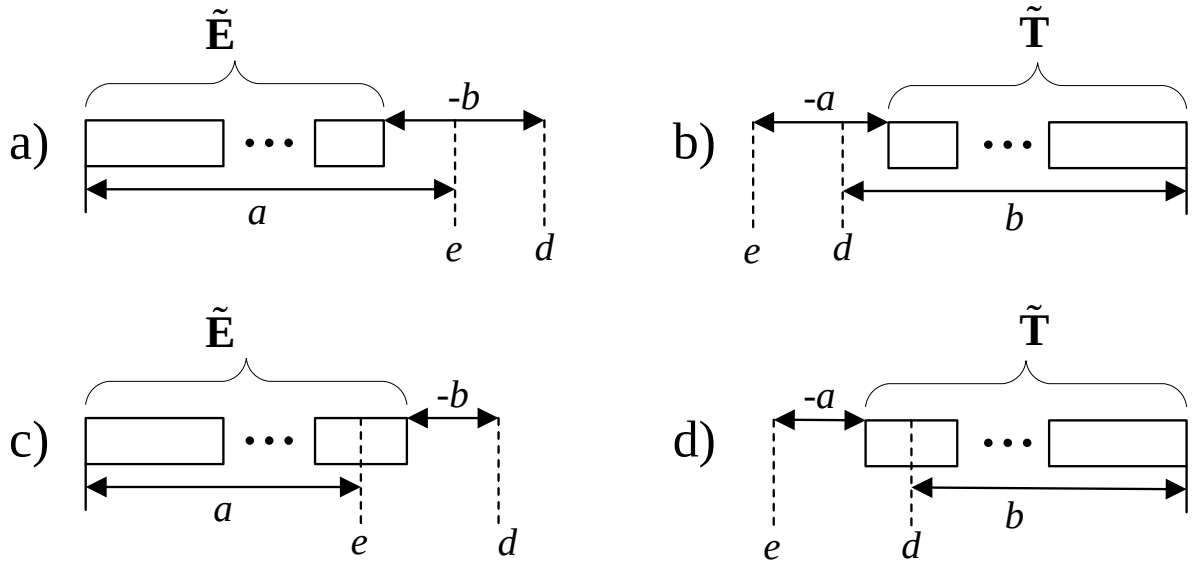
Rysunek 6.4: Dwa przykładowe harmonogramy cząstkowe σ_{k-1} i σ_k.

- z harmonogramu w stanie $(k-1, a, b)$ poprzez przyporządkowanie zadania k do zbioru **WI**. W tym przypadku mamy $F_k(a,b) = F_{k-1}(a,b)$.
- z harmonogramu w stanie $(k-1, a, b-p_k)$ poprzez przyporządkowanie zadania k do zbioru $\tilde{\mathbf{T}}$. W tym przypadku mamy $F_k(a,b) = F_{k-1}(a, b-p_k) + f_T(b)$.

Rozważmy teraz jakie wartości zmiennych stanu a i b opisują harmonogramy cząstkowe, które mogą być rozszerzone do harmonogramu optymalnego. Łatwo zauważyć, że zmienna stanu a przyjmuje wartości zarówno dodatnie jak i ujemne (patrz Rysunek 6.5). Jeżeli jednak w harmonogramie cząstkowym zbiór $\tilde{\mathbf{E}}$ nie jest zbiorem pustym, to wartość zmiennej stanu a jest dodatnia. Przyporządkowanie zadania j do zbioru $\tilde{\mathbf{E}}$ powoduje zwiększenie wartości zmiennej stanu a o czas jego wykonywania, czyli o p_j. Wynika z tego, że jest celowe rozważanie tylko takich stanów (k, a, b), w których $a > -\max_j p_j = -p_n$. W przypadku zmiennej stanu b mamy do czynienia z analogiczną sytuacją, tzn. celowe jest rozważanie tylko takich stanów (k, a, b), w których $b > -\max_j p_j = -p_n$.

Łatwo zauważyć, że szerokość przedziału $[e, d]$ w harmonogramach będących w stanie (k, a, b) wynosi $\tau_k - b - a$ (patrz Rysunek 6.3). Aby zapewnić nieujemną szerokość tego przedziału, musi być spełniona następująca nierówność: $a + b \leqslant \tau_n - D_{min}$ (w szczególności dla $k = 0$ mamy $a + b \leqslant 0$). Skoro $b > -p_n$ i $a > -p_n$, to $a \leqslant \tau - b < \tau_k + p_n$ oraz $b \leqslant \tau - a < \tau_k + p_n$. Ponadto, skoro szerokość przedziału $[e, d]$ w kompletnym harmonogramie musi być nie mniejsza niż D_{min}, to $\tau_n - b - a \geqslant D_{min}$, a w konsekwencji $a + b \leqslant \tau_n - D_{min}$. Łatwo zauważyć, że w rozwiązaniu optymalnym $a \geqslant 0$ i $b \geqslant 0$. Muszą być więc spełnione następujące zależności: $a \leqslant \tau_n - D_{min}$ i $b \leqslant \tau_n - D_{min}$. Podsumowując, w każdej iteracji k algorytmu możemy ograniczyć przedział wartości zmiennych stanu w sposób następujący: $-p_n < a \leqslant \min\{\tau_k + p_n, \tau_n - D_{min}\}$ oraz $-p_n < b \leqslant \min\{\tau_k + p_n, \tau_n - D_{min}\}$.

Poniżej zaprezentowano formalny opis algorytmu stworzonego na bazie powyższych rozważań.



Rysunek 6.5: Cztery przykładowe harmonogramy z zaznaczonymi zmiennymi stanu.

Algorytm A2.1

Krok 1 (*Inicjalizacja*) Dla $a = -p_n + 1, \ldots, \tau_n - D_{min}$; $b = -p_n + 1, \ldots, \tau_n - D_{min}$, podstaw:

$$F_0(a,b) := \begin{cases} 0, & \text{jeżeli } a+b \leqslant 0, \\ +\infty, & \text{w przeciwnym przypadku.} \end{cases}$$

Podstaw $k := 1$.

Krok 2 (*Rekursja*) Dla $a = -p_n + 1, \ldots, \min\{\tau_k + p_n, \tau_n - D_{min}\}$;
$b = -p_n + 1, \ldots, \min\{\tau_k + p_n, \tau_n - D_{min}\}$, wylicz:

$$F_k(a,b) := \min \begin{cases} F_{k-1}(a,b); & \\ F_{k-1}(a, b-p_k) + f_T(b), & \text{jeżeli } b > 0; \\ F_{k-1}(a-p_k, b) + f_E([a-p_k]^+) + f_T([t-\tau_{k-1}]^+), & \text{jeżeli } a > 0. \end{cases}$$

Jeżeli $k = n$, to idź do Kroku 3. W przeciwnym przypadku podstaw $k := k+1$ i powtórz Krok 2.

Krok 3 (*Rozwiązanie optymalne*) Wyznacz optymalną wartość kryterium:

$$F^* = \min\{F_n(a,b) + f_W(\tau_n - a - b) + f_D(a) : \\ a \geqslant 0;\ b \geqslant 0;\ D_{min} \leqslant \tau_n - a - b \leqslant D_{max}\}$$

oraz skonstruuj odpowiadające rozwiązanie optymalne metodą przeglądu wstecznego (ang. backtracking). Początek i koniec przedziału $[e, d]$ mogą być wyliczone w sposób następujący: $e = a$, $d = \tau_n - b$.

Twierdzenie 2. *Algorytm* **A2.1** *rozwiązuje optymalnie problem* **P2.1** *w czasie* $O\left(n\left(\sum p_j - D_{min}\right)^2\right)$.

Dowód. Łatwo zauważyć, że złożoność obliczeniowa algorytmu wynika bezpośrednio z rozmiaru przestrzeni stanów. Tak więc, skoro $k = 1, \ldots, n$, $a = -p_n + 1, \ldots, \tau_n - D_{min}$ i $b = -p_n + 1, \ldots, \tau_n - D_{min}$, to złożoność obliczeniowa algorytmu **A2.1** wynosi:

$$O\left(n\left(\tau_n - D_{min} + p_n\right)^2\right) = O\left(n\left(\sum_{j=1}^{n} p_j + p_n - D_{min}\right)^2\right) = O\left(n\left(\sum_{j=1}^{n} p_j - D_{min}\right)^2\right).$$

Rozważmy parę harmonogramów cząstkowych w tym samym stanie. Łatwo zaobserwować, że dla każdego harmonogramu w stanie (k, a, b) kary za początek i za szerokość przedziału $[e, d]$ wynoszą odpowiednio $f_D(a)$ i $f_W(\tau_k - a - b)$. Harmonogram z mniejszą wartością całkowitej kary za nieterminowość wykonania zadań będzie miał mniejszą wartość całkowitej kary za nieterminowość wykonania zadań po rozszerzeniu go o nie uszeregowane zadania w ten sam sposób, w jaki rozszerzony zostanie drugi harmonogram będący w tym stanie. Z powyższego wynika, że do rozszerzenia o kolejne nie uszeregowane zadania należy wybrać spośród harmonogramów będących w tym samym stanie, harmonogram o najmniejszej wartości całkowitej kary za nieterminowość wykonania zadań.

Optymalność algorytmu **A2.1** wynika bezpośrednio z powyższych rozważań oraz zasady optymalności ogólnej metody programowania dynamicznego. □

Skoro $D_{min} \geqslant 0$, to złożoność obliczeniową algorytmu **A2.1** będziemy dla uproszczenia wyrażać również w skróconej formie: $O(n(\sum p_j)^2)$.

Algorytm dla problemu P2

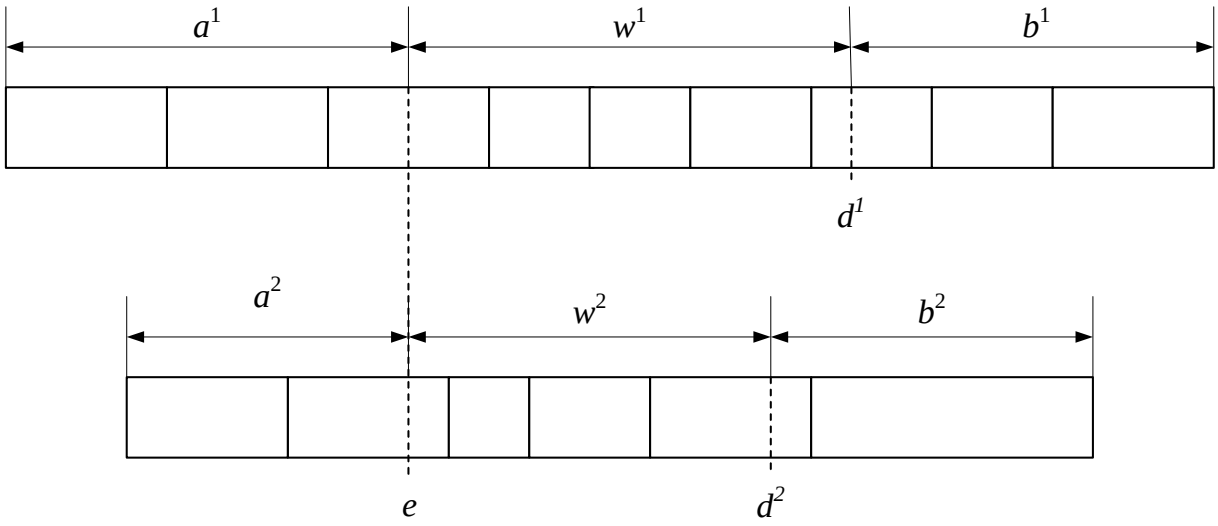
W niniejszym punkcie zostanie przedstawiony algorytm optymalny dla problemu **P2**.

Algorytm programowania dynamicznego **A2** dla problemu **P2** został skonstruowany w oparciu o Własności 6.7-6.12. Algorytm ten jest modyfikacją algorytmu **A2.1**.

Łatwo zauważyć, że aby uzyskać harmonogram optymalny wystarczy określić, które zadania należą do zbioru $\tilde{\mathbf{E}}^i$, które do zbioru $\tilde{\mathbf{T}}^i$, a które do zbioru $\mathbf{WI}^i$ $(i = 1, \ldots, m)$. Tak więc, w celu znalezienia rozwiązania należy dla każdego zadania podjąć decyzję czy należy ono do zbioru $\tilde{\mathbf{E}}^i$, czy do zbioru $\mathbf{WI}^i$, czy do zbioru $\tilde{\mathbf{T}}^i$ $(i = 1, \ldots, m)$.

Poniżej opisane jest działanie Algorytmu **A2**. Rozpoczynamy od harmonogramu zerowego σ_0, czyli takiego, który nie zawiera żadnego zadania. Następnie generujemy kolejne harmonogramy cząstkowe σ_k (harmonogramy zawierające k pierwszych zadań) z harmonogramów σ_{k-1} poprzez przyporządkowanie zadania k do zbioru $\tilde{\mathbf{E}}^i$ lub do zbioru $\mathbf{WI}^i$, lub do zbioru $\tilde{\mathbf{T}}^i$.

Niech $\tilde{S}^i$ oraz $\tilde{C}^i$ oznaczają odpowiednio moment rozpoczęcia pierwszego zadania i moment zakończenia ostatniego zadania na procesorze i w pewnym harmonogramie cząstkowym. Będziemy mówić, że harmonogram ten jest w stanie $(k, \bar{a}, \bar{w}, \bar{b})$, gdzie $\bar{a} = (a^1, a^2, \ldots, a^m)$, $\bar{w} = (w^1, w^2, \ldots, w^m)$ i $\bar{b} = (b^1, b^2, \ldots, b^m)$, jeżeli zawiera k pierwszych zadań, natomiast $a^i = e - \tilde{S}^i$, $b^i = \tilde{C}^i - d$ i $w^i = \sum_{j \in \mathbf{J}^i} p_j - a - b$. Wartość zmiennej stanu w^i będzie nam wygodnie interpretować jako szerokość pożądanego przedziału zakończenia wykonywania zadań. Łatwo zauważyć, że przy takiej interpretacji musimy dopuścić różne szerokości tego przedziału na poszczególnych procesorach - oczywiście jest to dopuszczalne tylko w harmonogramach cząstkowych. W związku z tym, niech d^i oznacza koniec pożądanego przedziału zakończenia wykonywania zadań na procesorze i. Zdefiniowane zmienne stanu zostały zilustrowane na Rysunku 6.6.



Rysunek 6.6: Przykładowy harmonogram cząstkowy z zaznaczonymi zmiennymi stanu.

W celu uproszczenia dalszych rozważań, zdefiniujmy $\bar{x}^i$ jako wektor o rozmiarze m z wartością 1 na pozycji i oraz wartością 0 na pozostałych pozycjach.

Niech $F_k(\bar{a}, \bar{w}, \bar{b})$ oznacza minimalną wartość sumy kar za nieterminowe wykonanie zadań spośród harmonogramów w stanie $(k, \bar{a}, \bar{w}, \bar{b})$. Zauważmy, że kara za początek i szerokość przedziału $[e, d]$ jest jednakowa dla wszystkich harmonogramów w tym samym stanie i wynosi odpowiednio $f_E(\max_i a^i)$ i $f_w(w^1)$, przy czym o karze za szerokość tego przedziału możemy mówić tylko w przypadku harmonogramów, w których $w^1 = w^2 = \ldots = w^m$.

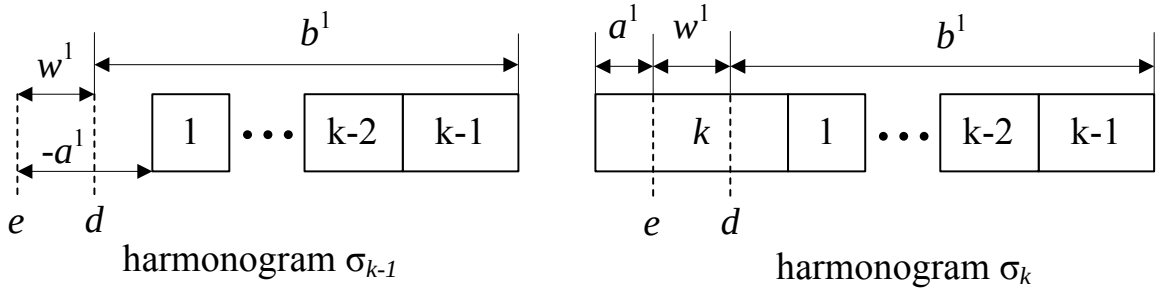
Łatwo zauważyć, że optymalna wartość funkcji celu wynosi:

$$\min\{F_n(\bar{a}, \bar{w}\bar{b}) + f_W(w^1) + f_D(\max_i a^i) : D_{min} \leqslant w^1 = w^2 = \ldots = w_m \leqslant D_{max}\}.$$

Załóżmy, że mamy harmonogram cząstkowy znajdujący się w stanie $(k-1, \bar{a}, \bar{w}, \bar{b})$.

- Jeżeli zadanie k przyporządkujemy do zbioru $\tilde{\mathbf{E}}^i$, to uzyskamy harmonogram w stanie $(k, \bar{a} + \bar{x}^i p_k, \bar{w}, \bar{b})$. Z Własności 6.9 wynika, że w otrzymanym harmonogramie zadanie k kończy wykonywać się w momencie $e - a^i$ (patrz Rysunek 6.6).
  - Jeżeli zadanie k kończy wykonywać się przed momentem d, czyli $k \neq h^i$, to mamy $F_k(\bar{a} + \bar{x}^i p_k, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b}) + f_E(\min\{a^i, 0\})$.
  - Jeżeli natomiast zadanie k kończy wykonywać się za momentem d, czyli $k = h^i$, to $F_k(\bar{a} + \bar{x}^i p_k, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b}) + f_T(-a^i - w^i)$ (patrz Rysunek 6.7).

  Łatwo zauważyć, że wartość wyrażenia $-a^i - w^i$ jest dodatnia tylko i wyłącznie gdy $k = h^i$. Tak więc, uwzględniając obydwa powyższe przypadki, otrzymujemy: $F_k(\bar{a} + \bar{x}^i p_k, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b}) + f_E(\max\{a^i, 0\}) + f_T(\max\{-a^i - w^i, 0\})$.
- Jeżeli zadanie k przyporządkujemy do zbioru $\mathbf{WI}^i$, to uzyskamy harmonogram w stanie $(k, \bar{a}, \bar{w} + \bar{x}^i p_k, \bar{b})$, natomiast $F_k(\bar{a}, \bar{w} + \bar{x}^i p_k, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b})$.
- Jeżeli zadanie k przyporządkujemy do zbioru $\tilde{\mathbf{T}}^i$, to uzyskamy harmonogram w stanie $(k, \bar{a}, \bar{w}, \bar{b} + \bar{x}^i p_k)$. Z Własności 6.10 wynika, że w otrzymanym harmonogramie zadanie k kończy wykonywać się w momencie $d + b^i$ (patrz Rysunek 6.6). Mamy wówczas $F_k(\bar{a}, \bar{w}, \bar{b} + \bar{x}^i p_k) = F_{k-1}(\bar{a}, \bar{w}, \bar{b}) + f_T(b^i + p_k)$, niezależnie od tego czy $k = h^i$, czy nie.



Rysunek 6.7: Dwa przykładowe harmonogramy cząstkowe σ_{k-1} i σ_k.

Z powyższych rozważań wynika, że harmonogram w stanie $(k, \bar{a}, \bar{w}, \bar{b})$ możemy uzyskać:

- z harmonogramu w stanie $(k-1, \bar{a} - p_k\bar{x}^i, \bar{w}, \bar{b})$ poprzez przyporządkowanie zadania k do zbioru $\tilde{\mathbf{E}}^i$ $(i = 1, \ldots, m)$. W tym przypadku mamy $F_k(\bar{a}, \bar{w}, \bar{b}) = F_{k-1}(\bar{a} - p_k\bar{x}^i, \bar{w}, \bar{b}) + f_E(\max\{a^i - p_k, 0\}) + f_T(\max\{-a^i - w^i, 0\})$.

- z harmonogramu w stanie $(k-1, \bar{a}, \bar{w}-p_k\bar{x}^i, \bar{b})$ poprzez przyporządkowanie zadania k do zbioru $\mathbf{WI}^i$ $(i = 1, \ldots, m)$. W tym przypadku mamy $F_k(\bar{a}, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w} - p_k\bar{x}^i, \bar{b})$.

- z harmonogramu w stanie $(k-1, \bar{a}, \bar{w}, \bar{b}-p_k\bar{x}^i)$ poprzez przyporządkowanie zadania k do zbioru $\tilde{\mathbf{T}}^i$ $(i = 1, \ldots, m)$. W tym przypadku mamy $F_k(\bar{a}, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b} - p_k\bar{x}^i) + f_T(b^i)$.

Rozważmy teraz jakie wartości zmiennych stanu a^i, w^i i b^i opisują harmonogramy cząstkowe, które mogą być rozszerzone do harmonogramu optymalnego. Podobnie jak miało to miejsce w przypadku problemu jednoprocesorowego, celowe jest rozważanie tylko takich stanów $(k, \bar{a}, \bar{w}, \bar{b})$, w których $a^i > -p_n$ i $b^i > -p_n$. Ponadto, łatwo zauważyć, że zmienna stanu w^i przyjmuje tylko wartości nieujemne.

Oczywiste jest, że $a^i + b^i + w^i = \sum_{j\in\mathbf{J}^i} p_j$. W konsekwencji, jeżeli żadne zadnie nie jest uszeregowane na procesorze i, to $a^i + b^i + w^i = 0$. Skoro rozważamy tylko całkowite harmonogramy, dla których $w^1 = w^2 = \ldots = w^m$, to wartość zmiennej stanu w^i nie może być większa niż $\min\{\frac{\tau_n}{m}, D_{max}\}$. Zauważmy, że w harmonogramie optymalnym nierówność $\min_i a^i \geqslant \max_i a^i + p_n$ musi być zawsze spełniona. W przeciwnym przypadku przesunięcie zadania, które zaczyna wykonywać się w momencie $\max_i a^i$ na inny procesor spowodowałoby zmniejszenie wartości kryterium. Ponadto, skoro minimalna szerokość przedziału $[e, d]$ wynosi D_{min}, to musi być spełniona następująca nierówność $\sum a^i \leqslant \tau_n - mD_{min}$. Wynika z tego, że wartość zmiennej stanu a^i nie może być większa niż $\frac{\tau_n}{m} - D_{min} + p_n$. Podobnie wartość zmiennej stanu b^i nie może być większa niż $\frac{\tau_n}{m} - D_{min} + p_n$. Niech zatem $\hat{w}_k = \min\{\tau_k, \lfloor\frac{\tau_n}{m}\rfloor, D_{max}\}$, $\hat{a}_k = \min\{\tau_k + p_n, \lfloor\frac{\tau_n}{m}\rfloor - D_{min} + p_n\}$, $\hat{b}_k = \min\{\tau_k - \hat{a}_k, \lfloor\frac{\tau_n}{m}\rfloor - D_{min} + p_n\}$ oznaczają odpowiednio maksymalne akceptowalne wartości zmiennych stanu w^i, a^i oraz b^i w iteracji k algorytmu.

Poniżej zaprezentowano formalny opis algorytmu skonstruowanego na bazie powyższych rozważań.

Algorytm A2

Krok1. (*Inicjalizacja*) Dla $w^i = 0, \ldots, D_{max}$; $a^i = -p_n + 1, \ldots, \tau_n - D_{min}$; $b^i = -p_n + 1, \ldots, \tau_n - D_{min}$ $(i = 1, \ldots, m)$, podstaw:

$$F_0(\bar{a}, \bar{w}, \bar{b}) = \begin{cases} 0, & \text{jeżeli } \forall_i\ w^i = -a^i - b^i; \\ \infty, & \text{w przeciwnym przypadku.} \end{cases}$$

Podstaw $k := 1$.

Krok 2. (*Rekursja*) Dla $a^i = -p_n + 1, \ldots, \hat{a}_k$; $b^i = -p_n + 1, \ldots, \hat{b}_k$; $w^i = 0, \ldots, \hat{w}_k$ $(i = 1, \ldots, m)$, jeżeli $\sum_{i=1}^{m}(a^i + b^i + w^i) = \tau_k$, wylicz:

$$F_k(\bar{a}, \bar{w}, \bar{b}) =$$
$$\min_{i=1,\ldots,m} \begin{cases} F_{k-1}(\bar{a} - p_k\bar{x}^i, \bar{w}, \bar{b}) + f_E([a^i - p_k]^+) + f_T([-a^i - w^i]^+), & \text{jeżeli } a^i > 0, \\ F_{k-1}(\bar{a}, \bar{w} - p_k\bar{x}^i, \bar{b}), & \\ F_{k-1}(\bar{a}, \bar{w}, \bar{b} - p_k\bar{x}^i) + f_T(t^i), & \text{jeżeli } b^i > 0. \end{cases}$$

Jeżeli $k = n$, to idź do Kroku 3. W przeciwnym przypadku podstaw $k := k + 1$ i powtórz Krok 2.

Krok 3. (*Rozwiązanie optymalne*) Wyznacz optymalną wartość kryterium:

$$F^* = \min_{\bar{a},\bar{w},\bar{b}}\{F_n(\bar{a}, \bar{w}, \bar{b}) + f_W(w^1) + f_D(\max_i a^i) :$$
$$D_{min} \leqslant w^1 = w^2 = \ldots = w^m \leqslant D_{max}\}$$

oraz skonstruuj odpowiadające mu rozwiązanie optymalne metodą przeglądu wstecznego (ang. backtracking). Początek i koniec pożądanego przedziału zakończenia wykonywania zadań może być wyznaczony następująco : $e = \max_i a^i$; $d = e + w^1$.

Twierdzenie 3. *Algorytm* **A2** *rozwiązuje optymalnie problem* **P2** *w czasie* $O\left(n\left(\frac{\tau_n}{m} - D_{min} + p_{max}\right)^{2m-1}\left(\min\left\{\frac{\tau_n}{m}, D_{max}\right\}\right)^m\right)$.

Dowód. Łatwo zauważyć, że złożoność obliczeniowa algorytmu wynika bezpośrednio z rozmiaru przestrzeni stanów. Tak więc skoro $k = 1, \ldots, n$; $-p_n < a^i \leqslant \lfloor\frac{\tau_n}{m}\rfloor - D_{min} + p_n$; $-p_n \leqslant b^i \leqslant \lfloor\frac{\tau_n}{m}\rfloor - D_{min} + p_n$; $0 \leqslant w^i \leqslant \min\left\{\frac{\tau_n}{m}, D_{max}\right\}$ (dla każdego $i = 1, \ldots, m$), to złożoność obliczeniowa algorytmu **A2** wynosi:

$$O\left(n\left(\frac{\tau_n}{m} - D_{min} + p_n\right)^{2m}\left(\min\left\{\frac{\tau_n}{m}, D_{max}\right\}\right)^m\right).$$

Skoro wyrażenie $\sum_{i=1}^{m}(a^i + b^i + w^i) = \sum_{j=1}^{k} p_j$ jest prawdziwe dla każdej iteracji algorytmu $k = 1, \ldots, n$, to możemy wyeliminować jedną ze zmiennych stanu, np. b^m. W efekcie zmniejszymy złożoność obliczeniową algorytmu do następującej wartości:

$$O\left(n\left(\frac{\tau_n}{m} - D_{min} + p_n\right)^{2m-1}\left(\min\left\{\frac{\tau_n}{m}, D_{max}\right\}\right)^m\right).$$

Rozważmy parę harmonogramów cząstkowych w tym samym stanie. Przypomnijmy, że kary za początek i za szerokość przedziału $[e, d]$ są identyczne dla wszystkich harmonogramów będących w tym samym stanie. Harmonogram z mniejszą wartością całkowitej

kary za nieterminowość wykonania zadań będzie miał mniejszą wartość całkowitej kary za nieterminowość wykonania zadań po rozszerzeniu go o nie uszeregowane zadania w ten sam sposób, w jaki rozszerzony zostanie drugi harmonogram będący w tym stanie. Z powyższego wynika, że do rozszerzenia o kolejne nie uszeregowane zadania należy wybrać spośród harmonogramów będących w tym samym stanie, harmonogram o najmniejszej wartości całkowitej kary za nieterminowość wykonania zadań.

Optymalność algorytmu **A2** wynika bezpośrednio z powyższych rozważań oraz zasady optymalności ogólnej metody programowania dynamicznego. □

Skoro $D_{min} \geqslant 0$, $\tau_n = \sum p_j$ oraz $\min\left\{\frac{\tau_n}{m}, D_{max}\right\} \leqslant \frac{\tau_n}{m}$, to złożoność obliczeniową algorytmu **A2** będziemy dla uproszczenia wyrażać również w następującej skróconej formie: $O\left(n\left(\frac{\sum p_j}{m} + p_{max}\right)^{3m-1}\right)$, gdzie $p_{max} = \max p_j$.

Uwaga 1. *Algorytm* **A2** *może być w prosty sposób zaadaptowany do rozwiązania rozszerzonej wersji problemu* **P2** *z procesorami jednorodnymi.*

Wyjaśnienie: W przypadku procesorów jednorodnych $p_{jl} = p_j/v_l$ oznacza czas wykonywania zadania j na procesorze l, gdzie v_l jest prędkością procesora l, $j = 1, \ldots, n$, $l = 1, \ldots, m$. Gdy zadania są przydzielone do procesorów, to czasy wykonywania poszczególnych zadań są ustalone. Łatwo zatem zauważyć, że Własności 6.7-6.12 są również spełnione dla problemu z procesorami jednorodnymi. Wystarczy zatem dokonać kilku drobnych zmian w algorytmie **A2** w celu zaadaptowania go do rozwiązania tego problemu.

6.6 Podsumowanie rozdziału

W niniejszym rozdziale analizowano wieloprocesorowy problem szeregowania z doborem pożądanego przedziału zakończenia wykonywania zadań przy kryterium minimalizacji sumy kar za nieterminowe wykonanie zadań oraz kar związanych z początkiem i szerokością tego przedziału. Wszystkie te kary są opisane identycznymi dla wszystkich zadań, dowolnymi funkcjami nierosnącymi. Wykazano, że problem ten jest silnie NP-trudny w ogólnym przypadku oraz że jest on NP-trudny w zwykłym sensie dla dowolnej ustalonej liczby procesorów. Ponadto wykazano liczne własności rozwiązania optymalnego. Na ich podstawie skonstruowano algorytmy oparte na metodzie programowania dynamicznego, które rozwiązują optymalnie badany problem oraz jego szczególny przypadek z pojedynczym procesorem.

Rozdział 7

Minimalizacja sumy ważonych opóźnień i przyspieszeń wykonania zadań

7.1 Wstęp

W niniejszym rozdziale przedstawione zostaną nowe wyniki, uzyskane przez autora niniejszej pracy, dotyczące wieloprocesorowego problemu szeregowania zadań z doborem pożądanego przedziału zakończenia wykonywania zadań, przy zadanym dolnym i górnym ograniczeniu na szerokość tego przedziału. Minimalizacji podlegać będzie suma ważonych nieterminowości wykonania zadań oraz kary związanej z szerokością pożądanego przedziału zakończenia wykonywania zadań. Jednostkowe kary za nieterminowe wykonanie zadań będą różne dla poszczególnych zadań.

Przedstawione w niniejszym rozdziale rezultaty zostały częściowo opublikowane w pracy [57].

Na wstępie należy wyodrębnić pewien szczególny przypadek tego problemu, mianowicie taki, w którym poniższa zależność jest prawdziwa dla każdych dwóch zadań j i l:

$$\frac{p_j}{\alpha_j} < \frac{p_l}{\alpha_l} \Rightarrow \frac{p_j}{\beta_j} \leqslant \frac{p_l}{\beta_l},$$

gdzie p_j oznacza czas wykonywania zadania j, natomiast α_j i β_j to odpowiednio jednostkowy koszt przyspieszenia/opóźnienia wykonania zadania j. Ten szczególny przypadek będzie określony mianem przypadku ze *zgodnymi ilorazami wag*.

W dalszej części tego rozdziału przedstawiona jest precyzyjna definicja analizowanego problemu (Podrozdział 7.2). Następnie zaprezentowane są niezbędne oznaczenia, wykazane własności rozwiązania optymalnego i ustalona złożoność obliczeniowa analizowanego problemu oraz jego szczególnego przypadku z pojedynczym procesorem (Podrozdział 7.3). Dla szczególnego przypadku problemu ze zgodnymi ilorazami wag skonstruowano,

na podstawie wykazanych własności, algorytm optymalny oparty na metodzie programowania dynamicznego (Podrozdział 7.4) W Podrozdziale 7.5 zaprezentowano w pełni wielomianowy schemat aproksymacyjny dla szczególnego przypadku problemu z pojedynczym procesorem, ze zgodnymi ilorazami wag oraz bez ograniczeń na szerokość pożądanego przedziału zakończenia wykonywania zadań. Następnie zaprezentowano trzy wielomianowo rozwiązywalne szczególne przypadki rozważanego problemu z jednostkowymi czasami wykonywania zadań (Podrozdział 7.6). Niniejszy rozdział zamyka krótkie podsumowanie uzyskanych wyników.

7.2 Sformułowanie problemu

Poniżej zostanie przedstawiona precyzyjna definicja problemu analizowanego w niniejszym rozdziale.

Zadany jest zbiór n niezależnych i niepodzielnych zadań $\mathbf{J} = \{1, \ldots, n\}$ do wykonania na m równoległych i identycznych procesorach. Każdy procesor może wykonywać jednocześnie tylko jedno zadanie. Każde zadanie $j \in \mathbf{J}$ o czasie wykonywania $p_j > 0$ jest dostępne w momencie zerowym. Wszystkie zadania mają wspólny pożądany przedział zakończenia wykonywania. Początek i koniec tego przedziału oznaczmy przez e i d.

Harmonogram zadań określa przydział zadań do procesorów, momenty zakończenia wykonywania zadań oraz początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

Przypomnijmy definicje następujących oznaczeń:

- S_j - moment rozpoczęcia wykonywania zadania j;
- C_j - moment zakończenia wykonywania zadania j;
- $E_j \triangleq \{e - C_j, 0\}$ - czas przedwczesnego wykonania zadania j (przyspieszenie wykonania zadania j);
- $T_j \triangleq \{C_j - d, 0\}$ - czas zbyt późnego wykonania zadania j (opóźnienie wykonania zadania j).

Przypomnijmy również, że w celu podkreślenia, iż konkretne oznaczenie (np. S_j) odnosi się do pewnego, określonego harmonogramu σ, stosowany będzie następujący zapis: $S_j(\sigma)$.

Szerokość przedziału $[e, d]$ jest ograniczona z dołu przez D_{min} oraz z góry przez D_{max}, tzn. $D_{min} \leqslant d - e \leqslant D_{max}$.

Celem problemu jest znalezienie takiego harmonogramu σ (spełniającego ograniczenie $D_{min} \leqslant d - e \leqslant D_{max}$), który minimalizuje wartość następującego kryterium:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d - e),$$

gdzie $\alpha_j \geqslant 0$, $\beta_j \geqslant 0$ i $\gamma \geqslant 0$ są pewnymi wagami, które przyjmują tylko wartości całkowite.

Zakładamy również, że czasy wykonywania zadań oraz początek i koniec pożądanego przedziału zakończenia wykonywania zadań są wartościami całkowitymi.

W trójpolowej notacji [34] problem może być przedstawiony następująco:

$$P|\langle e, d\rangle; D_{min} \leqslant d - e \leqslant D_{max}| \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d - e).$$

Badany problem będzie oznaczany również jako **P3**.

Problem **P3** jest problemem bardzo złożonym. Dlatego będzie analizowany również jego szczególny jednoprocesorowy przypadek, który w notacji trójpolowej będzie zdefiniowany w sposób następujący:

$$1|\langle e, d\rangle; D_{min} \leqslant d - e \leqslant D_{max}| \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d - e).$$

Problem ten będziemy oznaczać również jako **P3.1**.

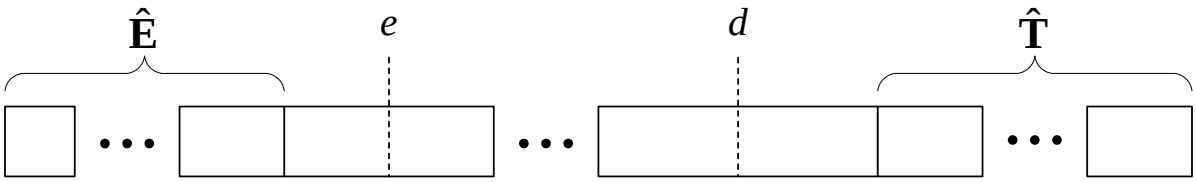
7.3 Własności rozwiązania optymalnego i złożoność obliczeniowa

W niniejszym podrozdziale zostaną wprowadzone pewne przydatne oznaczenia, wykazane własności rozwiązania optymalnego oraz ustalona złożoność obliczeniowa problemów **P3** i **P3.1**.

Dla zadanego harmonogramu σ zdefiniujemy następujące zbiory zadań:

- $\hat{\mathbf{E}} \triangleq \{j \in \mathbf{J} | C_j \leqslant e\}$ - zbiór zadań, które kończą wykonywać się w lub przed momentem e;
- $\hat{\mathbf{T}} \triangleq \{j \in \mathbf{J} | S_j \geqslant d\}$ - zbiór zadań, które rozpoczynają wykonywać się w lub po momencie d.

Zdefiniowane oznaczenia zostały zilustrowane na Rysunku 7.1.



Rysunek 7.1: Ilustracja zdefiniowanych oznaczeń.

Przypomnijmy również pewne zdefiniowane wcześniej zbiory zadań:

- $\mathbf{J}^i$ - zbiór zadań wykonywanych na procesorze i;
- $\mathbf{WI} \triangleq \{j \in \mathbf{J} : S_j \geqslant e \wedge C_j \leqslant d\}$ - zbiór zadań wykonanych całkowicie wewnątrz przedziału $[e, d]$;
- $\mathbf{T} \triangleq \{j \in \mathbf{J} : C_j > d\}$ - zbiór zadań, które kończą wykonywać się po momencie d.

Jeżeli $\mathbf{X}$ jest pewnym zbiorem zadań, to $\mathbf{X}^i = \mathbf{X} \cap \mathbf{J}^i$.

Własności problemu P3.1

Własność 7.1. *W optymalnym rozwiązaniu problemu* **P3.1** *nie ma okresów bezczynności procesora pomiędzy wykonywaniem poszczególnych zadań.*

Dowód tej własności zostanie pominięty, ponieważ byłby niemal identyczny jak dowód analogicznej Własności 5.1 dla problemu **P1**.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P3.1**, które spełniają Własność 7.1.

Własność 7.2. *Istnieje rozwiązanie optymalne problemu* **P3.1***, w którym przynajmniej jedno zadanie rozpoczyna wykonywać się w momencie* 0*.*

Dowód tej własności zostanie pominięty, ponieważ byłby niemal identyczny jak dowód analogicznej Własności 5.2 dla problemu **P1**.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P3.1**, które spełniają Własność 7.2.

Zauważmy, że dla rozwiązania spełniającego Własności 7.1 oraz 7.2, kolejność wykonania zadań implikuje momenty ich rozpoczęcia i zakończenia. Jeżeli poprzez π oznaczymy kolejność wykonywania zadań, to momenty zakończenia wykonywania zadań można wyliczyć z następującego wzoru:

$$C_{\pi(j)} = \sum_{l=1}^{j} p_{\pi(l)}.$$

Zapis $\sigma = (\pi, e, d)$ będzie oznaczał harmonogram σ, w którym π określa kolejność wykonywania zadań na procesorze, natomiast e i d to odpowiednio początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

Przypomnijmy zdefiniowane wcześniej oznaczenia:

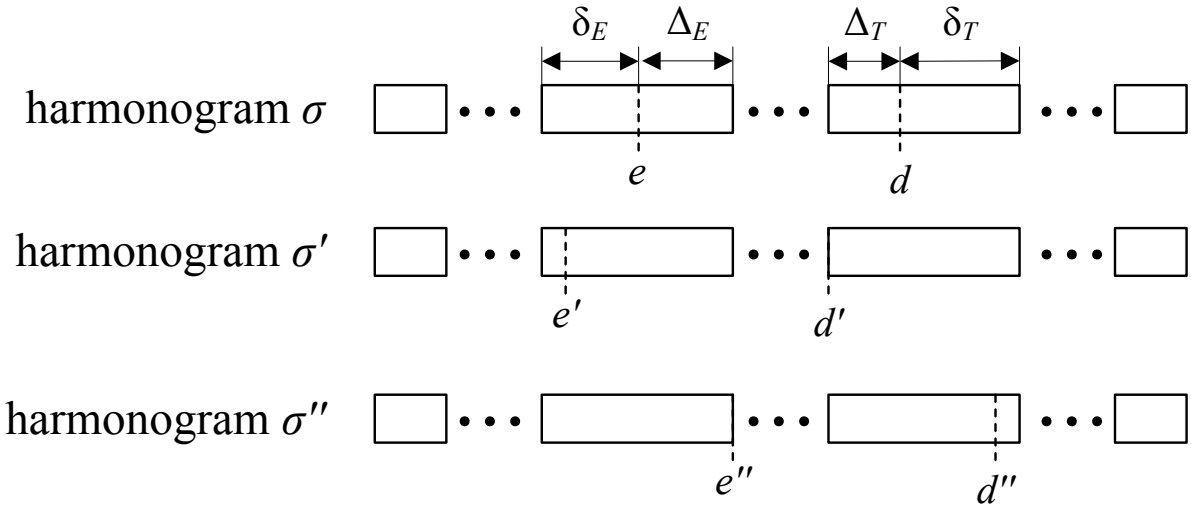
- $K_E \triangleq \min\{j : C_{\pi(j)} \geqslant e\}$ - pozycja pierwszego zadania, które kończy wykonywać się w lub po momencie e;
- $K_T \triangleq \max\{j : S_{\pi(j)} \leqslant d\}$ - pozycja ostatniego zadania, które rozpoczyna wykonywać się w lub przed momentem d;

- $\Delta_E \triangleq C_{\pi(K_E)} - e$ - część zadania $\pi(K_E)$, która wykonuje się wewnątrz przedziału $[e, d]$;
- $\delta_E \triangleq e - S_{\pi(K_E)}$ - część zadania $\pi(K_E)$, która wykonuje się przed przedziałem $[e, d]$;
- $\Delta_T \triangleq d - S_{\pi(K_T)}$ - część zadania $\pi(K_T)$, która wykonuje się wewnątrz przedziału $[e, d]$;
- $\delta_T \triangleq C_{\pi(K_T)} - d$ - część zadania $\pi(K_T)$, która wykonuje się za przedziałem $[e, d]$.

Własność 7.3. *Istnieje rozwiązanie optymalne problemu* **P3.1***, w którym jedno zadanie kończy wykonywać się w momencie e lub w momencie d.*

Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Załóżmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. nie istnieje żadne zadanie, które kończy wykonywać się w momencie e lub w momencie d. Oznacza to, że $\Delta_E \geqslant 0$ oraz $\Delta_T \geqslant 0$. W niniejszym dowodzie zakładamy, że oznaczenia $\hat{\mathbf{E}}$ i $\mathbf{T}$ odnoszą się do harmonogramu σ. Wartość kryterium dla rozwiązania σ wynosi:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d - e) =$$
$$\sum_{j \in \mathbf{J}} (\alpha_j \max\{e - C_j, 0\} + \beta_j \max\{C_j - d, 0\}) + \gamma(d - e) =$$
$$\sum_{j \in \hat{\mathbf{E}}} \alpha_j (e - C_j) + \sum_{j \in \mathbf{T}} \beta_j (C_j - d) + \gamma(d - e).$$



Rysunek 7.2: Przykładowe harmonogramy σ, σ' i σ''.

