Raport z projektu Systemy Zdarzeniowe

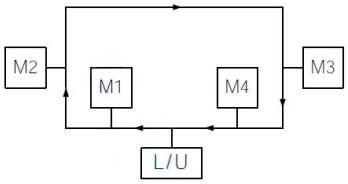
Marcin Ciopcia Daniel Gut Piotr Semberecki Hanna Sienkiewicz
Mateusz Stachowski

4 lutego 2015

1 Problem Projektu

1.1 Opis ogólny

Pierwszym celem projektu jest symulacja zautomatyzowanej linii produkcyjnej z wieloma gniazdami wytwórczymi, obsługiwanej przez wózki AGV. Linia produkcyjna ma kształt pierścienia, z ustalonym dozwolonym kierunkiem ruchu wózków AGV, poruszających się jak na rysunku 1.



Rysunek 1: Struktura ścieżek

Modelowany system posiada stacje załadowczo-rozładowczą (L/U), cztery stacje maszynowe (M_1, M_2, M_3, M_4), z których każda posiada bufor wejściowy i wyjściowy oraz robota wykonującego operację przeniesienia detalu z bufora wejściowego na stanowisko robocze maszyny oraz ze stanowiska roboczego do bufora wyjściowego. Opisywany system produkcyjny realizuje współbieżnie do sześciu typów produktów w podanej ilości. Operacje na danym produkcie zostają wykonane na rożnych maszynach w zadanej kolejności.

Następnie zostanie opracowany sterownik zdarzeniowy zarządzający logiką przepływu zadań w systemie składającą się z

- klasyfikatora zadań możliwych do realizacji,
- systemu agentowego zarządzającego kolejnością wykonywania działań,
- systemu przeciwdziałania zakleszczeniom zdarzeń w systemie.

Sterownik nadrzędny opisany zostanie za pomocą sieci Petriego, następnie zostanie oprogramowany w środowisku *Matlab*.

1.2 Zastosowanie systemów zdarzeniowych

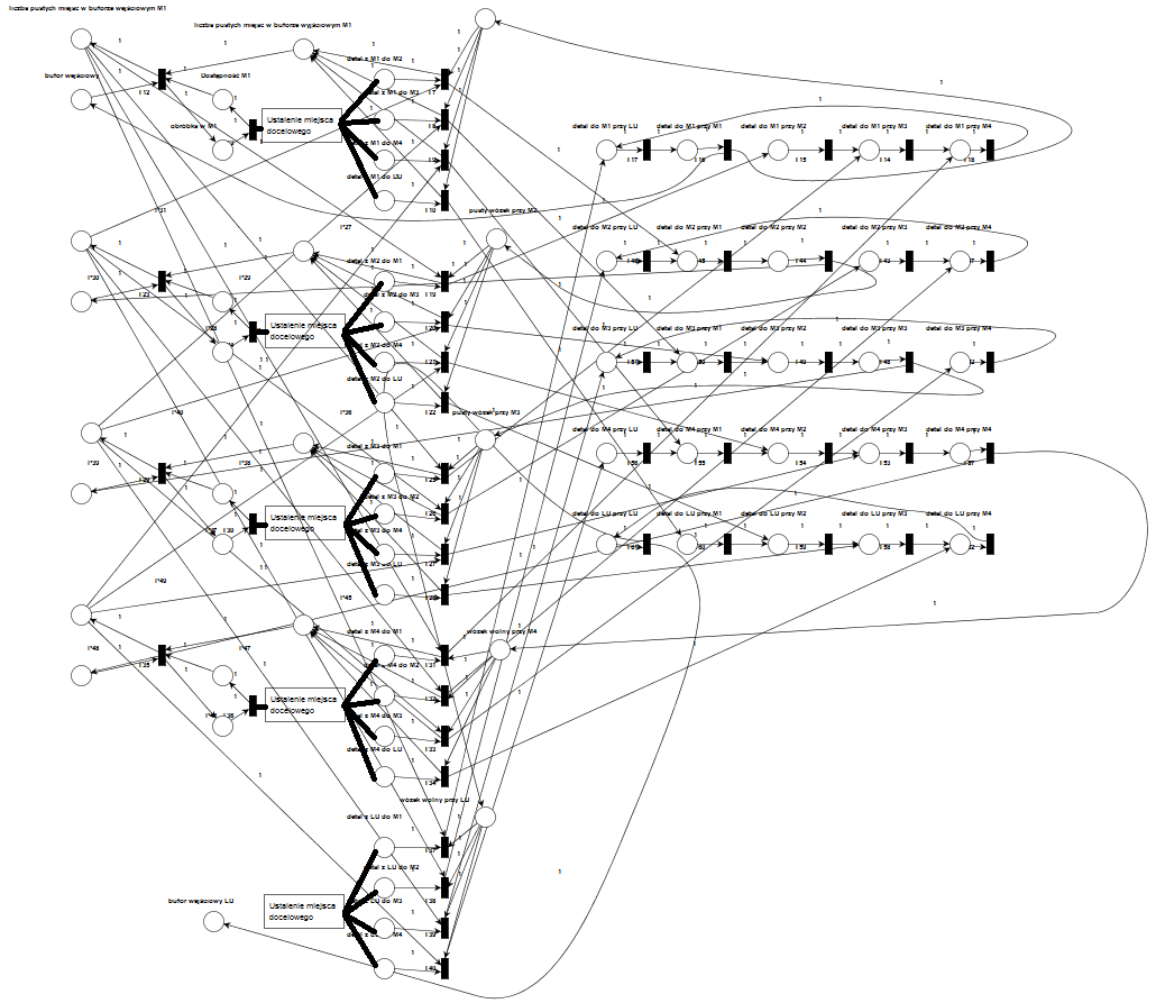
Podjęcie tego zagadnienie jest ważne ze względu na poszerzające się zastosowanie robotyzacji w wielu dziedzinach gospodarki. Za pomocą elastycznych komórek produkcyjnych (ESP) można modelować procesy produkcyjne, przepustowość sieci infrastruktury technicznej, systemy czasu rzeczywistego [1] , sterowniki logiczne [2] czy sieć komunikacji miejskiej. Modele różnorodnych systemów pozwalają ocenić ich efektywność funkcjonowania [3], co może wpłynąć na zmniejszenie kosztów, niwelując błędy już przy projektowaniu danego systemu. Za pomocą systemów zdarzeniowych można modelować również poruszanie się robotów mobilnych, zapobiegając przy tym zakleszczeniom, kolizjom, optymalizując czas wykonania danego zadania czy przejazdu przez daną ścieżkę.

