Politechnika Rzeszowska
Wydział Elektrotechniki i Informatyki
Katedra Informatyki i Automatyki

Laboratorium

Sterowanie procesami dyskretnymi

Stanowisko 3

Algorytmy harmonogramowania zadań
– pakiet LiSA

Rzeszów 2012

1. Wprowadzenie

W teorii szeregowania zadań do reprezentacji problemów harmonogramowania najczęściej używa się tzw. klasyfikacji Grahama. Polega ona na opisaniu problemu harmonogramowania za pomocą wyrażenia złożonego z trzech pól: $\alpha \mid \beta \mid \gamma$, których interpretacja jest następująca:

- α – **opisuje środowisko maszynowe**, tzn. ile maszyn (ogólnie „procesorów" przetwarzających zadania) jest w systemie, jakie są ich parametry i wzajemne powiązania; to pole nie może być puste,
- β – **reprezentuje dodatkowe ograniczenia** w systemie, inne niż te wynikające bezpośrednio ze struktury środowiska maszynowego, określonej polem α; to pole może być puste, jeśli nie ma dodatkowych ograniczeń,
- γ – **reprezentuje funkcję celu** (funkcję kryterialną), umożliwiającą ocenę porównawczą różnych wariantów harmonogramu i wybór najlepszego; to pole nie może być puste.

Wybrane, najczęściej spotykane wartości przypisywane polom α, β, γ zostały podane i omówione poniżej:

- **wartości pola α** (podane niżej warianty wzajemnie się wykluczają, może wystąpić tylko jeden z nich):
  - **1** – wszystkie zadania przetwarza pojedyncza maszyna,
  - **P** – dostępny jest zbiór identycznych maszyn równoległych (*identical machines*), każde zadanie może być przetworzone przez dowolną z maszyn,
  - **Q** – dostępny jest zbiór maszyn równoległych, każde zadanie może być przetworzone przez dowolną z maszyn, przy czym prędkość pracy każdej maszyny jest określana indywidualnie, ale jest jednakowa dla wszystkich zadań (*uniform machines*),

R – dostępny jest zbiór maszyn równoległych, każde zadanie może być przetworzone przez dowolną z maszyn, przy czym prędkość pracy każdej maszyny zależy zarówno od numeru maszyny jak i od numeru zadania (*unrelated machines*),

F – system przepływowy (*flow shop*); każde zadanie przetwarzane jest przez zbiór maszyn w ściśle określonym porządku, identycznym dla wszystkich zadań; konkretna maszyna może przetwarzać zadania w dowolnej kolejności,

J – system gniazdowy (*job shop*); każde zadanie przetwarzane jest przez zbiór maszyn w ściśle określonym porządku, który może być odmienny dla każdego zadań; konkretna maszyna może przetwarzać zadania w dowolnej kolejności,

O – system otwarty (*open shop*); każde zadanie przetwarzane jest przez zbiór maszyn, zarówno kolejność przetwarzania każdego z zadań na poszczególnych maszynach jak i sekwencja przetwarzania kolejnych zadań przez każdą z maszyn mogą być dowolne.

Podsumowanie: Wymienione powyżej rodzaje środowisk maszynowych można podzielić na 3 główne grupy:

- środowisko jednomaszynowe (*single machine problem*): 1,
- środowiska z maszynami równoległymi (*parallel machine problems*): P, Q, R,
- środowiska kolejnościowe (*shop problems*): F, J, O.

- **wartości pola β** (podane niżej warianty nie wykluczają się wzajemnie, może wystąpić wiele wariantów równocześnie, wymieniono tylko kilka ograniczeń spośród wielu rozważanych w teorii szeregowania):

pmtn – wywłaszczalność (*preemption*), operacje mogą być wywłaszczane, tzn. przerywane w czasie przetwarzania i wznawiane później,

$\mathbf{r_j}$ – dla zadań zdefiniowane są terminy/czasy gotowości (*release dates*), zadanie *j* nie może rozpocząć się wcześniej niż w chwili r_j,

d_j – dla zadań zdefiniowane są terminy/czasy pożądanego zakończenia (*due dates*), zdanie *j* powinno zostać ukończone najpóźniej w chwili d_j, ograniczenia te brane są pod uwagę w wyznaczaniu wartości funkcji kryterialnych uwzględniających opóźnienia i spóźnienia zadań (patrz pole γ),

no-wait – opóźnienia (przerwy) pomiędzy kolejnymi operacjami należącymi do tego samego zadania są zabronione,

s_{ijk} – zdefiniowane są czasy przygotowawczo-zakończeniowe (*setup times*), zmiana przygotowania maszyny *i* z w wykonywania zadania *j* do wykonywania zadania *k* zajmuje czas s_{ijk}.