Niech $\lambda_1 \triangleq \min\{\Delta_T(\sigma), \delta_E(\sigma)\}$. Utwórzmy harmonogram $\sigma' = (\pi, e', d')$ z harmonogramu σ taki, że $e' = e - \lambda_1$ i $d' = d - \lambda_1$. Zauważmy, że w harmonogramie σ' pewne

zadanie kończy wykonywać się w momencie e' (jeżeli $\delta_E(\sigma) \leqslant \Delta_T(\sigma)$) lub w momencie d' (jeżeli $\delta_E(\sigma) \geqslant \Delta_T(\sigma)$). Wartość kryterium dla harmonogramu σ' wynosi:

$$Z(\sigma') = \sum_{j\in\mathbf{J}}(\alpha_j E_j + \beta_j T_j) + \gamma(d' - e') =$$

$$\sum_{j\in\mathbf{J}}(\alpha_j \max\{e' - C_j, 0\} + \beta_j \max\{C_j - d', 0\}) + \gamma(d' - e') =$$

$$\sum_{j\in\mathbf{J}}(\alpha_j \max\{e - \lambda_1 - C_j, 0\} + \beta_j \max\{C_j - d + \lambda_1, 0\}) + \gamma(d - \lambda_1 - e + \lambda_1) =$$

$$\sum_{j\in\hat{\mathbf{E}}}\alpha_j(e - \lambda_1 - C_j) + \sum_{j\in\mathbf{T}}\beta_j(C_j - d + \lambda_1) + \gamma(d - e) =$$

$$\sum_{j\in\hat{\mathbf{E}}}\alpha_j(e - C_j) + \sum_{j\in\mathbf{T}}\beta_j(C_j - d) + \gamma(d - e) + \left(\sum_{j\in\mathbf{T}}\beta_j - \sum_{j\in\hat{\mathbf{E}}}\alpha_j\right)\lambda_1.$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma') = \left(\sum_{j\in\mathbf{T}}\beta_j - \sum_{j\in\hat{\mathbf{E}}}\alpha_j\right)\lambda_1.$$

Niech $\lambda_2 \triangleq \min\{\delta_T(\sigma), \Delta_E(\sigma)\}$. Utwórzmy harmonogram $\sigma'' = (\pi, e'', d'')$ z harmonogramu σ taki, że $e'' = e + \lambda_2$ i $d'' = d + \lambda_2$. Zauważmy, że w harmonogramie σ'' pewne zadanie kończy wykonywać się w momencie e'' (jeżeli $\Delta_E(\sigma) \leqslant \delta_T(\sigma)$) lub w momencie d'' (jeżeli $\Delta_E(\sigma) \geqslant \delta_T(\sigma)$). Wartość kryterium dla harmonogramu σ'' wynosi:

$$Z(\sigma'') = \sum_{j\in\mathbf{J}}(\alpha_j E_j + \beta_j T_j) + \gamma(d'' - e'') =$$

$$\sum_{j\in\mathbf{J}}(\alpha_j \max\{e'' - C_j, 0\} + \beta_j \max\{C_j - d'', 0\}) + \gamma(d'' - e'') =$$

$$\sum_{j\in\mathbf{J}}(\alpha_j \max\{e + \lambda_2 - C_j, 0\} + \beta_j \max\{C_j - d - \lambda_2, 0\}) + \gamma(d + \lambda_2 - e - \lambda_2) =$$

$$\sum_{j\in\hat{\mathbf{E}}}\alpha_j(e + \lambda_2 - C_j) + \sum_{j\in\mathbf{T}}\beta_j(C_j - d - \lambda_2) + \gamma(d - e) =$$

$$\sum_{j\in\hat{\mathbf{E}}}\alpha_j(e - C_j) + \sum_{j\in\mathbf{T}}\beta_j(C_j - d) + \gamma(d - e) + \left(\sum_{j\in\hat{\mathbf{E}}}\alpha_j - \sum_{j\in\mathbf{T}}\beta_j\right)\lambda_2.$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma'') = \left(\sum_{j\in\hat{\mathbf{E}}}\alpha_j - \sum_{j\in\mathbf{T}}\beta_j\right)\lambda_2.$$

Łatwo zauważyć, że:

- jeżeli $\sum_{j\in\mathbf{T}}\beta_j \geqslant \sum_{j\in\hat{\mathbf{E}}}\alpha_j$, to $Z(\sigma') \leqslant Z(\sigma)$;

- jeżeli $\sum_{j\in\mathbf{T}}\beta_j \leqslant \sum_{j\in\hat{\mathbf{E}}}\alpha_j$, to $Z(\sigma'') \leqslant Z(\sigma)$.

Z powyższego wynika, że możemy zawsze uzyskać harmonogram, dla którego rozważana własność jest prawdziwa, bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko takie rozwiązania problemu **P3.1**, które spełniają Własność 7.3.

Własność 7.4. *W rozwiązaniu optymalnym problemu* **P3.1** *zadania ze zbioru* $\hat{\mathbf{E}}$ *są uszeregowane w nierosnącym porządku wartości ilorazu* p_j/α_j.

Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Załóżmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. istnieją co najmniej dwa zadania $\pi(j)$ i $\pi(j+1)$ należące do zbioru $\hat{\mathbf{E}}$ takie, że $p_{\pi(j)}/\alpha_{\pi(j)} < p_{\pi(j+1)}/\alpha_{\pi(j+1)}$.

Wartość kryterium dla rozwiązania σ wynosi:

$$\begin{gathered}
Z(\sigma) = \sum_{j\in\mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d-e) = \\
Z_0 + \alpha_{\pi(j)} E_{\pi(j)} + \alpha_{\pi(j+1)} E_{\pi(j+1)} = \\
Z_0 + \alpha_{\pi(j)} \max\{e - C_{\pi(j)}, 0\} + \alpha_{\pi(j+1)} \max\{e - C_{\pi(j+1)}, 0\} = \\
Z_0 + \alpha_{\pi(j)}(e - C_{\pi(j)}) + \alpha_{\pi(j+1)}(e - C_{\pi(j+1)}) = \\
Z_0 + \alpha_{\pi(j)} \left(e - \left(S_{\pi(j)} + p_{\pi(j)}\right)\right) + \alpha_{\pi(j+1)} \left(e - \left(S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)}\right)\right),
\end{gathered}$$

gdzie $Z_0 = \sum_{j\in\mathbf{J}\setminus\{\pi(j),\pi(j+1)\}} (\alpha_j E_j + \beta_j T_j) + \gamma(d-e)$.

Utwórzmy harmonogram $\sigma' = (\pi', e, d)$ z harmonogramu σ taki, że zadania $\pi(j)$ i $\pi(j+1)$ wykonywane są w odwrotnej kolejności, tzn. $\pi'(j) = \pi(j+1)$ oraz $\pi'(j+1) = \pi(j)$. Zauważmy, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się. W związku z tym, wartość kryterium wynosi:

$$\begin{gathered}
Z(\sigma') = \sum_{j\in\mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d-e) = \\
Z_0 + \alpha_{\pi'(j)} E_{\pi'(j)} + \alpha_{\pi'(j+1)} E_{\pi'(j+1)} = \\
Z_0 + \alpha_{\pi'(j)} \max\{e - C_{\pi'(j)}, 0\} + \alpha_{\pi'(j+1)} \max\{e - C_{\pi'(j+1)}, 0\} = \\
Z_0 + \alpha_{\pi'(j)}(e - C_{\pi'(j)}) + \alpha_{\pi'(j+1)}(e - C_{\pi'(j+1)}) = \\
Z_0 + \alpha_{\pi(j+1)} \left(e - \left(S_{\pi(j)} + p_{\pi(j+1)}\right)\right) + \alpha_{\pi(j)} \left(e - \left(S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)}\right)\right).
\end{gathered}$$

Różnica pomiędzy wartościami kryterium dla rozwiązań σ i σ' wynosi:

$$Z(\sigma) - Z(\sigma') = \alpha_{\pi(j)} \left(e - (S_{\pi(j)} + p_{\pi(j)})\right) + \alpha_{\pi(j+1)} \left(e - (S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)})\right) - \left[\alpha_{\pi(j+1)} \left(e - (S_{\pi(j)} + p_{\pi(j+1)})\right) + \alpha_{\pi(j)} \left(e - (S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)})\right)\right] =$$
$$\alpha_{\pi(j)} p_{\pi(j+1)} - \alpha_{\pi(j+1)} p_{\pi(j)} =$$
$$\alpha_{\pi(j)} \alpha_{\pi(j+1)} \left(p_{\pi(j+1)}/\alpha_{\pi(j+1)} - p_{\pi(j)}/\alpha_{\pi(j)}\right).$$

Uwzględniając nierówność $p_{\pi(j)}/\alpha_{\pi(j)} < p_{\pi(j+1)}/\alpha_{\pi(j+1)}$, otrzymujemy:

$$Z(\sigma) \geqslant Z(\sigma').$$

Ostatecznie możemy stwierdzić, że zamiana kolejności wykonywania zadań $\pi(j)$ i $\pi(j+1)$ nie zwiększy wartości funkcji celu. Możemy zatem uzyskać harmonogram, w którym zadania ze zbioru $\hat{\mathbf{E}}$ są uszeregowane w nierosnącym porządku wartości ilorazu p_j/α_j, bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko takie rozwiązania problemu **P3.1**, które spełniają Własność 7.4.

Własność 7.5. *W rozwiązaniu optymalnym problemu* **P3.1** *zadanie ze zbioru* $\hat{\mathbf{T}}$ *są uszeregowane w niemalejącym porządku wartości ilorazu* p_j/β_j.

Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Załóżmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. istnieją co najmniej dwa zadania $\pi(j)$ i $\pi(j+1)$ należące do zbioru $\hat{\mathbf{T}}$ takie, że $p_{\pi(j)}/\beta_{\pi(j)} > p_{\pi(j+1)}/\beta_{\pi(j+1)}$. Wartość kryterium dla rozwiązania σ wynosi:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d - e) =$$
$$Z_0 + \beta_{\pi(j)} T_{\pi(j)} + \beta_{\pi(j+1)} T_{\pi(j+1)} =$$
$$Z_0 + \beta_{\pi(j)} \max\{C_{\pi(j)} - d, 0\} + \beta_{\pi(j+1)} \max\{C_{\pi(j+1)} - d, 0\} =$$
$$Z_0 + \beta_{\pi(j)}(S_{\pi(j)} + p_{\pi(j)} - d) + \beta_{\pi(j+1)}(S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)} - d),$$

gdzie $Z_0 = \sum_{j \in \mathbf{J} \setminus \{\pi(j), \pi(j+1)\}} (\alpha_j E_j + \beta_j T_j) + \gamma(d - e)$.

Utwórzmy harmonogram $\sigma' = (\pi', e, d)$ z harmonogramu σ taki, że zadania $\pi(j)$ i $\pi(j+1)$ wykonywane są w odwrotnej kolejności, tzn. $\pi'(j) = \pi(j+1)$ oraz $\pi'(j+1) = \pi(j)$. Zauważmy, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozo-

stałych zadań nie zmienią się. W związku z tym, wartość kryterium wynosi:

$$Z(\sigma') = \sum_{j \in \mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d - e) =$$
$$Z_0 + \beta_{\pi'(j)} T_{\pi'(j)} + \beta_{\pi'(j+1)} T_{\pi'(j+1)} =$$
$$Z_0 + \beta_{\pi'(j)} \max\{C_{\pi'(j)} - d, 0\} + \beta_{\pi'(j+1)} \max\{C_{\pi'(j+1)} - d, 0\} =$$
$$Z_0 + \beta_{\pi(j+1)} \left(S_{\pi(j)} + p_{\pi(j+1)} - d\right) + \beta_{\pi(j)} \left(S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)} - d\right).$$

Różnica pomiędzy wartościami kryterium dla rozwiązań σ i σ' wynosi:

$$Z(\sigma) - Z(\sigma') = \beta_{\pi(j)}(S_{\pi(j)} + p_{\pi(j)} - d) + \beta_{\pi(j+1)}(S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)} - d) -$$
$$\left[\beta_{\pi(j+1)}(S_{\pi(j)} + p_{\pi(j+1)} - d) + \beta_{\pi(j)}(S_{\pi(j)} + p_{\pi(j)} + p_{\pi(j+1)} - d)\right] =$$
$$\beta_{\pi(j+1)} p_{\pi(j)} - \beta_{\pi(j)} p_{\pi(j+1)} =$$
$$\beta_{\pi(j+1)} \beta_{\pi(j)} \left(p_{\pi(j)}/\beta_{\pi(j)} - p_{\pi(j+1)}/\beta_{\pi(j+1)}\right).$$

Uwzględniając nierówność $p_{\pi(j)}/\beta_{\pi(j)} > p_{\pi(j+1)}/\beta_{\pi(j+1)}$, otrzymujemy:

$$Z(\sigma) \geqslant Z(\sigma').$$

Ostatecznie możemy stwierdzić, że zamiana kolejności wykonywania zadań $\pi(j)$ i $\pi(j+1)$ nie zwiększy wartości funkcji celu. Możemy zatem uzyskać harmonogram, w którym zadania ze zbioru $\hat{\mathbf{T}}$ są uszeregowane w niemalejącym porządku wartości ilorazu p_j/β_j, bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko takie rozwiązania problemu **P3.1**, które spełniają Własność 7.5.

Własność 7.6. *W rozwiązaniu optymalnym problemu* **P3.1** *zadania ze zbioru* **WI** *są uszeregowane w dowolnej kolejności.*

Dowód. Prawdziwość tej własności wynika bezpośrednio z faktu, że wartość kryterium nie zależy od kolejności wykonywania zadań ze zbioru **WI**. □

Podsumujmy teraz wszystkie wykazane dotychczas własności rozwiązania optymalnego problemu **P3.1**:

- nie ma okresów przestoju pomiędzy wykonywaniem poszczególnych zadań;
- pierwsze zadanie rozpoczyna wykonywać się w momencie 0;
- pewne zadanie kończy wykonywać się w momencie e lub w momencie d;

- zadania ze zbioru $\hat{\mathbf{E}}$ są uszeregowane w nierosnącym porządku wartości ilorazu p_j/α_j;
- zadania ze zbioru $\hat{\mathbf{T}}$ są uszeregowane w niemalejącym porządku wartości ilorazu p_j/β_j;
- zadania ze zbioru **WI** są uszeregowane w dowolnej kolejności.

Złożoność obliczeniowa problemu P3.1

Łatwo zauważyć, że jeżeli $D_{min} = D_{max} = 0$ oraz $\alpha_j = \beta_j$ $(j = 1, \ldots, n)$, to problem **P3.1** redukuje się do NP-trudnego problem $1|d_j = d|\sum \alpha_j (E_j + T_j)$ [38]. Potwierdza to prawdziwość Wniosku 3.

Wniosek 3. *Problem* **P3.1** *jest NP-trudny.*

Własności problemu P3

Poniżej zostaną przedstawione własności rozwiązania optymalnego problemu **P3**.

Dowody Własności 7.7 i 7.8 zostaną pominięte, ponieważ byłyby one niemal identyczne jak dowody analogicznych Własności 5.1 i 5.2.

Własność 7.7. *W optymalnym rozwiązaniu problemu* **P3** *nie ma okresów bezczynności pomiędzy wykonywaniem poszczególnych zadań.*

Własność 7.8. *Istnieje rozwiązanie optymalne* **P3**, *w którym przynajmniej jedno zadanie rozpoczyna wykonywać się w momencie* 0.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P3**, które spełniają Własności 7.7 i 7.8.

Załóżmy, że kolejność (permutacja) wykonywania zadań na procesorze i oznaczona jest przez π_i, zatem zdefiniujmy:

- $K_E^i \triangleq \min\{j \in \mathbf{J}^i : C_j \geqslant e\}$ - pozycja pierwszego zadania ze zbioru $\mathbf{J}^i$, które kończy wykonywać się nie wcześniej niż w momencie e;
- $K_T^i \triangleq \max\{j \in \mathbf{J}^i : S_j \leqslant d\}$ - pozycja ostatniego zadania ze zbioru $\mathbf{J}^i$, które rozpoczyna wykonywać się nie wcześniej niż w momencie d;
- $\Delta_E^i \triangleq C_{\pi_i(K_E^i)} - e$ - część zadania $\pi_i(K_E^i)$ wykonana wewnątrz przedziału $[e, d]$;
- $\Delta_T^i \triangleq d - S_{\pi_i(K_T^i)}$ - część zadania $\pi_i(K_T^i)$ wykonana wewnątrz przedziału $[e, d]$;
- $\delta_E^i \triangleq e - S_{\pi_i(K_E^i)}$ - część zadania $\pi_i(K_E^i)$ wykonana przed przedziałem $[e, d]$;
- $\delta_T^i \triangleq C_{\pi_i(K_T^i)} - d$ - część zadania $\pi_i(K_T^i)$ wykonana za przedziałem $[e, d]$.

Zauważmy, że jeżeli przesuniemy cały harmonogram w prawo na osi czasu o dowolną wartość $\varepsilon > 0$, czyli zwiększymy wartości momentów rozpoczęcia wszystkich zadań oraz wartości parametrów e i d o ε, to wartość kryterium nie ulegnie zmianie. Nie byłoby to prawdą gdyby w kryterium uwzględnić dodatkowo karę za początek pożądanego przedziału zakończenia wykonywania zadań.

Powyższe spostrzeżenie zostanie wykorzystane w dowodzie Własności 7.9

Własność 7.9. *Istnieje rozwiązanie optymalne problemu* **P3***, w którym jedno zadanie na każdym procesorze kończy wykonywać się w momencie e lub d.*

Dowód. Na wstępie zaznaczmy, że dowód ten jest podobny do dowodu Własności 7.3.

Załóżmy, że znane jest rozwiązanie optymalne σ. Załóżmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. na pewnym procesorze i nie istnieje żadne zadanie, które kończy wykonywać się w momencie e lub w momencie d. Oznacza to, że $\Delta_E^i(\sigma) \geqslant 0$ oraz $\Delta_T^i(\sigma) \geqslant 0$. W niniejszym dowodzie zakładamy, że oznaczenia $\hat{\mathbf{E}}^i$ oraz $\mathbf{T}^i$ będą odnosiły się do harmonogramu σ. Wartość kryterium dla rozwiązania σ wynosi:

$$Z(\sigma) = \sum_{j\in\mathbf{J}}(\alpha_j E_j + \beta_j T_j) + \gamma(d-e) =$$
$$Z_0 + \sum_{j\in\mathbf{J}^i}(\alpha_j \max\{e-C_j,0\} + \beta_j \max\{C_j-d,0\}) + \gamma(d-e) =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}^i}\alpha_j(e-C_j) + \sum_{j\in\mathbf{T}^i}\beta_j(C_j-d) + \gamma(d-e),$$

gdzie $Z_0 = \sum_{j\in\mathbf{J}\setminus\mathbf{J}^i}(\alpha_j E_j + \beta_j T_j)$.

Niech $\lambda_1 \triangleq \min\{\Delta_T^i(\sigma), \delta_E^i(\sigma)\}$. Utwórzmy harmonogram σ' z harmonogramu σ taki, że $C_j(\sigma') = C_j(\sigma) - \lambda_1$ dla każdego zadania $j \in \mathbf{J}\setminus\mathbf{J}^i$ oraz $e' = e - \lambda_1$ i $d' = d - \lambda_1$, gdzie e' i d' oznaczają odpowiednio początek i koniec pożądanego przedziału zakończenia wykonywania zadań w harmonogramie σ'. Skoro przesuwamy momenty zakończenia wykonywania zadań, a tym samym momenty ich rozpoczęcia, w lewo na osi czasu, to może się zdarzyć, że pierwsze zadanie rozpocznie wykonywać się przed momentem 0 w harmonogramie σ'. Mimo tego będziemy harmonogram ten traktować jako dopuszczalny, ponieważ możemy dokonać przesunięcia całego harmonogramu w prawo na osi czasu o dowolną wartość, bez wzrostu wartości kryterium. Zauważmy, że w harmonogramie σ' pewne zadanie ze zbioru $\mathbf{J}^i$ kończy wykonywać się w momencie e' (jeżeli $\delta_E^i(\sigma) \leqslant \Delta_T^i(\sigma)$) lub w momencie d' (jeżeli $\delta_E^i(\sigma) \geqslant \Delta_T^i(\sigma)$). Dla $j \in \mathbf{J}\setminus\mathbf{J}^i$ mamy:

$$E_j(\sigma') = \max\{e'-C_j(\sigma'),0\} = \max\{e-\lambda_1-C_j(\sigma)+\lambda_1,0\} = \max\{e-C_j(\sigma),0\} = E_j(\sigma);$$

$$T_j(\sigma') = \max\{C_j(\sigma')-d',0\} = \max\{C_j(\sigma)-\lambda_1-d+\lambda_1,0\} = \max\{C_j(\sigma)-d,0\} = T_j(\sigma).$$

W związku z powyższym, wartość kryterium dla harmonogramu σ' wynosi:

$$Z(\sigma') = \sum_{j\in\mathbf{J}}(\alpha_j E_j + \beta_j T_j) + \gamma(d' - e') =$$
$$Z_0 + \sum_{j\in\mathbf{J}^i}(\alpha_j \max\{e' - C_j, 0\} + \beta_j \max\{C_j - d', 0\}) + \gamma(d' - e') =$$
$$Z_0 + \sum_{j\in\mathbf{J}^i}(\alpha_j \max\{e - \lambda_1 - C_j, 0\} + \beta_j \max\{C_j - d + \lambda_1, 0\}) + \gamma(d - e) =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}^i}\alpha_j(e - \lambda_1 - C_j) + \sum_{j\in\mathbf{T}^i}\beta_j(C_j - d + \lambda_1) + \gamma(d - e) =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}^i}\alpha_j(e - C_j) + \sum_{j\in\mathbf{T}^i}\beta_j(C_j - d) + \gamma(d - e) + \left(\sum_{j\in\mathbf{T}^i}\beta_j - \sum_{j\in\hat{\mathbf{E}}^i}\alpha_j\right)\lambda_1.$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma') = \left(\sum_{j\in\mathbf{T}^i}\beta_j - \sum_{j\in\hat{\mathbf{E}}^i}\alpha_j\right)\lambda_1.$$

Niech $\lambda_2 \triangleq \min\{\delta_T^i(\sigma), \Delta_E^i(\sigma)\}$. Utwórzmy teraz harmonogram σ'' z harmonogramu σ taki, że $C_j(\sigma'') = C_j(\sigma) + \lambda_2$ dla każdego zadania $j \in \mathbf{J} \setminus \mathbf{J}^i$ oraz $e'' = e + \lambda_2$ i $d'' = d + \lambda_2$, gdzie e'' i d'' oznaczają odpowiednio początek i koniec pożądanego przedziału zakończenia wykonywania zadań w harmonogramie σ''. Zauważmy, że w harmonogramie σ'' pewne zadanie kończy wykonywać się w momencie e'' (jeżeli $\Delta_E^i(\sigma) \leqslant \delta_T^i(\sigma)$) lub w momencie d'' (jeżeli $\Delta_E^i(\sigma) \geqslant \delta_T^i(\sigma)$). Dla $j \in \mathbf{J} \setminus \mathbf{J}^i$ mamy:

$$E_j(\sigma'') = \max\{e'' - C_j(\sigma''), 0\} = \max\{e + \lambda_2 - C_j(\sigma) - \lambda_2, 0\} = \max\{e - C_j(\sigma), 0\} = E_j(\sigma);$$

$$T_j(\sigma'') = \max\{C_j(\sigma'') - d'', 0\} = \max\{C_j(\sigma) + \lambda_2 - d - \lambda_2, 0\} = \max\{C_j(\sigma) - d, 0\} = T_j(\sigma).$$

W związku z powyższym, wartość kryterium dla harmonogramu σ'' wynosi:

$$Z(\sigma'') = \sum_{j\in\mathbf{J}}(\alpha_j E_j + \beta_j T_j) + \gamma(d'' - e'') =$$
$$Z_0 + \sum_{j\in\mathbf{J}^i}(\alpha_j \max\{e'' - C_j, 0\} + \beta_j \max\{C_j - d'', 0\}) + \gamma(d'' - e'') =$$
$$Z_0 + \sum_{j\in\mathbf{J}^i}(\alpha_j \max\{e + \lambda_2 - C_j, 0\} + \beta_j \max\{C_j - d - \lambda_2, 0\}) + \gamma(d - e) =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}^i}\alpha_j(e + \lambda_2 - C_j) + \sum_{j\in\mathbf{T^i}}\beta_j(C_j - d - \lambda_2) + \gamma(d - e) =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}^i}\alpha_j(e - C_j) + \sum_{j\in\mathbf{T}^i}\beta_j(C_j - d) + \gamma(d - e) + \left(\sum_{j\in\hat{\mathbf{E}}^i}\alpha_j - \sum_{j\in\mathbf{T}^i}\beta_j\right)\lambda_2.$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma'') = \left(\sum_{j \in \hat{\mathbf{E}}^i} \alpha_j - \sum_{j \in \mathbf{T}^i} \beta_j \right) \lambda_2.$$

Łatwo zauważyć, że:

- jeżeli $\sum_{j \in \mathbf{T}^i} \beta_j \geqslant \sum_{j \in \hat{\mathbf{E}}^i} \alpha_j$, to $Z(\sigma') \leqslant Z(\sigma)$;
- jeżeli $\sum_{j \in \mathbf{T}^i} \beta_j \leqslant \sum_{j \in \hat{\mathbf{E}}^i} \alpha_j$, to $Z(\sigma'') \leqslant Z(\sigma)$.

Z powyższego wynika, że możemy zawsze uzyskać harmonogram, dla którego rozważana własność jest prawdziwa, bez wzrostu wartości kryterium. □

Łatwo zauważyć, że gdy zadania są przydzielone do procesorów i ustalony jest przedział $[e, d]$, to uszeregowanie zadań na każdym z procesorów może być rozpatrywane indywidualnie. Oznacza to, że prawdziwość Własności 7.4-7.6 implikuje prawdziwość Własności 7.10-7.12.

Własność 7.10. *W rozwiązaniu optymalnym problemu* **P3** *zadania ze zbioru* $\hat{\mathbf{E}}^i$ *(*$i = 1, \ldots, m$*) są uszeregowane w nierosnącym porządku wartości ilorazu* p_j/α_j.

Własność 7.11. *W rozwiązaniu optymalnym problemu* **P3** *zadanie ze zbioru* $\hat{\mathbf{T}}^i$ *(*$i = 1, \ldots, m$*) są uszeregowane w niemalejącym porządku wartości ilorazu* p_j/β_j.

Własność 7.12. *W rozwiązaniu optymalnym problemu* **P3** *zadania ze zbioru* $\mathbf{WI}^i$ *(*$i = 1, \ldots, m$*) są uszeregowane w dowolnej kolejności.*

Złożoność obliczeniowa problemu P3

Przejdźmy teraz do ustalenia złożoności obliczeniowej problemu **P3**. Janiak i Marek [54] wykazali, że problem $P|\langle e, d\rangle|\sum(E_j + T_j) + (d - e)$ jest silnie NP-trudny w ogólnym przypadku oraz NP-trudny w zwykłym sensie dla ustalonej liczby procesorów większej lub równej 2. Skoro $P|\langle e, d\rangle|\sum(E_j + T_j) + (d - e)$ jest szczególnym przypadkiem problemu **P3**, to mamy:

Wniosek 4. *Problem* **P3** *jest silnie NP-trudny. Jest on NP-trudny w zwykłym sensie dla dowolnej ustalonej ilości procesorów* $m \geqslant 2$.

7.4 Algorytmy programowania dynamicznego

W niniejszym podrozdziale zostaną zaprezentowane dwa algorytmy optymalne oparte na metodzie programowania dynamicznego dla szczególnych przypadków problemów **P3**

i **P3.1**, w których dla każdych dwóch zadań $j, l \in \mathbf{J}$ zachodzi następująca zależność:

$$\frac{p_j}{\alpha_j} < \frac{p_l}{\alpha_l} \Rightarrow \frac{p_j}{\beta_j} \leqslant \frac{p_l}{\beta_l}.$$

W notacji trójpolowej, problemy te może przedstawić w sposób następujący:

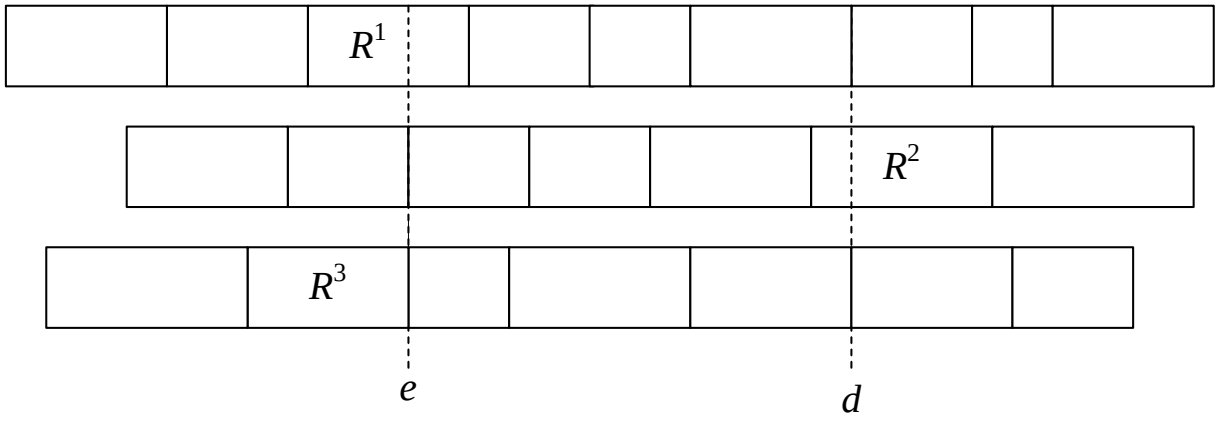
$$P|\langle e; d\rangle; D_{min} \leqslant d - e \leqslant D_{max}; \frac{p_j}{\alpha_j} \uparrow \frac{p_j}{\beta_j} \uparrow | \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d - e).$$

$$1|\langle e; d\rangle; D_{min} \leqslant d - e \leqslant D_{max}; \frac{p_j}{\alpha_j} \uparrow \frac{p_j}{\beta_j} \uparrow | \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d - e).$$

W celu skrócenia notacji, będziemy oznaczać je odpowiednio jako **P3.2** i **P3.3**.
Zakładamy, że zadania są ponumerowane tak, że $p_1/\alpha_1 \leqslant \ldots \leqslant p_n/\alpha_n$.
Zanim przedstawimy algorytmy, zdefiniujmy:

- $\tilde{R}^i$ - zadanie, które wykonuje się częściowo wewnątrz, a częściowo na zewnątrz przedziału $[e, d]$, tzn. $(S_{\tilde{R}^i} < e < C_{\tilde{R}^i}) \vee (S_{\tilde{R}^i} < d < C_{\tilde{R}^i})$.
- $\hat{R}^i$ - zadanie, które kończy wykonywać się w momencie e, tzn. $C_{\hat{R}^i} = e$
- $R^i \triangleq \begin{cases} \tilde{R}^i, & \text{jeżeli zadanie } \tilde{R}^i \text{ istnieje;} \\ \hat{R}^i, & \text{w przeciwnym przypadku.} \end{cases}$
- $\mathbf{R} \triangleq \{R^1, R^2, \ldots, R^m\}$.

Oznaczenie R^i zostało zilustrowane na Rysunku 7.3.



Rysunek 7.3: Przykładowy harmonogram z zaznaczonymi zadaniami R^i.

Zgodnie z Własnością 7.9 na każdym procesorze i istnieje zadanie R^i. Ponadto oczywiste jest, że $\hat{\mathbf{E}} \cup \hat{\mathbf{T}} \cup \mathbf{WI} \cup \mathbf{R} = \mathbf{J}$.

Dla problemu **P3.3** górny indeks "i" w powyższych oznaczeniach będzie pomijany.

Algorytm programowania dynamicznego dla problemu P3.3

W pierwszej kolejności zostanie przedstawiony algorytm programowania dynamicznego dla problemu **P3.3**, ponieważ jest on mniej skomplikowany.

Podzielmy problem **P3.3** na n podproblemów **P3.3**(R). W podproblemie **P3.3**(R) zadanie R jest ustalone. Łatwo zauważyć, że w celu optymalnego rozwiązania problemu **P3.3** wystarczy rozwiązać optymalnie problem **P3.3**(R) dla każdego $R = 1, \ldots, n$ a następnie wybrać rozwiązanie, dla którego wartość kryterium jest najmniejsza.

W związku z powyższym, w celu rozwiązania problemu **P3.3** wystarczy skonstruować algorytm optymalny dla problemu **P3.3**(R).

Zdefiniujmy:

- $\tau_k = \sum_{j \in \{1,\ldots,k\} \cup \{R\}} p_j$.

Na podstawie wykazanych własności możemy stwierdzić, że aby uzyskać harmonogram optymalny wystarczy określić, które zadania należą do zbioru $\hat{\mathbf{E}}$, które do zbioru $\hat{\mathbf{T}}$ a które do zbioru **WI**. Tak więc w celu znalezienia rozwiązania należy dla każdego zadania, z wyjątkiem zadania R, podjąć decyzję, czy należy ono do zbioru $\hat{\mathbf{E}}$, czy do zbioru **WI**, czy do zbioru $\hat{\mathbf{T}}$.

Poniżej opisano działanie algorytmu **A3.3**(R), który rozwiązuje optymalnie problem **P3.3**(R). Rozpoczynamy od harmonogramu zerowego σ_0, czyli takiego, który zawiera tylko zadanie R. Następnie generujemy kolejne harmonogramy cząstkowe σ_k (harmonogramy zawierające zadania $\{1, \ldots, k\} \cup \{R\}$) z harmonogramów σ_{k-1} poprzez przyporządkowanie zadania k do zbioru $\hat{\mathbf{E}}$ lub do zbioru **WI**, lub do zbioru $\hat{\mathbf{T}}$.

Będziemy mówić, że harmonogram jest w stanie (k, a, b) jeżeli:

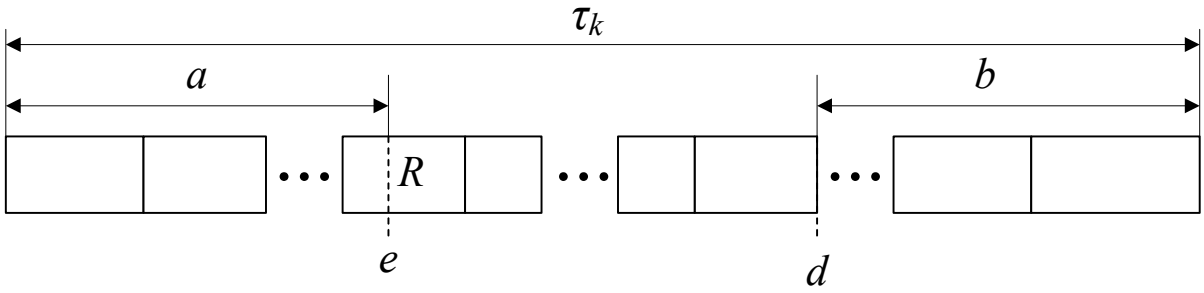
- zawiera zadania $\{1, \ldots, k\} \cup \{R\}$,
- różnica pomiędzy początkiem przedziału $[e, d]$ a momentem rozpoczęcia wykonywania pierwszego zadania jest równa a oraz
- różnica pomiędzy zakończeniem wykonywania ostatniego zadania a końcem przedziału $[e, d]$ jest równa b.

Parametry k, a i b będziemy nazywać zmiennymi stanu. Przykładowy harmonogram w stanie (k, a, b) został przedstawiony na Rysunku 7.4.

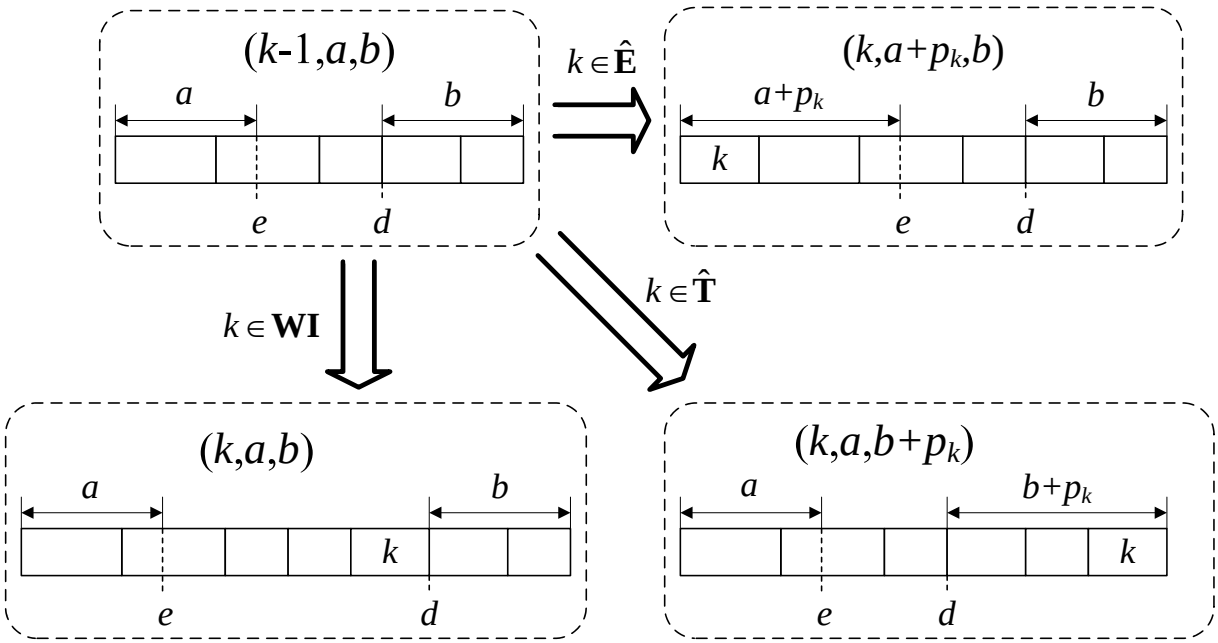
Niech $F_k(a, b)$ oznacza minimalną wartość kryterium dla harmonogramów w stanie (k, a, b). Łatwo zauważyć, że optymalna wartość funkcji celu wynosi:

$$\min\{F_n(a, b) : a = 0, \ldots, \tau_n;\ b = 0, \ldots, \tau_n; D_{min} \leqslant \tau_n - a - b \leqslant D_{max}\}.$$

Załóżmy, że mamy harmonogram cząstkowy znajdujący się w stanie $(k-1, a, b)$. Jeżeli $k = R$, to zadanie k jest już uszeregowane. Uzyskujemy zatem harmonogram w stanie (k, a, b), natomiast $F_k(a, b) = F_{k-1}(a, b)$. W przeciwnym przypadku:



Rysunek 7.4: Przykładowy harmonogram w stanie (k, a, b) z zaznaczonym zadaniem R.



Rysunek 7.5: Przykład generowania harmonogramów z harmonogramu w stanie $(k-1, a, b)$.

- Jeżeli zadanie k przyporządkujemy do zbioru $\hat{\mathbf{E}}$, to uzyskamy harmonogram w stanie $(k, a + p_k, b)$. Z Własności 7.4 wynika, że w otrzymanym harmonogramie zadanie k kończy wykonywać się w momencie $e - a$ (patrz Rysunek 7.5). Mamy wówczas $F_k(a + p_k, b) = F_{k-1}(a, b) + \alpha_k a$.

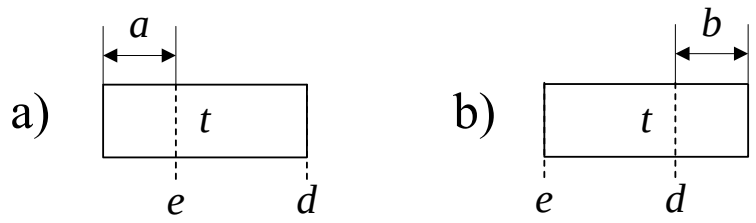
- Jeżeli zadanie k przyporządkujemy do zbioru $\mathbf{WI}$, to uzyskamy harmonogram w stanie (k, a, b), natomiast $F_k(a, b) = F_{k-1}(a, b) + \gamma p_k$ (patrz Rysunek 7.5).

- Jeżeli zadanie k przyporządkujemy do zbioru $\hat{\mathbf{T}}$, to uzyskamy harmonogram w stanie $(k, a, b + p_k)$. Z Własności 7.5 wynika, że w otrzymanym harmonogramie zadanie k kończy wykonywać się w momencie $d + b$ (patrz Rysunek 7.5). Mamy wówczas $F_k(a, b + p_k) = F_{k-1}(a, b) + \beta_k(b + p_k)$.

Z powyższych rozważań wynika, że jeżeli $k = R$, to harmonogram w stanie (k, a, b) uzyskujemy z harmonogramu $(k-1, a, b)$, natomiast $F_k(a, b) = F_{k-1}(a, b)$. W przeciwnym przypadku, harmonogram w stanie (k, a, b) możemy uzyskać:

- z harmonogramu w stanie $(k-1, a-p_k, b)$ poprzez przyporządkowanie zadania k do zbioru $\hat{\mathbf{E}}$. W tym przypadku mamy $F_k(a, b) = F_{k-1}(a-p_k, b) + \alpha_k(a-p_k)$.
- z harmonogramu w stanie $(k-1, a, b)$ poprzez przyporządkowanie zadania k do zbioru **WI**. W tym przypadku mamy $F_k(a, b) = F_{k-1}(a, b) + \gamma p_k$.
- z harmonogramu w stanie $(k-1, a, b-p_k)$ poprzez przyporządkowanie zadania k do zbioru $\hat{\mathbf{T}}$. W tym przypadku mamy $F_k(a, b) = F_{k-1}(a, b-p_k) + \beta_k b$.

$F_0(a, b)$ jest, zgodnie z definicją, równe karze za opóźnienie zadania R oraz za szerokość przedziału $[e, d]$. Zatem, $F_0(a, b) = \beta_R b + \gamma(p_R - a - b)$ (patrz Rysunek 7.6).



Rysunek 7.6: Dwa przykładowe harmonogramy σ_0.

Rozważmy teraz jakie wartości zmiennych stanu a i b opisują harmonogramy cząstkowe, które mogą być rozszerzone do harmonogramu optymalnego. Łatwo zauważyć, że zmienne stanu a i b przyjmują jedynie wartości nieujemne oraz musi być spełniona następująca nierówność $a + b \leqslant \tau_k$. W szczególności, dla $k = 0$, otrzymujemy $a + b \leqslant p_R$.

Z Własności 7.3 wynika, że w harmonogramie optymalnym nie istnieje zadanie, które rozpoczyna wykonywać się przed momentem e oraz kończy wykonywać się po momencie d. Wynika z tego, że $a = 0$ lub $b = 0$ w harmonogramie zerowym σ_0 (patrz Rysunek 7.6). Ponadto z dolnego ograniczenia na szerokość przedziału $[e, d]$ wynika, że należy rozpatrywać tylko takie zmienne stanu a i b, że $a \leqslant \tau_n - D_{min}$ oraz $b \leqslant \tau_n - D_{min}$.

Poniżej przedstawiony jest formalny opis algorytmu **A3.3** (R).

Algorytm A3.3 (R)

Krok 1. (Inicjalizacja). Ponumeruj zadania tak, że $p_1/\alpha_1 \leqslant \ldots \leqslant p_n/\alpha_n$. Podstaw:

$$F_0(a, b) := \begin{cases} \beta_R b + \gamma(p_R - a - b), & \text{jeżeli } (a = 0 \wedge 0 < b < p_R) \\ & \vee (b = 0 \wedge 0 < a \leqslant p_R); \\ +\infty, & \text{w przeciwnym przypadku.} \end{cases}$$

Następnie podstaw $k := 1$.

Step 2. (Rekursja). Dla każdego $a = 0, \ldots, \min\{\tau_k, \tau_n - D_{min}\}$ i $b = 0, \ldots, \min\{\tau_k, \tau_n - D_{min}\}$ wylicz:

- jeżeli $k = R$, to:
$$F_k(a, b) := F_{k-1}(a, b);$$
- w przeciwnym przypadku:
$$F_k(a, b) := \min \begin{cases} F_{k-1}(a - p_k, b) + \alpha_k(a - p_k), \\ F_{k-1}(a, b) + \gamma p_k, \\ F_{k-1}(a, b - p_k) + \beta_k b. \end{cases}$$

Jeżeli $k = n$, to idź do Kroku 3, w przeciwnym przypadku podstaw $k := k + 1$ i powtórz Krok 2.

Krok 3. (Rozwiązanie optymalne). Wyznacz optymalną wartość kryterium:

$$F^* := \min_{a,b} \{F_n(a, b) : D_{min} \leqslant \tau_n - a - b \leqslant D_{max}\}$$

oraz skonstruuj odpowiadające mu rozwiązanie optymalne metodą przeglądu wstecznego (ang. backtracking). Parametry przedziału $[e, d]$ mogą zostać wyliczone w sposób następujący: $e = a$; $d = \tau_n - b$.

Lemat 3. *Algorytm* **A3.3**(R) *rozwiązuje optymalnie problem* **P3.3**(R) *w czasie* $O(n(\sum p_j - D_{min})^2)$.

Dowód. Łatwo zauważyć, że złożoność obliczeniowa algorytmu wynika bezpośrednio z rozmiaru przestrzeni stanów. Tak więc skoro $k = 1, \ldots, n$; $a = 0, \ldots, \tau_n - D_{min}$ i $b = 0, \ldots, \tau_n - D_{min}$, to złożoność obliczeniowa algorytmu **A2.1** wynosi:

$$O\left(n\left(\tau_n - D_{min}\right)^2\right) = O\left(n\left(\sum_{j=1}^{n} p_j - D_{min}\right)^2\right).$$

Rozważmy parę harmonogramów cząstkowych w tym samym stanie. Harmonogram z mniejszą wartością kryterium będzie miał mniejszą wartość kryterium po rozszerzeniu go o nie uszeregowane zadania w ten sam sposób, w jaki rozszerzony zostanie drugi harmonogram będący w tym stanie. Z powyższego wynika, że do rozszerzenia o kolejne nie uszeregowane zadania należy wybrać spośród harmonogramów będących w tym samym stanie, harmonogram o najmniejszej wartości kryterium.

Optymalność algorytmu **A2.1** wynika bezpośrednio z powyższych rozważań oraz zasady optymalności ogólnej metody programowania dynamicznego [12]. □

Z wcześniejszych rozważań wynika, że problem **P3.3** może zostać rozwiązany przy pomocy algorytmu **A3.3**, który został przedstawiony poniżej.

Algorytm A3.3

Krok 1. Podstaw $R := 1$.

Krok 2. Rozwiąż problem **P3.3**(R) przy pomocy algorytmu **A3.3**(R). Uzyskane rozwiązanie optymalne oznaczmy przez σ^R. Podstaw $R := R+1$. Jeżeli $R \leqslant n$, to powtórz Krok 2.

Krok 3. Znajdź rozwiązanie σ^*, takie że $Z(\sigma^*) = \max\{Z(\sigma^R) : R = 1, \ldots, n\}$. Rozwiązanie σ^* jest rozwiązaniem optymalnym problemu **P3.3**.

Twierdzenie 4. *Algorytm* **A3.3** *rozwiązuje optymalnie problem* **P3.3** *w czasie* $O(n^2(\sum p_j - D_{min})^2)$.

Prawdziwość powyższego twierdzenia wynika bezpośrednio z Lematu 3.

Skoro $D_{min} \geqslant 0$, to złożoność obliczeniową algorytmu **A3.3** będziemy dla uproszczenia wyrażać również w skróconej formie: $O(n^2(\sum p_j)^2)$.

Algorytm programowania dynamicznego dla problemu P3.2

Poniżej zostanie przedstawiony algorytm **A3.2**, który rozwiązuje optymalnie problem **P3.2**.

Podzielmy problem **P3.2** na szereg podproblemów **P3.2**(**R**). W podproblemie **P3.2**(**R**) zbiór zadań **R** jest z góry zadany. Łatwo zauważyć, że w celu optymalnego rozwiązania problemu **P3.2** wystarczy rozwiązać optymalnie problemy **P3.2**(**R**) dla wszystkich możliwych zbiorów **R** a następnie wybrać rozwiązanie, dla którego wartość kryterium jest najmniejsza.

W związku z powyższym, w celu rozwiązania problemu **P3.2** wystarczy skonstruować algorytm optymalny dla problemu **P3.2**(**R**).

Zdefiniujmy:

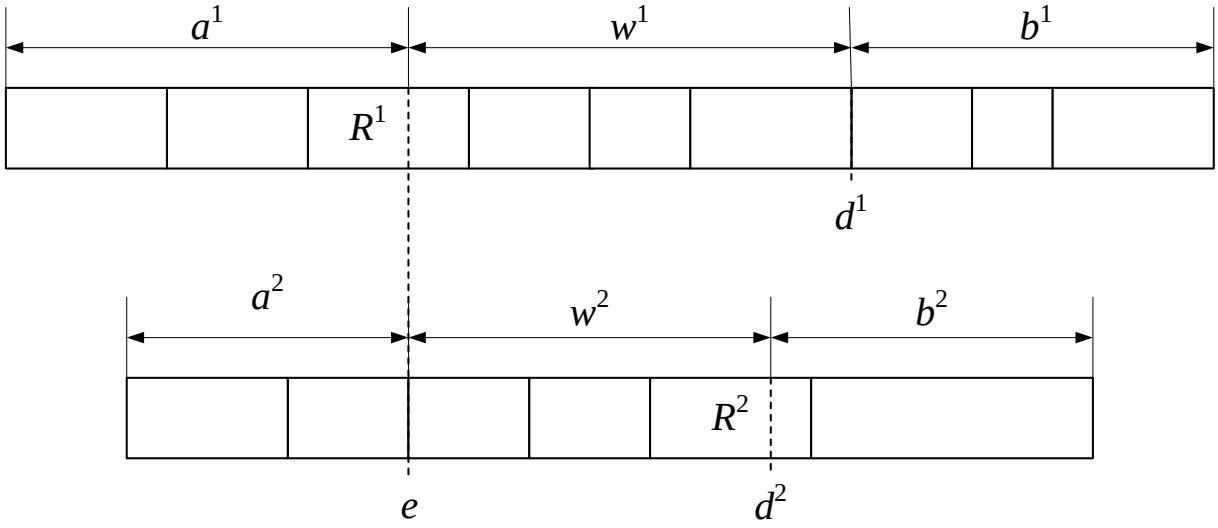
- $\tau_k \triangleq \sum_{j \in \{1,\ldots,k\} \cup \mathbf{R}} p_j$.

Algorytm programowania dynamicznego **A3.2** dla problemu **P3.2** został skonstruowany w oparciu o Własności 7.7-7.12. Algorytm ten jest modyfikacją algorytmu **A3.3**.

Łatwo zauważyć, że aby uzyskać harmonogram optymalny wystarczy określić, które zadania należą do zbioru $\hat{\mathbf{E}}^i$, które do zbioru $\hat{\mathbf{T}}^i$, a które do zbioru $\mathbf{WI}^i$ $(i = 1, \ldots, m)$. Tak więc w celu znalezienia rozwiązania należy dla każdego zadania podjąć decyzję czy należy ono do zbioru $\hat{\mathbf{E}}^i$, czy do zbioru $\mathbf{WI}^i$, czy do zbioru $\hat{\mathbf{T}}^i$ $(i = 1, \ldots, m)$.

Poniżej opisane jest działanie Algorytmu **A3.2**(**R**). Rozpoczynamy od harmonogramu zerowego σ_0, czyli takiego, który zawiera tylko zadania ze zbioru **R**. Następnie generujemy kolejne harmonogramy cząstkowe σ_k (harmonogramy zawierające zadania $\{1, 2, \ldots, k\} \cup \mathbf{R}$) z harmonogramów σ_{k-1} poprzez przyporządkowanie zadania k do zbioru $\hat{\mathbf{E}}^i$ lub do zbioru $\mathbf{WI}^i$, lub do zbioru $\hat{\mathbf{T}}^i$ $(i = 1, \ldots, m)$.

Niech $\tilde{S}^i$ oraz $\tilde{C}^i$ oznacza odpowiednio moment rozpoczęcia pierwszego zadania i moment zakończenia ostatniego zadania na procesorze i w pewnym harmonogramie cząstkowym. Będziemy mówić, że harmonogram ten jest w stanie $(k, \bar{a}, \bar{w}, \bar{b})$, gdzie $\bar{a} = (a^1, a^2, \ldots, a^m)$, $\bar{w} = (w^1, w^2, \ldots, w^m)$ i $\bar{b} = (b^1, b^2, \ldots, b^m)$, jeżeli zawiera zadania $\{1, 2, \ldots, k\} \cup \mathbf{R}$, natomiast $a^i = e - \tilde{S}^i$, $b^i = \tilde{C}^i - d$ i $w^i = \sum_{j \in \mathbf{J}^i} p_j - a - b$. Wartość zmiennej stanu w^i będzie nam wygodnie interpretować jako szerokość pożądanego przedziału zakończenia wykonywania zadań. Łatwo zauważyć, że przy takiej interpretacji musimy dopuścić różne szerokości tego przedziału na poszczególnych procesorach - oczywiście jest to dopuszczalne tylko w harmonogramach cząstkowych. W związku z tym, niech d^i oznacza koniec pożądanego przedziału zakończenia wykonywania zadań na procesorze i. Zdefiniowane zmienne stanu zostały zilustrowane na Rysunku 7.7.



Rysunek 7.7: Przykładowy harmonogram z zaznaczonymi zmiennymi stanu.

W celu uproszczenia dalszych rozważań, zdefiniujmy $\bar{x}^i$ jako wektor o rozmiarze m

z wartością 1 na pozycji i oraz wartością 0 na pozostałych pozycjach.

Niech $F_k(\bar{a}, \bar{w}, \bar{b})$ oznacza minimalną wartość sumy kar za nieterminowe wykonanie zadań spośród harmonogramów w stanie $(k, \bar{a}, \bar{w}, \bar{b})$. Zauważmy, że kara za szerokość przedziału $[e, d]$ jest jednakowa dla wszystkich harmonogramów w tym samym stanie i wynosi γw^1, przy założeniu, że $w^1 = w^2 = \ldots = w^m$.

Łatwo zauważyć, że optymalna wartość funkcji celu wynosi:

$$\min\{F_n(\bar{a}, \bar{w}\bar{b}) + \gamma w^1 : D_{min} \leqslant w^1 = w^2 = \ldots = w_m \leqslant D_{max}\}.$$

Załóżmy, że mamy harmonogram cząstkowy znajdujący się w stanie $(k-1, \bar{a}, \bar{w}, \bar{b})$. Jeżeli $k \in \mathbf{R}$, to zadania k jest już uszeregowane. Uzyskujemy zatem harmonogram w stanie $(k, \bar{a}, \bar{w}, \bar{b})$, natomiast $F_k(\bar{a}, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b})$. W przeciwnym przypadku:

- Jeżeli zadanie k przyporządkujemy do zbioru $\hat{\mathbf{E}}^i$, to uzyskamy harmonogram w stanie $(k, \bar{a} + \bar{x}^i p_k, \bar{w}, \bar{b})$. Z Własności 7.10 wynika, że w otrzymanym harmonogramie zadanie k kończy wykonywać się w momencie $e - a^i$ (patrz Rysunek 7.7). Mamy wówczas $F_k(\bar{a} + \bar{x}^i p_k, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b}) + \alpha_k a^i$.

- Jeżeli zadanie k przyporządkujemy do zbioru $\mathbf{WI}^i$, to uzyskamy harmonogram w stanie $(k, \bar{a}, \bar{w} + \bar{x}^i p_k, \bar{b})$, natomiast $F_k(\bar{a}, \bar{w} + \bar{x}^i p_k, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b})$.

- Jeżeli zadanie k przyporządkujemy do zbioru $\hat{\mathbf{T}}^i$, to uzyskamy harmonogram w stanie $(k, \bar{a}, \bar{w}, \bar{b} + \bar{x}^i p_k)$. Z Własności 7.11 wynika, że w otrzymanym harmonogramie zadanie k kończy wykonywać się w momencie $d + b^i$ (patrz Rysunek 7.7). Mamy wówczas $F_k(\bar{a}, \bar{w}, \bar{b} + \bar{x}^i p_k) = F_{k-1}(\bar{a}, \bar{w}, \bar{b}) + \beta_k(b^i + p_k)$.

Z powyższych rozważań wynika, że jeżeli $k \in \mathbf{R}$, to harmonogram w stanie $(k, \bar{a}, \bar{w}, \bar{b})$ uzyskujemy z harmonogramu $(k-1, \bar{a}, \bar{w}, \bar{b})$, natomiast $F_k(\bar{a}, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b})$. W przeciwnym przypadku harmonogram w stanie $(k, \bar{a}, \bar{w}, \bar{b})$ możemy uzyskać:

- z harmonogramu w stanie $(k-1, \bar{a} - p_k \bar{x}^i, \bar{w}, \bar{b})$ poprzez przyporządkowanie zadania k do zbioru $\hat{\mathbf{E}}^i$ $(i = 1, \ldots, m)$. W tym przypadku mamy $F_k(\bar{a}, \bar{w}, \bar{b}) = F_{k-1}(\bar{a} - p_k \bar{x}^i, \bar{w}, \bar{b}) + \alpha_k a^i$.

- z harmonogramu w stanie $(k-1, \bar{a}, \bar{w} - p_k \bar{x}^i, \bar{b})$ poprzez przyporządkowanie zadania k do zbioru $\mathbf{WI}^i$ $(i = 1, \ldots, m)$. W tym przypadku mamy $F_k(\bar{a}, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w} - p_k \bar{x}^i, \bar{b})$.

- z harmonogramu w stanie $(k-1, \bar{a}, \bar{w}, \bar{b} - p_k \bar{x}^i)$ poprzez przyporządkowanie zadania k do zbioru $\hat{\mathbf{T}}^i$ $(i = 1, \ldots, m)$. W tym przypadku mamy $F_k(\bar{a}, \bar{w}, \bar{b}) = F_{k-1}(\bar{a}, \bar{w}, \bar{b} - p_k \bar{x}^i) + \beta_k b^i$.

$F_0(\bar{a}, \bar{w}, \bar{b})$ jest, zgodnie z definicją, równe sumie kar za nieterminowe wykonanie zadań ze zbioru $\mathbf{R}$. W związku z tym $F_0(\bar{e}, \bar{w}, \bar{t}) = \sum_{i=1}^{m} \beta_{R^i} t_i$, ponieważ zadania te nie ponoszą kary za przedwczesne wykonanie.

Rozważmy teraz jakie wartości zmiennych stanu opisują harmonogramy cząstkowe, które mogą być rozszerzone do harmonogramu optymalnego. Łatwo zauważyć, że zmienne stanu a^i, w^i i b^i $(i = 1, \ldots, m)$ przyjmują jedynie wartości nieujemne. Oczywiste jest, że $a^i + b^i + w^i = \sum_{j \in \mathbf{J}^i} p_j$. W konsekwencji, dla harmonogramu σ_0 mamy $a^i + b^i + w^i = p_{R^i}$ $(i = 1, \ldots, m)$. Z Własności 7.9 wynika, że w harmonogramie optymalnym nie istnieje zadanie, które rozpoczyna wykonywać się przed momentem e a kończy wykonywać się po momencie d. Wynika z tego, że $a = 0$ lub $b = 0$ w harmonogramie σ_0.

Skoro rozważamy tylko całkowite harmonogramy, dla których $w^1 = w^2 = \ldots = w^m$, to wartość zmiennej stanu w^i nie może być większa niż $\min\{\frac{\tau_n}{m}, D_{max}\}$. Zauważmy, że w harmonogramie optymalnym nierówność $\min_i a^i \geqslant \max_i a^i + p_n$ musi być zawsze spełniona. W przeciwnym przypadku przesunięcie zadania, które zaczyna wykonywać się w momencie $\max_i a^i$ na inny procesor spowodowałoby zmniejszenie wartości kryterium. Ponadto, skoro minimalna szerokość przedziału $[e, d]$ wynosi D_{min}, to musi być spełniona następująca nierówność $\sum a^i \leqslant \tau_n - mD_{min}$. Wynika z tego, że wartość zmiennej stanu a^i nie może być większa niż $\frac{\tau_n}{m} - D_{min} + p_n$. Podobnie wartość zmiennej stanu b^i nie może być większa niż $\frac{\tau_n}{m} - D_{min} + p_n$. Niech zatem $\hat{w}_k = \min\{\tau_k, \left\lfloor \frac{\tau_n}{m} \right\rfloor, D_{max}\}$, $\hat{a}_k = \min\{\tau_k + p_n, \left\lfloor \frac{\tau_n}{m} \right\rfloor - D_{min} + p_n\}$, $\hat{b}_k = \min\{\tau_k - \hat{a}_k, \left\lfloor \frac{\tau_n}{m} \right\rfloor - D_{min} + p_n\}$ oznaczają odpowiednio maksymalne akceptowalne wartości zmiennych stanu w^i, a^i oraz b^i w iteracji k algorytmu.

Poniżej zaprezentowany jest szczegółowy opis algorytmu **A3.2** (**R**).

Algorytm A3.2 (R)

Krok 1. (Inicjalizacja). Ponumeruj zadania tak, że $p_1/\alpha_1 \leqslant \ldots \leqslant p_n/\alpha_n$. Podstaw:

$$F_0(\bar{a}, \bar{w}, \bar{b}) := \begin{cases} \sum_{i=1}^{m} \beta_{R^i} b^i, & \text{jeżeli dla każdego } i = 1, \ldots, m \\ & a^i + w^i + b^i = p_{R^i} \wedge (a^i = 0 \vee b^i = 0) \\ +\infty, & \text{w przeciwnym przypadku.} \end{cases}$$

Następnie podstaw $k := 1$.

Step 2. (Rekursja). Dla każdego $a^i = 0, \ldots, \hat{a}_k$; $w^i = 0, \ldots, \hat{w}_k$ i $b^i = 0, \ldots, \hat{b}_k$ $(i = 1, \ldots, m)$ wylicz:

- jeżeli $k \in \mathbf{R}$, to:

$$F_k(\bar{a}, \bar{w}, \bar{b}) := F_{k-1}(\bar{a}, \bar{w}, \bar{b});$$

- w przeciwnym przypadku:

$$F_k(\bar{a},\bar{w},\bar{b}) := \min_{i=1,\ldots,m} \begin{cases} F_{k-1}(\bar{a}-p_k\cdot\bar{x}^i,\bar{w},\bar{b})+\alpha_k(a^i-p_j), \\ F_{k-1}(\bar{a},\bar{w}-p_k\cdot\bar{x}^i,\bar{b}), \\ F_{k-1}(\bar{a},\bar{w},\bar{t}-p_k\cdot\bar{x}^i)+\beta_k b^i. \end{cases}$$

Jeżeli $k=n$, to idź do Kroku 3, w przeciwnym przypadku podstaw $k := k+1$ i powtórz Krok 2.

Krok 3. (Rozwiązanie optymalne). Wylicz optymalną wartość kryterium:

$$F^* := \min_{\bar{a},\bar{b},\bar{w}} \left\{ F_n(\bar{a},\bar{w},\bar{b}) + \gamma w^1 : D_{min} \leqslant w^1 = w^2 = \ldots = w^m \leqslant D_{max} \right\}$$

oraz skonstruuj odpowiadające rozwiązanie optymalne metodą przeglądu wstecznego (ang. backtracking). Parametry przedziału $[e,d]$ mogą zostać wyliczone w sposób następujący: $e = \max_i a^i$; $d = e + w^1$.

Lemat 4. *Algorytm* **A3.2** (**R**) *rozwiązuje optymalnie problem* **P3.2** (**R**) *w czasie* $O\left(n\left(\frac{\tau_n}{m}-D_{min}+p_n\right)^{2m-1}\left(\min\left\{\frac{\tau_n}{m},D_{max}\right\}\right)^m\right)$.

Dowód. Łatwo zauważyć, że złożoność obliczeniowa algorytmu wynika bezpośrednio z rozmiaru przestrzeni stanów. Tak więc skoro $k=1,\ldots,n$; $a^i=0,1,\ldots,\left\lfloor\frac{\tau_n}{m}\right\rfloor-D_{min}+p_n$; $b^i=0,1,\ldots,\left\lfloor\frac{\tau_n}{m}\right\rfloor-D_{min}+p_n$; $w^i=0,1,\ldots,\min\left\{\frac{\tau_n}{m},D_{max}\right\}$ (dla każdego $i=1,\ldots,m$), to złożoność obliczeniowa algorytmu **A3.2** (**R**) wynosi:

$$O\left(n\left(\frac{\tau_n}{m}-D_{min}+p_n\right)^{2m}\left(\min\left\{\frac{\tau_n}{m},D_{max}\right\}\right)^m\right).$$

Skoro wyrażenie $\sum_{i=1}^m(a^i+b^i+w^i)=\sum_{j=1}^k p_j$ jest prawdziwe dla każdej iteracji algorytmu $k=1,\ldots,n$, to możemy wyeliminować jedną ze zmiennych stanu, np. b^m. W efekcie zmniejszymy złożoność obliczeniową algorytmu do następującej wartości:

$$O\left(n\left(\frac{\tau_n}{m}-D_{min}+p_n\right)^{2m-1}\left(\min\left\{\frac{\tau_n}{m},D_{max}\right\}\right)^m\right).$$

Rozważmy parę harmonogramów cząstkowych w tym samym stanie. Przypomnijmy, że kara za szerokość przedziału $[e,d]$ jest identyczna dla wszystkich harmonogramów będących w tym samym stanie. Harmonogram z mniejszą wartością całkowitej kary za nieterminowość wykonania zadań będzie miał mniejszą wartość całkowitej kary za nieterminowość wykonania zadań po rozszerzeniu go o nie uszeregowane zadania w ten sam sposób, w jaki rozszerzony zostanie drugi harmonogram będący w tym stanie. Z powyższego wynika,

że do rozszerzenia o kolejne nie uszeregowane zadania należy wybrać spośród harmonogramów będących w tym samym stanie, harmonogram o najmniejszej wartości całkowitej kary za nieterminowość wykonania zadań.

Optymalność algorytmu **A2** wynika bezpośrednio z powyższych rozważań oraz zasady optymalności ogólnej metody programowania dynamicznego [12]. □

Poniżej przedstawiony będzie opis algorytmu, który rozwiązuje optymalnie problem **P3.2**.

Algorytm A3.2

Krok 1. Wygeneruj zbiór wszystkich możliwych zbiorów $\mathbf{R}$. Oznaczmy wygenerowany zbiór przez $\mathbf{\Omega}$.

Krok 2. Rozwiąż problem **P3.2**($\mathbf{R}$) przy pomocy algorytmu **A3.2**($\mathbf{R}$) dla każdego $\mathbf{R} \in \mathbf{\Omega}$. Uzyskane rozwiązanie optymalne oznaczmy przez $\sigma^{\mathbf{R}}$.

Krok 3. Znajdź rozwiązanie σ^*, takie że $Z(\sigma^*) = \max\{Z(\sigma^{\mathbf{R}}) : \mathbf{R} \in \mathbf{\Omega}\}$. Rozwiązanie σ^* jest rozwiązaniem optymalnym problemu **P3.2**.

Twierdzenie 5. *Algorytm* **A3.2** *rozwiązuje optymalnie problem* **P3.2** *w czasie* $O\left(n^{m+1}\left(\frac{\tau_n}{m} - D_{min} + p_n\right)^{2m-1}\left(\min\left\{\frac{\tau_n}{m}, D_{max}\right\}\right)^m\right)$.

Skoro ilość różnych zbiorów $\mathbf{R}$ nie przekracza n^m, to z Lematu 4 wynika poprawność powyższego dowodu.

Skoro $D_{min} \geqslant 0$, $\tau_n = \sum p_j$ oraz $\min\left\{\frac{\tau_n}{m}, D_{max}\right\} \leqslant \frac{\tau_n}{m}$, to złożoność obliczeniową algorytmu **A3.2** będziemy dla uproszczenia wyrażać również w skróconej formie:

$$O\left(n^{m+1}\left(\frac{\sum p_j}{m} + p_{max}\right)^{3m-1}\right).$$

Uwaga 2. *Algorytm* **A3.2** *może być w łatwy sposób przekształcony tak, aby rozwiązywał optymalnie rozszerzoną wersję problemu* **P3.2** *z jednorodnymi procesorami równoległymi.*

Wyjaśnienie: W przypadku procesorów jednorodnych $p_{jl} = p_j/v_l$ oznacza czas wykonywania zadania j na procesorze l, gdzie v_l oznacza prędkość procesora l, $j = 1, \ldots, n$, $l = 1, \ldots, m$. Gdy zadania są przydzielone do procesorów, to czasy wykonywania poszczególnych zadań są ustalone. Łatwo zatem zauważyć, że Własności 7.7 - 7.12 są również spełnione dla problemu z procesorami jednorodnymi. Wystarczy zatem dokonać kilku drobnych zmian w algorytmie **A3.2** w celu zaadaptowania go do rozwiązania tego problemu.

7.5 W pełni wielomianowy schemat aproksymacyjny

W niniejszym podrozdziale zostanie zaprezentowany w pełni wielomianowy schemat aproksymacyjny dla szczególnego przypadku problemu **P3.1**, w którym dla każdych dwóch zadań j i k prawdziwa jest zależność

$$\frac{p_j}{\alpha_j} < \frac{p_k}{\alpha_k} \Rightarrow \frac{p_j}{\beta_j} \leqslant \frac{p_k}{\beta_k}$$

a szerokość przedziału $[e, d]$ nie jest ograniczona ani z dołu ani z góry. Oznaczmy ten problem jako **P3.4**. W notacji trójpolowej, może być zdefiniowany w sposób następujący:

$$1|\langle e, d\rangle; \frac{p_j}{\alpha_j} \uparrow \frac{p_j}{\beta_j} \uparrow | \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d - e).$$

Najpierw zostanie wykazana własność problemu, która będzie wykorzystana przy konstrukcji schematu aproksymacyjnego.

Własność 7.13. *Istnieje rozwiązanie optymalne problemu* **P3.4***, w którym pewne zadanie kończy wykonywać się w momencie e oraz pewne zadanie kończy wykonywać się w momencie d.*

Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Przyjmijmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. żadne zadnie nie kończy wykonywać się w momencie e lub żadne zadanie nie kończy wykonywać się w momencie d.

Zgodnie z Własnością 7.3, w rozwiązaniu optymalnym pewne zadanie kończy wykonywać się w momencie e lub w momencie d. Musimy zatem rozważyć dwa możliwe przypadki.

Przypadek 1. Załóżmy, że w harmonogramie σ pewne zadanie kończy wykonywać się w momencie d.

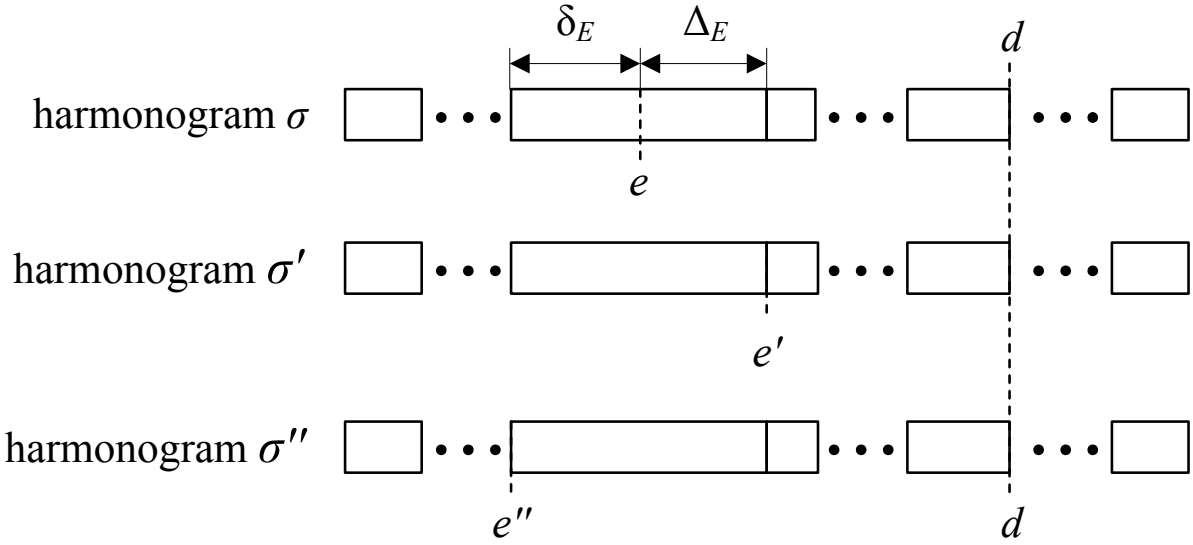
Z powyższego założenia wynika, że w harmonogramie σ żadne zadanie nie kończy wykonywać się w momencie e, czyli $\Delta_E(\sigma) > 0$. W niniejszym dowodzie zakładamy, że oznaczenia $\hat{\mathbf{E}}$, Δ_E i δ_E odnoszą się do harmonogramu σ.

Wartość funkcji celu dla harmonogramu σ wynosi:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d - e) =$$
$$Z_0 + \sum_{j \in \mathbf{J}} \alpha_j E_j + \gamma(d - e) = Z_0 + \sum_{j \in \mathbf{J}} \alpha_j \max\{e - C_j, 0\} + \gamma(d - e) =$$
$$Z_0 + \sum_{j \in \hat{\mathbf{E}}} \alpha_j (e - C_j) + \gamma(d - e),$$

gdzie $Z_0 = \sum_{j \in \mathbf{J}} \beta_j T_j$.

Utwórzmy harmonogram $\sigma' = (\pi, e', d)$ z harmonogramu σ taki, że $e' = e + \Delta_E$. Łatwo zauważyć, że $T_j(\sigma) = T_j(\sigma')$. Ponadto $C_j(\sigma') < e$ tylko dla zadań $j \in \hat{\mathbf{E}}$. Tak więc wartość



Rysunek 7.8: Przykładowe harmonogramy σ, σ' i σ''.

funkcji celu dla harmonogramu σ' wynosi:

$$Z(\sigma') = \sum_{j\in\mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d - e') =$$
$$Z_0 + \sum_{j\in\mathbf{J}} \alpha_j E_j + \gamma(d - e') = Z_0 + \sum_{j\in\mathbf{J}} \alpha_j \max\{e' - C_j, 0\} + \gamma(d - e') =$$
$$Z_0 + \sum_{j\in\mathbf{J}} \alpha_j \max\{e + \Delta_E - C_j, 0\} + \gamma(d - e - \Delta_E) =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}} \alpha_j (e + \Delta_E - C_j) + \gamma(d - e - \Delta_E) =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}} \alpha_j (e - C_j) + \gamma(d - e) + \left(\sum_{j\in\hat{\mathbf{E}}} \alpha_j - \gamma\right) \Delta_E.$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma') = \left(\sum_{j\in\hat{\mathbf{E}}} \alpha_j - \gamma\right) \Delta_E.$$

Utwórzmy teraz harmonogram $\sigma'' = (\pi, e'', d)$ z harmonogramu σ taki, że $e'' = e - \delta_E$. Łatwo zauważyć, że $T_j(\sigma) = T_j(\sigma'')$. Ponadto $C_j(\sigma'') \leqslant e$ tylko dla zadań $j \in \hat{\mathbf{E}}$. Tak więc

wartość funkcji celu dla harmonogramu σ'' wynosi:

$$Z(\sigma'') = \sum_{j\in\mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d - e'') =$$
$$Z_0 + \sum_{j\in\mathbf{J}} \alpha_j E_j + \gamma(d - e'') = Z_0 + \sum_{j\in\mathbf{J}} \alpha_j \max\{e'' - C_j, 0\} + \gamma(d - e'') =$$
$$Z_0 + \sum_{j\in\mathbf{J}} \alpha_j \max\{e - \delta_E - C_j, 0\} + \gamma(d - e + \delta_E) =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}} \alpha_j (e - \delta_E - C_j) + \gamma(d - e + \delta_E) =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}} \alpha_j (e - C_j) + \gamma(d - e) + \left(\gamma - \sum_{j\in\hat{\mathbf{E}}} \alpha_j\right) \delta_E.$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma'') = \left(\gamma - \sum_{j\in\hat{\mathbf{E}}} \alpha_j\right) \delta_E.$$

Łatwo zauważyć, że:

- jeżeli $\sum_{j\in\hat{\mathbf{E}}} \alpha_j \geqslant \gamma$, to $Z(\sigma') \leqslant Z(\sigma)$;
- jeżeli $\sum_{j\in\hat{\mathbf{E}}} \alpha_j \leqslant \gamma$, to $Z(\sigma'') \leqslant Z(\sigma)$.

Przypadek 2. Załóżmy, że w harmonogramie σ pewne zadanie kończy wykonywać się w momencie e.

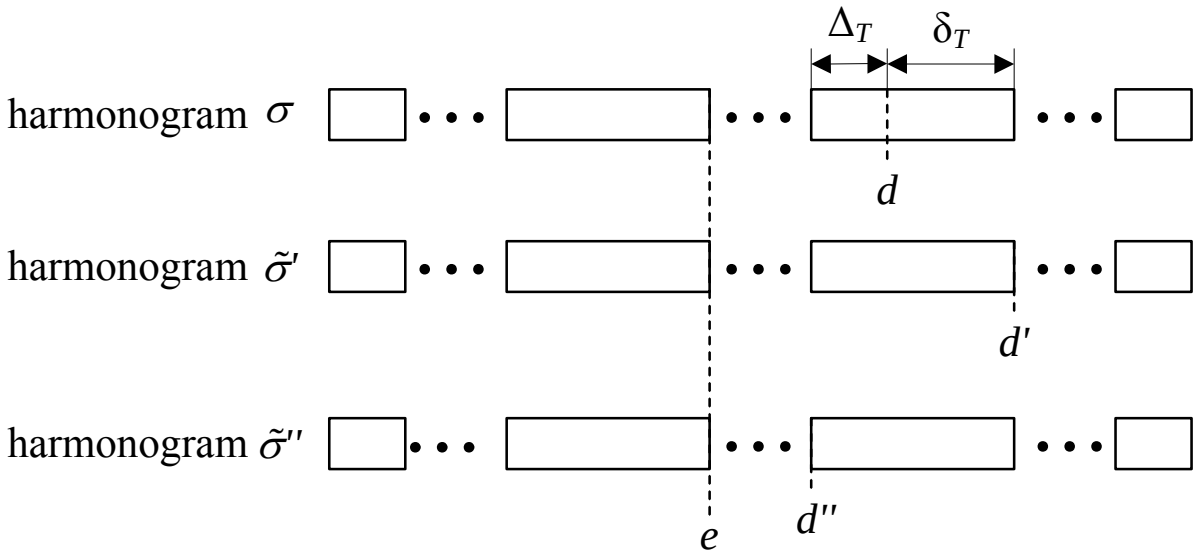
Z powyższego założenia wynika, że w harmonogramie σ żadne zadanie nie kończy wykonywać się w momencie d, czyli $\Delta_T(\sigma) > 0$. W niniejszym dowodzie zakładamy, że oznaczenia $\mathbf{T}$, Δ_T i δ_T będą odnosiły się do harmonogramu σ.

Wartość funkcji celu dla harmonogramu σ wynosi:

$$Z(\sigma) = \sum_{j\in\mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d - e) =$$
$$\tilde{Z}_0 + \sum_{j\in\mathbf{J}} \beta_j T_j + \gamma(d - e) = \tilde{Z}_0 + \sum_{j\in\mathbf{J}} \beta_j \max\{C_j - d, 0\} + \gamma(d - e) =$$
$$\tilde{Z}_0 + \sum_{j\in\hat{\mathbf{E}}} \beta_j (C_j - d) + \gamma(d - e),$$

gdzie $\tilde{Z}_0 = \sum_{j\in\mathbf{J}} \alpha_j E_j$.

Utwórzmy harmonogram $\tilde{\sigma}' = (\pi, e, d')$ z harmonogramu σ taki, że $d' = d + \delta_T$. Łatwo zauważyć, że $E_j(\sigma) = E_j(\tilde{\sigma}')$. Ponadto $C_j(\tilde{\sigma}') \geqslant d$ tylko dla zadań $j \in \mathbf{T}$. Tak więc



Rysunek 7.9: Przykładowe harmonogramy σ, $\tilde{\sigma}'$ i $\tilde{\sigma}''$.

wartość funkcji celu dla harmonogramu $\tilde{\sigma}'$ wynosi:

$$Z(\tilde{\sigma}') = \sum_{j\in\mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d' - e) =$$
$$\tilde{Z}_0 + \sum_{j\in\mathbf{J}} \beta_j T_j + \gamma(d' - e) = \tilde{Z}_0 + \sum_{j\in\mathbf{J}} \beta_j \max\{C_j - d', 0\} + \gamma(d' - e) =$$
$$\tilde{Z}_0 + \sum_{j\in\mathbf{J}} \beta_j \max\{C_j - d - \delta_T, 0\} + \gamma(d + \delta_T - e) =$$
$$\tilde{Z}_0 + \sum_{j\in\mathbf{T}} \beta_j (C_j - d - \delta_T,) + \gamma(d + \delta_T - e) =$$
$$\tilde{Z}_0 + \sum_{j\in\mathbf{T}} \beta_j (e - C_j) + \gamma(d - e) + \left(\gamma - \sum_{j\in\mathbf{T}} \beta_j\right) \delta_T.$$

Mamy zatem:

$$Z(\sigma) - Z(\tilde{\sigma}') = \left(\gamma - \sum_{j\in\mathbf{T}} \beta_j\right) \delta_T.$$

Utwórzmy teraz harmonogram $\tilde{\sigma}'' = (\pi, e, d'')$ z harmonogramu σ taki, że $d'' = d - \Delta_T$. Łatwo zauważyć, że $T_j(\sigma) = T_j(\tilde{\sigma}'')$. Ponadto $C_j(\tilde{\sigma}'') > d$ tylko dla zadań $j \in \mathbf{T}$. Tak więc

wartość funkcji celu dla harmonogramu $\tilde{\sigma}''$ wynosi:

$$Z(\tilde{\sigma}'') = \sum_{j\in\mathbf{J}} (\alpha_j E_j + \beta_j T_j) + \gamma(d'' - e) =$$
$$\tilde{Z}_0 + \sum_{j\in\mathbf{J}} \beta_j T_j + \gamma(d'' - e) = \tilde{Z}_0 + \sum_{j\in\mathbf{J}} \beta_j \max\{C_j - d'', 0\} + \gamma(d'' - e) =$$
$$\tilde{Z}_0 + \sum_{j\in\mathbf{J}} \beta_j \max\{C_j - d + \Delta_T, 0\} + \gamma(d - \Delta_T - e) =$$
$$\tilde{Z}_0 + \sum_{j\in\mathbf{T}} \beta_j (C_j - d + \Delta_T) + \gamma(d - e - \Delta_T) =$$
$$\tilde{Z}_0 + \sum_{j\in\mathbf{T}} \beta_j (C_j - d) + \gamma(d - e) + \left(\sum_{j\in\hat{\mathbf{E}}} \beta_j - \gamma\right) \Delta_T.$$

Mamy zatem:

$$Z(\sigma) - Z(\tilde{\sigma}'') = \left(\sum_{j\in\hat{\mathbf{E}}} \beta_j - \gamma\right) \Delta_T.$$

Łatwo zauważyć, że:

- jeżeli $\gamma \geqslant \sum_{j\in\mathbf{T}} \beta_j$, to $Z(\tilde{\sigma}') \leqslant Z(\sigma)$;
- jeżeli $\gamma \leqslant \sum_{j\in\mathbf{T}} \beta_j$, to $Z(\tilde{\sigma}'') \leqslant Z(\sigma)$.

Z powyższego wynika, że możemy zawsze uzyskać harmonogram spełniający dowodzoną własność bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P3.4**, które spełniają Własność 7.13.

Łatwo zauważyć, że jeżeli $\gamma > \sum_{j\in\mathbf{J}} \max\left\{\alpha_j, \beta_j\right\}$, to $e = d$ w rozwiązaniu optymalnym problemu **P3.4**. Jeżeli zatem $\alpha_j = \beta_j$ dla każdego zadania $j = 1, \ldots, n$ oraz $\gamma > \sum_{j=1}^{n} \alpha_j$, to problem **P3.4** redukuje się do NP-trudnego problemu $1|d_j = d|\sum \alpha_j (E_j + T_j)$ [38]. Z powyższego wynika prawdziwość Wniosku 5.

Wniosek 5. *Problem* **P3.4** *jest NP-trudny.*

Przytoczmy najpierw pewne definicje związane ze schematami aproksymacyjnymi.

Niech σ^* oznacza rozwiązanie optymalne, a σ^H rozwiązanie uzyskane przy pomocy pewnego algorytmu aproksymacyjnego H_ε. Będziemy mówili, że H_ε jest algorytmem $(1+\varepsilon)$-aproksymacyjnym, jeżeli $Z(\sigma^H) \leqslant (1+\varepsilon)Z(\sigma^*)$ dla wszystkich instancji rozpatrywanego problemu.

Będziemy mówili, że rodzina algorytmów $\{H_\varepsilon\}$ jest *w pełni wielomianowym schematem aproksymacyjnym*, jeśli dla dowolnego $\varepsilon > 0$, H_ε jest $(1+\varepsilon)$-aproksymacyjnym algorytmem o złożoności wielomianowej zarówno od długości instancji problemu, jak i wyrażenia $1/\varepsilon$.

Poniżej zostanie przedstawiony w pełni wielomianowy schemat aproksymacyjny dla problemu **P3.4**, który został skonstruowany na bazie schematu aproksymacyjnego dla problemu $1|d_j = d|\sum \alpha_j (E_j + T_j)$ zaproponowanego przez *Kovalyov i Kubiak* [74].