Kryteria harmonogramowania (pole γ) konstruuje się najczęściej w oparciu o następujące parametry:

C_j – czas zakończenia wykonywania zadania *j* (*completion time*),

L_j – opóźnienie (*lateness*) zakończenia zadania *j*, tj. różnica pomiędzy zaplanowanym i pożądanym czasem ukończenia zadania: $L_j = C_j - d_j$,

T_j – spóźnienie (*tardiness*) zakończenia zadania *j*, tj. różnica pomiędzy zaplanowanym i pożądanym czasem ukończenia zadania, naliczana jednak tylko przy wartości dodatniej: $T_j = \max\{0, C_j - d_j\}$,

U_j – opóźnienie jednostkowe zadania *j*, przyjmuje wartość 1 jeśli zadanie jest opóźnione i wartość 0 w przeciwnym wypadku: U_j = if $C_j > d_j$ then 1 else 0.

- **wartości pola γ** (podane niżej warianty wzajemnie się wykluczają, może wystąpić tylko jeden z nich):

Cmax – minimalizacja momentu zakończenia ostatniego zadania (tzw. długość uszeregowania, *makespan*): $C_{max} = \max\{C_j\} \rightarrow \min$,

Lmax – minimalizacja opóźnienia maksymalnego: $L_{max} = \max\{L_j\} \rightarrow \min$,

SumCj – minimalizacja sumy czasów ukończenia wszystkich zadań: $\sum C_j \rightarrow \min$,

SumTj – minimalizacja sumy czasów spóźnień wszystkich zadań: $\sum T_j \rightarrow \min$,

SumUj – minimalizacja liczby spóźnionych zadań: $\sum U_j \rightarrow \min$,

Przykład 1.1.

Pewne przedsiębiorstwo musi zaplanować produkcję sześciu detali metalowych. Produkcja każdego z nich wymaga pięciu operacji (frezowanie, wiercenie, szlifowanie, transport, pakowanie), przy czym szczegółowa kolejność tych operacji jest różna dla różnych produktów (tab. 1.1). Np. detal A podlega kolejno: wierceniu, frezowaniu, szlifowaniu, transportowi i pakowaniu.

	Frezowanie (1)	**Wiercenie (2)**	**Szlifowanie (3)**	**Transport (4)**	**Pakowanie (5)**
Detal A (1)	2	1	3	4	5
Detal B (2)	2	1	3	5	4
Detal C (3)	1	2	3	4	5
Detal D (4)	2	3	1	4	5
Detal E (5)	3	1	2	4	5
Detal F (6)	2	3	1	5	4

Tab. 1.1. Sekwencje operacji technologicznych przykładowego problemu harmonogramowania

	Frezowanie (1)	**Wiercenie (2)**	**Szlifowanie (3)**	**Transport (4)**	**Pakowanie (5)**
Detal A (1)	26	42	11	21	13
Detal B (2)	28	34	9	34	11
Detal C (3)	31	47	13	23	14
Detal D (4)	27	39	8	19	15
Detal E (5)	34	51	17	13	10
Detal F (6)	23	43	11	22	11

Tab. 1.2. Czasy operacji przykładowego problemu harmonogramowania

Dla każdego typu operacji dostępna jest tylko jedna maszyna lub pracownik, który może tę operację wykonać. Czasy poszczególnych operacji zdefiniowane są w Tabeli 1.2.

Dla detali zdefiniowane są terminy gotowości rozpoczęcia produkcji (np. może to być czas dostawy wymaganych materiałów) oraz pożądane daty ukończenia produkcji (tab. 1.3).

	Detal A (1)	**Detal B (2)**	**Detal C (3)**	**Detal D (4)**	**Detal E (5)**	**Detal F (6)**
Terminy gotowości	32	46	27	4	18	21
Pożądane daty ukończenia	215	255	240	230	230	250

Tab. 1.3. Terminy gotowości i pożądane daty ukończenia zadań

Przedsiębiorstwo potrzebuje takiego planu produkcyjnego, który zapewni minimalną liczbę spóźnionych zadań, niezależnie od tego jak duże będą spóźnienia.

Pytanie

Jak sklasyfikować przedstawiony problem harmonogramowania za pomocą notacji Grahama?