Załóżmy, że zadania są ponumerowane tak, że $p_1/\alpha_1 \leqslant \ldots \leqslant p_n/\alpha_n$. Niech:

$$h(a,b) \triangleq \begin{cases} 1, & \text{jeżeli } a = b; \\ 0, & \text{w przeciwnym przypadku.} \end{cases}$$

Dla każdego harmonogramu spełniającego Własności 7.1- 7.6 i Własność 7.13 definiujemy:

$$x_j \triangleq \begin{cases} 0, & \text{jeżeli } C_j \leqslant e; \\ 1, & \text{jeżeli } e < C_j \leqslant d; \\ 2, & \text{jeżeli } C_j > d. \end{cases}$$

Niech $\mathbf{X}$ oznacza zbiór wszystkich wektorów $x = (x_1, \ldots, x_n)$, gdzie $x_j \in \{0, 1, 2\}$. Zdefiniujmy:
$\mathbf{X}_j \triangleq \{x \in \mathbf{X} | x_i = 0, i = j+1, \ldots, n\}$,
$W_0(x) \triangleq P_0^E(x) \triangleq P_0^T(x) \triangleq 0$,
$P_j^E(x) \triangleq P_{j-1}^E(x) + p_j h(x_j, 0)$,
$P_j^T(x) \triangleq P_{j-1}^T(x) + p_j h(x_j, 2)$,
$W_j(x) \triangleq W_{j-1}(x) + \alpha_j P_{j-1}^E(x) h(x_j, 0) + \beta_j P_j^T(x) h(x_j, 2) + \gamma p_j h(x_j, 1)$,
$x \in \mathbf{X}_j$, $j = 1, \ldots, n$.

Pierwszych j elementów każdego wektora $x \in \mathbf{X}_j$ odpowiada uszeregowaniu zadań $1, \ldots, j$. $W_j(x)$ jest całkowitym kosztem dla tego harmonogramu. Natomiast $P_j^E(x)$ i $P_j^T(x)$ oznaczają odpowiednio sumy czasów wykonywania zadań ze zbiorów $\mathbf{E}$ i $\mathbf{T}$. Z powyższego wynika, że problem **P3.4** redukuje się do problemu minimalizacji wartości funkcji $W_n(x)$. Będziemy oznaczać wektor $x \in \mathbf{X}$, który minimalizuje funkcję $W_n(x)$ jako $x^* = (x_1^*, \ldots, x_n^*)$.

W iteracji j algorytm dzieli zbiór rozwiązań cząstkowych (harmonogramów zawierających j pierwszych zadań) na rozdzielne podzbiory przy użyciu procedury $Partition(\mathbf{A}, G, \delta)$, która została opisana w pracy [74], gdzie $\mathbf{A} \subseteq \mathbf{X}$, G jest nieujemną funkcją całkowitą na zbiorze $\mathbf{X}_j$ oraz $0 < \delta \leqslant 1$. Procedura ta dzieli zbiór $\mathbf{A}$ na k rozdzielnych podzbiorów $\mathbf{A}_1^G, \mathbf{A}_2^G, \ldots, \mathbf{A}_k^G$ tak, że $|G(x) - G(x')| \leqslant \delta \min\{G(x), G(x')\}$ dla każdego $x, x' \in \mathbf{A}_i^G$, $i = 1, \ldots, k$. W pracy [74] wykazano również, że dla $0 < \delta \leqslant 1$ prawdziwa jest następująca zależność:

$$k \leqslant \log G_{max}/\delta + 2, \tag{7.1}$$

gdzie $G_{max} = \max_{x \in \mathbf{A}} G(x)$.

Algorytm wylicza kolejno sekwencje zbiorów $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$, gdzie $\mathbf{Y}_j \subseteq \mathbf{X}_j$, $j = 1, \ldots, n$. Wektor x^0 (rozwiązanie dostarczane przez algorytm) jest następnie wybierany ze zbioru $\mathbf{Y}_n$. Rozpoczynamy od zbioru $\mathbf{Y}_0 = \{(0, \ldots, 0)\}$. W każdej iteracji j ($j = 1, \ldots, n$)

konstruowany jest pewien zbiór poprzez generowanie trzech wektorów dla każdego wektora $x \in \mathbf{Y}_{j-1}$ dodając 0, 1 lub 2 na pozycję j wektora x. Zbiór ten jest następnie dzielony przy użyciu procedury *Partition* na rozdzielne podzbiory w taki sposób, że dla każdych dwóch wektorów x i x' z tego samego zbioru spełnione są następujące zależności:

$$|W_j(x) - W_j(x')| \leqslant \varepsilon \min\{W_j(x), W_j(x')\},$$

$$|P_j^E(x) - P_j^E(x')| \leqslant \frac{\varepsilon}{2n} \min\{P_j^E(x), P_j^E(x')\},$$

$$|P_j^T(x) - P_j^T(x')| \leqslant \frac{\varepsilon}{2n} \min\{P_j^T(x), P_j^T(x')\}.$$

Z każdego podzbioru wybierane są dwa rozwiązania w celu dołączenia ich do zbioru $\mathbf{Y}_j$; jedno z minimalną wartością funkcji $W_j(x)$ a drugie z maksymalną wartością funkcji $W_j(x)$. Wszystkie pozostałe rozwiązania znajdujące się w podzbiorach są odrzucane.

Formalny opis schematu aproksymacyjnego jest przedstawiony poniżej.

Algorytm A3.4$_\varepsilon$

Krok 1. (Inicjalizacja). Ponumeruj zadania tak, że $p_1/\alpha_1 \leqslant \ldots \leqslant p_n/\alpha_n$. Podstaw $\mathbf{Y}_0 := \{(0, \ldots, 0)\}$ oraz $j := 1$.

Krok 2. (Generacja zbiorów $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$). Dla każdego wektora $x \in \mathbf{Y}_{j-1}$ wygeneruj trzy wektory poprzez dodanie 0, 1 lub 2 na pozycji j wektora x. Oznaczmy wygenerowany zbiór wektorów jako $\mathbf{Y}'_j$. Dla każdego wektora $x \in \mathbf{Y}'_j$ wylicz $P_j^E(x)$, $P_j^T(x)$ i $W_j(x)$.

Jeżeli $j = n$, to podstaw $\mathbf{Y}_n := \mathbf{Y}'_n$ i idź do Kroku 3.

W przeciwnym przypadku wykonaj następujące obliczenia:

(Podział zbioru $\mathbf{Y}'_j$ ze względu na W_j) Wywołaj procedurę *Partition*$(\mathbf{Y}'_j, W_j, \varepsilon)$ aby podzielić zbiór $\mathbf{Y}'_j$ na rozdzielne podzbiory $\mathbf{Y}_1^W, \ldots, \mathbf{Y}_{k_W}^W$.

(Podział zbioru $\mathbf{Y}'_j$ ze względu na P_j^T) Wywołaj procedurę *Partition*$(\mathbf{Y}'_j, P_j^T, \varepsilon/(2n))$ aby podzielić zbiór $\mathbf{Y}'_j$ na rozdzielne podzbiory $\mathbf{Y}_1^T, \ldots, \mathbf{Y}_{k_T}^T$.

(Podział zbioru $\mathbf{Y}'_j$ ze względu na P_j^E) Wywołaj procedurę *Partition*$(\mathbf{Y}'_j, P_j^E, \varepsilon/(2n))$ aby podzielić zbiór $\mathbf{Y}'_j$ na rozdzielne podzbiory $\mathbf{Y}_1^E, \ldots, \mathbf{Y}_{k_E}^E$.

Dla wszystkich trójek (l_1, l_2, l_3) takich, że $\mathbf{Y}_{l_1}^W \cap \mathbf{Y}_{l_2}^E \cap \mathbf{Y}_{l_3}^T \neq \emptyset$, $1 \leqslant l_1 \leqslant k_W$, $1 \leqslant l_2 \leqslant k_E$ i $1 \leqslant l_3 \leqslant k_T$, podstaw $\mathbf{Z}_{(l_1,l_2,l_3),j} := \mathbf{Y}_{l_1}^W \cap \mathbf{Y}_{l_2}^E \cap \mathbf{Y}_{l_3}^T$.

W każdym podzbiorze $\mathbf{Z}_{q,j}$ wybierz wektory $x^{(q,min)}$ i $x^{(q,max)}$ takie, że:

$$W_j(x^{(q,min)}) = \min\{W_j(x) | x \in \mathbf{Z}_{q,j}\},$$

$$W_j(x^{(q,max)}) = \max\{W_j(x) | x \in \mathbf{Z}_{q,j}\}.$$

Podstaw $\mathbf{Y}_j := \{x^{(q,min)}, x^{(q,max)} | q$ jest trójką (l_1, l_2, l_3) taką, że $\mathbf{Y}_{l_1}^W \cap \mathbf{Y}_{l_2}^E \cap \mathbf{Y}_{l_3}^T \neq \emptyset$, $1 \leqslant l_1 \leqslant k_W$, $1 \leqslant l_2 \leqslant k_E$ i $1 \leqslant l_3 \leqslant k_T\}$ i $j := j + 1$. Powtórz Krok 2.

Krok 3. (Rozwiązanie). Wybierz wektor $x^0 \in \mathbf{Y}_n$ taki, że $W_n(x^0) = \min\{W_n(x) | x \in \mathbf{Y}_n\}$.

Lemat 5. *Liczba różnych trójek (l_1, l_2, l_3) takich, że $\mathbf{Y}_{l_1}^W \cap \mathbf{Y}_{l_2}^E \cap \mathbf{Y}_{l_3}^T \neq \emptyset$, $1 \leqslant l_1 \leqslant k_W$, $1 \leqslant l_2 \leqslant k_E$ oraz $1 \leqslant l_3 \leqslant k_T$ nie przekracza $k_W(k_E + k_T)$.*

Dowód. Dowód ten jest modyfikacją dowodu analogicznego lematu przedstawionego w pracy [74].

Wystarczy pokazać, że liczba różnych par (l_2, l_3) takich że $\mathbf{Y}_{l_2}^E \cap \mathbf{Y}_{l_3}^T \neq \emptyset$ nie przekracza $k_E + k_T$. Skojarzmy parę indeksów $(l_2(x), l_3(x))$ z każdym wektorem $x \in \mathbf{Y}_{l_2(x)}^E \cap \mathbf{Y}_{l_3(x)}^T$ i uporządkujmy wektory $x \in \mathbf{Y}_j'$ w niemalejącym porządku $P_j^E(x)$. Skoro $P_j^T(x) = \sum_{i=1}^{j} p_i - P_j^E(x)$, to te wektory są jednocześnie uporządkowane w nierosnącym porządku $P_j^T(x)$. Jeżeli zatem przeszukujemy wektory $x \in \mathbf{Y}_j'$ w niemalejącym porządku $P_j^E(x)$, to indeks $l_2(x)$ nigdy nie maleje a indeks $l_2(x)$ nigdy nie rośnie. Z powyższego wynika, że ilość różnych par (l_2, l_3) takich, że $\mathbf{Y}_{l_2}^E \cap \mathbf{Y}_{l_3}^T \neq \emptyset$ nie przekracza $k_E + k_T$. □

Lemat 6. *Dla każdych dwóch wektorów $x, x' \in \mathbf{Z}_{q,j}$ spełnione są następujące zależności:*

$$|W_j(x) - W_j(x')| \leqslant \varepsilon \min\{W_j(x), W_j(x')\};$$

$$|P_j^E(x) - P_j^E(x')| \leqslant \frac{\varepsilon}{2n} \min\{P_j^E(x), P_j^E(x')\};$$

$$|P_j^T(x) - P_j^T(x')| \leqslant \frac{\varepsilon}{2n} \min\{P_j^T(x), P_j^T(x')\}.$$

Prawdziwość powyższego lematu wynika bezpośrednio z (7.1).

Twierdzenie 6. *Algorytm $\mathbf{A3}.\mathbf{4}_\varepsilon$ znajduje rozwiązanie $x^0 \in \mathbf{X}$ takie, że $W_n(x^0) \leqslant (1 + \varepsilon) W_n(x^*)$.*

Dowód. Dowód ten jest modyfikacją dowodu analogicznego twierdzenia przedstawionego w pracy [74].

Załóżmy, że $x = (x_1^*, \ldots, x_j^*, 0, \ldots, 0) \in Z_{q,j} \subseteq \mathbf{Y}_j'$ dla pewnego j i q gdzie j pierwszych elementów wektora x jest takich samych jak pierwszych j elementów wektora x^*. Zgodnie z definicją algorytmu $\mathbf{A3}.\mathbf{4}_\varepsilon$, takie j zawsze istnieje, np. $j = 1$.

Algorytm $\mathbf{A3}.\mathbf{4}_\varepsilon$ może nie wybrać wektora x do dalszej konstrukcji. Z Lematu 6 wynika, że wektory $x^{(q,opt)}$, gdzie $opt \in \{min, max\}$ wybrane zamiast wektora x ze zbioru $\mathbf{Z}_{q,j}$ w j-tej iteracji spełniają następujące nierówności:

$$|W_j(x^*) - W_j(x^{(q,opt)})| \leqslant \varepsilon W_j(x^*), \tag{7.2}$$

$$|P_j^E(x^*) - P_j^E(x^{(q,opt)})| \leqslant \nu_1 P_j^E(x^*), \tag{7.3}$$

$$|P_j^T(x^*) - P_j^T(x^{(q,opt)})| \leqslant \nu_1 P_j^T(x^*), \tag{7.4}$$

gdzie $\nu_1 = \frac{\varepsilon}{2n} < \varepsilon$.

Podstawmy $\nu_{j+1} := \nu_j + (1+\nu_j)\frac{\varepsilon}{2n} = \frac{\varepsilon}{2n} + (1+\frac{\varepsilon}{2n})\nu_j$, $j = 1, \ldots, n-2$. Pokażemy, że powyższe trzy nierówności będą również spełnione, gdy zastąpimy j przez $j+1, \ldots, n-1$ a ν_1 zastąpimy odpowiednio przez $\nu_2, \ldots, \nu_{n-j}$. Wartość ν_j dla $j = 1, 2$ i 3 wynosi:

$$\nu_1 = \frac{\varepsilon}{2n};$$

$$\nu_2 = \frac{\varepsilon}{2n} + \left(1 + \frac{\varepsilon}{2n}\right)\frac{\varepsilon}{2n} = \frac{\varepsilon}{2n}\left[1 + \left(1 + \frac{\varepsilon}{2n}\right)\right];$$

$$\nu_3 = \frac{\varepsilon}{2n} + \left(1 + \frac{\varepsilon}{2n}\right)\frac{\varepsilon}{2n}\left[1 + \left(1 + \frac{\varepsilon}{2n}\right)\right] = \frac{\varepsilon}{2n}\left[1 + \left(1 + \frac{\varepsilon}{2n}\right) + \left(1 + \frac{\varepsilon}{2n}\right)^2\right].$$

Z powyższego można łatwo wywnioskować, że dla każdego $j = 1, \ldots, n-1$ mamy:

$$\nu_j = \frac{\varepsilon}{2n}\sum_{i=0}^{j}\left(1 + \frac{\varepsilon}{2n}\right)^i.$$

Wartość ν_j, $j = 2, \ldots, n-1$, może być oszacowana w sposób następujący:

$$\nu_j \leqslant \nu_{n-1} = \frac{\varepsilon}{2n}\sum_{i=0}^{n-1}\left(1 + \frac{\varepsilon}{2n}\right)^i = \sum_{i=0}^{n-1}\left(1 + \frac{\varepsilon}{2n}\right)^{i+1} - \sum_{i=0}^{n-1}\left(1 + \frac{\varepsilon}{2n}\right)^i =$$
$$\left(1 + \frac{\varepsilon}{2n}\right)^n - \left(1 + \frac{\varepsilon}{2n}\right)^0 = \left(1 + \frac{\varepsilon}{2n}\right)^n - 1 = \sum_{i=1}^{n}\begin{bmatrix} n \\ i \end{bmatrix}\left(\frac{\varepsilon}{2n}\right)^i.$$

Ponadto dla każdego $i = 1, \ldots, n$:

$$\sum_{i=1}^{n}\begin{bmatrix} n \\ i \end{bmatrix}\left(\frac{\varepsilon}{2n}\right)^i = \frac{n!}{i!(n-i)!}\left(\frac{\varepsilon}{2n}\right)^i = \frac{n(n-1)\cdots(n-i+1)}{i!n^i}\left(\frac{\varepsilon}{2}\right)^i \leqslant \frac{1}{i!}\left(\frac{\varepsilon}{2}\right)^i \leqslant \frac{1}{i!}\frac{\varepsilon}{2}.$$

Ostatnia z powyższych nierówności jest spełniona, ponieważ $\frac{\varepsilon}{2} \leqslant 1$. Z powyższego wynika następująca zależność:

$$\nu_j \leqslant \frac{\varepsilon}{2}\sum_{i=1}^{n}\frac{1}{i!} \leqslant \varepsilon$$

dla $j = 2, \ldots, n-1$.

Rozważmy wektory $\tilde{x} = \left(x_1^{(q,opt)}, \ldots, x_j^{(q,opt)}, x_{j+1}^*, 0, \ldots, 0\right)$, gdzie $opt \in \{min, max\}$, $\tilde{x} \in \mathbf{Y}'_{j+1}$, w których element x_{j+1}^* jest taki sam jak w wektorze x^*. Zgodnie z definicją P_{j+1}^E i P_{j+1}^T, mamy:

$$\left|P_{j+1}^E(x^*) - P_{j+1}^E(\tilde{x})\right| = \left|P_j^E(x^*) + p_{j+1}h(x_{j+1}^*, 0) - P_j^E(x^{(q,opt)}) - p_{j+1}h(x_{j+1}^*, 0)\right| =$$
$$\left|P_j^E(x^*) - P_j^E(x^{(q,opt)})\right| \leqslant \nu_1 P_j^E(x^*) \leqslant \nu_1 P_{j+1}^E(x^*);$$

$$\left|P_{j+1}^T(x^*) - P_{j+1}^T(\tilde{x})\right| = \left|P_j^T(x^*) + p_{j+1}h(x_{j+1}^*, 2) - P_j^T(x^{(q,opt)} - p_{j+1}h(x_{j+1}^*, 2))\right| =$$
$$\left|P_j^T(x^*) - P_j^T(x^{(q,opt)})\right| \leqslant \nu_1 P_j^T(x^*) \leqslant \nu_1 P_{j+1}^T(x^*). \tag{7.5}$$

W konsekwencji:

$$P_{j+1}^E(\tilde{x}) \leqslant (1+\nu_1)P_{j+1}^E(x^*);$$
$$P_{j+1}^T(\tilde{x}) \leqslant (1+\nu_1)P_{j+1}^T(x^*). \tag{7.6}$$

Ponadto, zgodnie z definicją W_{j+1}, mamy:

$$|W_{j+1}(x^*) - W_{j+1}(\tilde{x})| \leqslant |W_j(x^*) - W_j(x^{(q,opt)})| + \alpha_{j+1}h(x_{j+1}^*, 0)|P_j^E(x^*) - P_j^E(x^{(q,opt)})| +$$
$$\beta_{j+1}h(x_{j+1}^*, 2)|P_j^T(x^*) - P_j^T(x^{(q,opt)})| + \gamma p_{j+1}\left(h(x_{j+1}^*, 1) - h(x_{j+1}^*, 1)\right) \leqslant$$
$$\varepsilon W_j^* + \nu_1\left(\alpha_{j+1}h(x_{j+1}^*, 1)P_j^E(x^*) + \beta_{j+1}h(x_{j+1}^*, 0)P_j^T(x^*)\right) \leqslant \varepsilon W_{j+1}(x^*). \tag{7.7}$$

Ostatnia z powyższych nierówności jest spełniona, ponieważ $\nu_1 \leqslant \varepsilon$.

Załóżmy, że $\tilde{x} \in \mathbf{Z}_{l,j+1} \subseteq \mathbf{Y}'_{j+1}$ i algorytm $\mathbf{A3.4}_\varepsilon$ wybiera w iteracji $j+1$ wektory $x^{(l,min)}, x^{(l,max)} \in \mathbf{Z}_{l,j+1}$ zamiast wektora $\tilde{x}$ ze zbioru $\mathbf{Z}_{l,j+1}$. Wówczas, zgodnie z definicją $x^{(l,min)}$ i $x^{(l,max)}$, mamy:

$$W_{j+1}\left(x^{(l,min)}\right) \leqslant W_{j+1}(\tilde{x}) \leqslant W_{j+1}\left(x^{(l,max)}\right).$$

Z powyższych nierówności oraz nierówności (7.7) mamy:

$$|W_{j+1}(x^*) - W_{j+1}\left(x^{(l,min)}\right)| \leqslant \varepsilon W_{j+1}(x^*) \tag{7.8}$$

lub

$$|W_{j+1}(x^*) - W_{j+1}\left(x^{(l,max)}\right)| \leqslant \varepsilon W_{j+1}(x^*). \tag{7.9}$$

Teraz na podstawie (7.5) i (7.6) oraz Lematu 6 otrzymujemy:

$$\left|P_{j+1}^E(x^*) - P_{j+1}^E\left(x^{(l,opt)}\right)\right| = \left|P_{j+1}^E(x^*) - P_{j+1}^E(\tilde{x}) + P_{j+1}^E(\tilde{x}) - P_{j+1}^E\left(x^{(l,opt)}\right)\right| \leqslant$$
$$\left|P_{j+1}^E(x^*) - P_{j+1}^E(\tilde{x})\right| + \left|P_{j+1}^E(\tilde{x}) - P_{j+1}^E\left(x^{(l,opt)}\right)\right| \leqslant$$
$$\nu_1 P_{j+1}^E(x^*) + \frac{\varepsilon}{2n}P_{j+1}^E(\tilde{x}) \leqslant \nu_1 P_{j+1}^E(x^*) + \frac{\varepsilon}{2n}(1+\nu_1)P_{j+1}^E(x^*) =$$
$$\left(\nu_1 + \frac{\varepsilon}{2n}(1+\nu_1)\right)P_{j+1}^E(x^*)$$

oraz

$$\left|P_{j+1}^T(x^*) - P_{j+1}^T\left(x^{(l,opt)}\right)\right| = \left|P_{j+1}^T(x^*) - P_{j+1}^T(\tilde{x}) + P_{j+1}^T(\tilde{x}) - P_{j+1}^T\left(x^{(l,opt)}\right)\right| \leqslant$$
$$\left|P_{j+1}^T(x^*) - P_{j+1}^T(\tilde{x})\right| + \left|P_{j+1}^T(\tilde{x}) - P_{j+1}^T\left(x^{(l,opt)}\right)\right| \leqslant$$
$$\nu_1 P_{j+1}^T(x^*) + \frac{\varepsilon}{2n} P_{j+1}^T(\tilde{x}) \leqslant \nu_1 P_{j+1}^T(x^*) + \frac{\varepsilon}{2n}(1+\nu_1) P_{j+1}^T(x^*) =$$
$$\left(\nu_1 + \frac{\varepsilon}{2n}(1+\nu_1)\right) P_{j+1}^T(x^*).$$

Z powyższego wynika:

$$\left|P_{j+1}^E(x^*) - P_{j+1}^E\left(x^{(l,opt)}\right)\right| \leqslant \nu_2 P_{j+1}^E(x^*) \tag{7.10}$$

oraz

$$\left|P_{j+1}^T(x^*) - P_{j+1}^T\left(x^{(l,opt)}\right)\right| \leqslant \nu_2 P_{j+1}^T(x^*). \tag{7.11}$$

Jeżeli $\nu_j \leqslant \varepsilon$ dla $j = 2, \ldots, n-1$, co zostało wcześniej wykazane, to powyższe wyprowadzenie nierówności (7.8) - (7.11) dla iteracji $j+1$ z nierówności (7.2) - (7.4) dla iteracji j może być powtórzone dla $j+1, j+2, \ldots, n-1$. Przypomnijmy, że potrzebowaliśmy nierówności $\nu_1 \leqslant \varepsilon$ aby udowodnić (7.7). Poprzez indukcję mamy, że istnieje $x' \in \mathbf{Y}_n$ takie, że $|W_n(x^*) - W_n(x')| \leqslant \varepsilon W_n(x^*)$. Wynika z tego, że w trzecim kroku algorytmu $\mathbf{A3.4}_\varepsilon$ będzie wybrany taki wektor x^0, dla którego spełniona jest następująca nierówność:

$$W_n(x^0) \leqslant (1+\varepsilon) W^*.$$

□

Twierdzenie 7. *Algorytm* $\mathbf{A3.4}_\varepsilon$ *działa w czasie* $O(n^2 \log^3(\max_j\{n, 1/\varepsilon, p_j, \alpha_j, \beta_j, \gamma\})/\varepsilon^2)$.

Dowód. Najpierw ustalmy złożoność obliczeniową iteracji j Kroku 2. Najbardziej czasochłonną operacją jest podział zbioru $\mathbf{Y}'_j$ na podzbiory $\mathbf{Z}_{q,y}$. Ze złożoności obliczeniowej procedury *Partition* wynika, że podział ten może zostać dokonany w czasie $O(|\mathbf{Y}'_j| \log |\mathbf{Y}'_j|)$. W konsekwencji złożoność obliczeniowa iteracji j Kroku 2 wynosi $O(|\mathbf{Y}'_j| \log |\mathbf{Y}'_j|)$.

Ponieważ zbiór $\mathbf{Y}'_j$ uzyskujemy poprzez wygenerowanie trzech wektorów dla każdego wektora ze zbioru $\mathbf{Y}_{j-1}$, to $|\mathbf{Y}'_j| \leqslant 3|\mathbf{Y}_{j-1}|$. Niech f_j oznacza ilość różnych trójek $q = (l_1, l_2, l_3)$, $1 \leqslant l_1 \leqslant k_W, 1 \leqslant l_2 \leqslant k_E$, $1 \leqslant l_3 \leqslant k_T$, takich że $\mathbf{Y}_{l_1}^W \cap \mathbf{Y}_{l_2}^E \cap \mathbf{Y}_{l_3}^T \neq \emptyset$ w iteracji j. Łatwo zauważyć, że $|\mathbf{Y}_{j-1}| \leqslant 2f_{j-1}$, ponieważ co najwyżej dwa wektory są wybierane z każdego podzbioru $Z_{q,j-1}$. Zgodnie z Lematem 6, wartość f_j nie przekracza $k_W(k_E + k_T)$. Z powyższego wynika, że $|\mathbf{Y}'_j| \leqslant 6f_{j-1} \leqslant 6k_W(k_E + k_T)$.

Ostatecznie, z równania (7.1), otrzymujemy:

$$k_E, k_T \leqslant 2n \log\left(\max\left\{P^E_{j-1}(x), P^T_{j-1}(x)|\mathbf{Y}'_{j-1}\right\}\right)/\varepsilon + 2 \leqslant 2n \log\left(n \max_j p_j\right)/\varepsilon + 2 \leqslant 4n \log\left(\max_j\{n, p_j\}\right)/\varepsilon + 2$$

oraz

$$k_W \leqslant \log\left(\max\left\{W_{j-1}(x)|\mathbf{Y}_{j-1},\right\}\right)/\varepsilon + 2 \leqslant \log\left(n \max_j p_j \max_j\{\alpha_j, \beta_j, \gamma\}\right)/\varepsilon + 2 \leqslant 3 \log\left(\max_j\{n, p_j, \alpha_j, \beta_j, \gamma\}\right)/\varepsilon + 2$$

dla każdego $j = 1, \ldots, n$. Mamy zatem $|\mathbf{Y}'_j| \leqslant O(n \log^2(\max\{n, p_j, \alpha_j, \beta_j, \gamma\})/\varepsilon^2)$. Wynika z tego, że iteracja j Kroku 2 działa w czasie $O(n \log^3(\max\{n, 1/\varepsilon, p_j, \alpha_j, \beta_j, \gamma\})/\varepsilon^2)$ a całkowita złożoność obliczeniowa algorytmu wynosi $O(n^2 \log^3(\max\{n, 1/\varepsilon, p_j, \alpha_j, \beta_j, \gamma\})/\varepsilon^2)$. □

7.6 Wielomianowo rozwiązywalne przypadki

W niniejszym punkcie rozważane będą szczególne przypadki problemu **P3**, które można rozwiązać w czasie wielomianowym.

Niech **P3.5** oznacza szczególny przypadek problemu **P3** z pojedynczym procesorem oraz jednostkowymi czasami wykonywania. W notacji trójpolowej problem można zdefiniować w sposób następujący:

$$1|\langle e; d\rangle; D_{min} \leqslant d - e \leqslant D_{max}; p_j = 1|\sum(\alpha_j E_j + \beta_j T_j) + \gamma(d - e).$$

Łatwo zauważyć, że w rozwiązaniu optymalnym zachodzi następująca zależność $d - e \leqslant \sum p_j = n$. W związku z tym, w dalszych rozważaniach ograniczymy się do instancji problemu, w których $D_{max} \leqslant n$.

Z Własności 7.1 i 7.2 wynika, że $C_{\pi(j)} = j$ dla każdego $j = 1, \ldots, n$. W konsekwencji $E_{\pi(j)} = \max\{e - C_{\pi(j)}, 0\} = \max\{e - j, 0\}$ oraz $T_{\pi(j)} = \max\{C_{\pi(j)} - d, 0\} = \max\{j - d, 0\}$. Przypomnijmy, iż z założenia parametry e i d przyjmują jedynie wartości całkowite. Tak więc, wartość funkcji celu dla problemu **P3.5** wynosi:

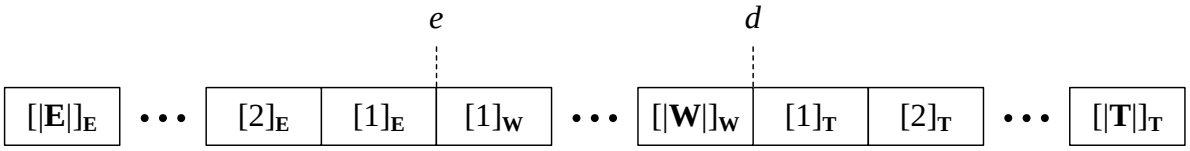
$$Z(\sigma) = \sum_{j\in\mathbf{J}}(\alpha_j E_j + \beta_j T_j) + \gamma(d - e) = \sum_{j=1}^{n}\left(\alpha_{\pi(j)} E_{\pi(j)} + \beta_{\pi(j)} T_{\pi(j)}\right) + \gamma(d - e) = \sum_{j=1}^{n}\left(\alpha_{\pi(j)} \max\{e - j, 0\} + \beta_{\pi(j)} \max\{j - d, 0\}\right) + \sum_{j=e+1}^{d} \gamma = \sum_{j=1}^{e} \alpha_{\pi(j)}(e - j) + \sum_{j=d+1}^{n} \beta_{\pi(j)}(j - d) + \sum_{j=e+D_{min}+1}^{d} \gamma + \gamma D_{min}.$$

Zauważmy, że wartość γD_{min} jest z góry zadana dla ustalonej instancji; w związku z czym nie ma znaczenia w procesie poszukiwania rozwiązania optymalnego.

Dla zadanego harmonogramu $\sigma = (\pi, e, d)$ zdefiniujmy:

$$[k]_{\mathbf{X}} \triangleq \begin{cases} \pi(e-k+1), & \text{jeżeli } \mathbf{X}{=}\mathbf{E}; \\ \pi(e+k), & \text{jeżeli } \mathbf{X}{=}\mathbf{W}; \\ \pi(d+k), & \text{jeżeli } \mathbf{X}{=}\mathbf{T}. \end{cases}$$

Oznaczenie $[k]_{\mathbf{X}}$ jest zilustrowane na Rysunku 7.10.



Rysunek 7.10: Ilustracja oznaczenia $[k]_{\mathbf{X}}$.

Możemy teraz wartość kryterium dla problemu **P3.5** wyrazić w sposób następujący:

$$Z(\sigma) = \sum_{j=1}^{|\mathbf{E}|} \alpha_{[j]_{\mathbf{E}}}(j-1) + \sum_{j=1}^{|\mathbf{T}|} \beta_{[j]_{\mathbf{T}}} j + \sum_{j=D_{min}+1}^{|\mathbf{W}|} \gamma + \gamma D_{min}. \tag{7.12}$$

Z powyższego wyrażenia wynika, że wartość funkcji celu możemy wyrazić jake sumę kosztów wnoszonych przez poszczególne zadania. Niech $cost_{j\mathbf{X}k}$ oznacza koszt jaki wnosi zadanie j, jeżeli $j = [k]_{\mathbf{X}}$. Z wyrażenia (7.12) mamy:

$$cost_{j\mathbf{X}k} = \begin{cases} \alpha_j(k-1), & \text{jeżeli } \mathbf{X} = \mathbf{E} \text{ oraz } k = 1, \ldots, |\mathbf{E}|; \\ 0, & \text{jeżeli } \mathbf{X} = \mathbf{W} \text{ oraz } k = 1, \ldots, D_{min}; \\ \gamma, & \text{jeżeli } \mathbf{X} = \mathbf{W} \text{ oraz } k = D_{min}+1, \ldots, |\mathbf{W}|; \\ \beta_j k, & \text{jeżeli } \mathbf{X} = \mathbf{T} \text{ oraz } k = 1, \ldots, |\mathbf{T}|. \end{cases}$$

Możemy teraz wartość kryterium wyrazić w sposób następujący:

$$Z(\sigma) = \sum_{j=1}^{|\mathbf{E}|} cost_{[j]_{\mathbf{E}}\mathbf{E}j} + \sum_{j=1}^{|\mathbf{T}|} cost_{[j]_{\mathbf{T}}\mathbf{T}j} + \sum_{j=1}^{|\mathbf{W}|} cost_{[j]_{\mathbf{W}}\mathbf{W}j} + \gamma D_{min} = \sum_{\mathbf{X}\in\{\mathbf{E},\mathbf{W},\mathbf{T}\}} \sum_{j=1}^{|\mathbf{X}|} cost_{[j]_{\mathbf{X}}\mathbf{X}j} + \gamma D_{min}. \tag{7.13}$$

Nie została wykazana żadna własność, która pozwalałaby na określenie ilość zadań znajdujących się w poszczególnych zbiorach **E**, **W** i **T**. Z ograniczenia na szerokość przedziału $[e, d]$ wynika jedynie, że: $D_{min} \leqslant |\mathbf{W}| \leqslant D_{max}$. W konsekwencji, ilość zadań w zbiorach **E** i **T** nie może być większa niż $n - D_{min}$. W celu skrócenia notacji, zdefiniujmy:

- $v_{\mathbf{E}} \triangleq n - D_{min}$ - maksymalna ilość zadań w zbiorze $|\mathbf{E}|$;
- $v_{\mathbf{W}} \triangleq D_{max}$ - maksymalna ilość zadań w zbiorze $|\mathbf{W}|$;
- $v_{\mathbf{T}} \triangleq n - D_{min}$ - maksymalna ilość zadań w zbiorze $|\mathbf{T}|$.

Mamy zatem: $|\mathbf{W}| \leqslant v_{\mathbf{W}}$, $|\mathbf{E}| \leqslant v_{\mathbf{E}}$ i $|\mathbf{T}| \leqslant v_{\mathbf{T}}$.

Z powyższych rozważań wynika, że problem **P3.5** redukuje się do problemu minimalizacji:

$$\sum_{j=1}^{n} \sum_{\mathbf{X} \in \{\mathbf{E},\mathbf{W},\mathbf{T}\}} \sum_{k=1}^{v_{\mathbf{X}}} cost_{j\mathbf{X}k}\, x_{j\mathbf{X}k}$$

przy ograniczeniach:

- $\sum\limits_{\mathbf{X} \in \{\mathbf{E},\mathbf{W},\mathbf{T}\}} \sum\limits_{k=1}^{v_{\mathbf{X}}} x_{j\mathbf{X}k} = 1 : (j = 1, \ldots, n)$,
- $\sum\limits_{j=1}^{n} x_{j\mathbf{X}k} \leqslant 1 : (\mathbf{X} \in \{\mathbf{E}, \mathbf{W}, \mathbf{T}\}; k = 1, \ldots, v_{\mathbf{X}})$,
- $x_{j\mathbf{X}k} = 0$ lub $1 : (j = 1, \ldots, n; \mathbf{X} \in \{\mathbf{E}, \mathbf{W}, \mathbf{T}\}; k = 1, \ldots, v_{\mathbf{X}})$.

Rozwiązanie dla przedstawionego powyżej problemu interpretujemy w sposób następujący:

- jeżeli $x_{j\mathbf{X}k} = 1$, to $[k]_{\mathbf{X}} = j$ $(j = 1, \ldots, n; \mathbf{X} \in \{\mathbf{E}, \mathbf{W}, \mathbf{T}\}; k = 1, \ldots, v_{\mathbf{X}})$;
- $e = \sum_{j=1}^{n} \sum_{k=1}^{v_{\mathbf{E}}} x_{j\mathbf{E}k}$;
- $d = e + \sum_{j=1}^{n} \sum_{k=1}^{v_{\mathbf{W}}} x_{j\mathbf{W}k}$.

Zadane ograniczenia gwarantują, że każde zadanie jest wykonywane dokładnie raz oraz każda pozycja jest zajęta przez co najwyżej jedno zadanie. Łatwo zauważyć, że tak zdefiniowany problem programowania liniowego jest jednocześnie znanym problemem skojarzenia, który może być rozwiązany w czasie $O(n^3)$ (patrz Lawler [78]).

Niech **P3.6** oznacza szczególny przypadek problemu **P3.1** z jednostkowymi czasami wykonywania. W notacji trójpolowej problem jest zdefiniowany następująco:

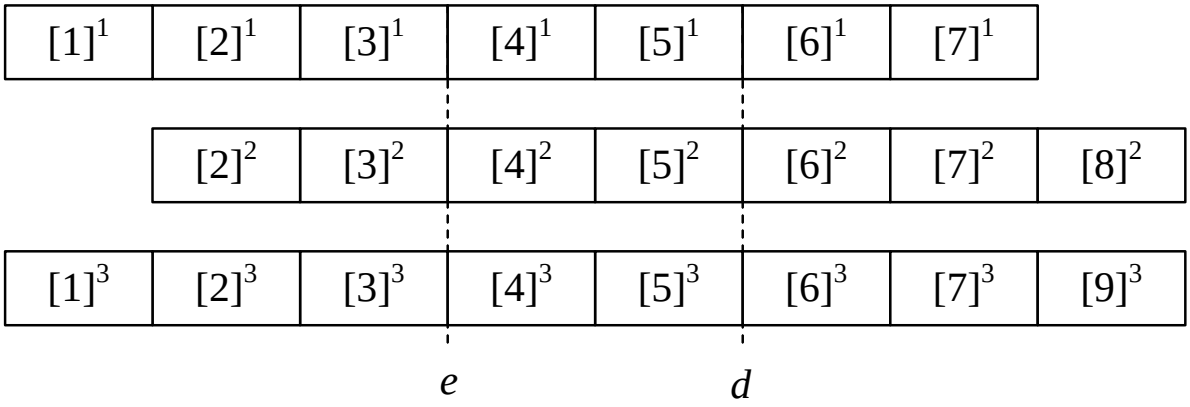
$$P|\langle e; d\rangle; D_{min} \leqslant d - e \leqslant D_{max}; p_j = 1| \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d - e).$$

Zauważmy, że problem **P3.6** jest wieloprocesorową wersją problemu **P3.5**.

Zgodnie z Własnościami 7.7, 7.8 i 7.9, $C_j \in \{1, 2, \ldots, n\}$ dla każdego zadania $j \in \mathbf{J}$. Niech $[j]^i$ oznacza zadanie, które kończy wykonywać się w momencie j na procesorze i - patrz Rysunek 7.11. W celu uproszczenia dalszych rozważań, będziemy mówili, że $[j]^i$ jest zadaniem na pozycji j na procesorze i. Ponadto, niech s^i oraz c^i oznaczają odpowiednio

moment zakończenia wykonywania pierwszego i ostatniego zadania na procesorze i. Skoro $C_{[j]^i} = j$, to $E_{[j]^i} = \max\{e - C_{[j]^i}, 0\} = \max\{e - j, 0\}$ oraz $T_{[j]^i} = \max\{C_{[j]^i} - d, 0\} = \max\{j - d, 0\}$. Przypomnijmy, iż z założenia parametry e i d przyjmują jedynie wartości całkowite. Wykorzystując wprowadzone oznaczenia, wartość funkcji celu możemy przedstawić w sposób następujący:

$$\begin{gathered} Z(\sigma) = \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d-e) = \\ \sum_{i=1}^{m}\left(\sum_{j=s^i}^{c^i}\left(\alpha_{[j]^i}E_{[j]^i} + \beta_{[j]^i}T_{[j]^i}\right)\right) + \gamma(d-e) = \\ \sum_{i=1}^{m}\left(\sum_{j=s^i}^{c^i}\left(\alpha_{[j]^i}\max\{e-j,0\} + \beta_{[j]^i}\max\{j-d,0\}\right)\right) + \gamma(d-e) = \\ \sum_{i=1}^{m}\left(\sum_{j=s^i}^{e}\alpha_{[j]^i}(e-j) + \sum_{j=d+1}^{c^i}\beta_{[j]^i}(j-d)\right) + \gamma(d-e). \end{gathered} \tag{7.14}$$



Rysunek 7.11: Przykładowy harmonogram dla problemu **P3.6**.

Zauważmy, że jeżeli szerokość przedziału $[e, d]$ jest równa lub większa niż $\left\lceil \frac{n}{m} \right\rceil$, to wystarczy uszeregować wszystkie zadania w dowolnej kolejności wewnątrz tego przedziału, aby uzyskać minimalną wartość kryterium. Z związku z tym w dalszych rozważaniach ograniczymy się tylko do tych instancji, w których $D_{max} \leqslant \left\lceil \frac{n}{m} \right\rceil$.

Niech **P3.6** (D) oznacza szczególny przypadek problemu **P3.6**, w którym $D_{min} = D_{max} = D$. Łatwo zauważyć, że w celu rozwiązania problemu **P3.6**, wystarczy rozwiązać problem **P3.6** (D) dla każdego $D = D_{min}, D_{min} + 1 \ldots, D_{max}$ a następnie wybrać rozwiązanie o najmniejszej wartości kryterium. Z tego powodu skupimy się teraz na znalezieniu algorytmu optymalnego dla problemu **P3.6** (D).