Rozwiązanie

Każde zadanie składa się z ciągu następujących po sobie operacji, a więc jest to problem kolejnościowy. Dodatkowo dla poszczególnych zadań kolejność operacji jest ściśle określona i odmienna, jest to więc środowisko maszynowe typu *job shop* ($\alpha = J$). Dla problemu zdefiniowane są dodatkowe ograniczenia w postaci terminów gotowości rozpoczęcia zadań (r_j) oraz pożądanych dat zakończenia (d_j), zatem $\beta = r_j;d_j$. Kryterium harmonogramowania stanowi minimalizacja liczby spóźnionych zadań ($\gamma = SumUj$). Ostatecznie, problem zostanie sklasyfikowany jako: $J \mid r_j;d_j \mid SumUj$.

2. Oprogramowanie LiSA

Oprogramowanie LiSA (*Library of Scheduling Algorithms*) jest pakietem przeznaczonym do rozwiązywania problemów harmonogramowania z użyciem różnych algorytmów. Możliwy jest wybór algorytmów wbudowanych lub implementacja własnych.

2.1. Harmonogramowanie z użyciem oprogramowania LiSA

Rozwiązywanie problemu harmonogramowania z wykorzystaniem oprogramowania LiSA składa się z czterech etapów:

1. Zdefiniowanie typu problemu w notacji Grahama.
2. Wprowadzenie szczegółowych parametrów problemu, wynikających z jego typu (np. czasy operacji, kolejności operacji, pożądane daty ukończenia itp.),
3. Uruchomienie algorytmu harmonogramowania. Można wybierać algorytmy z grupy dokładnych (*exact algorithms*) lub heurystycznych (*heuristic algorithms*).
4. Analiza i interpretacja wyników. Rezultaty mogą być zaprezentowane m.in. w postaci:
   - zasobowo zorientowanego wykresu Gantta,
   - grafu sekwencji,
   - tablicy zestawiającej czasy ukończenia operacji.

   Możliwy jest powrót do punktu 3 w celu powtórzenia procedury harmonogramującej z użyciem innego algorytmu.

2.2. Rozwiązanie przykładowego zadania

Zaprezentowany zostanie sposób rozwiązania z wykorzystaniem pakietu LiSA wcześniej przedstawionego problemu harmonogramowania (przykład 1.1). Główne okno programu LiSA przedstawione jest na rysunku 2.1. Aby zdefiniować nowy problem harmonogramowania należy wybrać z menu głównego opcję **File** → **New**. Ukaże się okno dialogowe definiowania typu

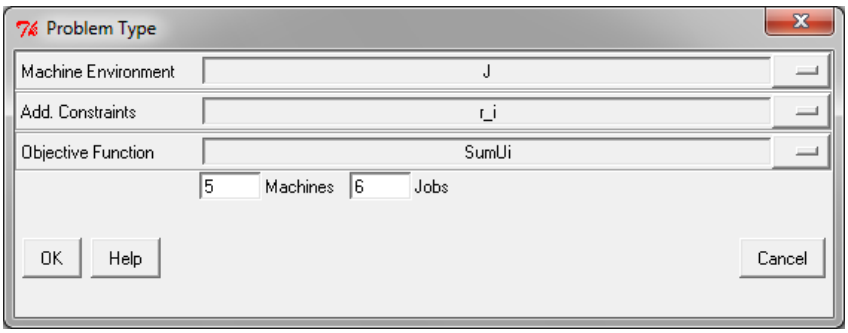
problemu harmonogramowania (rys. 2.2). Należy w nim wprowadzić poszczególne parametry z notacji Grahama: *Machine Environment* – α, *Add. Constraints* – β, *Objective Function* – γ. W oknie tym podaje się także liczbę maszyn (*Machines*) oraz zadań (*Jobs*).