Dla zadanego harmonogramu σ zdefiniujmy:

$$[k]^i_{\mathbf{X}} \triangleq \begin{cases} [e-k+1]^i, & \text{jeżeli } \mathbf{X}=\mathbf{E}; \\ [e+k]^i, & \text{jeżeli } \mathbf{X}=\mathbf{W}; \\ [d+k]^i, & \text{jeżeli } \mathbf{X}=\mathbf{T}. \end{cases}$$

Możemy teraz wartość kryterium dla problemu **P3.6**(D) wyrazić w sposób następujący:

$$Z(\sigma) = \sum_{i=1}^{m} \left[\sum_{j=1}^{|\mathbf{E}^i|} \alpha_{[j]^i_{\mathbf{E}}} (j-1) + \sum_{j=1}^{|\mathbf{T}^i|} \beta_{[j]^i_{\mathbf{T}}} j \right] + \gamma D. \tag{7.15}$$

Zauważmy, że γD jest wartością stałą i nie podlega optymalizacji. Z wyrażenia (7.15) wynika, że wartość funkcji celu możemy wyrazić jake sumę kosztów wnoszonych przez poszczególne zadania. Niech zatem $\widetilde{cost}_{j\mathbf{X}k}$ oznacza koszt jaki wnosi zadanie j do całkowitego kosztu harmonogramu, jeżeli $j = [k]^i_{\mathbf{X}}$ w tym harmonogramie. Na podstawie wyrażenia (7.15) wartość tego kosztu definiujemy w sposób następujący:

$$\widetilde{cost}_{jXk} \triangleq \begin{cases} \alpha_j(k-1), & \text{jeżeli } \mathbf{X}=\mathbf{E} \text{ oraz } k=1,\ldots,|\mathbf{E}|; \\ 0, & \text{jeżeli } \mathbf{X}=\mathbf{W} \text{ oraz } k=1,\ldots,|\mathbf{W}|; \\ \beta_j k, & \text{jeżeli } \mathbf{X}=\mathbf{T} \text{ oraz } k=1,\ldots,|\mathbf{T}|. \end{cases}$$

Korzystając z wprowadzonych oznaczeń możemy wartość kryterium wyrazić w sposób następujący:

$$\begin{aligned} Z(\sigma) = \sum_{i=1}^{m} \left(\sum_{j=1}^{|\mathbf{E}^i|} \widetilde{cost}_{[j]^i_{\mathbf{E}}\mathbf{E}j} + \sum_{j=1}^{|\mathbf{T}^i|} \widetilde{cost}_{[j]^i_{\mathbf{T}}\mathbf{T}j} \right) + \gamma D = \\ \sum_{i=1}^{m} \sum_{\mathbf{X}\in\{\mathbf{E},\mathbf{W},\mathbf{T}\}} \sum_{j=1}^{|\mathbf{X}^i|} \widetilde{cost}_{[j]^i_{\mathbf{X}}\mathbf{X}j} + \gamma D. \end{aligned} \tag{7.16}$$

Skoro czasy wykonywania zadań są jednostkowe, to w rozwiązaniu optymalnym dokładnie mD zadań wykonuje się wewnątrz przedziału $[e,d]$ dla problemu **P3.6**(D). Oznacza to, że co najwyżej $n-mD$ zadań może wykonać się przed tym przedziałem oraz co najwyżej $n-mD$ zadań może wykonać się za tym przedziałem. Łatwo zauważyć, że w rozwiązaniu optymalnym $|\mathbf{E}^i| \leqslant |\mathbf{E}^v|+1$ i $|\mathbf{T}^i| \leqslant |\mathbf{T}^v|+1$ dla każdych dwóch procesorów i oraz v. Z powyższego wynika, że $|\mathbf{E}^i| \leqslant \left\lceil \frac{n-mD}{m} \right\rceil = \left\lceil \frac{n}{m} \right\rceil - D$ i $|\mathbf{T}^i| \leqslant \left\lceil \frac{n-mD}{m} \right\rceil = \left\lceil \frac{n}{m} \right\rceil - D$. W celu skrócenia notacji zdefiniujmy:

- $\tilde{v}_{\mathbf{E}} \triangleq \left\lceil \frac{n}{m} \right\rceil - D$ - maksymalna ilość zadań w zbiorze $|\mathbf{E}^i|$ $(i=1,\ldots,m)$;
- $\tilde{v}_{\mathbf{W}} \triangleq D$ - maksymalna ilość zadań w zbiorze $|\mathbf{W}^i|$ $(i=1,\ldots,m)$;
- $\tilde{v}_{\mathbf{T}} \triangleq \left\lceil \frac{n}{m} \right\rceil - D$ - maksymalna ilość zadań w zbiorze $|\mathbf{T}^i|$ $(i=1,\ldots,m)$.

Mamy zatem $|\mathbf{E}^i| \leqslant \tilde{v}_{\mathbf{E}}$, $|\mathbf{T^i}| \leqslant \tilde{v}_{\mathbf{T}}$ oraz $|\mathbf{W^i}| = \tilde{v}_{\mathbf{W}}$ w każdym rozwiązaniu dopuszczalnym. Zauważmy, że:

$$\sum_{i=1}^{m}\left(|\mathbf{E}^i| + |\mathbf{T}^i| + |\mathbf{W}^i|\right) \leqslant \sum_{i=1}^{m}(\tilde{v}_{\mathbf{E}} + \tilde{v}_{\mathbf{T}} + \tilde{v}_{\mathbf{W}}) = m(\tilde{v}_{\mathbf{E}} + \tilde{v}_{\mathbf{T}} + \tilde{v}_{\mathbf{W}}) =$$
$$m\left(\left\lceil\frac{n}{m}\right\rceil - D + \left\lceil\frac{n}{m}\right\rceil - D + D\right) \leqslant m\left(2\left\lceil\frac{n}{m}\right\rceil\right) = O(n).$$

Z powyższych rozważań wynika, że problem **P3.6**(D) redukuje się do problemu minimalizacji:

$$\sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{\mathbf{X}\in\{\mathbf{E},\mathbf{W},\mathbf{T}\}}\sum_{k=1}^{\tilde{v}_{\mathbf{X}}}\widetilde{cost}_{j\mathbf{X}k}\; x^i_{j\mathbf{X}k}$$

przy ograniczeniach:

- $\sum\limits_{i=1}^{m}\sum\limits_{\mathbf{X}\in\{\mathbf{E},\mathbf{W},\mathbf{T}\}}\sum\limits_{k=1}^{\tilde{v}_{\mathbf{X}}} x^i_{j\mathbf{X}k} = 1 : (j = 1,\ldots,n)$,
- $\sum\limits_{j=1}^{n} x^i_{j\mathbf{X}k} \leqslant 1 : (i = 1,\ldots,m;\ \mathbf{X} \in \{\mathbf{E},\mathbf{W},\mathbf{T}\};\ k = 1,\ldots,\tilde{v}_{\mathbf{X}})$,
- $x^i_{j\mathbf{X}k} = 0$ lub $1 : (i = 1,\ldots,m;\ j = 1,\ldots,n;\ \mathbf{X} \in \{\mathbf{E},\mathbf{W},\mathbf{T}\};\ k = 1,\ldots,\tilde{v}_{\mathbf{X}})$.

Rozwiązanie dla przedstawionego powyżej problemu interpretujemy w sposób następujący:

- jeżeli $x_{j\mathbf{X}k} = 1$, to $[k]_{\mathbf{X}} = j$ $(i = 1,\ldots,m;\ j = 1,\ldots,n;\ \mathbf{X} \in \{\mathbf{E},\mathbf{W},\mathbf{T}\};\ k = 1,\ldots,\tilde{v}_{\mathbf{X}})$
- $e = \max\left\{\sum_{j=1}^{n}\sum_{k=1}^{\tilde{v}_{\mathbf{E}}} x^i_{j\mathbf{E}k} : i = 1,\ldots,m\right\}$;
- $d = e + D$.

Zastosowane ograniczenia gwarantują, że każde zadanie jest wykonywane dokładnie raz i każda pozycja jest zajęta przez co najwyżej jedno zadanie. Łatwo zauważyć, że zdefiniowany powyżej problem jest dobrze znanym problemem skojarzenia, który może być rozwiązany optymalnie w czasie $O(n^3)$ (patrz Lawler [78]). Przypomnijmy, że w celu rozwiązania problemu **P3.6** wystarczy rozwiązać problem **P3.6** (D) dla każdego $D = D_{min}, D_{min}+1,\ldots,D_{max}$. Przypomnijmy również, że założyliśmy, iż ograniczamy się do instancji, w których $D_{min} \leqslant \left\lceil\frac{n}{m}\right\rceil$. Skoro problem **P3.6** (D) można rozwiązać w czasie $O((v_{\mathbf{E}} + v_{\mathbf{W}} + v_{\mathbf{T}})^3) = O(n^3)$, to problem **P3.6** można rozwiązać w czasie $O(n^3(D_{max} - D_{min})) = O\left(\frac{n^4}{m}\right)$.

Niech **P3.7** oznacza szczególny przypadek problemu **P3.6**, w którym $\alpha_j = \beta_j$ dla każdego zadania $j = 1,\ldots,n$. W notacji trójpolowej problem ten można przedstawić

następująco:

$$P|\langle e;d\rangle; D_{min} \leqslant d-e \leqslant D_{max}; p_j = 1|\sum \alpha_j(E_j+T_j)+\gamma(d-e).$$

Podobnie jak miało to miejsce w przypadku problemu **P3.6**, ograniczymy się w naszych rozważaniach do instancji problemu, w których $D_{max} \leqslant \left\lceil \frac{n}{m} \right\rceil$.

Wartość funkcji celu dla problemu **P3.7** wynosi:

$$Z(\sigma) = \sum_{i=1}^{m} \left(\sum_{j=s^i}^{e} (e-j)\alpha_{[j]^i} + \sum_{j=d+1}^{c^i} (j-d)\alpha_{[j]^i} \right) + \gamma(d-e) = \sum_{i=1}^{m} \sum_{j=s^i}^{c^i} w_j \alpha_{[j]^i} + \gamma(d-e), \tag{7.17}$$

gdzie w_j oznacza wagę pozycji j i jest zdefiniowane w sposób następujący:

$$w_j \triangleq \begin{cases} (e-j), & \text{jeżeli } j \leqslant e; \\ 0, & \text{jeżeli } e < j \leqslant d; \\ (j-d), & \text{jeżeli } j > d. \end{cases}$$

Przypomnijmy, że w tym podrozdziale poprzez zadanie na pozycji j rozumiemy zdanie kończące wykonywać się w momencie j.

Jeżeli szerokość przedziału $[e,d]$ byłaby zadana, to w celu optymalizacji kryterium, należałoby znaleźć taki harmonogram, który minimalizowałby wartość następującego wyrażenia: $\sum_{i=1}^{m} \sum_{j=s^i}^{c^i} w_j \alpha_{[j]^i}$.

Zauważmy, że waga pozycji nie zależy od procesora. Ponadto zachodzą następujące relacje pomiędzy wagami pozycji:

$$w_e = w_{e+1} = \ldots = w_{d-1} = w_d = 0 \tag{7.18}$$

$$1 = w_{e-1} = w_{d+1} < w_{e-2} = w_{d+2} < w_{e-3} = w_{d+3} < \ldots \tag{7.19}$$

Przypomnijmy w tym momencie Lemat 2 zaprezentowany w Rozdziale 5.

Lemat 2. [40] *Niech* $\bar{a} = (a_1, a_2, \ldots, a_n)$ *oraz* $\bar{b} = (b_1, b_2, \ldots, b_n)$ *oznaczają dwa wektory o dodatnich współczynnikach. Iloczyn skalarny wektorów a i b jest minimalny, gdy współczynniki tych wektorów są odwrotnie uporządkowane, czyli* $a_1 \leqslant a_2 \leqslant \ldots \leqslant a_n$; $b_1 \geqslant b_2 \geqslant \ldots \geqslant b_n$ *lub* $a_1 \geqslant a_2 \geqslant \ldots \geqslant a_n$; $b_1 \leqslant b_2 \leqslant \ldots \leqslant b_n$.

Z Lematu 2 wynika prawdziwość Własności 7.14.

Własność 7.14. *W rozwiązaniu optymalnym problemu* **P3.7**, *dla zadanej szerokości przedziału* $[e,d]$, *zadania są uszeregowane w taki sposób, że zadanie o największej wartości parametru* α_j *jest przyporządkowane do pozycji o najmniejszej wadze, zadanie o kolejnej*

największej wartości parametru α_j jest przyporządkowane do pozycji o kolejnej najmniejszej wadze, itd.

W dalszych rozważaniach będziemy brać pod uwagę tylko te rozwiązania problemu **P3.7**, które spełniają Własność 7.14.

Załóżmy, że zadania są ponumerowane tak, że $\alpha_1 \geqslant \alpha_2 \geqslant \ldots \geqslant \alpha_m$.

Jeżeli szerokość przedziału $[e, d]$ byłaby z góry zadana, to moglibyśmy skonstruować rozwiązanie optymalne, w oparciu o Własność 7.14 oraz zależności (7.18) i (7.19), w następujący sposób:

- uszereguj $m(d-e)$ pierwszych zadań (czyli zadań o najmniejszych wartościach parametru α_j) na pozycjach $e, \ldots, d$ na wszystkich m procesorach;
- uszereguj m kolejnych zadań na pozycji $d+1$ na wszystkich m procesorach;
- uszereguj m kolejnych zadań na pozycji $e-1$ na wszystkich m procesorach;
- uszereguj m kolejnych zadań na pozycji $d+2$ na wszystkich m procesorach;
- i tak dalej, aż do uszeregowania wszystkich zadań.

Jeżeli założymy, że szerokość przedziału $[e, d]$ jest równa D, to wartości parametrów e i d dla harmonogramu skonstruowanego w przedstawiony powyżej sposób możemy wyliczyć następująco:

$$e = \left\lceil \frac{n - Dm}{2m} \right\rceil ; \; d = e + D. \tag{7.20}$$

Niech $\sigma(D)$ oznacza rozwiązanie optymalne problemu **P3.7** dla ustalonej szerokości przedziału $[e, d]$ równej D. Załóżmy, że rozwiązanie to zostało skonstruowane w sposób, który został przedstawiony powyżej. Wartość funkcji celu dla harmonogramu $\sigma(D)$ wynosi:

$$\begin{gathered} Z(\sigma(D)) = \gamma D + \sum_{i=1}^{m} \sum_{j=s^i}^{c^i} w_j \alpha_{[j]^i} = \\ \gamma D + \sum_{j=(D+1)m+1}^{(D+2)m} w_{d+1}\alpha_j + \sum_{j=(D+2)m+1}^{(D+3)m} w_{e-1}\alpha_j + \\ \sum_{j=(D+3)m+1}^{(D+4)m} w_{d+2}\alpha_j + \sum_{j=(D+4)m+1}^{(D+5)m} w_{e-2}\alpha_j + \ldots = \\ \gamma D + \sum_{j=(D+1)m+1}^{(D+3)m} \alpha_j + \sum_{j=(D+3)m+1}^{(D+5)m} 2\alpha_j + \ldots \end{gathered} \tag{7.21}$$

Musimy zatem znaleźć taką szerokość pożądanego przedziału zakończenia wykonywania zadań $D^* = D_{min}, \ldots, D_{max}$, dla której wartość wyrażenia $Z(\sigma(D^*))$ jest minimalna. W tym celu wystarczy wyliczyć wartość $Z(\sigma(D))$ dla każdego $D = D_{min}, D_{min} +$

$1, \ldots, D_{max}$, a następnie wybrać takie D^*, że:

$$D^* = \arg\min\{Z(\sigma(D)) : D = D_{min}, \ldots, D_{max}\}.$$

Znalezienie optymalnej szerokości przedziału $[e, d]$ w ten sposób zajęłoby $O(n(D_{max} - D_{min}))$ czasu. Wykorzystując pewne prawidłowości występujące w wyrażeniu (7.21), cel ten można osiągnąć również w czasie $O(n)$. Szczegóły zostaną przedstawione poniżej.

Pierwszym krokiem jest zbadanie jak wartość kryterium zmienia się kiedy zwiększamy szerokość tego przedziału o 1. Zdefiniujmy zatem $Z_\Delta(D) \triangleq Z(\sigma(D+1)) - Z(\sigma(D))$. Podstawiając do wzoru (7.21) otrzymujemy:

$$\begin{gathered} Z_\Delta(D) = Z(\sigma(D+1)) - Z(\sigma(D)) = \\ \left[\gamma(D+1) + \sum_{j=(D+2)m+1}^{(D+4)m} \alpha_j + \sum_{j=(D+4)m+1}^{(D+6)m} 2\alpha_j + \ldots\right] - \\ \left[\gamma D + \sum_{j=(D+1)m+1}^{(D+3)m} \alpha_j + \sum_{j=(D+3)m+1}^{(D+5)m} 2\alpha_j + \ldots\right] = \\ \gamma - \sum_{j=(D+1)m+1}^{(D+2)m} \alpha_j - \sum_{j=(D+3)m+1}^{(D+4)m} \alpha_j + \ldots \end{gathered} \tag{7.22}$$

Własność 7.15. *Istnieje rozwiązanie optymalne problemu* **P3.7***, w którym:*

$$d - e = \begin{cases} D_{min}, & \text{jeżeli } Z_\Delta(D_{min}) \geqslant 0; \\ \max\{D : D \leqslant D_{max}; Z_\Delta(D-1) < 0\}, & \text{w przeciwnym przypadku.} \end{cases}$$

Dowód. Na wstępie zostanie wykazane, że $Z_\Delta(D) \geqslant Z_\Delta(D-1)$ dla każdego $D = D_{min} + 1, \ldots, D_{max}$. Z wyrażenia (7.22) otrzymujemy:

$$\begin{gathered} Z_\Delta(D) - Z_\Delta(D-1) = \\ \left(\gamma - \sum_{j=(D+1)m+1}^{(D+2)m} \alpha_j - \sum_{j=(D+3)m+1}^{(D+4)m} \alpha_j - \ldots\right) - \left(\gamma - \sum_{j=Dm+1}^{(D+1)m} \alpha_j - \sum_{j=(D+2)m+1}^{(D+3)m} \alpha_j - \ldots\right) = \\ \sum_{j=Dm+1}^{(D+1)m} \alpha_j - \sum_{j=(D+1)m+1}^{(D+2)m} \alpha_j + \sum_{j=(D+2)m+1}^{(D+3)m} \alpha_j - \sum_{j=(D+3)m+1}^{(D+4)m} \alpha_j + \ldots \end{gathered}$$

Przypomnijmy, że zadania są ponumerowane tak, że $\alpha_1 \geqslant \alpha_2 \geqslant \ldots \geqslant \alpha_n$. Wynika z tego, że dla każdego $k \geqslant 1$ prawdziwa jest następująca zależność:

$$\sum_{j=(D+k)m+1}^{(D+k+1)m} \alpha_j \geqslant \sum_{j=(D+k+1)m+1}^{(D+k+2)m} \alpha_j.$$

Możemy więc stwierdzić, że $Z_\Delta(D) - Z_\Delta(D-1) \geqslant 0$ dla dowolnego $D = D_{min}, \ldots, D_{max}$.

Rozważmy teraz dwa możliwe przypadki.

Przypadek 1. Załóżmy, że $Z_\Delta(D_{min}) \geqslant 0$.

Skoro $Z_\Delta(D) \geqslant Z_\Delta(D-1)$, to $Z_\Delta(D_{max}) \geqslant Z_\Delta(D_{max}-1) \geqslant \ldots \geqslant Z_\Delta(D_{min}) \geqslant 0$. Wynika z tego, że $Z(\sigma(D_{max})) \geqslant Z(\sigma(D_{max}-1)) \geqslant \ldots \geqslant Z(\sigma(D_{min}))$. Możemy zatem stwierdzić, że $d-e = D_{min}$ w rozwiązaniu optymalnym.

Przypadek 2. Załóżmy, że $Z_\Delta(D_{min}) < 0$.

Niech $D^* = \max\{D : D \leqslant D_{max}; Z_\Delta(D-1) < 0\}$. Skoro $Z_\Delta(D-1) \leqslant Z_\Delta(D)$, to:

- $Z_\Delta(D) \geqslant 0$ dla każdego $D = D^*, \ldots, D_{max}$;
- $Z_\Delta(D) < 0$ dla każdego $D = D_{min}, \ldots, D^*-1$.

Skoro $Z_\Delta(D) \triangleq Z(\sigma(D+1)) - Z(\sigma(D))$, to:

$$Z(\sigma(D_{min})) > \ldots > Z(\sigma(D^*-1)) > Z(\sigma(D^*)) \leqslant Z(\sigma(D^*+1)) \leqslant \ldots \leqslant Z(\sigma(D_{max})).$$

Możemy zatem stwierdzić, że $d-e = D^*$ w rozwiązaniu optymalnym.

Powyższe rozważania potwierdzają prawdziwość dowodzonej własności. □

Wyliczenie wartości $Z_\Delta(D)$ na podstawie wyrażenia (7.22) zajmuje $O(n)$ czasu. Niemniej jednak wystarczy zauważyć, że wartość $Z_\Delta(D)$ może być również wyliczona rekurencyjnie poprzez wykorzystanie następującej zależności:

$$Z_\Delta(D) - Z_\Delta(D-2) = \left[\gamma - \sum_{j=(D+1)m+1}^{(D+2)m} \alpha_j - \sum_{j=(D+3)m+1}^{(D+4)m} \alpha_j - \ldots\right] - \left[\gamma - \sum_{j=(D-1)m+1}^{Dm} \alpha_j - \sum_{j=(D+1)m+1}^{(D+2)m} \alpha_j - \sum_{j=(D+3)m+1}^{(D+4)m} \alpha_j - \ldots\right] = \sum_{j=(D-1)m+1}^{Dm} \alpha_j.$$

Zatem:

$$Z_\Delta(D) = Z_\Delta(D-2) + \sum_{j=(D-1)m+1}^{Dm} \alpha_j. \tag{7.23}$$

Wystarczy więc wyliczyć wartości $Z_\Delta(D_{min})$ oraz $Z_\Delta(D_{min}+1)$ z wyrażenia (7.22) a następnie, dla każdego $D > D_{min}+1$, wartość$Z_\Delta(D)$ może zostać wyliczona z wyrażenia (7.23) w czasie $O(m)$. Skoro, zgodnie z wcześniejszym założeniem, ograniczamy się w naszych rozważaniach do instancji, w których $D_{max} \leqslant \left\lceil \frac{n}{m} \right\rceil$, to wyliczenie wartości $Z_\Delta(D)$ dla każdego $D = D_{min}, \ldots, D_{max}$ zajmie $2O(n) + (D_{max} - D_{min})\,O(m) = 2O(n) + O\left(\frac{n}{m}\right) O(m) = O(n)$ czasu.

Oznaczmy przez **A3.7** algorytm rozwiązujący optymalnie problem **P3.7**. Algorytm ten działa w trzech etapach. W pierwszym etapie (Kroki 1-2) ustalana jest optymalna

szerokość przedziału $[e, d]$. W kolejnym etapie (Krok 3) znajdowany jest początek i koniec przedziału $[e, d]$ zgodnie ze wzorem (7.20). W ostatnim etapie (Kroki 4-6) zadania szeregowane są zgodnie z Własnością 7.14.

Poniżej przedstawiony jest szczegółowy opis algorytmu **A3.7**. Zwrot "uszereguj zadanie j na pozycji k" będzie oznaczał "uszereguj zadanie j tak, że $C_j = k$".

Algorytm A3.7

Krok 1. Ponumeruj zadania według nierosnącego porządku wartości α_j, tzn. $\alpha_1 \geqslant \alpha_2 \geqslant \ldots \geqslant \alpha_n$. Podstaw $D := D_{min}$. Wylicz $Z_\Delta(D)$ i $Z_\Delta(D+1)$ na podstawie wyrażenia (7.22).

Krok 2. Jeżeli $Z_\Delta(D) < 0$ i $D < D_{max}$, to wylicz $Z_\Delta(D+1)$ na podstawie wyrażenia (7.23), podstaw $D := D + 1$ i powtórz Krok 2.

Krok 3. Podstaw $e := \left\lceil \frac{n-Dm}{2m} \right\rceil$ i $d := e + D$.

Krok 4. Uszereguj zadania $1, \ldots, (D+1)m$ w dowolnej kolejności na pozycjach $e, \ldots, d$ na procesorach $1, \ldots, m$. Podstaw $j := (D+1)m$, $x := e - 1$ i $y := d + 1$.

Krok 5. Uszereguj zadania $j+1, j+2, \ldots, \min\{j+m, n\}$ na pozycji y na procesorach $1, \ldots, \min\{m, n-j\}$. Podstaw $j := j + m$ oraz $y := y + 1$. Jeżeli $j \geqslant n$, to STOP.

Krok 6. Uszereguj zadania $j+1, j+2, \ldots, \min\{j+m, n\}$ na pozycji x na procesorach $1, \ldots, \min\{m, n-j\}$. Podstaw $j := j + m$ oraz $x := x - 1$. Jeżeli $j \geqslant n$, to STOP; w przeciwnym przypadku idź do Kroku 5.

Twierdzenie 8. *Algorytm* **A3.7** *rozwiązuje optymalnie problem* **P3.7** *w czasie* $O(n \log n)$.

Dowód. Najpierw przeanalizujmy złożoność obliczeniową algorytmu. Krok 1 potrzebuje czasu $O(n \log n)$. Krok 2 potrzebuje czasu $O(m)$ i jest powtarzany $O\left(\frac{n}{m}\right)$ razy. Krok 3 działa w czasie $O(1)$. W Kroku 4 wykonywanych jest $O(n)$ operacji. Kroki 5 i 6 potrzebują czasu $O(m)$ i są powtarzane $O\left(\frac{n}{m}\right)$ razy. Z powyższego wynika, że całkowita złożoność obliczeniowa algorytmu wynosi $O(n \log n) + O(m)O\left(\frac{n}{m}\right) + O(1) + O(n) + O(m)O\left(\frac{n}{m}\right) = O(n \log n)$.

Wyznaczenie optymalnej szerokości przedziału $[e, d]$ odbywa się w Krokach 1-2 zgodnie z Własnością 7.15. Następnie, w Kroku 3 parametry przedziału $[e, d]$ są ustalane w oparciu o wyrażenie (7.20). W krokach 4-6, zadania są szeregowane zgodnie z Własnością 7.14. W związku z tym, rozwiązanie dostarczone przez Algorytm **A3.7** jest rozwiązaniem optymalnym. □

7.7 Podsumowanie rozdziału

W niniejszym rozdziale analizowano wieloprocesorowy problem szeregowania zadań z doborem pożądanego przedziału zakończenia ich wykonywania, przy zadanym dolnym i górnym ograniczeniu na szerokość tego przedziału. Minimalizacji podlegała suma ważonych nieterminowości wykonania zadań oraz kary związanej z szerokością tego przedziału. Wykazano wiele własności rozwiązania optymalnego oraz pokazano, że problem ten jest silnie NP-trudny.

Dla szczególnego przypadku tego problemu ze zgodnymi ilorazami wag skonstruowano optymalny algorytm oparty na metodzie programowania dynamicznego. Ponadto zaproponowano w pełni wielomianowy schemat aproksymacyjny dla szczególnego przypadku z pojedynczym procesorem, ze zgodnymi ilorazami wag oraz bez ograniczeń na szerokość pożądanego przedziału zakończenia wykonywania zadań. Pokazano ponadto, że przy jednostkowych czasach wykonywania zadań, problem ten można rozwiązać w czasie $O(\frac{n^4}{m})$. Wykazano również, że czas rozwiązania problemu z jednostkowymi czasami wykonywania można dodatkowo zredukować do $O(n^3)$ dla przypadku z pojedynczym procesorem oraz do $O(n \log n)$ dla przypadku z symetrycznymi kosztami jednostkowymi nieterminowego wykonania zadań.

Rozdział 8

Minimalizacja ważonej liczby zadań opóźnionych i przedwcześnie wykonanych

8.1 Wstęp

W niniejszym rozdziale przedstawione zostaną nowe wyniki, uzyskane przez autora niniejszej pracy, dotyczące dwóch jednoprocesorowych problemów szeregowania zadań przy kryterium minimalizacji ważonej liczby opóźnionych i przedwcześnie wykonanych zadań. W pierwszym analizowanym problemie zarówno początek jak i koniec pożądanego przedziału zakończenia wykonywania zadań będą zmiennymi decyzyjnymi. Natomiast w drugim problemie początek pożądanego przedziału zakończenia wykonywania zadań będzie zmienną decyzyjną, podczas gdy szerokość tego przedziału będzie z góry zadana.

W niniejszym rozdziale zaprezentowana zostaną precyzyjne definicje analizowanych problemów (Podrozdział 8.2). Następnie przedstawione zostaną niezbędne oznaczenia oraz własności rozwiązań optymalnych obu problemów (Podrozdział 8.3). W Podrozdziale 8.4 zostanie zaprezentowany wielomianowy algorytm optymalny dla pierwszego problemu oraz dwa pseudowielomianowe algorytmy optymalne dla drugiego problemu. W pełni wielomianowy schemat aproksymacyjny dla drugiego problemu zostanie opisany w Podrozdziale 8.5. Niniejszy rozdział zamyka krótkie podsumowanie uzyskanych wyników.

8.2 Sformułowanie problemu

Poniżej zostanie przedstawiona definicja pierwszego z rozważanych problemów.

Zadany jest zbiór n niezależnych i niepodzielnych zadań $\mathbf{J} = \{1, \ldots, n\}$ do wykonania na pojedynczym procesorze. Procesor ten może wykonywać jednocześnie tylko jedno zadanie. Każde zadanie $j \in \mathbf{J}$, o czasie wykonywania p_j, jest dostępne w momencie 0.

Wszystkie zadania mają wspólny pożądany przedział zakończenia wykonywania. Początek i koniec tego przedziału oznaczmy przez e i d.

Harmonogram zadań określa momenty zakończenia wykonywania zadań oraz początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

Przypomnijmy definicje następujących oznaczeń:

- S_j - moment rozpoczęcia wykonywania zadania j;
- C_j - moment zakończenia wykonywania zadania j;
- $V_j \triangleq \begin{cases} 1, & \text{jeżeli } C_j < e; \\ 0, & \text{w przeciwnym przypadku;} \end{cases}$
- $U_j \triangleq \begin{cases} 1, & \text{jeżeli } C_j > d; \\ 0, & \text{w przeciwnym przypadku.} \end{cases}$

Przypomnijmy również, że w celu podkreślenia, iż konkretne oznaczenie (np. S_j) odnosi się do pewnego, określonego harmonogramu σ, stosowany będzie następujący zapis: $S_j(\sigma)$.

Celem jest znalezienie takiego harmonogramu σ, który minimalizuje wartość następującego kryterium:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j) + \gamma(d - e) + \theta d,$$

gdzie $\alpha_j \geqslant 0$, $\beta_j \geqslant 0$, $\gamma \geqslant 0$ i $\theta \geqslant 0$ są zadanymi wagami, które przyjmują tylko wartości całkowite.

W trójpolowej notacji [34], problem może być przedstawiony następująco:

$$1|\langle e, d\rangle| \sum(\alpha_j V_j + \beta_j U_j) + \theta d.$$

W celu skrócenia notacji, badany problem będziemy oznaczać również jako **P4.1**.

Drugi z badanych problemów, który będziemy oznaczać jako **P4.2**, jest bardzo podobny do problemu **P4.1**. Różnica polega jedynie na tym, że w problemie **P4.2** szerokość przedziału jest z góry zadana i wynosi D. W konsekwencji, kara za szerokość przedziału nie jest uwzględniana w kryterium. Tak więc w przypadku problemu **P4.2** wartość funkcji celu dla harmonogramu σ wynosi:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j) + \theta d.$$

W notacji trójpolowej [34], problem **P4.2** może być przedstawiony następująco:

$$1|\langle e, e + D\rangle| \sum(\alpha_j V_j + \beta_j U_j) + \theta d.$$

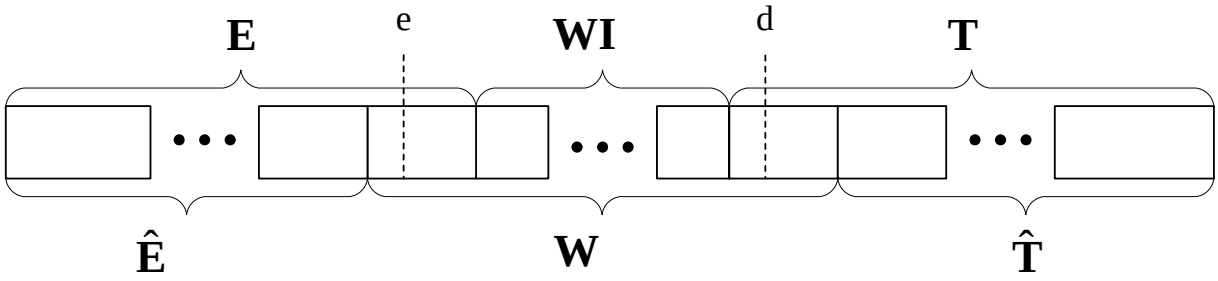
8.3 Własności rozwiązania optymalnego

W niniejszym podrozdziale zostaną zaprezentowane własności dla problemów **P4.1** i **P4.2**.

Na wstępie przypomnijmy zdefiniowane wcześniej zbiory zadań:

- $\mathbf{E} \triangleq \{j \in \mathbf{J} : S_j < e\}$ - zbiór zadań, które rozpoczynają wykonywać się przed momentem e;
- $\hat{\mathbf{E}} \triangleq \{j \in \mathbf{J} | C_j \leqslant e\}$ - zbiór zadań, które kończą wykonywać się w lub przed momentem e;
- $\mathbf{T} \triangleq \{j \in \mathbf{J} : C_j > d\}$ - zbiór zadań, które kończą wykonywać się po momencie d;
- $\hat{\mathbf{T}} \triangleq \{j \in \mathbf{J} | S_j \geqslant d\}$ - zbiór zadań, które rozpoczynają wykonywać się w lub po momencie d;
- $\mathbf{WI} \triangleq \{j \in \mathbf{J} : S_j \geqslant e \wedge C_j \leqslant d\}$ - zbiór zadań wykonanych całkowicie wewnątrz przedziału $[e, d]$;
- $\mathbf{W} \triangleq \{j \in \mathbf{J} : S_j \leqslant d \wedge C_j \geqslant e\}$ - zbiór zadań wykonanych całkowicie lub częściowo wewnątrz przedziału $[e, d]$.

Powyższe oznaczenia są zilustrowane na Rysunku 8.1

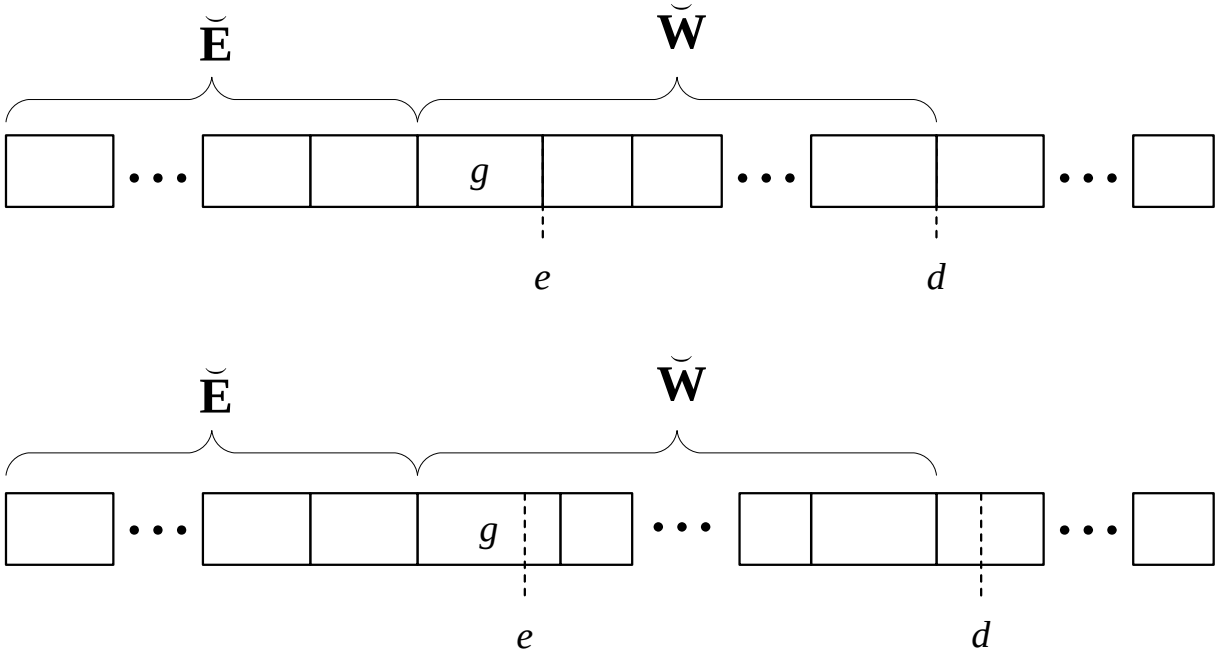


Rysunek 8.1: Przykładowy harmonogram z zaznaczonymi zbiorami zadań.

W celu uproszczenia dalszych rozważań zdefiniujmy dodatkowo następujące oznaczenia:

- g - zadanie, które rozpoczyna wykonywać się przed momentem e, kończy natomiast wykonywać się po lub w momencie e, tzn. $S_g < e$ i $C_g \geqslant e$ (jeżeli $e = 0$, to zadanie g nie istnieje);
- $\breve{\mathbf{E}} \triangleq \hat{\mathbf{E}} \setminus \{g\}$;
- $\breve{\mathbf{W}} \triangleq \mathbf{WI} \cup \{g\}$.

Zdefiniowane oznaczenia zostały zilustrowane na Rysunku 8.2.



Rysunek 8.2: Dwa przykładowe harmonogramy z zaznaczonymi zbiorami $\breve{\mathbf{E}}$ i $\breve{\mathbf{W}}$ oraz zadaniem g.

Własności rozwiązania optymalnego problemu P4.1

Poniżej zostanie wykazanych szereg własności problemu **P4.1**, które posłużą do skonstruowania wielomianowego algorytmu optymalnego.

Własność 8.1. *Istnieje rozwiązanie optymalne problemu* **P4.1***, w którym nie ma okresów bezczynności procesora.*

Dowód tej własności zostanie pominięty, ponieważ byłby on niemal identyczny jak dowód analogicznej Własności 5.1 problemu **P1** $(1|\langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max}|\sum(\alpha E_j + \beta T_j) + \theta e + f_W(d-e))$.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P4.1**, które spełniają Własność 8.1.

Własność 8.2. *Istnieje rozwiązanie optymalne problemu* **P4.1***, w którym pierwsze zadanie rozpoczyna wykonywać się w momencie* 0*.*

Dowód tej własności zostanie pominięty, ponieważ byłby on niemal identyczny jak dowód analogicznej Własności 5.2 problemu **P1** $(1|\langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max}|\sum(\alpha E_j + \beta T_j) + \theta e + f_W(d-e))$.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P4.1**, które spełniają Własność 8.2.

Zauważmy, że dla rozwiązania spełniającego Własności 8.1 oraz 8.2, kolejność wykonania zadań implikuje momenty ich rozpoczęcia i zakończenia. Zatem przedział $[e, d]$ oraz kolejność wykonywania zadań określają precyzyjnie harmonogram dla problemu **P4.1**.

Zapis $\sigma = (\pi, e, d)$ będzie oznaczał harmonogram σ, w którym π określa kolejność wykonywania zadań na procesorze, natomiast e i d to odpowiednio początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

W dowodach Własności 8.3 i 8.4 wykorzystywane będą następujące, zdefiniowane wcześniej oznaczenia:

- $K_E \triangleq \min\{j : C_{\pi(j)} \geqslant e\}$ - pozycja pierwszego zadania, które kończy wykonywać się w lub po momencie e;
- $K_T \triangleq \max\{j : S_{\pi(j)} \leqslant d\}$ - pozycja ostatniego zadania, które rozpoczyna wykonywać się w lub przed momentem d;
- $\Delta_E \triangleq C_{\pi(K_E)} - e$ - część zadania $\pi(K_E)$, która wykonuje się wewnątrz przedziału $[e, d]$;
- $\delta_E \triangleq e - S_{\pi(K_E)}$ - część zadania $\pi(K_E)$, która wykonuje się przed przedziałem $[e, d]$;
- $\Delta_T \triangleq d - S_{\pi(K_T)}$ - część zadania $\pi(K_T)$, która wykonuje się wewnątrz przedziału $[e, d]$;
- $\delta_T \triangleq C_{\pi(K_T)} - d$ - część zadania $\pi(K_T)$, która wykonuje się za przedziałem $[e, d]$.

Własność 8.3. *Istnieje rozwiązanie optymalne problemu* **P4.1***, w którym jedno zadanie kończy lub rozpoczyna wykonywać się w momencie d.*

Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Przyjmijmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. żadne zadanie nie kończy ani nie rozpoczyna wykonywać się w momencie d, czyli $\Delta_T(\sigma) > 0$. Załóżmy w niniejszym dowodzie, że oznaczenie Δ_T będzie odnosić się do harmonogramu σ. Rozważmy dwa możliwe przypadki.

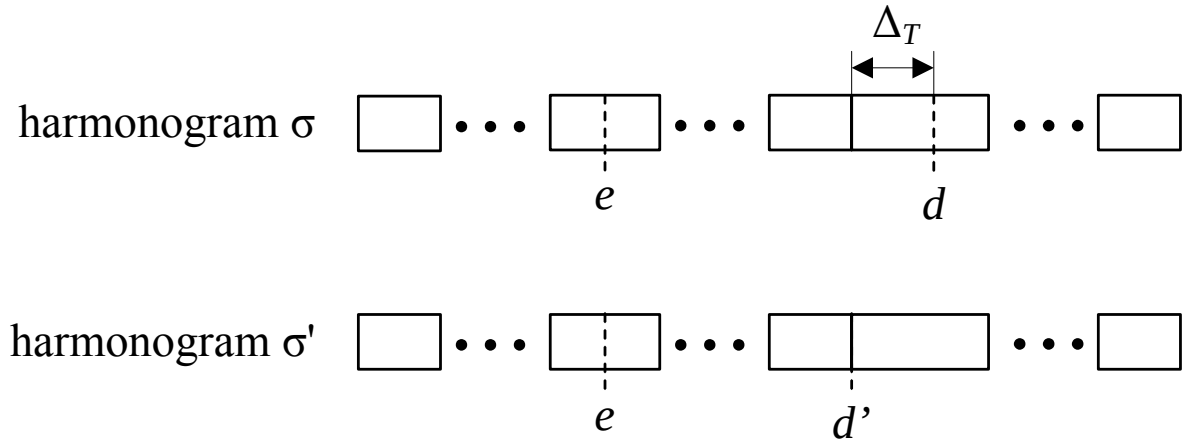
Przypadek 1. Załóżmy, że $d - e \geqslant \Delta_T$.

Wartość kryterium dla harmonogramu σ możemy wyrazić w następujący sposób:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j) + \gamma(d - e) + \theta d = Z_0 + \gamma(d - e) + \theta d,$$

gdzie $Z_0 = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j)$.

Utwórzmy teraz harmonogram $\sigma' = (\pi, e, d')$ z harmonogramu σ taki, że $d' = d - \Delta_T$. W tak powstałym harmonogramie pewne zadanie rozpoczyna wykonywać się w momencie



Rysunek 8.3: Przykładowe harmonogramy σ i σ' przy założeniu, że $d - e \geqslant \Delta_T$.

d' (patrz Rysunek 8.3). Skoro założyliśmy, że warunek $d - e \geqslant \Delta_T$ jest spełniony, to $d' = d - \Delta_T \geqslant e$. Łatwo zauważyć, że $V_j(\sigma) = V_j(\sigma')$ oraz $U_j(\sigma) = U_j(\sigma')$ dla każdego zadania $j \in \mathbf{J}$ (patrz Rysunek 8.3). Zatem wartość funkcji celu dla harmonogramu σ' wynosi:

$$\begin{gathered} Z(\sigma') = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j) + \gamma(d' - e) + \theta d' = \\ Z_0 + \gamma(d - \Delta_T - e) + \theta(d - \Delta_T) = \\ Z_0 + \gamma(d - e) + \theta d - (\gamma + \theta)\Delta_T. \end{gathered}$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma') = [\gamma(d - e) + \theta d] - [\gamma(d - e) + \theta d - (\gamma + \theta)\Delta_T] = (\gamma + \theta)\Delta_T \geqslant 0,$$

a stąd:

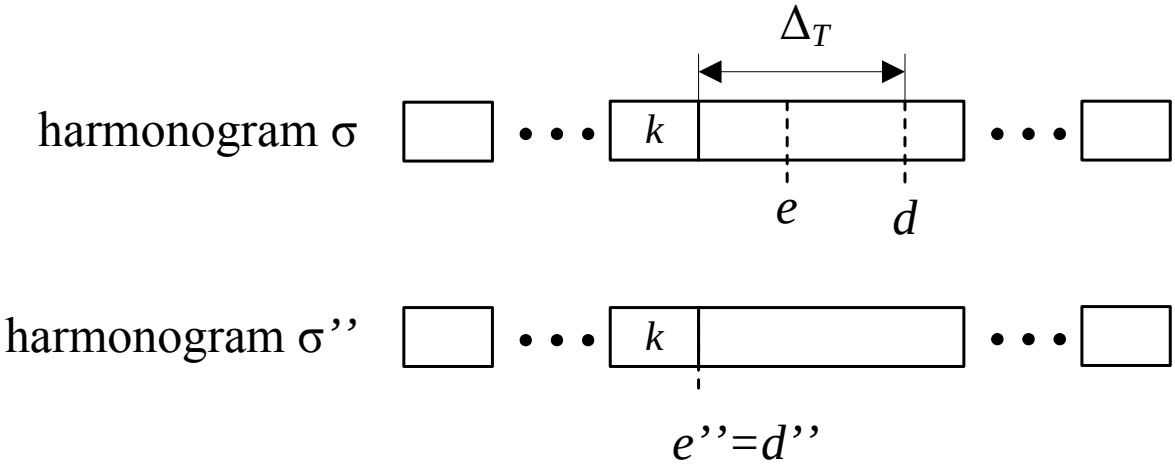
$$Z(\sigma) \geqslant Z(\sigma').$$

Przypadek 2. Załóżmy, że $d - e < \Delta_T$.