Rys. 2.1. Okno główne programu LiSA

Uwaga

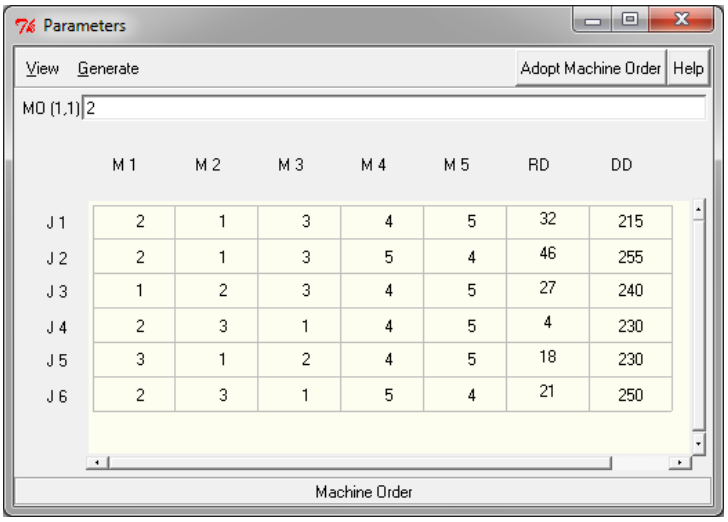
W przypadku problemu harmonogramowania z ograniczeniem d_j (*due dates*), w pakiecie LiSA nie należy przypisywać tego ograniczenia do pola β. Oprogramowanie automatycznie rozpoznaje konieczność wprowadzenia parametrów d_j, jeśli wymaga ich kryterium harmonogramowania.

Po poprawnym wprowadzeniu parametrów określających typ problemu harmonogramowania właściwy dla przykładu 1.1, okno dialogowe powinno być wypełnione jak na rysunku 2.2. Ustawienia należy zaakceptować wciskając OK.



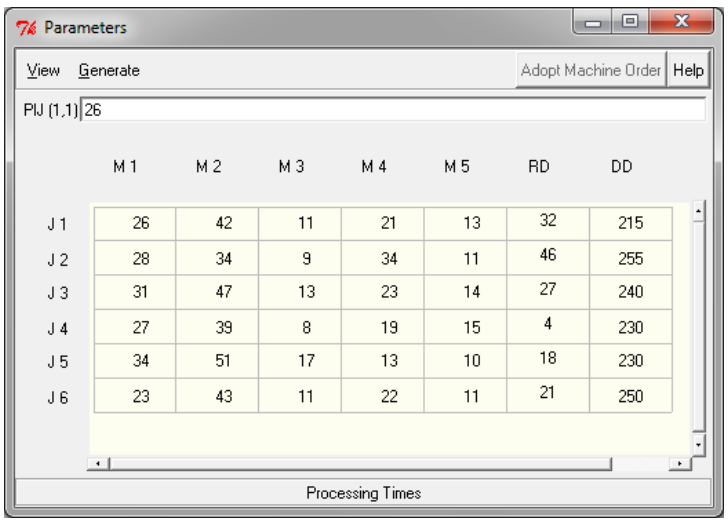
Rys. 2.2. Okno definiowania typu problemu harmonogramowania

Następnie należy wywołać opcję **Edit → Parameters**. Spowoduje to otwarcie okna edycji szczegółowych parametrów problemu harmonogramowania. W oknie tym, po wybraniu **View → Machine Order**, ukaże się macierz reprezentująca kolejność operacji (maszyn) dla poszczególnych zadań (rys. 2.3). Należy ją uzupełnić zgodnie z zawartością tabeli 1.1. Na podstawie tabeli 1.3 należy także uzupełnić kolumny RD (*release dates*) oraz DD (*due dates*).



Rys. 2.3. Okno definiowania kolejności operacji, czasów gotowości oraz pożądanych czasów zakończenia

Następnie, w tym samym oknie, należy wybrać **View → Processing Times**, pojawi się macierz z czasami operacji (rys. 2.4), która powinna zostać uzupełniona na podstawie tabeli 1.2.



Rys. 2.4. Okno definiowania czasów operacji

Po definiowaniu parametrów szczegółowych, okno **Parameters** należy zamknąć. Zdefiniowany problem harmonogramowania należy zapisać wybierając opcję **File → Save as**, używając wybranej przez siebie nazwy i lokalizacji.

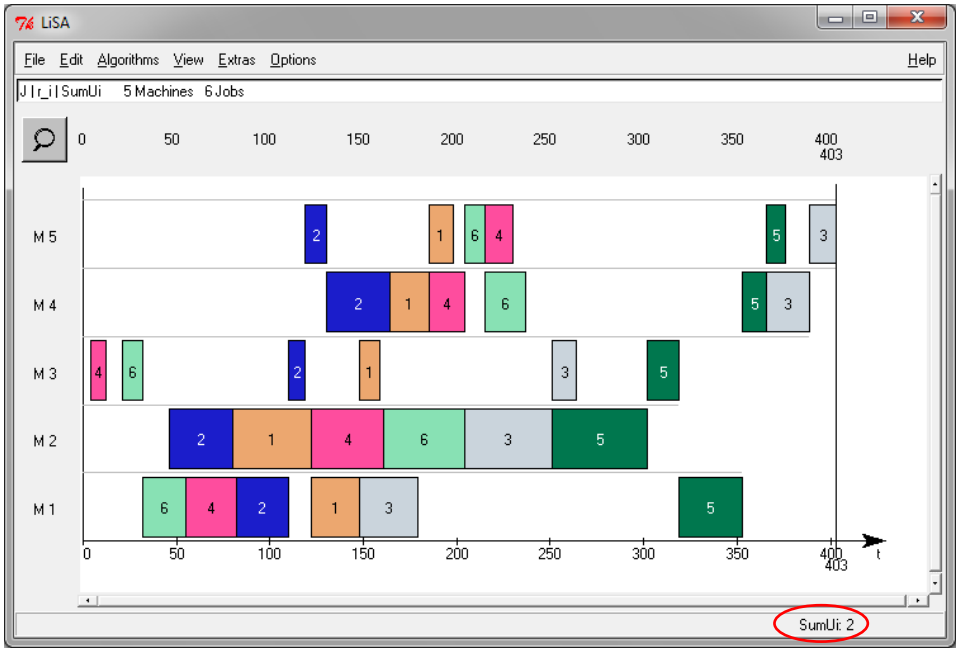
Jeżeli problem harmonogramowania został zdefiniowany poprawnie, w opcjach **Algorithms → Exact Algorithms** oraz **Algorithms → Heuristic Algorithms** powinny być dostępne listy algorytmów harmonogramujących, właściwych dla danego problemu.

Dla przetestowania np. algorytmu szukania z zabronieniami, należy wybrać opcję **Algorithms → Heuristic Algorithms → Tabu Search**. Jeżeli po wykonaniu algorytmu rezultat nie zostanie przedstawiony w postaci wykresu Gantta (rys. 2.5), to należy przełączyć się to tego widoku opcją **View → Gantt Chart**. W przypadku pokazanym na rysunku 2.5, algorytm szukania z zabronieniami zdołał zminimalizować liczbę spóźnionych zadań do 3.

Rys. 2.5. Rezultat harmonogramowania z użyciem algorytmu *Tabu Search* – wykres Gantta

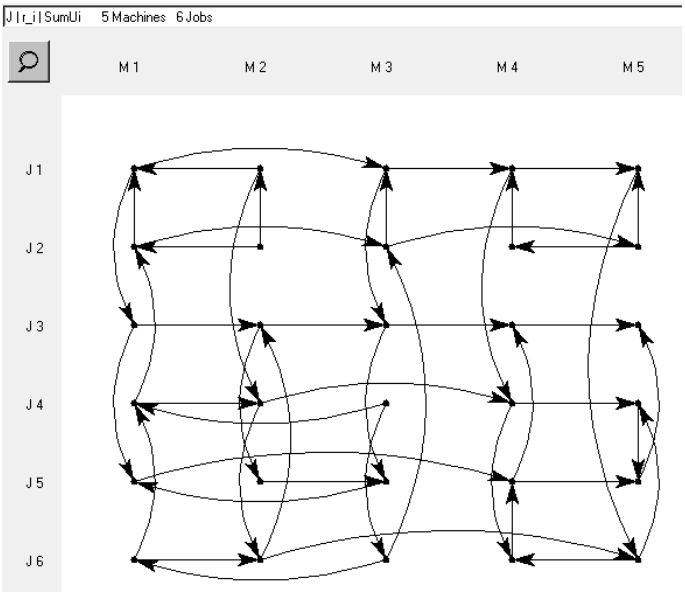
Użycie do tego samego problemu algorytmu dokładnego (w rozważanym przypadku algorytm typu *Branch & Bound*) **Algorithms → Exact Algorithms → Tabu Search → Branch & Bound**, doprowadziło do uzyskania wartości kryterium równego 2 (rys. 2.6). Algorytmy dokładne gwarantują odnalezienie najlepszego (optymalnego) rozwiązania, zatem należy przyjąć, że

żaden inny algorytm heurystyczny nie będzie w stanie wygenerować lepszego wyniku. Wadą algorytmów dokładnych jest gwałtowny wzrost czasu obliczeń wraz z rozmiarem problemu. Ponieważ rozmiar problemu harmonogramowania z przykładu 1.1 jest bardzo mały, algorytm dokładny działa szybko.