Niech k oznacza zadanie wykonujące się na pozycji $K_E - 1$ w harmonogramie σ, czyli $k = \pi(K_E - 1)$. Zauważmy, że $C_k(\sigma) < e$, czyli $V_k(\sigma) = 1$ i $U_k(\sigma) = 0$. Wartość kryterium dla harmonogramu σ możemy wyrazić w następujący sposób:

$$\begin{gathered} Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j) + \gamma(d - e) + \theta d = \\ Z'_0 + \alpha_k + \gamma(d - e) + \theta d, \end{gathered}$$

gdzie $Z'_0 = \sum_{j \in \mathbf{J} \setminus \{k\}} (\alpha_j V_j + \beta_j U_j)$.



Rysunek 8.4: Przykładowe harmonogramy σ i σ'' przy założeniu, że $d - e < \Delta_T$.

Utwórzmy teraz harmonogram $\sigma'' = (\pi, e'', d'')$ z harmonogramu σ taki, że $d'' = e'' = d - \Delta_T$. W tak powstałym harmonogramie pewne zadanie kończy lub rozpoczyna wykonywać się w momencie d'' (patrz Rysunek 8.4). Łatwo zauważyć, że $V_j(\sigma) = V_j(\sigma'')$ oraz $U_j(\sigma) = U_j(\sigma'')$ dla każdego zadania $j \in \mathbf{J} \setminus \{k\}$ (patrz Rysunek 8.4). Ponadto $C_k(\sigma'') = e''$, czyli $V_k(\sigma'') = U_k(\sigma'') = 0$. Zatem wartość funkcji celu dla harmonogramu σ'' wynosi:

$$Z(\sigma'') = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j) + \gamma(d'' - e'') + \theta d'' =$$

$$\sum_{j \in \mathbf{J} \setminus \{k\}} (\alpha_j V_j + \beta_j U_j) + \theta(d - \Delta_T) = Z'_0 + \theta(d - \Delta_T) = Z'_0 + \theta d - \theta \Delta_T.$$

Mamy zatem:

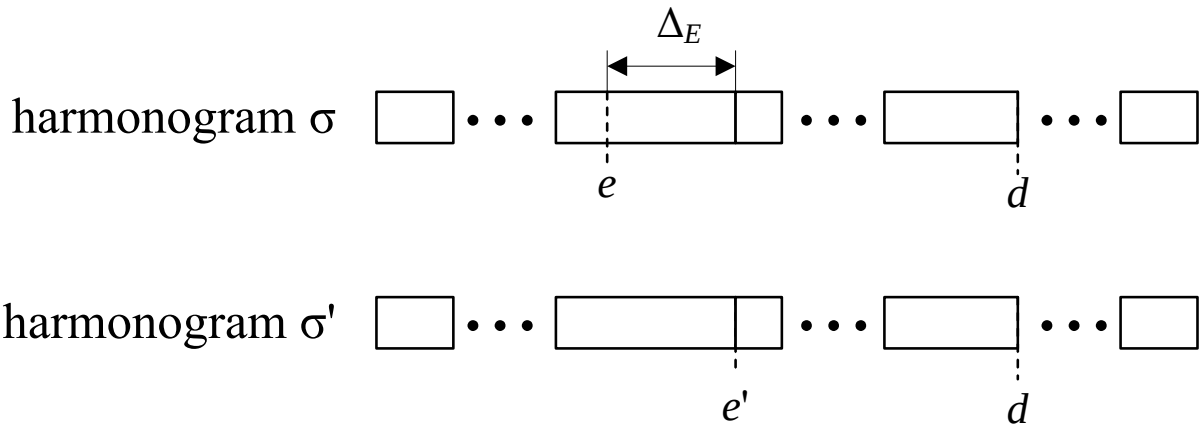
$$Z(\sigma) - Z(\sigma'') = [\alpha_k + \gamma(d - e) + \theta d] - [\theta d - \theta \Delta_T] = \alpha_k + \gamma(d - e) + \theta \Delta_T \geqslant 0,$$

a stąd

$$Z(\sigma) \geqslant Z(\sigma'').$$

Możemy zatem uzyskać harmonogram spełniający dowodzoną własność bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P4.1**, które spełniają Własność 8.3.



Rysunek 8.5: Przykładowe harmonogramy σ i σ' dla przypadku gdy $d - e \geqslant \Delta_E$.

Własność 8.4. *Istnieje rozwiązanie optymalne problemu* **P4.1***, w którym jedno zadanie kończy lub rozpoczyna wykonywać się w momencie e.*

Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Przyjmijmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. żadne zadanie nie kończy ani nie rozpoczyna wykonywać się w momencie e, czyli $\Delta_E(\sigma) > 0$. Załóżmy w niniejszym dowodzie, że oznaczenie Δ_E odnosi się do harmonogramu σ. Wartość kryterium dla harmonogramu σ wynosi:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j) + \gamma(d - e) + \theta d = Z_0 + \gamma(d - e),$$

gdzie $Z_0 = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j) + \theta d$.

Utwórzmy teraz harmonogram $\sigma' = (\pi, e', d)$ z harmonogramu σ taki, że $e' = e + \Delta_E$. Zgodnie z Własnością 8.3 pewne zadanie kończy wykonywać się w momencie d, co gwarantuje nam, że $e' \leqslant d$. W tak powstałym harmonogramie pewne zadanie kończy wykonywać się w momencie e' (patrz Rysunek 8.5). Łatwo zauważyć, że $V_j(\sigma) = V_j(\sigma')$ oraz $U_j(\sigma) = U_j(\sigma')$ dla każdego zadania $j \in \mathbf{J}$ (patrz Rysunek 8.5). Zatem wartość funkcji celu dla harmonogramu σ' wynosi:

$$\begin{aligned} Z(\sigma') = \sum_{j \in \mathbf{J}} (\alpha_j V_j + \beta_j U_j) + \gamma(d - e') + \theta d = Z_0 + \gamma(d - e') = \\ Z_0 + \gamma(d - e - \Delta_E) = Z_0 + \gamma(d - e) - \gamma\Delta_E. \end{aligned}$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma') = \gamma(d - e) - [\gamma(d - e) - \gamma\Delta_E] = \gamma\Delta_E \geqslant 0,$$

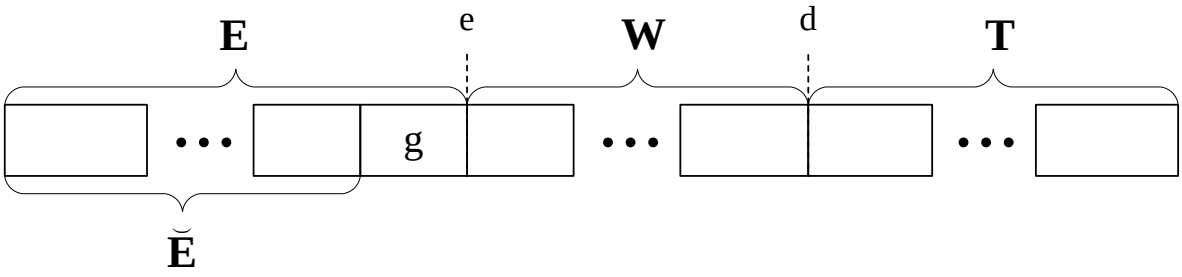
a stąd:

$$Z(\sigma) \geqslant Z(\sigma').$$

Ostatecznie możemy stwierdzić, że przesunięcie początku przedziału $[e, d]$ w prawo o Δ_E nie spowoduje wzrostu wartości funkcji celu. Możemy zatem uzyskać harmonogram, w którym pewne zadanie kończy lub rozpoczyna wykonywać się w momencie e, bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P4.1**, które spełniają Własność 8.4.

Zauważmy, że skoro pewne zadanie kończy lub rozpoczyna wykonywać się w momencie e oraz pewne zadanie kończy lub rozpoczyna wykonywać się w momencie d, to $\hat{\mathbf{T}} = \mathbf{T}$ oraz $\mathbf{WI} = \mathbf{W}$. Zadanie g natomiast kończy wykonywać się w momencie e.



Rysunek 8.6: Przykładowy harmonogram z zaznaczonymi zbiorami **E**, **T**, **W** i $\check{\mathbf{E}}$ oraz zadaniem g.

Z Własności 8.3 i 8.4 wynika, że wartości parametrów e i d możemy wyrazić w sposób następujący (patrz Rysunek 8.6):

$$e = \sum_{j \in \mathbf{E}} p_j;$$

$$d = \sum_{j \in \mathbf{E} \cup \mathbf{W}} p_j.$$

W związku z tym, wartość funkcji celu dla harmonogramu σ wynosi:

$$Z(\sigma) = \sum(\alpha_j V_j + \beta_j U_j) + \gamma(d - e) + \theta d =$$

$$\sum_{j \in \check{\mathbf{E}}} \alpha_j + \sum_{j \in \mathbf{T}} \beta_j + \gamma \left(\sum_{j \in \mathbf{E} \cup \mathbf{W}} p_j - \sum_{j \in \mathbf{E}} p_j \right) + \theta \sum_{j \in \mathbf{E} \cup \mathbf{W}} p_j =$$

$$\sum_{j \in \check{\mathbf{E}}} \alpha_j + \sum_{j \in \mathbf{T}} \beta_j + \gamma \sum_{j \in \mathbf{W}} p_j + \theta \sum_{j \in \check{\mathbf{E}}} p_j + \theta p_g + \theta \sum_{j \in \mathbf{W}} p_j =$$

$$\sum_{j \in \check{\mathbf{E}}} (\alpha_j + \theta p_j) + \theta p_g + \sum_{j \in \mathbf{W}} (\gamma + \theta) p_j + \sum_{j \in \mathbf{T}} \beta_j.$$

Na podstawie powyższego wyrażenia możemy powiedzieć jaki koszt wnosi każde zada-

nie j do całkowitego kosztu uszeregowania σ. Oznaczmy ten koszt przez $cost_j(\sigma)$.

Możemy teraz wartość funkcji celu wyrazić w sposób następujący:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} cost_j(\sigma), \tag{8.1}$$

gdzie

$$cost_j(\sigma) \triangleq \begin{cases} \alpha_j + \theta p_j, & \text{jeżeli } j \in \breve{\mathbf{E}}; \\ \theta p_j, & \text{jeżeli } j = g; \\ (\theta + \gamma) p_j, & \text{jeżeli } j \in \mathbf{W}; \\ \beta_j, & \text{jeżeli } j \in \mathbf{T}. \end{cases}$$

Własność 8.5. *W rozwiązaniu optymalnym problemu* **P4.1**:

- *zadania ze zbioru* $\breve{\mathbf{E}}$ *są wykonywane w dowolnej kolejności;*
- *zadania ze zbioru* $\mathbf{W}$ *są wykonywane w dowolnej kolejności;*
- *zadania ze zbioru* $\mathbf{T}$ *są wykonywane w dowolnej kolejności.*

Dowód. Prawdziwość dowodzonej własności wynika bezpośrednio z faktu, że kolejność wykonywania zadań ze zbioru $\breve{\mathbf{E}}$ ani $\mathbf{W}$, ani $\mathbf{T}$ nie wpływa na wartość funkcji celu. □

Własność 8.6. *W rozwiązaniu optymalnym problemu* **P4.1***:*

- *jeżeli* $\alpha_j + \theta p_j < \min\{(\theta + \gamma) p_j, \beta_j\}$*, to zadnie* j *nie należy do zbioru* $\mathbf{W} \cup \mathbf{T}$*;*
- *jeżeli* $(\theta + \gamma) p_j < \min\{\alpha_j + \theta p_j, \beta_j\}$*, to zadnie* j *nie należy do zbioru* $\breve{\mathbf{E}} \cup \mathbf{T}$*;*
- *jeżeli* $\beta_j < \min\{\alpha_j + \theta p_j, (\theta + \gamma) p_j\}$*, to zadnie* j *nie należy do zbioru* $\mathbf{W} \cup \breve{\mathbf{E}}$*.*

Dowód. Zauważmy, że ilość zadań w zbiorach $\breve{\mathbf{E}}$, $\mathbf{W}$ i $\mathbf{T}$ nie jest w żaden sposób ograniczona. W związku z tym prawdziwość dowodzonej własności wynika bezpośrednio z wyrażenia (8.1). □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P4.1**, które spełniają Własności 8.5 i 8.6.

Łatwo zauważyć, że dla rozwiązania σ spełniającego powyższą własność mamy:

$$cost_j(\sigma) = \begin{cases} \theta, & \text{jeżeli } j = g; \\ \min\{\alpha_j + \theta p_j, (\theta + \gamma) p_j, \beta_j\}, & \text{jeżeli } j \neq g. \end{cases}$$

Możemy zatem wartość kryterium zapisać w następującej postaci:

$$Z(\sigma) = \sum_{j \in \mathbf{J} \setminus \{g\}} \min\{\alpha_j + \theta, \theta + \gamma, \beta_j\} + \theta p_g =$$
$$\sum_{j \in \mathbf{J}} \min\{\alpha_j + \theta p_j, (\theta + \gamma) p_j, \beta_j\} + (\theta p_g - \min\{\alpha_g + \theta p_g, (\theta + \gamma) p_g, \beta_g\}) =$$
$$\sum_{j \in \mathbf{J}} \min\{\alpha_j + \theta p_j, (\theta + \gamma) p_j, \beta_j\} - \min\{\alpha_g, \gamma p_g, \beta_g - \theta p_g\}.$$

Łatwo zauważyć, że wartość wyrażenia $\sum_{j \in \mathbf{J}} \min\{\theta p_j + \alpha_j, (\theta + \gamma) p_j, \beta_j\}$ jest stała dla danej instancji. Wynika z tego, że w celu minimalizacji kryterium należy znaleźć takie zadanie $j \in \mathbf{J}$, które maksymalizuje wartość wyrażenia $\min\{\alpha_j, \gamma p_j, \beta_j - \theta p_j\}$.

Zdefiniujmy:

- $v(j) \triangleq \min\{\alpha_j, \gamma p_j, \beta_j - \theta p_j\}$;
- $v_{max} \triangleq \max_j v(j)$.

Z powyższych rozważań wynika, że w rozwiązaniu optymalnym nierówność $v(g) \geqslant v(j)$ jest spełniona dla wszystkich zadań $j \in \mathbf{J}$. Przypomnijmy, że jeżeli $e = 0$, to zadanie g nie istnieje.

Skoro parametry α_j, p_j i γ przyjmują jedynie wartości nieujemne, to $v(j) < 0$ $(j \in \mathbf{J})$ wtedy i tylko wtedy, gdy $\beta_j < \theta p_j$. Z Własności 8.6 wynika zatem, że jeżeli $v_{max} < 0$, to żadne zadanie nie należy do zbioru $\mathbf{E} \cup \mathbf{W}$, a w konsekwencji $e = d = 0$.

Z powyższych rozważań wynika bezpośrednio prawdziwość Własności 8.7.

Własność 8.7. *Istnieje rozwiązanie optymalne problemu* **P4.1***, w którym*

- *jeżeli* $v_{max} < 0$*, to* $e = d = 0$*;*
- *jeżeli* $v_{max} \geqslant 0$*, to zadanie* k *takie, że* $v(k) = v_{max}$ *kończy wykonywać się w momencie* e *(czyli* $g = k$*).*

Własności rozwiązania optymalnego problemu P4.2

Poniżej przedstawiono własności problemu **P4.2**, których prawdziwość została wykazana przez Yeung i in. [103].

Własność 8.8. [103] *Istnieje rozwiązanie optymalne problemu* **P4.2***, w którym*
(i) pierwsze zadanie rozpoczyna wykonywać się w momencie 0*;*
(ii) zadania są wykonywane bez przestojów pomiędzy nimi.

Własność 8.9. [103] *Istnieje rozwiązanie optymalne problemu* **P4.2***, w którym*
(i) jedno zadanie kończy wykonywać się w momencie d *lub*
(ii) $e = 0$*.*

Własność 8.10. [103] *Istnieje rozwiązanie optymalne problemu* **P4.2**, *w którym*

- *zadania ze zbioru* $\breve{\mathbf{E}}$ *są wykonywane w dowolnej kolejności;*
- *zadania ze zbioru* $\mathbf{WI}$ *są wykonywane w dowolnej kolejności;*
- *zadania ze zbioru* $\hat{\mathbf{T}}$ *są wykonywane w dowolnej kolejności.*

Własność 8.11. [103] *W rozwiązaniu optymalnym problemu* **P4.2**

- *jeżeli* $\alpha_j + \gamma p_j < \beta_j$, *to zadanie* j *nie należy do zbioru* $\hat{\mathbf{T}}$;
- *jeżeli* $\alpha_j + \gamma p_j > \beta_j$, *to zadanie* j *nie należy do zbioru* $\breve{\mathbf{E}}$.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P4.2**, które spełniają Własności 8.8 - 8.11.

W pracy [103] zaprezentowano algorytm programowania dynamicznego o złożoności obliczeniowej $O(n^2 D p_{max})$ rozwiązujący optymalnie problem **P4.2**. W dalszej części rozdziału zostanie wykazane, że problem **P4.2** można rozwiązać w czasie $O(n(D+p_{max}))$. Do skonstruowania szybszego algorytmu potrzebna jest jeszcze jedna własność rozwiązania optymalnego, która zostanie przedstawiona poniżej.

Własność 8.12. *Istnieje rozwiązanie optymalne problemu* **P4.2**, *w którym zadanie* g *ma najdłuższy czas wykonywania spośród zadań ze zbioru* $\breve{\mathbf{W}}$.

Dowód. Załóżmy, że znane jest rozwiązanie optymalne σ. Załóżmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. istnieje co najmniej jedno takie zadanie $j \in \breve{\mathbf{W}}$, że $p_j > p_g$. Skoro $\breve{\mathbf{W}} = \mathbf{WI} \cup \{g\}$, to zadanie j należy do zbioru $\mathbf{WI}$. Bez utraty ogólności możemy założyć, że zadanie j wykonuje się bezpośrednio po zadaniu g. Załóżmy w niniejszym dowodzie, że oznaczenie g odnosi się do harmonogramu σ. Zauważmy, że $e \leqslant C_j(\sigma) \leqslant d$ oraz $e \leqslant C_g(\sigma) \leqslant d$. Wynika z tego, że $V_j(\sigma) = U_j(\sigma) = 0$ i $V_g(\sigma) = U_g(\sigma) = 0$.

Utwórzmy harmonogram σ' z harmonogramu σ taki, że zadania j i g wykonywane są w odwrotnej kolejności. Zauważmy, że po takiej zamianie, terminy rozpoczęcia i zakończenia wykonywania pozostałych zadań nie zmienią się. Ponadto, łatwo zauważyć, że spełnione są następujące zależności $e \leqslant C_j(\sigma) \leqslant d$ oraz $e \leqslant C_g(\sigma) \leqslant d$. Wynika z tego, że $V_j(\sigma') = U_j(\sigma') = 0$ i $V_g(\sigma') = U_g(\sigma') = 0$, a w konsekwencji $V_k(\sigma') = V_k(\sigma)$ oraz $U_k(\sigma') = U_k(\sigma)$ dla każdego zadania $k \in \mathbf{J}$. Mamy zatem:

$$Z(\sigma) = Z(\sigma').$$

Z powyższego wynika, że zamiana kolejności wykonywania zadań j i g nie zwiększy wartości funkcji celu. Możemy zatem uzyskać harmonogram, w którym zadanie g będzie miało najdłuższy czas wykonywania spośród zadań ze zbioru $\breve{\mathbf{W}}$, bez wzrostu wartości kryterium. □

8.4 Algorytmy optymalne

W niniejszym podrozdziale zostaną zaprezentowane algorytmy optymalne dla problemów **P4.1** i **P4.2**.

Algorytm optymalny dla problemu P4.1

Poniżej przedstawiony jest algorytm **A4.1**, który rozwiązuje optymalnie problem **P4.1**. Algorytm ten działa w oparciu o Własności 8.1 - 8.7.

Algorytm A4.1

Krok 1. Wylicz wartości $v(j) := \min\{\alpha_j, \gamma p_j, \beta_j - \theta p_j\}$ $(j = 1, 2, \ldots, n)$ oraz $v_{max} = \max_j v(j)$.

- Jeżeli $v_{max} < 0$, to wylicz parametry przedziału $[e, d]$ w sposób następujący: $e = d = 0$, uszereguj zadania w dowolnym porządku i zakończ działanie algorytmu.
- Jeżeli $v_{max} \geqslant 0$, to uszereguj zadanie j takie, że $v(j) = v_{max}$, jako zadanie g.

Krok 2. Uszereguj pozostałe zadania w sposób następujący:

- Jeżeli $\alpha_j + \theta p_j < \min\{\beta_j, (\theta + \gamma)p_j\}$, to przydziel zadanie j do zbioru $\breve{\mathbf{E}}$.
- Jeżeli $(\theta + \gamma)p_j < \min\{\alpha_j + \theta p_j, \beta_j\}$, to przydziel zadanie j do zbioru $\mathbf{W}$.
- Jeżeli $\beta_j < \min\{\alpha_j + \theta, \theta + \gamma\}$, to przydziel zadanie j do zbioru $\mathbf{T}$.

Krok 3. Wylicz parametry przedziału $[e, d]$ w sposób następujący: $e = \sum_{j \in \mathbf{E}} p_j$, $d = e + \sum_{j \in \mathbf{W}} p_j$.

Twierdzenie 9. *Algorytm* **A4.1** *rozwiązuje optymalnie problem* **P4.1** *w czasie* $O(n)$.

Dowód. Optymalność algorytmu **A4.1** wynika bezpośrednio z Własności 8.1-8.7.

Łatwo zauważyć, że każdy z trzech kroków algorytmu wymaga czasu $O(n)$. Zatem całkowita złożoność obliczeniowa algorytmu jest równa $O(n)$. □

Algorytm optymalny dla problemu P4.2

Poniżej omówiony zostanie algorytm optymalny dla problemu **P4.2**.

Załóżmy, że zadania są ponumerowane według niemalejących czasów wykonywania, tzn. $p_1 \leqslant \ldots \leqslant p_n$.

Rozbijmy problem **P4.2** na dwa następujące podproblemy:

- **P4.2.1** : Zakładamy, że w rozwiązaniu optymalnym tego podproblemu $e = 0$.
- **P4.2.2** : Zakładamy, że w rozwiązaniu optymalnym tego podproblemu pewne zadanie kończy wykonywać się w momencie d.

Z Własności 8.9 wynika, że w celu rozwiązania problemu **P4.2** wystarczy rozwiązać problemy **P4.2.1** i **P4.2.2** a następnie wybrać rozwiązanie o minimalnej wartości kryterium.

Własność 8.13. *Problemy* **P4.2.1** *i* $1|d_j = \hat{d}|\sum \beta_j U_j$ *są sobie równoważne w tym sensie, że rozwiązanie optymalne jednego z nich jest rozwiązaniem optymalnym drugiego a wartość kryterium różni się jedynie o stałą.*

Dowód. Skoro $e = 0$ w rozwiązaniu optymalnym problemu **P4.2.1**, to $d = D$ a $V_j = 0$ dla każdego zadania $j \in \mathbf{J}$. Wynika z tego, że w problemie **P4.2.1** należy znaleźć takie uszeregowanie zadań (kolejność wykonywania zadań), które minimalizuje następujące kryterium:

$$\sum_{j \in \mathbf{J}} \beta_j U_j + \theta D.$$

Z kolei w problemie $1|d_j = \hat{d}|\sum \beta_j U_j$ należy znaleźć takie uszeregowanie zadań, które minimalizuje wartość następującego kryterium:

$$\sum_{j \in \mathbf{J}} \beta_j U_j.$$

Skoro wartość θD jest wartością stałą dla danej instancji, to optymalne uszeregowanie problemu **P4.2.1** będzie również optymalnym uszeregowaniem dla problemu $1|d_j = \hat{d}|\sum \beta_j U_j$ (przy założeniu, że $\hat{d} = D$) i na odwrót, natomiast wartości ich kryteriów różnią się jedynie o stałą. □

Z powyższej własności wynika, że optymalny algorytm o złożoności $O(nD)$ dla problemu $1|d_j = \hat{d}|\sum \alpha_j U_j$ [79] może zostać zastosowany do rozwiązania problemu **P4.2.1**.

Zajmijmy się teraz problemem **P4.2.2**.

Skoro pewne zadanie kończy wykonywać się w momencie d, to wartość parametru d możemy wyrazić w sposób następujący:

$$d = \sum_{j \in \check{\mathbf{E}} \cup \check{\mathbf{W}}} p_j.$$

Wartość funkcji celu dla problemu **P4.2.2** wynosi zatem:

$$Z(\sigma) = \sum_{j\in\mathbf{J}}(\alpha_j V_j + \beta_j U_j) + \theta d = \sum_{j\in\check{\mathbf{E}}} \alpha_j + \sum_{j\in\hat{\mathbf{T}}} \beta_j + \theta \sum_{j\in\check{\mathbf{E}}\cup\check{\mathbf{W}}} p_j = \sum_{j\in\check{\mathbf{E}}}(\alpha_j + \theta p_j) + \sum_{j\in\check{\mathbf{W}}} \theta p_j + \sum_{j\in\hat{\mathbf{T}}} \beta_j.$$

Uwzględniając Własność 8.11, otrzymujemy:

$$Z(\sigma) = \sum_{j\in\check{\mathbf{E}}\cup\hat{\mathbf{T}}} \min\{\alpha_j + \theta p_j, \beta_j\} + \sum_{j\in\check{\mathbf{W}}} \theta p_j.$$

W celu skrócenia notacji, zdefiniujmy:

- $\psi_j \triangleq \min\{\alpha_j + \theta p_j, \beta_j\}$.

Teraz wartość funkcji celu możemy wyrazić następująco:

$$Z(\sigma) = \sum_{j\in\check{\mathbf{E}}\cup\hat{\mathbf{T}}} \psi_j + \sum_{j\in\check{\mathbf{W}}} \theta p_j. \tag{8.2}$$

Zauważmy, że ilość zadań w zbiorach $\check{\mathbf{E}}$ oraz $\hat{\mathbf{T}}$ nie jest w żaden sposób ograniczona, natomiast w przypadku zbioru $\check{\mathbf{W}}$ musi być spełniona następująca nierówność:

$$\sum_{j\in\mathbf{WI}} p_j = \sum_{j\in\check{\mathbf{W}}} p_j - p_g \leqslant D.$$

Z Własności 8.12 wynika, że $p_g = \max\{p_j : j \in \check{\mathbf{W}}\}$. Możemy zatem powyższą nierówność zapisać w sposób następujący:

$$\sum_{j\in\check{\mathbf{W}}} p_j - \max\{p_j : j \in \check{\mathbf{W}}\} \leqslant D. \tag{8.3}$$

Poniżej opisano działanie Algorytmu $\mathbf{A4.2.2}_1$, który rozwiązuje optymalnie problem **P4.2.2**. Rozpoczynamy od harmonogramu zerowego σ_0, czyli takiego, który nie zawiera żadnego zadania. Następnie generujemy kolejne harmonogramy cząstkowe σ_k (harmonogramy zawierające k pierwszych zadań) z harmonogramów σ_{k-1} poprzez przyporządkowanie zadania k do zbioru $\check{\mathbf{E}} \cup \hat{\mathbf{T}}$ lub do zbioru $\check{\mathbf{W}}$.

Będziemy mówić, że harmonogram cząstkowy jest w stanie (k, w), jeżeli zawiera pierwszych k zadań, a suma czasów wykonywania zadań ze zbioru $\check{\mathbf{W}}$ wynosi w.

Niech $F_k(w)$ oznacza minimalną wartość funkcji celu dla harmonogramów znajdujących się w stanie (k, w). Łatwo zauważyć, że wartość funkcji celu dla rozwiązania optymalnego jest równa:

$$\min\{F_n(w) : w = 0, 1, \ldots, D + p_n\}.$$

Zauważmy, że zgodnie z definicją funkcji $F_k(w)$, mamy $F_0(0) = 0$.

Załóżmy, że mamy harmonogram cząstkowy znajdujący się w stanie $(k-1, w)$. Jeżeli zadanie k przyporządkujemy do zbioru $\check{\mathbf{E}} \cup \hat{\mathbf{T}}$, to uzyskamy harmonogram w stanie (k, w). W rezultacie wartość kryterium wzrośnie o ψ_k, czyli $F_k(w) = F_{k-1}(w) + \psi_k$; patrz wyrażenie (8.2). Jeżeli natomiast zadanie k przyporządkujemy do zbioru $\check{\mathbf{W}}$, to uzyskamy harmonogram w stanie $(k, w + p_k)$. Wartość funkcji celu wzrośnie o θp_k, czyli $F_k(w + p_k) = F_{k-1}(w) + \theta p_k$. Zadanie k możemy przyporządkować do zbioru $\check{\mathbf{W}}$, tylko jeżeli uzyskany harmonogram będzie spełniał warunek (8.3), czyli $w \leqslant D$.

Z powyższego wynika, że harmonogram w stanie (k, w) możemy uzyskać:

- z harmonogramu w stanie $(k-1, w)$ poprzez przyporządkowanie zadania k do zbioru $\check{\mathbf{E}} \cup \hat{\mathbf{T}}$. W tym przypadku mamy $F_k(w) = F_{k-1}(w) + \psi_k p_k$.
- z harmonogramu w stanie $(k-1, w-p_k)$ poprzez przyporządkowanie zadania k do zbioru $\check{\mathbf{W}}$. W tym przypadku mamy $F_k(w) = F_{k-1}(w-p_k) + \theta p_k$. Zadanie k możemy przyporządkować do zbioru $\check{\mathbf{W}}$ tylko jeżeli $w - p_k \leqslant D$.

Formalny opis algorytmu $\mathbf{A4.2.2}_1$ jest podany poniżej.

Algorytm $\mathbf{A4.2.2}_1$

Krok 1. (Inicjalizacja). Ponumeruj zadania według niemalejących czasów ich wykonywania czyli tak, że $p_1 \leqslant \ldots \leqslant p_n$. Podstaw

$$F_0(w) := \begin{cases} 0, & \text{jeżeli } w = 0; \\ +\infty, & \text{w przeciwnym przypadku.} \end{cases}$$

Następnie podstaw $k := 1$.

Krok 2. (Rekurencja). Dla każdego $w = 0, \ldots, D + p_n$, wyznacz:

$$F_k(w) := \min \begin{cases} F_{k-1}(w) + \psi_k; \\ F_{k-1}(w - p_k) + \theta p_k, & \text{jeżeli } w - p_k \leqslant D. \end{cases}$$

Jeżeli $k = n$, to idź do Kroku 3, w przeciwnym przypadku podstaw $k := k + 1$ i powtórz Krok 2.

Krok 3. (Rozwiązanie optymalne). Wyznacz optymalną wartość kryterium:

$$F^* := \min \{F_n(w) : w = 0, \ldots, D + p_n\}$$

oraz skonstruuj odpowiadające rozwiązanie optymalne metodą przeglądu wstecznego (ang. backtracking).

Lemat 7. *Algorytm* $\mathbf{A4.2.2}_1$ *rozwiązuje optymalnie problem* **P4.2.2** *w czasie* $O(n(D + p_{max}))$.

Dowód. Złożoność obliczeniowa algorytmu wynika bezpośrednio z rozmiaru przestrzeni stanów. Skoro $k = 1, 2, \ldots, n$ a $w = 0, 1, \ldots, D + p_{max}$, to złożoność obliczeniowa algorytmu jest równa: $O(n(D + p_{max}))$.

Rozważmy parę harmonogramów cząstkowych znajdujących się w tym samym stanie. Harmonogram o mniejszej wartości kryterium będzie nadal miał mniejszą wartość kryterium po rozszerzeniu go o nie uszeregowane zadania w ten sam sposób, co drugi harmonogram będący w tym samym stanie. Wynika z tego, że tylko harmonogram o minimalnej wartości kryterium spośród harmonogramów będących w tym samym stanie należy wybrać do rozszerzania o kolejne nie uszeregowane zadania.

Optymalność algorytmu $\mathbf{A4.2.2}_1$ wynika bezpośrednio z powyższych rozważań oraz optymalności ogólnej techniki programowania dynamicznego [12]. □

Algorytm optymalny dla problemu P4.2

Poniżej przedstawiony został algorytm rozwiązujący optymalnie problem **P4.2**.

Algorytm A4.2

Krok 1. Rozwiąż optymalnie problem **P4.2.1** przy pomocy algorytmu optymalnego dla problemu $1|d_j = \hat{d}|\sum \beta_j U_j$. Otrzymane rozwiązanie oznaczmy przez σ^1.

Krok 2. Rozwiąż optymalnie problem **P4.2.2** przy pomocy algorytmu $\mathbf{A4.2.2}_1$. Otrzymane rozwiązanie oznaczmy przez σ^2.

Krok 3. Jeżeli $Z(\sigma^1) < Z(\sigma^2)$, to harmonogram σ^1 jest optymalnym rozwiązaniem problemu **P4.2**, a w przeciwnym przypadku harmonogram σ^2 jest optymalnym rozwiązaniem problemu **P4.2**.

Z Lematu 7 oraz Własności 8.13 wynika prawdziwość Twierdzenia 10.

Twierdzenie 10. *Algorytm* **A4.2** *rozwiązuje optymalnie problem* **P4.2** *w czasie* $O(nD + p_{max})$.

Alternatywny algorytm optymalny dla problemu P4.2.2

Poniżej zostanie zaprezentowany drugi algorytm programowania dynamicznego dla problemu **P4.2.2**, który będzie później wykorzystany przy konstrukcji w pełni wielomianowego schematu aproksymacyjnego.

Niech UB oznacza górne oszacowanie wartości funkcji celu, czyli $UB \geqslant Z(\sigma^*)$, gdzie σ^* oznacza rozwiązanie optymalne. Poniżej opisano działanie algorytmu $\mathbf{A4.2.2}_2$, który rozwiązuje optymalnie problem **P4.2.2** w czasie $O(nUB)$.

Rozpoczynamy od harmonogramu zerowego σ_0, czyli takiego, który nie zawiera żadnego zadania. Następnie generujemy kolejne harmonogramy cząstkowe σ_k (harmonogramy zawierające k pierwszych zadań) z harmonogramów σ_{k-1} poprzez przyporządkowanie zadania k do zbioru $\breve{\mathbf{E}} \cup \hat{\mathbf{T}}$ lub do zbioru $\breve{\mathbf{W}}$.

Będziemy mówić, że harmonogram cząstkowy jest w stanie (k, v) jeżeli zawiera pierwszych k zadań a wartość kryterium jest równa v.

Niech $P_k(v)$ oznacza minimalną wartość sumy czasów wykonywania zadań ze zbioru $\breve{\mathbf{W}}$ dla harmonogramów znajdujących się w stanie (k, v). Jeżeli żaden harmonogram nie znajduje się w stanie (k, v), to $P_k(v) = +\infty$. Łatwo zauważyć, że wartość funkcji celu dla rozwiązania optymalnego jest równa:

$$\min \{v : P_n(v) < +\infty; v = 0, 1, \ldots, UB\}.$$

Zauważmy, że zgodnie z definicją funkcji $P_k(v)$, mamy $P_0(0) = 0$.

Załóżmy, że mamy harmonogram cząstkowy znajdujący się w stanie $(k-1, v)$. Jeżeli zadanie k przyporządkujemy do zbioru $\breve{\mathbf{E}} \cup \hat{\mathbf{T}}$, to uzyskamy harmonogram w stanie $(k, v + \psi_k)$; patrz wyrażenie (8.2). Nie spowoduje to wzrostu wartości sumy czasów wykonywania zadań ze zbioru $\breve{\mathbf{W}}$, więc $P_k(v + \psi_k) = P_{k-1}(v)$. Jeżeli natomiast zadanie k przyporządkujemy do zbioru $\breve{\mathbf{W}}$, to uzyskamy harmonogram w stanie $(k, v + \theta p_k)$. Natomiast wartość sumy czasów wykonywania zadań ze zbioru $\breve{\mathbf{W}}$ wzrośnie o θp_k, czyli $P_k(v+\theta p_k) = P_{k-1}(v)+p_k$. Zadanie k możemy przyporządkować do zbioru $\breve{\mathbf{W}}$, tylko jeżeli $P_k(v+\theta p_k)-p_k = P_{k-1}(v) \leqslant D$ (przypomnijmy, że $p_k \geqslant p_j$ dla każdego $j = 1, \ldots, k-1$). W przeciwnym przypadku uzyskane rozwiązanie nie spełniałoby warunku (8.3).

Z powyższego wynika, że harmonogram w stanie (k, v) możemy uzyskać:

- z harmonogramu w stanie $(k-1, v-\psi_k)$ poprzez przyporządkowanie zadania k do zbioru $\breve{\mathbf{E}} \cup \hat{\mathbf{T}}$. W tym przypadku mamy $P_k(v) = P_{k-1}(v-\psi_k)$.

- z harmonogramu w stanie $(k-1, v-\theta p_k)$ poprzez przyporządkowanie zadania k do zbioru $\breve{\mathbf{W}}$. W tym przypadku mamy $P_k(v) = P_{k-1}(v-\theta p_k)+p_k$. Zadanie k możemy przyporządkować do zbioru $\breve{\mathbf{W}}$ tylko jeżeli $P_k(v) - p_k = P_{k-1}(v-\theta p_k) \leqslant D$.

Formalny opis algorytmu $\mathbf{A4.2.2}_2$ jest podany poniżej.

Algorytm A4.2.2$_2$

Krok 1. (Inicjalizacja). Ponumeruj zadania według niemalejących czasów ich wykonywania czyli tak, że $p_1 \leqslant \ldots \leqslant p_n$. Podstaw

$$P_k(v) := \begin{cases} 0, & \text{jeżeli } v = 0 \text{ i } k = 0; \\ +\infty, & \text{w przeciwnym przypadku.} \end{cases}$$

Następnie podstaw $k := 1$.

Krok 2. (Rekursja). Dla każdego $v = 0, \ldots, UB$, wyznacz:

$$P_k(v) := \min \begin{cases} P_{k-1}(v - \psi_k); \\ P_{k-1}(v - \theta p_k) + p_k, \quad \text{jeżeli } P_{k-1}(v - \theta p_k) \leqslant D. \end{cases}$$

Jeżeli $k = n$, to idź do Kroku 3, w przeciwnym przypadku podstaw $k := k + 1$ i powtórz Krok 2.

Krok 3. (Rozwiązanie optymalne). Wyznacz optymalną wartość kryterium:

$$\min \{v : P_n(v) < +\infty; v = 0, 1, \ldots, UB\}$$

oraz skonstruuj odpowiadające rozwiązanie optymalne σ^* metodą przeglądu wstecznego (ang. backtracking).

Lemat 8. *Algorytm* **A4.2.2**$_2$ *rozwiązuje optymalnie problem* **P4.2.2** *w czasie* $O(nUB)$.

Dowód. Złożoność obliczeniowa algorytmu wynika bezpośrednio z rozmiaru przestrzeni stanów. Skoro $k = 1, 2, \ldots, n$ a $v = 0, 1, \ldots, UB$, to złożoność obliczeniowa algorytmu jest równa: $O(nUB)$.

Optymalności algorytmu **A4.2.2**$_1$ można dowieść używając tej samej argumentacji, która została użyta w dowodzie Lematu 7. □

8.5 W pełni wielomianowy schemat aproksymacyjny

W niniejszym podrozdziale zostanie zaprezentowany w pełni wielomianowy schemat aproksymacyjny dla problemu **P4.2**.

Przypomnijmy najpierw pewne definicje związane ze schematami aproksymacyjnymi.

Niech σ^* oznacza rozwiązanie optymalne, a σ^H rozwiązanie uzyskane przy pomocy pewnego algorytmu aproksymacyjnego H_ε. Będziemy mówili, że H_ε jest algorytmem $(1+\varepsilon)$-aproksymacyjnym, jeżeli $Z(\sigma^H) \leqslant (1+\varepsilon)Z(\sigma^*)$ dla wszystkich instancji rozpatrywanego problemu.

Będziemy mówili, że rodzina algorytmów $\{H_\varepsilon\}$ jest *w pełni wielomianowym schematem aproksymacyjnym*, jeśli dla dowolnego $\varepsilon > 0$, H_ε jest $(1+\varepsilon)$-aproksymacyjnym algorytmem o złożoności wielomianowej zarówno od długości instancji problemu, jak i wyrażenia $1/\varepsilon$.

Podobnie jak miało to miejsce w przypadku algorytmów optymalnych, w celu uzyskania rozwiązania dla problemu **P4.2** rozwiążemy dwa podproblemy: **P4.2.1** i **P4.2.2**.

Schemat aproksymacyjny dla problemu P4.2.1

Niech $Z'(\sigma)$ oznacza wartość funkcji celu dla harmonogramu σ problemu $1|d_j = \hat{d}|\sum \beta_j U_j$:

$$Z'(\sigma) = \sum \beta_j U_j.$$

Z Własności 8.13 wynika, że dla każdego harmonogramu σ prawdziwa jest następująca relacja:

$$Z(\sigma) = Z'(\sigma) + \theta D.$$

Lemat 9. *Każdy algorytm $(1+\varepsilon)$-aproksymacyjny dla problemu $1|d_j = \hat{d}|\sum \beta_j U_j$ jest również $(1+\varepsilon)$-aproksymacyjnym dla problemu* **P4.2.1**.

Dowód. Niech σ^* będzie rozwiązaniem optymalnym problemu $1|d_j = \hat{d}|\sum \beta_j U_j$. Zgodnie z Własnością 8.13, σ^* jest jednocześnie rozwiązaniem optymalnym problemu **P4.2.1**. Niech σ^0 oznacza rozwiązanie dostarczone przez algorytm $(1+\varepsilon)$-aproksymacyjny dla problemu $1|d_j = \hat{d}|\sum \beta_j U_j$. Łatwo zauważyć, że prawdziwe są poniższe relacje:

$$\frac{Z(\sigma^0)}{Z(\sigma^*)} \leqslant \frac{Z(\sigma^0) - \theta D}{Z(\sigma^*) - \theta D} = \frac{Z'(\sigma^0)}{Z'(\sigma^*)} \leqslant \varepsilon.$$

Z powyższego wynika, że algorytm $(1+\varepsilon)$-aproksymacyjny dla problemu $1|d_j = \hat{d}|\sum \beta_j U_j$ jest również algorytmem $(1+\varepsilon)$-aproksymacyjnym dla problemu **P4.2.1**. □

W pracy [29] zaproponowano w pełni wielomianowy schemat aproksymacyjny dla problemu $1|d_j = \hat{d}|\sum \beta_j U_j$. Każdy algorytm $(1+\varepsilon)$-aproksymacyjny wchodzący w skład tego schematu działa w czasie $O(n^2 \log n + n^2/\varepsilon)$. Zgodnie z Lematem 9 algorytm ten jest również algorytmem $(1+\varepsilon)$-aproksymacyjnym dla problemu **P4.2.1**. Oznaczmy ten algorytm jako $\mathbf{H4.2.1}_\varepsilon$.

Schemat aproksymacyjny dla problemu P4.2.2

Zgodnie z Własnościami 8.8-8.12 oraz wcześniejszymi rozważaniami, problem **P4.2.1** może być sformułowany w sposób następujący:

$$\min Z(x) = \sum_{j=1}^{n} \left(x_j \theta p_j + (1 - x_j)\psi_j\right),$$

przy ograniczeniach:

$$\sum_{j=1}^{n} (x_j p_j) - \max_j \{p_j x_j\} \leqslant D, \tag{8.4}$$

$$x_j \in \{0, 1\},\ (j = 1, \ldots, n). \tag{8.5}$$

Jeżeli $x_j = 1$, to zadanie j jest przyporządkowane do zbioru $\breve{\mathbf{W}}$; jeżeli $x_j = 0$, to zadanie j jest przyporządkowane do zbioru $\breve{\mathbf{E}} \cup \hat{\mathbf{T}}$.

Załóżmy, że dane są wielkości LB oraz UB takie, że $0 < LB \leqslant Z(x^*) \leqslant UB$, gdzie x^* oznacza rozwiązanie optymalne. Niech $\lambda \triangleq \varepsilon LB/n$. Zdefiniujmy następujący *problem zaokrąglony*:

$$\min z(x) = \sum_{j=1}^{n} \left(x_j \left\lfloor \frac{\theta p_j}{\lambda} \right\rfloor + (1 - x_j) \left\lfloor \frac{\psi_j}{\lambda} \right\rfloor\right),$$

przy ograniczeniach (8.4) i (8.5).

Lemat 10. *Każdy dokładny algorytm dla problemu zaokrąglonego jest algorytmem* $(1+\varepsilon)$*-aproksymacyjnym dla problemu* **P4.2.1**.

Dowód. Niech x^* oznacza rozwiązanie optymalne dla problemu **P4.2.1** oraz niech x^0 oznacza rozwiązanie optymalne dla problemu zaokrąglonego. Łatwo zauważyć, że każde dopuszczalne rozwiązanie problemu **P4.2.1** jest rozwiązaniem dopuszczalnym problemu zaokrąglonego i na odwrót.

Pozostaje pokazać, że $Z(x^0) \leqslant (1 + \varepsilon)Z(x^*)$. Mamy:

$$Z(x^0) = \sum_{j=1}^{n} \left(x_j^0 \theta p_j + (1 - x_j^0)\psi_j\right) = \lambda \sum_{j=1}^{n} \left(x_j^0 \frac{\theta p_j}{\lambda} + (1 - x_j^0)\frac{\psi_j}{\lambda}\right) \leqslant$$

$$\lambda \sum_{j=1}^{n} \left(x_j^0 \left\lfloor \frac{\theta p_j}{\lambda} \right\rfloor + (1 - x_j^0) \left\lfloor \frac{\psi_j}{\lambda} \right\rfloor\right) + n\lambda = \lambda z(x^0) + n\lambda \leqslant$$

$$\lambda z(x^*) + n\lambda = \lambda \sum_{j=1}^{n} \left(x_j^* \left\lfloor \frac{\theta p_j}{\lambda} \right\rfloor + (1 - x_j^*) \left\lfloor \frac{\psi_j}{\lambda} \right\rfloor\right) + n\lambda \leqslant$$

$$\lambda \sum_{j=1}^{n} \left(x_j^* \frac{\theta p_j}{\lambda} + (1 - x_j^*)\frac{\psi_j}{\lambda}\right) + n\lambda = \sum_{j=1}^{n} \left(x_j^* \theta p_j + (1 - x_j^*)\psi_j\right) + n\lambda =$$

$$Z(x^*) + \lambda n = Z(x^*) + \frac{\varepsilon LB}{n} n = Z(x^*) + \varepsilon LB \leqslant Z(x^*) + \varepsilon Z(x^*) = (1 + \varepsilon)Z(x^*). \quad \square$$

Zaprezentujemy teraz algorytm programowania dynamicznego dla problemu zaokrąglonego, który jest modyfikacją algorytmu $\mathbf{A4.2.2}_2$.

Algorytm H4.2.2$_\varepsilon$

Krok 1. (Inicjalizacja). Ponumeruj zadania według niemalejących czasów ich wykonywania czyli tak, że $p_1 \leqslant \ldots \leqslant p_n$. Podstaw

$$P_k(v) := \begin{cases} 0, & \text{jeżeli } v = 0 \text{ i } k = 0; \\ +\infty, & \text{w przeciwnym przypadku.} \end{cases}$$

Następnie podstaw $k := 1$.

Krok 2. (Rekursja). Dla każdego $v = 0, \ldots, \lfloor UB/\lambda \rfloor$, wyznacz:

$$P_k(v) := \min \begin{cases} P_{k-1}\left(v - \left\lfloor \frac{\psi_k}{\lambda} \right\rfloor\right); \\ P_{k-1}\left(v - \left\lfloor \frac{\theta p_k}{\lambda} \right\rfloor\right) + p_k, \quad \text{jeżeli } P_{k-1}\left(v - \left\lfloor \frac{\theta p_k}{\lambda} \right\rfloor\right) \leqslant D. \end{cases}$$

Jeżeli $k = n$, to idź do Kroku 3, w przeciwnym przypadku podstaw $k := k + 1$ i powtórz Krok 2.

Krok 3. (Rozwiązanie optymalne). Wyznacz optymalną wartość kryterium:

$$\min \{v : P_n(v) < +\infty; v = 0, 1, \ldots, \lfloor UB/\lambda \rfloor\}$$

oraz skonstruuj odpowiadające rozwiązanie optymalne σ^* metodą przeglądu wstecznego (ang. backtracking).

Złożoność obliczeniowa powyższego algorytmu wynosi:

$$O\left(n\frac{UB}{\lambda}\right) = O\left(nUB\frac{n}{LB\varepsilon}\right) = O\left(n^2\frac{UB}{LB}\frac{1}{\varepsilon}\right).$$

Łatwo zauważyć, że możemy przyjąć następujące wartości ograniczeń LB oraz UB:

$$LB = \theta D,$$

$$UB = n \max_j \beta_j + \theta D.$$

Wartość LB jest większa od zera dla tych instancji, w których $D > 0$. Zauważmy, że jeżeli $D = 0$, to problem **P4.2.1** redukuje się do szczególnego przypadku problemu **P4.1** $(1|\langle e, d\rangle| \sum(\alpha_j V_j + \beta_j V_j))$ z $\gamma \geqslant \max_j\{\alpha_j, \beta_j\}$. Skoro problem **P4.1** można rozwiązać w czasie $O(n)$, to w dalszych rozważaniach ograniczymy się do tych instancji problemu **P4.2.1**, w których $D > 0$.

Algorytm $\mathbf{H4.2.2}_\varepsilon$ posiada własności, które pozwalają na zastosowanie tzw. Procedury Poprawy Ograniczeń (ang. Bound Improvement Procedure) [73] w celu znalezienia wartości Y^0 takiej, że $Y^0 \leqslant Z(x^*) \leqslant 3Y^0$. W procedurze tej binarne poszukiwanie jest dokonywane na przedziale $[\theta D, n \max_j \beta_j + \theta D]$. Dla każdej próbkowanej wartości $Y = 2^{k-1}\theta D$, $k = 1, 2, \ldots$, stosowany jest algorytm $\mathbf{H4.2.2}_\varepsilon$ przy $\varepsilon = 1$ dla $LB = Y$ oraz $UB = 2Y$. Złożoność obliczeniowa algorytmu $\mathbf{H4.2.2}_\varepsilon$ przy $LB = Y$ i $UB = Y$ wynosi $O(n^2/\varepsilon)$ [73]. Jeżeli znajduje on rozwiązanie z wartością Y', wówczas $Y \leqslant Z(x^*) \leqslant Y' \leqslant 3Y$ i podstawiamy $Y^0 := Y$. Jeżeli nie znajdzie żadnego rozwiązania, tzn. $P_n(v) = \infty$ dla wszystkich wartości v, wówczas $Z(x^*) > 2Y$ i procedura jest kontynuowana dla $k := k+1$. Złożoność obliczeniowa tej procedury jest następująca:

$$O\left(n^2 \log\left(\frac{UB}{LB}\right)\right) = O\left(n^2 \log\left(\frac{n \max_j \beta_j + \theta D}{\theta D}\right)\right) \leqslant O\left(n^2 \log\left(n \max_j \beta_j\right)\right) =$$
$$O\left(n^2\left(\log n + \log\left(\max_j \beta_j\right)\right)\right) \leqslant O\left(n^2 \log\left(\max_j\{n, \beta_j\}\right)\right).$$

Reasumując, kompletny w pełni wielomianowy schemat aproksymacyjny dla problemu **P4.2.2** stosuje kolejno Procedurę Poprawy Ograniczeń [73] oraz algorytm $\mathbf{H4.2.2}_\varepsilon$. Łatwo zauważyć, że dla ograniczeń znalezionych przez tę procedurę złożoność obliczeniowa algorytmu $\mathbf{H4.2.2}_\varepsilon$ wynosi:

$$O\left(n^2 \frac{1}{\varepsilon}\right).$$

Schemat aproksymacyjny dla problemu P4.2

Poniżej zostanie przedstawiony kompletny schemat aproksymacyjny dla problemu **P4.2**.

Algorytm $\mathbf{H4.2}_\varepsilon$

Krok 1. Rozwiąż problem **P4.2.1** przy pomocy algorytmu $\mathbf{H4.2.1}_\varepsilon$. Uzyskane rozwiązanie oznaczmy przez σ^1.

Krok 2. Rozwiąż problem **P4.2.2** przy pomocy Procedury Poprawy Ograniczeń oraz algorytmu $\mathbf{H4.2.2}_\varepsilon$. Uzyskane rozwiązanie oznaczmy przez σ^2.

Krok 3. Jeżeli $Z(\sigma^1) < Z(\sigma^2)$, to zwróć rozwiązanie σ^1. W przeciwnym przypadku zwróć rozwiązanie σ^2.

Twierdzenie 11. *Niech* σ^* *oznacza rozwiązanie optymalne problemu* **P4.2**. *Algorytm* $\mathbf{H4.2}_\varepsilon$ *dostarcza w czasie* $O\left(n^2 \log\left(\max_j\{n, \beta_j\}\right) + n^2/\varepsilon\right)$ *rozwiązanie* σ^0 *takie, że* $Z(\sigma^0) \leqslant (1+\varepsilon)Z(\sigma^*)$.

Dowód. Z Lematów 9 i 10 wynika, że $Z(\sigma^0) \leqslant (1+\varepsilon)Z(\sigma^*)$.

Złożoność obliczeniowa algorytmów $\mathbf{H4.2.1}_\varepsilon$ i $\mathbf{H4.2.2}_\varepsilon$ to odpowiednio $O\left(n^2 \log n + n^2/\varepsilon\right)$ oraz $O\left(n^2/\varepsilon\right)$. Procedura Poprawy Ograniczeń działa natomiast w czasie $O\left(n^2 \log\left(\max_j\{n, \beta_j\}\right)\right)$. Ostatecznie całkowita złożoność obliczeniowa algorytmu $\mathbf{H4.2}_\varepsilon$ wynosi $O\left(n^2 \log\left(\max_j\{n, \beta_j\}\right) + n^2/\varepsilon\right)$. □

8.6 Podsumowanie rozdziału

W Rozdziale 8 analizowano dwa jednoprocesorowe problemy szeregowania z doborem pożądanego przedziału zakończenia wykonywania zadań, przy kryterium minimalizacji ważonej liczby nieterminowo wykonanych zadań. W pierwszym problemie w procesie optymalizacji dobierany był zarówno początek jak i koniec tego przedziału. W drugim problemie optymalizowany był natomiast tylko początek tego przedziału przy zadanej z góry jego szerokości. Wykazano liczne własności rozwiązań optymalnych obu problemów. Na ich podstawie skonstruowano wielomianowy algorytm optymalny dla pierwszego problemu, natomiast dla drugiego problemu stworzono pseudowielomianowy algorytm optymalny oparty na metodzie programowania dynamicznego oraz w pełni wielomianowy schemat aproksymacyjny.

Rozdział 9

Minimalizacja ważonej liczby opóźnionych zadań oraz sumy ważonych przyspieszeń wykonania zadań

9.1 Wstęp

W niniejszym rozdziale przedstawione zostaną nowe wyniki, uzyskane przez autora niniejszej pracy, dotyczące jednoprocesorowego problemu szeregowania z optymalnym doborem pożądanego przedziału zakończenia wykonywania zadań. Minimalizacji podlegać będą: suma ważonych przyspieszeń wykonania zadań, ważona liczba opóźnionych zadań oraz kary związane z końcem i szerokością tego przedziału.

Rezultaty przedstawione w niniejszym rozdziale zostały częściowo opublikowane w pracy [60].

Na początku rozdziału zaprezentowana jest definicja badanego problemu (Podrozdział 9.2). Następnie przedstawione są istotne definicje oznaczeń oraz własności rozwiązania optymalnego (Podrozdział 9.3). W Podrozdziale 9.4 zostanie wykazana NP-trudność problemu poprzez transformację wielomianową z klasycznego problemu podziału. Pseudowielomianowy algorytm optymalny oparty na metodzie programowania dynamicznego będzie opisany w Podrozdziale 9.5. Niniejszy rozdział zamyka krótkie podsumowanie uzyskanych wyników.

9.2 Sformułowanie problemu

Poniżej zostanie przedstawiona definicja badanego problemu.

Zadany jest zbiór n niezależnych i niepodzielnych zadań $\mathbf{J} = \{1, \ldots, n\}$ do wykona-

nia na pojedynczym procesorze. Procesor ten może wykonywać jednocześnie tylko jedno zadanie. Każde zadanie $j \in \mathbf{J}$, o czasie wykonywania p_j ($p_j > 0$), jest dostępne w momencie 0. Zakładamy, że czasy wykonywania przyjmują tylko wartości całkowite. Wszystkie zadania mają wspólny pożądany przedział zakończenia wykonywania. Początek i koniec tego przedziału oznaczmy odpowiednio przez e i d.

Harmonogram zadań określa momenty zakończenia wykonywania zadań oraz początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

Przypomnijmy definicje następujących oznaczeń:

- S_j - moment rozpoczęcia wykonywania zadania j;
- C_j - moment zakończenia wykonywania zadania j;
- $E_j \triangleq \{e - C_j, 0\}$ - czas przedwczesnego wykonania zadania j (przyspieszenie wykonania zadania j);
- $U_j \triangleq \begin{cases} 1 & \text{jeżeli } C_j > d; \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$

Celem jest znalezienie takiego harmonogramu σ, który minimalizuje wartość następującego kryterium:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j E_j + \beta_j U_j) + \gamma(d - e) + \theta d,$$

gdzie $\alpha_j \geqslant 0$, $\beta_j \geqslant 0$, $\gamma \geqslant 0$ i $\theta \geqslant 0$ są zadanymi wagami, które przyjmują tylko wartości całkowite.

W trójpolowej notacji [34] problem może być przedstawiony następująco:

$$1|\langle e, d\rangle| \sum(\alpha_j E_j + \beta_j U_j) + \gamma(d - e) + \theta d.$$

W celu skrócenia notacji, badany problem będziemy oznaczać również jako **P5**.

9.3 Własności rozwiązania optymalnego

W tym podrozdziale zostanie zaprezentowanych szereg własności rozwiązania optymalnego problemu **P5**.

Dowody dwóch poniższych własności zostaną pominięte, ponieważ byłyby one niemal identyczne jak dowody analogicznych Własności 5.1 i 5.2 dla Problemu $\mathbf{P1}(1|\langle e, d\rangle; D_{min} \leqslant d - e \leqslant D_{max}| \sum(\alpha E_j + \beta T_j) + \theta e + f_W(d - e).)$.

Własność 9.1. *Istnieje rozwiązanie optymalne problemu* **P5**, *w którym nie ma okresów bezczynności procesora pomiędzy wykonywaniem poszczególnych zadań.*

Własność 9.2. *Istnieje rozwiązanie optymalne problemu* **P5**, *w którym przynajmniej jedno zadanie rozpoczyna wykonywać się w momencie* 0.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P5**, które spełniają Własności 9.1 i 9.2.

Zauważmy, że dla rozwiązania spełniającego Własności 9.1 oraz 9.2, kolejność wykonania zadań implikuje momenty ich rozpoczęcia i zakończenia. Zatem przedział $[e, d]$ oraz kolejność wykonywania zadań określają precyzyjnie harmonogram dla problemu **P5**.

Zapis $\sigma = (\pi, e, d)$ będzie oznaczał harmonogram σ, w którym π określa kolejność wykonywania zadań na procesorze, natomiast e i d to odpowiednio początek i koniec pożądanego przedziału zakończenia wykonywania zadań.

Przypomnijmy teraz zdefiniowane wcześniej oznaczenia:

- $\mathbf{E} \triangleq \{j \in \mathbf{J} : S_j < e\}$ - zbiór zadań, które rozpoczynają wykonywać się przed momentem e;
- $\hat{\mathbf{E}} \triangleq \{j \in \mathbf{J} | C_j \leqslant e\}$ - zbiór zadań, które kończą wykonywać się w lub przed momentem e;
- $\mathbf{T} \triangleq \{j \in \mathbf{J} : C_j > d\}$ - zbiór zadań, które kończą wykonywać się po momencie d;
- $\hat{\mathbf{T}} \triangleq \{j \in \mathbf{J} | S_j \geqslant d\}$ - zbiór zadań, które rozpoczynają wykonywać się w lub po momencie d;
- $\mathbf{W} \triangleq \{j \in \mathbf{J} : S_j \leqslant d \wedge C_j \geqslant e\}$ - zbiór zadań wykonanych całkowicie lub częściowo wewnątrz przedziału $[e, d]$;
- $\mathbf{WI} \triangleq \{j \in \mathbf{J} : S_j \geqslant e \wedge C_j \leqslant d\}$ - zbiór zadań wykonanych całkowicie wewnątrz przedziału $[e, d]$;
- $K_E \triangleq \min\{j : C_{\pi(j)} \geqslant e\}$ - pozycja pierwszego zadania, które kończy wykonywać się w lub po momencie e;
- $\Delta_E \triangleq C_{\pi(K_E)} - e$ - część zadania $\pi(K_E)$, która wykonuje się wewnątrz przedziału $[e, d]$;
- $\delta_E \triangleq e - S_{\pi(K_E)}$ - część zadania $\pi(K_E)$, która wykonuje się na zewnątrz przedziału $[e, d]$.

Własność 9.3. *Istnieje rozwiązanie optymalne problemu* **P5***, w którym pewne zadanie kończy lub rozpoczyna wykonywać się w momencie e.*

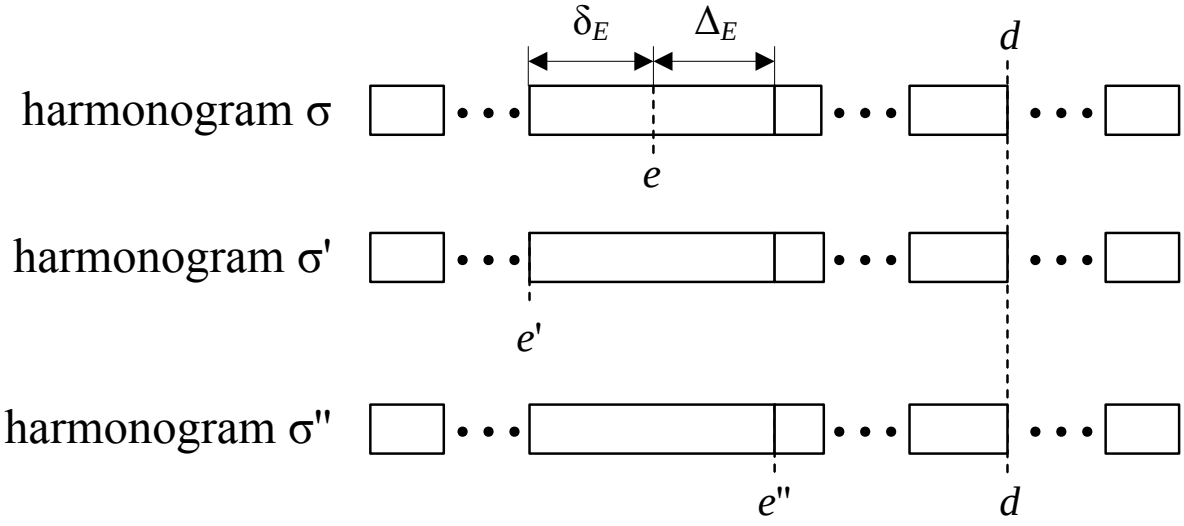
Dowód. Załóżmy, że znane jest rozwiązanie optymalne $\sigma = (\pi, e, d)$. Przyjmijmy również, że rozwiązanie to nie spełnia dowodzonej własności, tzn. żadne zadanie nie kończy ani nie rozpoczyna wykonywać się w momencie e, czyli $\Delta_E(\sigma) > 0$. Przyjmijmy w niniejszym dowodzie, że oznaczenia Δ_E oraz δ_E odnoszą się do harmonogramu σ. Rozważmy dwa możliwe przypadki.

Przypadek 1. Załóżmy, że $d - e \geqslant \Delta_E$.

Wartość kryterium dla harmonogramu σ możemy wyrazić w następujący sposób:

$$Z(\sigma) = \sum_{j\in\mathbf{J}}(\alpha_j E_j + \beta_j U_j) + \gamma(d-e) + \theta d = Z_0 + \sum_{j\in\mathbf{J}} \alpha_j E_j + \gamma(d-e) =$$
$$Z_0 + \sum_{j\in\mathbf{J}} \alpha_j \max\{e - C_j, 0\} + \gamma(d-e) = Z_0 + \sum_{j\in\hat{\mathbf{E}}} \alpha_j (e - C_j) + \gamma(d-e),$$

gdzie $Z_0 = \sum_{j\in\mathbf{J}} \beta_j U_j + \theta d$.



Rysunek 9.1: Przykładowe harmonogramy σ, σ' i σ'' przy założeniu, że $d - e \geqslant \Delta_E$.

Utwórzmy teraz harmonogram $\sigma' = (\pi, e', d)$ z harmonogramu σ taki, że $e' = e - \delta_E$. W tak powstałym harmonogramie pewne zadanie kończy lub rozpoczyna wykonywać się w momencie e' (patrz Rysunek 9.1). Łatwo zauważyć, że $U_j(\sigma) = U_j(\sigma')$ dla każdego zadania $j \in \mathbf{J}$ (patrz Rysunek 9.1). Zatem wartość funkcji celu dla harmonogramu σ' wynosi:

$$Z(\sigma') = \sum_{j\in\mathbf{J}}(\alpha_j E_j + \beta_j U_j) + \gamma(d-e') + \theta d = Z_0 + \sum_{j\in\mathbf{J}} \alpha_j E_j + \gamma(d-e') =$$
$$Z_0 + \sum_{j\in\mathbf{J}} \alpha_j \max\{e' - C_j, 0\} + \gamma(d-e') = Z_0 + \sum_{j\in\hat{\mathbf{E}}} \alpha_j (e' - C_j) + \gamma(d-e') =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}} \alpha_j (e - \delta_E - C_j) + \gamma(d - e + \delta_E).$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma') =$$
$$\left[\sum_{j\in\hat{\mathbf{E}}} \alpha_j(e - C_j) + \gamma(d - e)\right] - \left[\sum_{j\in\hat{\mathbf{E}}} \alpha_j(e - \delta_E - C_j) + \gamma(d - e + \delta_E)\right] =$$
$$\delta_E \sum_{j\in\hat{\mathbf{E}}} \alpha_j - \delta_E\gamma = \delta_E \left(\sum_{j\in\hat{\mathbf{E}}} \alpha_j - \gamma\right).$$

Utwórzmy teraz harmonogram $\sigma'' = (\pi, e'', d)$ z harmonogramu σ taki, że $e'' = e + \Delta_E$. W tak powstałym harmonogramie pewne zadanie kończy wykonywać się w momencie e'' (patrz Rysunek 9.1). Skoro założyliśmy, że warunek $d - e \geqslant \Delta_E$ jest spełniony, to $d \geqslant e + \Delta_E = e''$. Łatwo zauważyć, że $U_j(\sigma) = U_j(\sigma'')$ dla każdego zadania $j \in \mathbf{J}$ (patrz Rysunek 9.1). Zatem wartość funkcji celu dla harmonogramu σ'' wynosi:

$$Z(\sigma'') = \sum_{j\in\mathbf{J}} (\alpha_j E_j + \beta_j U_j) + \gamma(d - e'') + \theta d = Z_0 + \sum_{j\in\mathbf{J}} \alpha_j E_j + \gamma(d - e'') =$$
$$Z_0 + \sum_{j\in\mathbf{J}} \alpha_j \max\{e'' - C_j, 0\} + \gamma(d - e'') = Z_0 + \sum_{j\in\hat{\mathbf{E}}} \alpha_j(e'' - C_j) + \gamma(d - e'') =$$
$$Z_0 + \sum_{j\in\hat{\mathbf{E}}} \alpha_j(e + \Delta_E - C_j) + \gamma(d - e + \Delta_E).$$

Mamy zatem:

$$Z(\sigma) - Z(\sigma'') =$$
$$\left[\sum_{j\in\hat{\mathbf{E}}} \alpha_j(e - C_j) + \gamma(d - e)\right] - \left[\sum_{j\in\hat{\mathbf{E}}} \alpha_j(e + \Delta_E - C_j) + \gamma(d - e - \Delta_E)\right] =$$
$$-\Delta_E \sum_{j\in\hat{\mathbf{E}}} \alpha_j + \Delta_E\gamma = \Delta_E \left(\gamma - \sum_{j\in\hat{\mathbf{E}}} \alpha_j\right).$$

Łatwo zauważyć, że:

- jeżeli $\gamma \leqslant \sum_{j\in\hat{\mathbf{E}}} \alpha_j$, to $Z(\sigma') \leqslant Z(\sigma)$;
- jeżeli $\gamma \geqslant \sum_{j\in\hat{\mathbf{E}}} \alpha_j$, to $Z(\sigma'') \leqslant Z(\sigma)$.

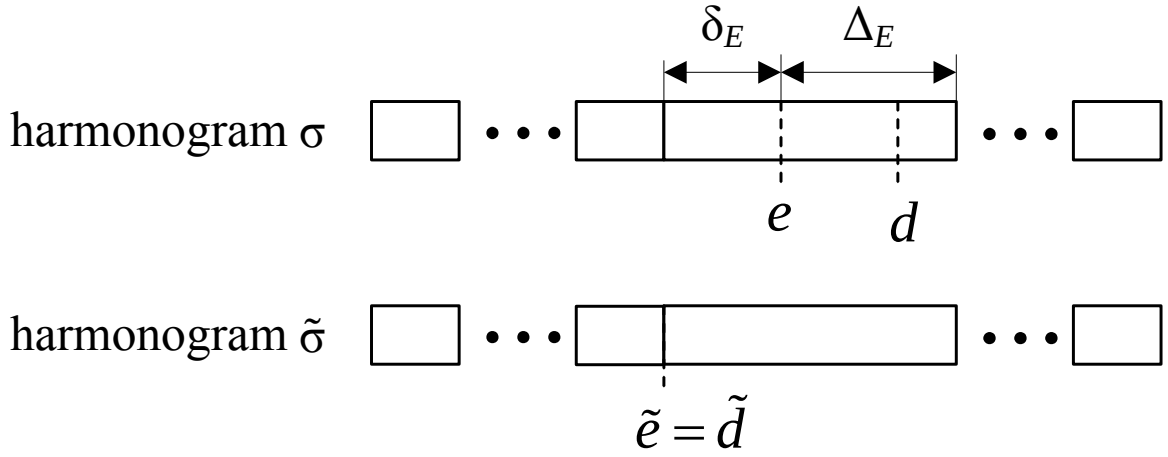
Z powyższego wynika, że $Z(\sigma') \leqslant Z(\sigma)$ lub $Z(\sigma'') \leqslant Z(\sigma)$.

Przypadek 2. Załóżmy, że $d - e < \Delta_E$.

Wartość kryterium dla harmonogramu σ możemy wyrazić w następujący sposób:

$$Z(\sigma) = \sum_{j \in \mathbf{J}} (\alpha_j E_j + \beta_j U_j) + \gamma(d-e) + \theta d = \tilde{Z}_0 + \sum_{j \in \mathbf{J}} \alpha_j E_j + \gamma(d-e) + \theta d =$$
$$\tilde{Z}_0 + \sum_{j \in \mathbf{J}} \alpha_j \max\{e - C_j, 0\} + \gamma(d-e) + \theta d =$$
$$\tilde{Z}_0 + \sum_{j \in \hat{\mathbf{E}}} \alpha_j (e - C_j) + \gamma(d-e) + \theta d,$$

gdzie $\tilde{Z}_0 = \sum_{j \in \mathbf{J}} \beta_j U_j$.



Rysunek 9.2: Przykładowe harmonogramy σ i σ'' przy założeniu, że $d - e < \Delta_T$.

Utwórzmy teraz harmonogram $\tilde{\sigma} = (\pi, \tilde{e}, \tilde{d})$ z harmonogramu σ taki, że $\tilde{d} = \tilde{e} = e - \delta_E$. W tak powstałym harmonogramie pewne zadanie kończy lub rozpoczyna wykonywać się w momencie $\tilde{e}$ (patrz Rysunek 9.2). Łatwo zauważyć, że $U_j(\sigma) = U_j(\tilde{\sigma})$ dla każdego zadania $j \in \mathbf{J}$ (patrz Rysunek 9.2). Zatem wartość funkcji celu dla harmonogramu $\tilde{\sigma}$ wynosi:

$$Z(\tilde{\sigma}) = \sum_{j \in \mathbf{J}} (\alpha_j E_j + \beta_j U_j) + \gamma(\tilde{d} - \tilde{e}) + \theta \tilde{d} =$$
$$\tilde{Z}_0 + \sum_{j \in \mathbf{J}} \alpha_j E_j + \gamma(\tilde{d} - \tilde{e}) + \theta \tilde{d} = \tilde{Z}_0 + \sum_{j \in \mathbf{J}} \alpha_j \max\{\tilde{e} - C_j, 0\} + \theta \tilde{d} =$$
$$\tilde{Z}_0 + \sum_{j \in \hat{\mathbf{E}}} \alpha_j (\tilde{e} - C_j) + \theta \tilde{d} = \tilde{Z}_0 + \sum_{j \in \hat{\mathbf{E}}} \alpha_j (e - \delta_E - C_j) + \theta(d - \delta_E).$$

Mamy zatem:

$$Z(\sigma) - Z(\tilde{\sigma}) =$$
$$\left[\sum_{j\in\hat{\mathbf{E}}} \alpha_j(e - C_j) + \gamma(d-e) + \theta d\right] - \left[\sum_{j\in\hat{\mathbf{E}}} \alpha_j(e - \delta_E - C_j) + \theta(d-\delta_E)\right] =$$
$$\delta_E \sum_{j\in\hat{\mathbf{E}}} \alpha_j + \gamma(d-e) + \delta_E\theta = \gamma(d-e) + \delta_E\left(\sum_{j\in\hat{\mathbf{E}}} \alpha_j + \theta\right) \geqslant 0,$$

a stąd:

$$Z(\sigma) \geqslant Z(\tilde{\sigma}).$$

Z powyższego wynika, że możemy uzyskać harmonogram, w którym pewne zadanie kończy lub rozpoczyna wykonywać się w momencie e, bez wzrostu wartości kryterium. □

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P5**, które spełniają Własność 9.3.

Własność 9.4. *Istnieje rozwiązanie optymalne problemu* **P5***, w którym jedno zadanie kończy lub rozpoczyna wykonywać się w momencie d.*

Dowód powyższej własności zostanie pominięty, ponieważ byłby on niemal identyczny jak dowód analogicznej Własności 8.3 dla problemu **P4.1** $(1|\langle e,d\rangle|\sum(\alpha_j V_j + \beta_j T_j) + \gamma(d-e) + \theta d)$.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P5**, które spełniają Własność 9.4.

Z Własności 9.3 i 9.4 wynika, że w rozwiązaniu optymalnym problemu **P5** prawdziwe są następujące zależności: $\hat{\mathbf{E}} = \mathbf{E}$, $\hat{\mathbf{T}} = \mathbf{T}$ oraz $\mathbf{W} = \mathbf{WI}$.

Własność 9.5. *W rozwiązaniu optymalnym problemu* **P5** *zadania ze zbioru* $\mathbf{E}$ *są uszeregowane według nierosnącego porządku wartości ilorazu p_j/α_j.*

Dowód tej własności zostanie pominięty, ponieważ byłby niemal identyczny jak dowód Własności 7.4 dla problemu **P3.1** $(1|\langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max}|\sum(\alpha_j E_j + \beta_j T_j) + \gamma(d-e))$.

W dalszych rozważaniach będą brane pod uwagę tylko te rozwiązania problemu **P5**, które spełniają Własność 9.5.

Własność 9.6. *W rozwiązaniu optymalnym problemu* **P5**

- *zadania ze zbioru* $\mathbf{W}$ *są uszeregowane w dowolnej kolejności;*

- *zadania ze zbioru* $\mathbf{T}$ *są uszeregowane w dowolnej kolejności.*

Dowód. Prawdziwość tej własności wynika bezpośrednio z faktu, że kolejność wykonywania zadań ze zbioru $\mathbf{W}$ ani kolejność wykonywania zadań ze zbioru $\mathbf{T}$ nie wpływa na wartość funkcji celu. □

Z Własności 9.3 i 9.4 wynika, że początek i koniec pożądanego przedziału zakończenia wykonywania zadań można wyrazić w sposób następujący:

$$e = \sum_{j \in \mathbf{E}} p_j;$$

$$d = \sum_{j \in \mathbf{E} \cup \mathbf{W}} p_j = e + \sum_{j \in \mathbf{W}} p_j.$$

Korzystając z powyższych wzorów, wartość funkcji celu dla problemu **P5** możemy zapisać w następującej postaci:

$$\begin{gathered} Z(\sigma) = \sum \alpha_j E_j + \sum \beta_j U_j + \gamma(d-e) + \theta d = \\ \sum_{j \in \mathbf{E}} \alpha_j E_j + \sum_{j \in \mathbf{T}} \beta_j + \gamma \sum_{j \in \mathbf{W}} p_j + \theta \sum_{j \in \mathbf{E} \cup \mathbf{W}} p_j = \\ \sum_{j \in \mathbf{E}} (\alpha_j E_j + \theta p_j) + \sum_{j \in \mathbf{W}} (\gamma + \theta) p_j + \sum_{j \in \mathbf{T}} \beta_j. \end{gathered}$$

Z powyższego wyrażenia wynika bezpośrednio prawdziwość następującej własności.

Własność 9.7. *W optymalnym rozwiązaniu problemu* **P5**

- $\beta_j \leqslant \gamma p_j$ *dla każdego zadania* $j \in \mathbf{T}$ *oraz*
- $\gamma p_j \leqslant \beta_j$ *dla każdego zadania* $j \in \mathbf{W}$.

W dalszych rozważaniach będziemy brać pod uwagę tylko te rozwiązania problemu **P4.2**, dla których Własność 9.7 jest spełniona.

Powyższa własność pozwala nam zdecydować, czy zadanie j należy do zbioru $\mathbf{W}$ czy do zbioru $\mathbf{T}$ jeżeli wiemy, że należy ono do zbioru $\mathbf{W} \cup \mathbf{T}$.

Wartość funkcji celu dla harmonogramu σ spełniającego wszystkie wymienione powyżej własności wynosi:

$$\begin{gathered} Z(\sigma) = \sum \alpha_j E_j + \sum \beta_j U_j + \gamma(d-e) = \\ \sum_{j \in \mathbf{E}} (\alpha_j E_j + \theta p_j) + \sum_{j \in \mathbf{W} \cup \mathbf{T}} \min\{\beta_j, (\gamma + \theta) p_j\}. \end{gathered} \tag{9.1}$$

Podsumujmy teraz przedstawione powyżej własności rozwiązania optymalnego problemu **P5**:

- nie ma okresów bezczynności procesora pomiędzy wykonywaniem kolejnych zadań;

- pierwsze zadanie rozpoczyna wykonywać się w momencie 0;
- pewne zadnie kończy lub rozpoczyna wykonywać się w momencie e;
- pewne zadanie kończy lub rozpoczyna wykonywać się w momencie d;
- zadania ze zbioru $\mathbf{E}$ są uszeregowane według nierosnącego porządku wartości ilorazu p_j/α_j;
- kolejność wykonywania zadań ze zbioru $\mathbf{W}$ jest dowolna;
- kolejność wykonywania zadań ze zbioru $\mathbf{T}$ jest dowolna;
- $\beta_j \leqslant \gamma p_j$ dla każdego zadania $j \in \mathbf{T}$;
- $\gamma p_j \leqslant \beta_j$ dla każdego zadania $j \in \mathbf{W}$.

9.4 Analiza złożoności obliczeniowej

Poniżej wykażemy, że problem **P5** jest NP-trudny.

Twierdzenie 12. *Problem* **P5** *jest NP-trudny.*

Dowód. W dowodzie powyższego twierdzenia zostanie wykorzystany NP-zupełny Problem Podziału (PP), który można sformułować następująco [28]:

Dany jest zbiór $\widehat{n}$ dodatnich liczb całkowitych: $\mathbf{X} = \{x_1, x_2, \ldots, x_{\widehat{n}}\}$, których suma wynosi $\sum_{j=1}^{\widehat{n}} x_j = 2B$. Pytanie brzmi, czy istnieje taki podział zbioru $\mathbf{X}$ na dwa rozdzielne podzbiory $\mathbf{X}_1$ i $\mathbf{X}_2$, że $\sum_{x_j \in \mathbf{X}_1} x_j = \sum_{x_j \in \mathbf{X}_2} x_j = B$?

Odpowiednia instancja rozpatrywanego problemu szeregowania zdefiniowana jest poprzez dane problemu PP w następujący sposób.
Danych jest $n = \widehat{n} + 1$ zadań, wśród których jest $\widehat{n}$ zadań *podziału* oraz jedno zadanie *dodatkowe* z. Parametry zadań są następujące:

$$p_j = x_j; \qquad \alpha_j = 2x_j; \qquad \beta_j = 2Bx_j + 2x_j - x_j^2; \qquad j = 1, \ldots, \widehat{n};$$
$$p_z = 1; \qquad \alpha_z = B + 1; \qquad \beta_z = B + 1.$$

Ponadto, $\gamma = B + 1$ oraz $\theta = 0$.

Łatwo zauważyć, że powyższa transformacja jest transformacją wielomianową. Zatem należy teraz wykazać, że problem PP posiada rozwiązanie wtedy i tylko wtedy, gdy istnieje także rozwiązanie dla skonstruowanej powyżej instancji rozpatrywanego problemu szeregowania z następującą wartością kryterium:

$$\sum_{j \in \mathbf{J}} (\alpha_j E_j + \beta_j U_j) + \gamma(d - e) + \theta d \leqslant y = 4B - \sum_{x_j \in \mathbf{X}} x_j^2 + 3B^2.$$

(i) Załóżmy, że PP posiada rozwiązanie, tzn. istnieje podział zbioru $\mathbf{X}$ na dwa rozłączne podzbiory $\mathbf{X}_1$ oraz $\mathbf{X}_2$ taki, że zachodzi $\sum_{x_j \in \mathbf{X}_1} x_j = \sum_{x_j \in \mathbf{X}_2} x_j = B$. Niech $\mathbf{J}_1$ oraz $\mathbf{J}_2$ oznaczają zbiory zawierające zadania utworzone odpowiednio na podstawie elementów z podzbiorów $\mathbf{X}_1$ oraz $\mathbf{X}_2$. Załóżmy, bez straty ogólności, że zbiory $\mathbf{J}_1$ oraz $\mathbf{J}_2$ zawierają odpowiednio k oraz $\widehat{n} - k$ zadań ($0 \leqslant k \leqslant \widehat{n}$). Rozwiązanie dla problemu **P5** jest dane przez uszeregowanie π, w którym w pierwszej kolejności procesor wykonuje w dowolnej kolejności zadania ze zbioru $\mathbf{J}_1$, następnie zadanie dodatkowe a dalej zadania ze zbioru $\mathbf{J}_2$, również w dowolnej kolejności. Parametry przedziału $[e, d]$ są zadane natomiast w sposób następujący: $e = d = \sum_{j \in \mathbf{J}_1 \cup \{z\}} p_j = 1 + \sum_{j \in \mathbf{J}_1} p_j$.

Suma kar za przedwczesne wykonanie zadań w harmonogramie $\sigma = (\pi, e, d)$ wynosi:

$$\sum_{j \in \mathbf{J}} \alpha_j E_j = \sum_{j \in \mathbf{J}_1} \alpha_j E_j = \sum_{j=1}^{k} \alpha_{\pi(j)} E_{\pi(j)} = \sum_{j=1}^{k} \alpha_{\pi(j)} (e - C_{\pi(j)}) =$$
$$\sum_{j=1}^{k} \left[\alpha_{\pi(j)} \left(1 + \sum_{l \in \mathbf{J}_1} p_l - C_{\pi(j)} \right) \right] = \sum_{j=1}^{k} \left[\alpha_{\pi(j)} \left(1 + \sum_{l=1}^{k} p_{\pi(l)} - \sum_{l=1}^{j} p_{\pi(l)} \right) \right] =$$
$$\sum_{j=1}^{k} \left[\alpha_{\pi(j)} \left(1 + \sum_{l=j+1}^{k} p_{\pi(l)} \right) \right] = \sum_{j=1}^{k} \alpha_{\pi(j)} + \sum_{j=1}^{k} \left[\alpha_{\pi(j)} \left(\sum_{l=j+1}^{k} p_{\pi(l)} \right) \right].$$

Ze względu na to, że $p_j = x_j$ oraz $\alpha_j = 2x_j$, wartość powyższego wyrażenia jest równa:

$$\sum_{j \in \mathbf{J}} \alpha_j E_j = \sum_{j=1}^{k} 2x_j + \sum_{j=1}^{k} \left[x_j \left(\sum_{l=j+1}^{k} 2x_l \right) \right] =$$
$$2 \sum_{j=1}^{k} x_j + 2 \left[x_1 (x_2 + x_3 + \ldots + x_k) + x_2 (x_3 + x_4 + \ldots + x_k) + \ldots + x_{k-1} x_k \right] =$$
$$2 \sum_{j=1}^{k} x_j + 2 \sum_{0 \leqslant j < l \leqslant k} x_j x_l.$$

Korzystając z następującej, znanej z algebry, zależności:

$$\left(\sum_{j=1}^{k} x_j \right)^2 = \sum_{j=1}^{k} x_j^2 + 2 \sum_{0 \leqslant j < l \leqslant k} x_j x_l$$

otrzymujemy:

$$\sum_{j \in \mathbf{J}} \alpha_j E_j = 2 \sum_{j=1}^{k} x_j + \left(\sum_{j=1}^{k} x_j \right)^2 - \sum_{j=1}^{k} x_j^2 = 2 \sum_{x_j \in \mathbf{X}_1} x_j + \left(\sum_{x_j \in \mathbf{X}_1} x_j \right)^2 - \sum_{x_j \in \mathbf{X}_1} x_j^2.$$

Suma kar za zbyt późne wykonanie zadań w harmonogramie $\sigma = (\pi, e, d)$, przy

uwzględnieniu, że $p_j = x_j$ oraz $\beta_j = 2Bx_j + 2x_j - x_j^2$, wynosi:

$$\sum_{j\in\mathbf{J}} \beta_j U_j = \sum_{j\in\mathbf{J}_2} \beta_j = \sum_{x_j\in\mathbf{X}_2} (2Bx_j + 2x_j - x_j^2) = 2B \sum_{x_j\in\mathbf{X}_2} x_j + 2 \sum_{x_j\in\mathbf{X}_2} x_j - \sum_{x_j\in\mathbf{X}_2} x_j^2.$$

Skoro $e = d$ oraz $\theta = 0$, to kary związane z szerokością i początkiem przedziału $[e, d]$ są równe 0.