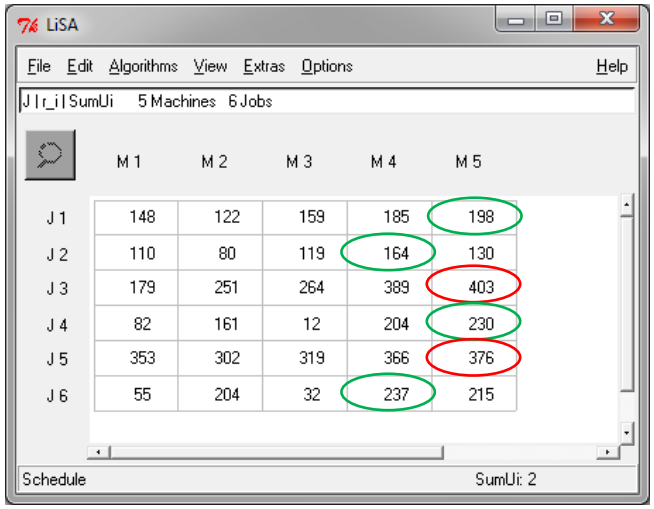
Rys. 2.6. Rezultat harmonogramowania z użyciem algorytmu dokładnego – wykres Gantta

Można dokonać przełączenia widoku (**View → Sequence Graph**), aby zobaczyć rozwiązanie w postaci grafu sekwencji (rys. 2.7).



Rys. 2.7. Rezultat harmonogramowania z użyciem algorytmu dokładnego – graf sekwencji

Inny przydatny widok (**View → Schedule**) przedstawia tablicę zawierającą czasy zakończenia wszystkich operacji (rys. 2.8). Łatwo zauważyć, że dwa opóźnione względem pożądanych czasów zakończenia zadania to zadanie nr 3 oraz zdanie nr 5.



Rys. 2.8. Rezultat harmonogramowania z użyciem algorytmu dokładnego – czasy zakończeń

2.3. Dodatkowe uwagi

W praktycznym użytkowaniu oprogramowania LiSA należy uwzględnić następujące dodatkowe uwagi:

1. Każdorazowo po poprawnym wprowadzeniu opisu problemu harmonogramowania, tj. po uzupełnieniu okien dialogowych **Problem Type** oraz **Parameters**, należy projekt zapisać. Przed każdym zastosowaniem nowego algorytmu harmonogramowania projekt należy na nowo wczytać opcją **File → Open**.
2. Jeżeli listy dostępnych harmonogramów są puste (opcje **Algorithms → Exact Algorithms** oraz **Algorithms → Heuristic Algorithms**), oznacza to, że w pakiecie LiSA nie ma harmonogramu odpowiedniego dla danego problemu.
3. Jeżeli listy dostępnych harmonogramów są nieaktywne (opcje **Algorithms → Exact Algorithms** oraz **Algorithms → Heuristic Algorithms**), oznacza to, że w programie nie zostały wczytane parametry szczegółowe. Należy

skontrolować ustawienie tych parametrów (**Edit → Parameters**). Niekiedy wystarczy tylko otworzyć i zamknąć okno **Parameters**.

3. Zadania do samodzielnej realizacji

1. Rozwiązać przykładowy problem harmonogramowania (przykład 1.1) za pomocą pakietu LiSA, realizując tok postępowania opisany w punkcie 2.2, przy czym dodatkowo należy:
   a) Wykorzystać przynajmniej trzy różne algorytmy harmonogramowania.
   b) Dla każdego z algorytmów zastosować trzy kryteria: SumUj, Lmax oraz Cmax.
   c) Wykonać tabelaryczne zestawienie prezentujące zmiany wartości kryteriów zależnie od użytego algorytmu.
   d) Pokazać na przykładzie, że zmiana kryterium optymalizacji powoduje z reguły znaczne pogorszenie (zwiększenie) wartości parametru, który przestaje być kryterium – np. dokonanie zmiany z kryterium Cmax na SumUj powinno spowodować wyraźne zwiększenie wartości Cmax i zmniejszenie wartości SumUj.
2. Zaproponować własny problem harmonogramowania, inny niż *job shop*, ale sformułowany w podobny sposób jak zadanie z przykładu 1.1. Podać jego klasyfikację w notacji Grahama. Rozwiązać z użyciem pakietu LiSA. Przykłady zaproponowane przez poszczególne zespoły laboratoryjne powinny być różne.

W sprawozdaniu należy udokumentować wszystkie etapy realizacji ćwiczenia, przedstawić wyniki, sformułować wnioski.