W związku z powyższym, wartość kryterium dla harmonogramu $\sigma = (\pi, e, d)$ wynosi:

$$\sum_{j\in\mathbf{J}} (\alpha_j E_j + \beta_j U_j) + \gamma(d - e) + \theta d =$$

$$2 \sum_{x_j\in\mathbf{X}_1} x_j + \left(\sum_{x_j\in\mathbf{X}_1} x_j \right)^2 - \sum_{x_j\in\mathbf{X}_1} x_j^2 + 2B \sum_{x_j\in\mathbf{X}_2} x_j + 2 \sum_{x_j\in\mathbf{X}_2} x_j - \sum_{x_j\in\mathbf{X}_2} x_j^2 =$$

$$2 \sum_{x_j\in\mathbf{X}} x_j - \sum_{x_j\in\mathbf{X}} x_j^2 + \left(\sum_{x_j\in\mathbf{X}_1} x_j \right)^2 + 2B \sum_{x_j\in\mathbf{X}_2} x_j =$$

$$4B - \sum_{x_j\in\mathbf{X}} x_j^2 + \left(\sum_{x_j\in\mathbf{X}_1} x_j \right)^2 + 2B \sum_{x_j\in\mathbf{X}_2} x_j. \tag{9.2}$$

Z przyjętego założenia $\sum_{x_j\in\mathbf{X}_1} x_j = \sum_{x_j\in\mathbf{X}_2} x_j = B$ wynika, że wartość kryterium jest równa wymaganej wartości:

$$Z(\sigma) = 4B - \sum_{j\in\mathbf{X}} x_j^2 + B^2 + 2B^2 = y.$$

(ii) Załóżmy teraz, że PP nie posiada rozwiązania, co oznacza, że dowolny podział zbioru $\mathbf{X}$ na dwa rozłączne podzbiory $\mathbf{X}_1$ oraz $\mathbf{X}_2$ daje $\sum_{x_j\in\mathbf{X}_1} x_j \neq \sum_{x_j\in\mathbf{X}_2} x_j$. Bez straty ogólności możemy założyć, że $\sum_{x_j\in\mathbf{X}_1} x_j = B-\lambda$ oraz $\sum_{x_j\in\mathbf{X}_2} x_j = B+\lambda$, gdzie λ jest pewną dodatnią liczbą całkowitą. Skoro $\gamma = B+1$, to szerokość przedziału $[e, d]$ musi być równa 0. Zadanie dodatkowe z musi kończyć wykonywać się w momencie $e = d$ (w przeciwnym przypadku wartość kryterium byłaby większa niż y niezależnie od uszeregowania pozostałych zadań). Wynika z tego, że wyrażenie (9.2) jest prawdziwe dla dowolnego podziału zbioru $\mathbf{X}$ na podzbiory $\mathbf{X}_1$ i $\mathbf{X}_2$ (nawet dla przypadku, gdy jeden z tych podzbiorów jest pusty). Zatem, przy założeniu braku rozwiązania dla

PP, otrzymujemy:

$$Z(\sigma) = 4B - \sum_{x_j \in \mathbf{X}} x_j^2 + (B-\lambda)^2 + 2B\,(B+\lambda) =$$
$$4B - \sum_{x_j \in \mathbf{X}} x_j^2 + B^2 - 2B\lambda + \lambda^2 + 2B^2 + 2B\lambda =$$
$$4B - \sum_{x_j \in \mathbf{X}} x_j^2 + 3B^2 + \lambda^2 > y.$$

Z powyższego wynika, że brak rozwiązania dla PP sprawia, iż problem szeregowania **P5** również nie posiada rozwiązania.

Problem **P5** posiada zatem rozwiązanie wtedy i tylko wtedy, gdy problem podziału (PP) również je posiada. □

9.5 Algorytm optymalny

W niniejszym podrozdziale zostanie przedstawiony algorytm programowania dynamicznego, który rozwiązuje optymalnie problem **P5**.

Zauważmy, że na podstawie wykazanych własności rozwiązania optymalnego problemu **P5** możemy skonstruować rozwiązanie optymalne jeżeli wiemy, które zadania znajdują się w zbiorze **E**, a które w zbiorze $\mathbf{W} \cup \mathbf{T}$. W celu znalezienia rozwiązania należy więc dla każdego zadania podjąć decyzję czy należy ono do zbioru **E**, czy do zbioru $\mathbf{W} \cup \mathbf{T}$. Zauważmy, że przyporządkowanie zadania k do zbioru $\mathbf{W} \cup \mathbf{T}$ powoduje powstanie kosztu $\min\{(\theta+\gamma)p_k, \beta_k\}$, natomiast przyporządkowanie zadania k do zbioru **E** powoduje powstanie kosztu $\alpha_k a + \theta p_k$, gdzie a jest sumą czasów wykonywania zadań należących do zbioru **E** (przed uszeregowaniem zadania k) - patrz Własność 9.5.

Załóżmy, że zadania są ponumerowane tak, że $p_1/\alpha_1 \leqslant \ldots \leqslant p_n/\alpha_n$.

Poniżej opisane jest działanie Algorytmu **A5**, który rozwiązuje optymalnie problem **P5**. Rozpoczynamy od harmonogramu zerowego σ_0, czyli takiego, który nie zawiera żadnego zadania. Następnie generujemy kolejne harmonogramy cząstkowe σ_k (harmonogramy zawierające k pierwszych zadań) z harmonogramów σ_{k-1} poprzez przyporządkowanie zadania k do zbioru **E** lub do zbioru $\mathbf{W} \cup \mathbf{T}$.

Będziemy mówić, że harmonogram cząstkowy jest w stanie (k, a) jeżeli zawiera pierwszych k zadań, a suma czasów wykonywania zadań ze zbioru **E** wynosi a.

Niech $F_k(a)$ oznacza minimalną wartość funkcji celu dla harmonogramów znajdujących się w stanie (k, a).

Łatwo zauważyć, że wartość funkcji celu dla rozwiązania optymalnego równa jest:

$$\min\left\{F_n(a) : a = 0, 1, \ldots, \sum_{j \in \mathbf{J}} p_j\right\}.$$

Załóżmy, że mamy harmonogram cząstkowy znajdujący się w stanie $(k-1, a)$. Jeżeli zadanie k przyporządkujemy do zbioru $\mathbf{E}$ (zadanie k zakończy wykonywać się w momencie $e-a$), to uzyskamy harmonogram w stanie $(k, a+p_k)$. W rezultacie wartość kryterium wzrośnie o $\alpha_k a + \theta p_k$, czyli $F_k(a+p_k) = F_{k-1}(a) + \alpha_k a + \theta p_k$; patrz wyrażenie (9.1). Jeżeli natomiast zadanie k przyporządkujemy do zbioru $\mathbf{W} \cup \mathbf{T}$, to uzyskamy harmonogram w stanie (k, a). Wartość funkcji celu wzrośnie o $\min\{\beta_k, (\theta+\gamma)p_k\}$, czyli $F_k(a) = F_{k-1}(a) + \min\{\beta_k, (\theta+\gamma)p_k\}$.

Z powyższego wynika, że harmonogram w stanie (k, a) możemy uzyskać:

- z harmonogramu w stanie $(k-1, a-p_k)$ poprzez przyporządkowanie zadania k do zbioru $\mathbf{E}$. W tym przypadku mamy $F_k(a) = F_{k-1}(a-p_k) + \alpha_k(a-p_k) + \theta p_k$.
- z harmonogramu w stanie $(k-1, a)$ poprzez przyporządkowanie zadania k do zbioru $\mathbf{W} \cup \mathbf{T}$. W tym przypadku mamy $F_k(a) = F_{k-1}(a) + \min\{\beta_k, (\gamma+\theta)p_k\}$.

Formalny opis algorytmu **A5** jest podany poniżej.

Algorytm A5

Krok 1. (Inicjalizacja). Ponumeruj zadania tak, że $p_1/\alpha_1 \leqslant \ldots \leqslant p_n/\alpha_n$. Podstaw $k := 1$ oraz

$$F_0(a) := \begin{cases} 0, & \text{jeżeli } a = 0; \\ +\infty, & \text{w przeciwnym przypadku.} \end{cases}$$

Krok 2. (Rekursja). Dla każdego $a = 0, 1, \ldots, \sum_{j=1}^{k} p_j$ wylicz:

$$F_k(a) := \min \begin{cases} F_{k-1}(a-p_k) + \alpha_k(a-p_k) + \theta p_k, \\ F_{k-1}(a) + \min\{\beta_k, (\gamma+\theta)p_k\}. \end{cases}$$

Jeżeli $k = n$, to idź do Kroku 3; w przeciwnym przypadku podstaw $k := k+1$ i powtórz Krok 2.

Krok 3. (Rozwiązanie optymalne). Wyznacz optymalną wartość kryterium:

$$F^* := \min\left\{F_n(a) : a = 0, 1, \ldots, \sum_{j\in\mathbf{J}} p_j\right\}$$

oraz skonstruuj odpowiadające rozwiązanie optymalne metodą przeglądu wstecznego (ang. backtracking).

Własność 9.8. *Algorytm* **A5** *rozwiązuje optymalnie problem* **P5** *w czasie* $O\left(n\left(\sum p_j\right)\right)$.

Dowód. Najpierw przeanalizujmy czas działania algorytmu. Złożoność obliczeniowa algorytmu wynika bezpośrednio z rozmiaru przestrzeni stanów. Skoro $k = 1, 2, \ldots, n$ i $a = 0, 1, \ldots, \sum_{j=1}^{n} p_j$, to złożoność obliczeniowa algorytmu **A5** wynosi $O(n(\sum_{j=1}^{n} p_j))$.

Rozważmy parę harmonogramów cząstkowych znajdujących się w tym samym stanie. Harmonogram o mniejszej wartości kryterium będzie nadal miał mniejszą wartość kryterium po rozszerzeniu go o nie uszeregowane zadania, w ten sam sposób, w jaki rozszerzony zostanie drugi harmonogram będący w tym samym stanie. Z powyższego wynika, że do rozszerzenia o kolejne nie uszeregowane zadania należy wybrać spośród harmonogramów będących w tym samym stanie, harmonogram o najmniejszej wartości kryterium.

Optymalność algorytmu **A5** wynika bezpośrednio z powyższych rozważań oraz zasady optymalności ogólnej metody programowania dynamicznego. □

9.6 Podsumowanie rozdziału

W niniejszym rozdziale analizowany był jednoprocesorowy problem szeregowania zadań z doborem pożądanego przedziału zakończenia ich wykonywania. Minimalizacji podlegały: suma ważonych przyspieszeń wykonania zadań, ważona liczba opóźnionych zadań oraz kary związane z szerokością i końcem tego przedziału.

Wykazano wiele własności rozwiązania optymalnego oraz udowodniono, poprzez transformację wielomianową z klasycznego problemu podziału, że problem ten jest NP-trudny. Na podstawie udowodnionych własności, skonstruowano pseudowielomianowy algorytm oparty na metodzie programowania dynamicznego, który rozwiązuje optymalnie ten problem.

Rozdział 10

Podsumowanie pracy

Niniejsza praca została poświęcona problemom szeregowania z optymalnym doborem pożądanego przedziału zakończenia wykonywania zadań. Badane były zarówno problemy jednoprocesorowe jak i wieloprocesorowe. W rozważanych problemach minimalizacji podlegały następujące kryteria:

- $\sum(\alpha E_j + \beta T_j) + f_W(d-e) + \theta e$ - ważona suma nieterminowości wykonania zadań oraz kosztów związanych z położeniem i szerokością pożądanego przedziału zakończenia wykonywania zadań (f_W - dowolna wypukła funkcja niemalejąca);
- $\sum(f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e)$ - suma kosztów nieterminowości wykonania zadań opisanych dowolnymi identycznymi dla wszystkich zadań funkcjami niemalejącymi oraz kosztów związanych z szerokością i położeniem pożądanego przedziału zakończenia wykonywania zadań (f_E, f_T, f_W i f_D - dowolne funkcje niemalejące);
- $\sum(\alpha_j E_j + \beta_j T_j) + \gamma(d-e)$ - suma ważonych nieterminowości wykonania zadań oraz liniowego kosztu związanego z szerokością pożądanego przedziału zakończenia wykonywania zadań;
- $\sum(\alpha_j V_j + \beta_j U_j) + \theta d + \gamma(d-e)$ - suma ważonej liczby nieterminowo wykonanych zadań oraz liniowych kosztów związanych z szerokością i położeniem pożądanego przedziału zakończenia wykonywania zadań;
- $\sum(\alpha_j E_j + \beta_j U_j) + \gamma(d-e) + \theta d$ - suma ważonych przyspieszeń wykonania zadań, ważonej liczby opóźnionych zadań oraz liniowych kar związanych z szerokością i położeniem pożądanego przedziału zakończenia wykonywania zadań.

Analizowane problemy oraz uzyskane dla nich rezultaty zostały wyszczególnione poniżej.

- Jednoprocesorowy problem szeregowania z doborem początku i końca pożądanego przedziału zakończenia wykonywania zadań przy zadanym dolnym i górnym

ograniczeniu na szerokość tego przedziału, z następującym kryterium optymalizacji: $\sum(\alpha E_j + \beta T_j) + f_W(d-e) + \theta e$ (problem **P1**)

  - Skonstruowano wielomianowy algorytm, o złożoności $O(n \log n + \log p_{max})$, który rozwiązuje optymalnie ten problem.

- Wieloprocesorowy problem szeregowania z doborem początku i końca pożądanego przedziału zakończenia wykonywania zadań przy zadanym dolnym i górnym ograniczeniu na szerokość tego przedziału, z następującym kryterium optymalizacji: $\sum(f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e)$ (problem **P2**)
  - Wykazano, że problem ten jest silnie NP-trudny a jego jednoprocesorowy przypadek szczególny jest NP-trudny w zwykłym sensie.
  - Skonstruowano ponadwielomianowy algorytm rozwiązujący optymalnie ten problem oraz pseudowielomianowy algorytm rozwiązujący optymalnie jednoprocesorowy przypadek szczególny.

- Wieloprocesorowy problem szeregowania z doborem początku i końca pożądanego przedziału zakończenia wykonywania zadań przy zadanym dolnym i górnym ograniczeniu na szerokość tego przedziału, z następującym kryterium optymalizacji: $\sum(\alpha_j E_j + \beta_j T_j) + \gamma(d-e)$ (problem **P3**)
  - Wykazano, że problem ten jest silnie NP-trudny a jego jednoprocesorowy przypadek szczególny jest NP-trudny w zwykłym sensie.
  - Skonstruowano ponadwielomianowy algorytm rozwiązujący optymalnie ten problem oraz pseudowielomianowy algorytm rozwiązujący optymalnie jednoprocesorowy przypadek szczególny.
  - Skonstruowano w pełni wielomianowy schemat aproksymacyjny dla szczególnego przypadku z pojedynczym procesorem, ze zgodnymi ilorazami wag oraz bez ograniczeń na szerokość pożądanego przedziału zakończenia wykonywania zadań.
  - Skonstruowano wielomianowy algorytm, o złożoności $O\left(\frac{n^4}{m}\right)$, który rozwiązuje optymalnie szczególny przypadek tego problemu z jednostkowymi czasami wykonywania zadań.
  - Skonstruowano wielomianowy algorytm, o złożoności $O(n^3)$, który rozwiązuje optymalnie szczególny przypadek tego problemu z pojedynczym procesorem oraz jednostkowymi czasami wykonywania zadań.
  - Skonstruowano wielomianowy algorytm, o złożoności $O(n \log n)$, który rozwiązuje optymalnie szczególny przypadek tego problemu z jednostkowymi czasami

wykonywania zadań oraz symetrycznymi kosztami jednostkowymi nieterminowego wykonanie zadań, tzn. $\alpha_j = \beta_j$.

- Jednoprocesorowy problem szeregowania z doborem początku i końca pożądanego przedziału zakończenia wykonywania zadań przy następującym kryterium optymalizacji: $\sum(\alpha_j E_j + \beta_j T_j) + \gamma(d - e)$ (problem **P4.1**)
  - Skonstruowano wielomianowy algorytm, o złożoności $O(n)$, który rozwiązuje optymalnie ten problem.
- Jednoprocesorowy problem szeregowania z doborem początku oraz zadaną szerokością pożądanego przedziału zakończenia wykonywania zadań przy następującym kryterium optymalizacji: $\sum(\alpha_j E_j + \beta_j T_j) + \gamma(d - e)$ (problem **P4.2**)
  - Skonstruowano pseudowielomianowy algorytm, który rozwiązuje optymalnie ten problem.
  - Skonstruowano w pełni wielomianowy schemat aproksymacyjny dla tego problemu.
- Jednoprocesorowy problem szeregowania z doborem początku i końca pożądanego przedziału zakończenia wykonywania zadań przy następującym kryterium optymalizacji: $\sum(\alpha_j E_j + \beta_j U_j) + \gamma(d - e) + \theta d$ (problem **P5**)
  - Udowodniono NP-trudność tego problemu poprzez skonstruowanie wielomianowej transformacji z klasycznego problemu podziału.
  - Skonstruowano pseudowielomianowy algorytm, który rozwiązuje optymalnie ten problem.

Do **najważniejszych rezultatów pracy** można zaliczyć:

- Skonstruowanie wielomianowego algorytmu o niskiej złożoności obliczeniowej, który rozwiązuje optymalnie problem **P1**.
- Skonstruowanie pseudowielomianowego algorytmu rozwiązującego optymalnie jednoprocesorowy przypadek szczególny problemu **P2**.
- Skonstruowanie szybkich wielomianowych algorytmów optymalnych dla szczególnych przypadków problemu **P3** z jednostkowymi czasami wykonywania zadań.
- Skonstruowanie w pełni wielomianowego schematu aproksymacyjnego dla problemu **P4.2**.
- Wykazanie NP-trudności problemu **P5** poprzez skonstruowanie wielomianowej transformacji z klasycznego problemu podziału.

Tabela 10.1: Podsumowanie wyników uzyskanych w rozprawie

Lp.	**Problem**	**Złożoność obliczeniowa**	**Algorytmy**
1.	$1 \mid \langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max} \mid \sum(\alpha E_j + \beta T_j) + f_W(d-e) + \theta e$	P	alg. opt. $O(n \log n + \log p_{max})$
2.	$1 \mid \langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max} \mid \sum(f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e)$	NP-trudny	alg. opt. $O(n(\sum p_j)^2)$
3.	$P \mid \langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max} \mid \sum(f_E(E_j) + f_T(T_j)) + f_W(d-e) + f_D(e)$	s. NP-trudny	alg. opt. $O\left(n\left(\frac{\sum p_j}{m} + p_{max}\right)^{3m-1}\right)$
4.	$1 \left\vert \langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max}; \frac{p_j}{\alpha_j} \uparrow \frac{p_j}{\beta_j} \uparrow \right\vert \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d-e)$	NP-trudny	alg. opt. $O(n^2(\sum p_j)^2)$
5.	$P \left\vert \langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max}; \frac{p_j}{\alpha_j} \uparrow \frac{p_j}{\beta_j} \uparrow \right\vert \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d-e)$	s. NP-trudny	alg. opt. $O\left(n^{m+1}\left(\frac{\sum p_j}{m} + p_{max}\right)^{3m-1}\right)$
6.	$1 \left\vert \langle e,d\rangle; \frac{p_j}{\alpha_j} \uparrow \frac{p_j}{\beta_j} \uparrow \right\vert \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d-e)$	NP-trudny	w pełni wiel. sch. aproks.
7.	$1 \mid \langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max}; p_j = 1 \mid \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d-e)$	P	alg. opt. $O(n^3)$
8.	$P \mid \langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max}; p_j = 1 \mid \sum(\alpha_j E_j + \beta_j T_j) + \gamma(d-e)$	P	alg. opt. $O\left(\frac{n^4}{m}\right)$
9.	$P \mid \langle e,d\rangle; D_{min} \leqslant d-e \leqslant D_{max}; p_j = 1 \mid \sum(\alpha_j E_j + \alpha_j T_j) + \gamma(d-e)$	P	alg. opt. $O(n \log n)$
10.	$1 \mid \langle e,d\rangle \mid \sum(\alpha_j V_j + \beta_j U_j) + \theta d + \gamma(d-e)$	P	alg. opt. $O(n)$
11.	$1 \mid \langle e,e+D\rangle \mid \sum(\alpha_j V_j + \beta_j U_j) + \theta d$	NP-trudny	alg. opt. $O(n(D+p_{max}))$; w pełni wiel. sch. aproks.
12.	$1 \mid \langle e,d\rangle \mid \sum(\alpha_j E_j + \beta_j U_j) + \gamma(d-e) + \theta d$	NP-trudny	alg. opt. $O\left(n\left(\sum p_j\right)\right)$

W Tabeli 10 znajduje się podsumowanie rezultatów uzyskanych w niniejszej rozprawie. W pierwszym problemie f_W oznacza dowolną wypukłą funkcje niemalejącą. W drugim i trzecim problemie f_E, f_T, f_W oraz f_D oznaczają dowolne funkcje niemalejące. W czwartej kolumnie użyto następujących skrótów:

- "alg. opt." oznacza "algorytm optymalny";
- "w pełni wiel. sch. aproks." oznacza "w pełni wielomianowy schemat aproksymacyjny".

Z przedstawionego podsumowania wynika, że postawione w pracy cele zostały w pełni zrealizowane. Otrzymano szereg rezultatów teoretycznych, które są cenne w odniesieniu do dorobku naukowego prezentowanego w światowej literaturze przedmiotu. Skonstruowano wiele dokładnych algorytmów rozwiązania o złożoności zarówno wielomianowej jak i pseudowielomianowej. Przy tworzeniu tych drugich wykorzystano metodę programowania dynamicznego. Ponadto zostały skonstruowane dwa w pełni wielomianowe schematy aproksymacyjne. Liczne wykazane własności rozwiązania optymalnego mogą być w późniejszym etapie prac wykorzystane do konstrukcji efektywnych algorytmów przybliżonych. Dla wszystkich rozpatrywanych w niniejszej pracy problemów została ustalona ich złożoność obliczeniowa.

Podjęcie innych kierunków badań związanych z problemami szeregowania, w których występuje dobór pożądanych przedziałów zakończenia wykonywania zadań, uzależnione jest od potrzeb kreowanych przez otaczające nas realia. Badania te mogą dotyczyć innych kryteriów minimalizacji, bądź problemów, w których przedział ten jest wspólny tylko dla zadań wewnątrz pewnej grupy. Badania te mogą być również ukierunkowane na przypadki, w których zadania są podzielne lub na zadania zostały nałożone ograniczenia kolejnościowe w ich wykonywaniu. Warte rozpatrzenia są również problemy, w których występują procesy wielostadialne (np.: przepływowe, gniazdowe).

Niniejsza praca doktorska była w części finansowana ze środków budżetowych na naukę w latach 2005-2007 jako projekt badawczy nr 3 T11F 029 28.

Bibliografia

[1] G. I. Adamopoulos, C. P. Pappis. Scheduling jobs with different, job dependent earliness and tardiness penalties using the SLK method. *European Journal of Operational Research*, 88:334–344, 1996.

[2] G. I. Adamopoulos, C. P. Pappis. Scheduling under a common due-date on parallel unrelateed machines. *European Journal of Operational Research*, 105:494–501, 1998.

[3] B. Alidaee. Optimal assignment of slack due-dates and sequencing in a single machine. *Applied Mathematics Letters*, 4:9–11, 1991.

[4] B. Alidaee, S.S. Panwalkar. Single stage minimum absolute lateness problem with a common due date on non-identical machines. *Journal of the Operational Research Society*, 44:29–36, 1993.

[5] M. Azizoglu, S. Webster. Scheduling about an unrestricted common due window with arbitrary earliness/tardiness penalty rates. *IIE Transactions*, 29:1001–1006, 1997.

[6] A. Bachman, A. Janiak, A. Kozik, M. Winczaszek. Jednomaszynowy problem szeregowania zadań z potęgowymi funkcjami zmiany wartości zadań. *Zeszyty Naukowe Politechniki Śląskiej, s. Automatyka*, 134:2–12, 2002.

[7] A. Bachman, A. Janiak, A. Kozik, M. Winczaszek. Przybliżone algorytmy rozwiązywania jednomaszynowego problemu szeregowania zadań o zmiennych wartościach. *Zeszyty Naukowe Politechniki Śląskiej, s. Automatyka*, 134:13–22, 2002.

[8] U. Bagchi, R.S. Sullivan, Y.L Chang. Minimizing mean absolute deviations of completion times about a common due date. *Naval Research Logistics*, 33:227–240, 1986.

[9] U. Bagchi, R.S Sullivan, Chang Y.L. Minimizing mean squared deviation of completion times about a common due date. *Management Science*, 33:894–906, 1987.

[10] U. Bagchi, Chang Y.L., R.S Sullivan. Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties and a common due date. *Naval Research Logistics*, 34:739–751, 1987.

[11] K. R. Baker, G. D. Scudder. Sequencing with earliness and tardiness penalties: a review. *Operations Research*, 38:22–36, 1990.

[12] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[13] Z.-L. Chen, C.-Y. Lee. Parallel machine scheduling with a common due window. *European Journal of Operational Research*, 136:512–527, 2002.

[14] T.C.E. Cheng. Minimizing the maximum deviation of job completion time about a common due-date. *Computers and Mathematics with Applications*, 14:279–283, 1987.

[15] T.C.E. Cheng. Optimal common due-date with limited completion times deviation. *Computers and Operations Research*, 15:91–96, 1988.

[16] T.C.E. Cheng. Optimal assignment of slack due-dates and sequencing in a single machine shop. *Applied Mathematics Letters*, 2:333–335, 1989.

[17] T.C.E. Cheng. A note on a partial search algorithm for the single-machine optimal common due-date assignment and sequencing problem. *Computers and Operations Research*, 17:321–324, 1990.

[18] Chen Z.-L. Cheng T.C.E. Parallel-machine scheduling problems with earliness and tardiness penalties. *Journal of the Operational Research Society*, 45:685–695, 1994.

[19] P. De, J.B. Ghosh, C.E. Wells. A note on the minimization of mean squared deviation of completion times about a common due date. *Management Science*, 35:1143–1147, 1989.

[20] P. De, J.B. Ghosh, C.E. Wells. CON due-date determination and sequencing. *Computers and Operations Research*, 17:333–342, 1990.

[21] P. De, J.B. Ghosh, C.E. Wells. Optimal delivery time quotation and order sequencing. *Decision Science*, 22:379–390, 1991.

[22] P. De, J.B. Ghosh, C.E. Wells. On the minimization of completion time variance with a bicriteria extension. *Operations Research*, 40:1148–1155, 1992.

[23] P. De, J.B. Ghosh, C.E. Wells. On the general solution for a class of early/tardy problems. *Computers and Operations Research*, 20:141–149, 1993.

[24] P. De, J.B. Ghosh, C.E. Wells. Due-date assignment and early/tardy scheduling on identical parallel machines. *Naval Research Logistics*, 41:17–32, 1994.

[25] P. De, J.B. Ghosh, C.E. Wells. Solving a generalized model for CON due date assignment and sequencing. *International Journal of Production Economics*, 34:179–185, 1994.

[26] P. Dileepan. Common due date scheduling problem with separate earliness and tardiness penalties. *Computers and Operations Research*, 20:179–181, 1993.

[27] H. Emmons. Scheduling to a common due date on parallel uniform processors. *Naval Research Logistics*, 34:803–810, 1987.

[28] M.R. Garey, D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completness*. W.H. Freeman and Co., 1979.

[29] G.V. Gens, E.V. Levner. Fast approximation algorithm for job sequencing with deadlines. *Discrete Applied Mathematics*, 3:313–318, 1981.

[30] V. Gordon, J.-M. Proth, C. Chu. A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139:1–25, 2002.

[31] V. S. Gordon, J.-M. Proth, C. Chu. Due date assignment and scheduling: SLK, TWK and other due date assignment models. *Production Planning & Control*, 13:117–132, 2002.

[32] V. S. Gordon, V. A. Strusevich. Earliness penalties on a single machine subject to precedence constraints: SLK due date assignment. *Computers & Operations Research*, 26:157–177, 1999.

[33] V.S. Gordon. A note on optimal assignment of slack due-dates in single-machine scheduling. *European Journal of Operational Research*, 70:311–315, 1993.

[34] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematic*, 5:287–326, 1979.

[35] M.C. Gupta, Y.P. Gupta, A. Kumar. Minimizing flow time variance in a single machine system using genetic algorithms. *European Journal of Operational Research*, 70:289–303, 1993.

[36] Y.P. Gupta, C.R. Bector, M.C. Gupta. Optimal schedule on a single machine using various due date determination methods. *Computers in Industry*, 15:245–253, 1990.

[37] N. G. Hall, W. Kubiak, S. P. Sethi. Earliness-tardiness scheduling problems, II: deviation of completion times about a restrictive common due date. *Operations Research*, 39:847–856, 1991.

[38] N. G. Hall, M. E. Posner. Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date. *Operations Research*, 39:836–846, 1991.

[39] Q. Hao, Z. Yang, D. Wang, Z. Li. Common due date determination and sequencing using tabu search. *Computers and Operations Research*, 23:409–417, 1996.

[40] G.H. Hardy, J.E. Littlewood, G. Polya. *Inequalities*. Cambridge University Press, New York, 1934.

[41] J. A. Hoogeveen, H. Oosterhout ans S. L. van de Velde. New lower and upper bounds for scheduling around a small common due date. *Operations Research*, 42:102–111, 1994.

[42] J. A. Hoogeveen, S. L. van de Velde. Scheduling around a small common due date. *European Journal of Operational Research*, 55:237–242, 1991.

[43] J.R. Jackson. Scheduling a production line to minimize maximum tardiness. Management Science Research Project, UCLA, 1955.

[44] R. J. W. James. Using tabu search to solve the common due date early/tardy machine scheduling problem. *Computers Operations Research*, 24:199–208, 1997.

[45] A. Janiak, M.Y. Kovalyov, M. Marek. Soft due window assignment and scheduling on parallel machines. Accepted for publication in IEEE Transactions on Systems, Man, and Cybernetics, Part A.

[46] A. Janiak, A. Kozik, M. Winczaszek. Effective algorithms for a single processor scheduling problem with detoriarating job values. *Proceedings of 9th IEEE International Conference on Methods and Models in Automation and Robotics*, strony 1145–1148, Międzyzdroje, Polska, 25-28 Sierpień 2003.

[47] A. Janiak, T. Krysiak, M. Winczaszek. A single processor scheduling problem with a step function of change of job value. *Proceedings of 9th IEEE International Conference on Methods and Models in Automation and Robotics*, strony 1189–1191, Międzyzdroje, Polska, 25-28 Sierpień 2003.

[48] A. Janiak, M. Marek. *Jednomaszynowy problem szeregowania z optymalizowanymi przedziałamia zakończenia wykonywania zadań*, strony 145–155. Zeszyty Naukowe Politechniki Śląskiej, Seria Automatyka, Zeszyt 129. Gliwice, 2000.

[49] A. Janiak, M. Marek. *Optymalny dobór przedziałów czasowych zakończenia wykonywania zadań w wybranych jednomaszynowych problemach szeregowania*, strony 269–276. Zeszyty Naukowe AGH, Seria Automatyka, Tom 5, Zeszyt 1/2. Kraków, 2001.

[50] A. Janiak, M. Marek. *Wielomaszynowy problem szeregowania z optymalizowanymi przedziałami czasowymi zakończenia wykonywania zadań*, strony 277–281. Zeszyty Naukowe AGH, Seria Automatyka, Tom 5, Zeszyt 1/2. Kraków, 2001.

[51] A. Janiak, M. Marek. Multi-machine scheduling problem with optimal due interval assignment subject to generalized sum type criterion. *Operations Research Proceedings 2001*, strony 207–212. Springer-Verlag, 2002.

[52] A. Janiak, M. Marek. Multi-processor scheduling problem with due interval assignment subject to generalized min-max type criterion. *Proceedings of the 9th IEEE Inter. Confer. on Methods and Models in Automation and Robotics*, wolumen 2, strony 1169–1173, Miedzyzdroje, Poland, 25-28 August 2003.

[53] A. Janiak, M. Marek. Scheduling problems with optimal due interval assignment subject to some generalized criteria. *Operations Research Proceedings 2002*, strony 142–147. Springer-Verlag, 2003.

[54] A. Janiak, M. Marek. Parallel processor scheduling problems with optimal due interval assignment. *Proceedings of 3th International Conference on Parallel Computing Systems (PCS 2004)*, strony 383–387, Colima, Meksyk, 19 - 22 Wrzesień 2004. IEEE Press.

[55] A. Janiak, M. Marek. Property of symmetry for some single processor scheduling problems with due interval assignment. *Systems Science*, 30:97–107, 2004.

[56] A. Janiak, M. Winczaszek. An optimal algorithm for a single processor scheduling problem with a common due window. *Proceedings of 9th IEEE International Conference on Methods and Models in Automation and Robotics*, strony 1213–1216, Międzyzdroje, Polska, 25-28 Sierpień 2003.

[57] A. Janiak, M. Winczaszek. A single processor scheduling problem with a common due window assignment. *Operations Research Proceedings 2004, Selected Papers of the International Conference on Operations Research (OR2004)*, strony 213–220, Tilburg, Holandia, 1-3 Wrzesień 2004.

[58] A. Janiak, M. Winczaszek. Szeregowanie zadań z nieliniową funkcją kar za nieterminowość. Z. Bubnicki, O. Hryniewicz, J. Węglarz, redaktorzy, *Badania operacyjne i systemowe 2004. Zastosowania*, strony 261–270. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2004.

[59] A. Janiak, M. Winczaszek. Szeregowanie zadań z potęgowym modelem zmiany ich wartości. Z. Bubnicki, O. Hryniewicz, J. Węglarz, redaktorzy, *Badania operacyjne i systemowe 2004. Zastosowania*, strony 271–282. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2004.

[60] A. Janiak, M. Winczaszek. Common due window assignment in parallel processor scheduling problem. *Proceedings of the 11th IEEE International Conference on Methods and Models in Automation and Robotics*, strony 1079–1083, Międzyzdroje, Polska, 29 Sierpień - 1 Wrzesień 2005.

[61] A. Janiak, M. Winczaszek. Optimal algorithm for parallel processor scheduling problem with due window assignment. *Proceedings of the 11th IEEE International Conference on Methods and Models in Automation and Robotics*, strony 1085–1089, Międzyzdroje, Polska, 29 Sierpień - 1 Wrzesień 2005.

[62] A. Janiak, M. Winczaszek. Common due window assignment in parallel processor scheduling problem with nonlinear penalty functions. *Lecture Notes in Copmuter Science*, 3911:132–139, 2006.

[63] B. Jurish, W. Kubiak, J. Józefowska. Algorithms for miniclique scheduling problems. *Discrete Applied Mathematics*, 72:115–139, 1997.

[64] H.G. Kahlbacher. SWEAT - a program for a scheduling problem with earliness and tardiness penalties. *European Journal of Operational Research*, 43:111–112, 1989.

[65] H.G. Kahlbacher. Scheduling with monotonous earliness and tardiness penalties. *European Journal of Operational Research*, 64:258–277, 1991.

[66] H.G. Kahlbacher, T.C.E. Cheng. Parallel machine scheduling to minimize costs for earliness and number of tardy jobs. *Discrete Applied Mathematics*, 47:139–164, 1993.

[67] J. J. Kanet. Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly*, 28:643–651, 1981.

[68] N.I. Karacapilidis, C.P. Pappis. Optimal due date determination and sequencing of n jobs on a single machine using SLK method. *Computers in Industry*, 21:762–770, 1993.

[69] N.I. Karacapilidis, C.P. Pappis. Form similarities of the CON and SLK due date determination methods. *Journal of the Operational Research Society*, 46:762–770, 1995.

[70] C. Koulamas. The total tardiness problem: Review and extensions. *Operations Research*, 42:1025–1041, 1994.

[71] C. Koulamas. Single-machine scheduling with time windows and earliness/tardiness penalties. *European Journal of Operational Research*, 91:190–202, 1996.

[72] C. Koulamas. Maximizing the weighted number of one-time jobs in a single machine scheduling with time windows. *Mathematical Computer Modelling*, 25:57–62, 1997.

[73] M. Y. Kovalyov. Improving the complexities of approximation algorithms for optimization problems. *Operations Research Letters*, 17:85–87, 1995.

[74] M. Y. Kovalyov, W. Kubiak. A fully polynomial approximation scheme for the weighted earliness-tardiness problem. *Operations Research*, 42:757–761, 1999.

[75] F.-J. Kramer, C.-Y. Lee. Due window scheduling for parallel machines. *Mathematical and Computer Modeling*, 20:69–89, 1994.

[76] W. Kubiak. Completion time variance minimization on a single machine is difficult. *Operations Research Letters*, 14:49–59, 1993.

[77] W. Kubiak, S. Lou, S. Sethi. Equivalence of mean flow time problems and mean absolute deviation problems. *Operations Research Letters*, 9:371–374, 1990.

[78] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart & Winston, New York, 1976.

[79] E.L. Lawler, J.M. Moore. A functional equation and its application to resource application and resource application and sequenceing problems. *Management Science*, 16:77–84, 1969.

[80] C. Y. Lee, S. J. Kim. Parallel genetic algorithms for the earliness-tardiness job scheduling problem with general penalty weights. *Computers Industrial Engineering*, 28:231–243, 1995.

[81] C.-L. Li, T.C.E. Cheng. The parallel machine min-max weighted absolute lateness scheduling problem. *Naval Research Logistics*, 41:33–46, 1994.

[82] S. D. Liman, S. S. Panwalkar, S. Thongmee. Determination of common due window location in a single processor scheduling problem. *European Journal of Operational Research*, 93:68–74, 1996.

[83] S. D. Liman, S. S. Panwalkar, S. Thongmee. A single machine scheduling problem with common due window and controllable processing times. *Annals of Operations Research*, 70:145–154, 1997.

[84] S. D. Liman, S. S. Panwalkar, S. Thongmee. Common due window size and location determination in a single machine scheduling problem. *Journal of the Operational Research Society*, 49:1007–1010, 1998.

[85] S. D. Liman, S. Ramaswamy. Earliness-tardiness scheduling problems with a common delivery window. *Operations Research Letters*, 15:195–203, 1994.

[86] M. Marek. *Problemy szeregowania z optymalizowanymi przedziałami czasowymi zakończenia wykonywania zadań*. Praca doktorska, Politechnika Wrocławska, Instytut Cybernetyki Technicznej, 2004.

[87] G. Mosheiov. A due-window determination in minmax scheduling problems. *INFOR*, 39:719–732, 2001.

[88] N.G.Hall. Single- and multiple-processor models for minimizing completion time variance. *Naval Research Logistics Quarterly*, 33:49–54, 1986.

[89] S. S. Panwalkar, M. L. Smith, A. Seidmann. Common due date assignment to minimize total penalty for the one machine scheduling problem. *Operations Research*, 30:391–399, 1982.

[90] X. Qi, F.-S. Tu. Scheduling a single machine to minimize earliness penalties subject to the SLK due-date determination method. *European Journal of Operational Research*, 105:502–508, 1998.

[91] M. Raghavachari. Scheduling problems with non-regular penalty functions - a review. *Operations Research*, 25:144–164, 1988.

[92] A. Seidmann, S.S. Panwalkar. amd M.L. Smith. Optimal assignment of due-dates for a single processor scheduling problem. *International Journal of Production Research*, 19:393–399, 1981.

[93] T. Sen, S.K. Gupta. A stat-of-art survey of static scheduling research involving due dates. *OMEGA*, 12:63–76, 1984.

[94] J. B. Sidney. Optimal sinle-machine scheduling with earliness and tardiness penalties. *Operations Reseach*, 25:62–69, 1977.

[95] C. Smutnicki. *Optimization and control in Just-In-Time manufacturing systems*. Monografie. Wydawnictwo Politechniki Wrocławskiej, Wrocław, 1997.

[96] P.S. Sundararaghavan, M.U. Ahmed. Minimizing the sum of absolute lateness in single-machine and multimachine scheduling. *Naval Research Logistics Quarterly*, 31:325–333, 1984.

[97] W. Szwarc. Single-machine scheduling to minimize absolute deviation of completion times from a common due date. *Naval Research Logistics*, 36:663–673, 1989.

[98] J. A. Ventura, M. X. Weng. An improved dynamic programming algorithm for the single-machine mean absolute deviation problem with a restrictive common due date. *Operations Research Letters*, 17:149–152, 1995.

[99] G. Wan, B. P.-C. Yen. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, 142:271–281, 2002.

[100] M.X. Weng, J.A. Ventura. Scheduling about a large common due-date with tolerance to minimize mean absolute deviation of completion times. *Naval Research Logistics*, 41:843–851, 1994.

[101] Y. Wilamowsky, S. Epstein, B. Dickman. Optimal common due-date completion time tolerance. *Computers and Operations Research*, 23:1203–1210, 1996.

[102] Y. Wu, D. Wang. Optimal single-machine scheduling about a common due window with earliness/tardiness and additional penalties. *International Journal of Systems Sciences*, 30:1279–1284, 1999.

[103] W. K. Yeung, C. Oguz, T. C. E. Cheng. Minimizing weighted number of early and tardy jobs with a common due window involving location penalty. *Annals of Operations Research*, 108:33–54, 2001.

[104] W. K. Yeung, C. Oguz, T. C. E. Cheng. Single-machine scheduling with a common due window. *Computers & Operations Research*, 28:157–175, 2001.

[105] W. K. Yeung, C. Oguz, T. C. E. Cheng. Two-stage flowshop earliness and tardiness machine scheduling involving a common due window. *International Journal of Production Economics*, 90:421–434, 2004.

[106] W.-S. Yoo, L. A. Martin-Vega. Scheduling single-machine problems for on-time delivery. *Computers & Industrial Engineering*, 39:371–392, 2001.