1.3 Upowszechnienie wyników

Wyniki projektu będą upowszechnione, będą one zawierały

- archiwum z oprogramowaniem,
- dokumentację algorytmów,
- dokumentację oprogramowania dla użytkowników i deweloperów,
- przykład działania systemu,

1.4 Sieć Petriego

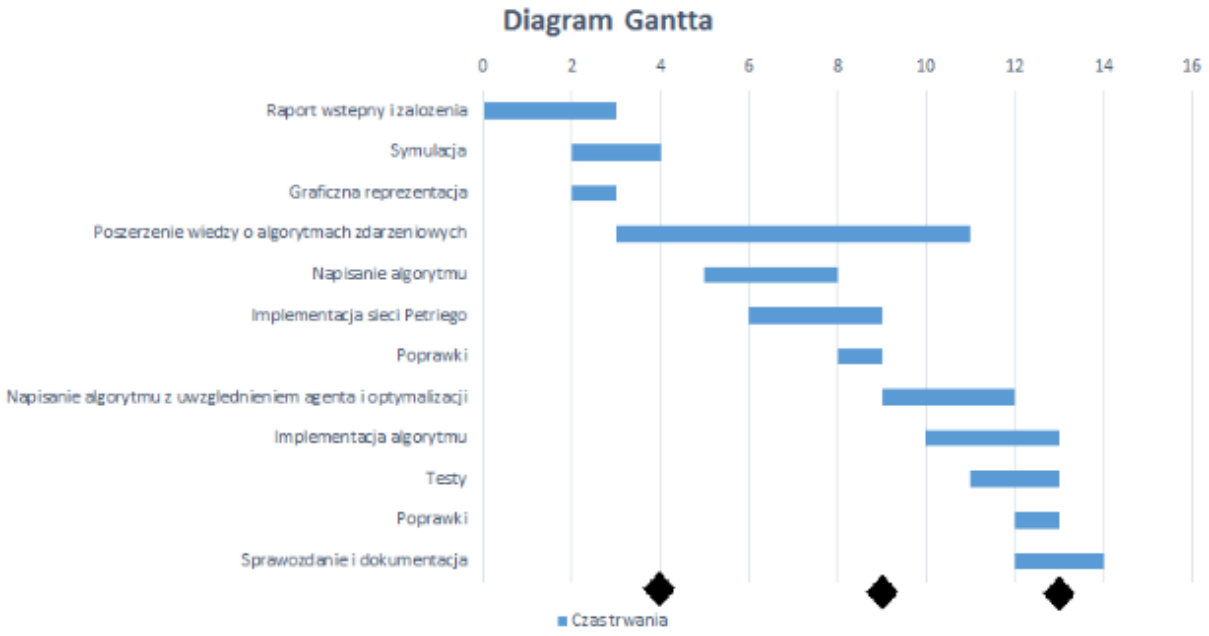


Rysunek 2: Sieć Petriego

Wykorzystaliśmy standardowe oznaczenia miejsc i tranzycji w sieciach Petriego. Wyróżniliśmy operacje maszynowe oraz realizowane przez wózki AGV, które są obserwowalne przez sterownik główny, po zakończeniu danego zadania maszyna/wózek informuje o tym kontroler główny, który może zlecić nowe zadanie dla maszyny/wózka (zdarzenie kontrolowalne).

2 Plan pracy i rozkład w czasie

Na rysunku 3 znajdują się wyselekcjonowane zadania oraz czas ich trwania jak również zaznaczone zostały kamienie milowe.



Rysunek 3: Diagram Gantta

3 Zarządzanie projektem

Spotkania zespołu odbywać się będą w terminie zajęć oraz po wcześniejszym umówieniu się członków zespołu przy użyciu *Skype'a* oraz grupy na portalu społecznościowym *Facebook*. Oficjalnym liderem grupy jest Hanna Sienkiewicz. Jest to osoba, która jest odpowiedzialna za rozliczanie z zadań członków grupy przed upływem wyznaczonych terminów. Podział zadań w grupie następuje zgodnie z umiejętnościami i oczekiwaniami poszczególnych członków grupy. Nie przewidujemy problemów z tym związanych, członkowie grupy czują się zobligowani do pomocy przy większych i trudniejszych zadaniach.

4 Zespół

W skład zespołu projektowego wchodzą

- Hanna Sienkiewicz, lider grupy, dane kontaktowe: 184184@student.pwr.edu.pl,
- Marcin Ciopcia,
- Daniel Gut,
- Piotr Semberecki,
- Mateusz Stachowski.

5 Struktura systemu

W systemie zdarzeniowym można wyróżnić

- sterownik główny,
- sterowniki maszyn,
- sieć Petriego,
- symulator zdarzeń.

Symulator ma za zadanie zasymulować uruchomienie zadania na maszynie, ładowanie z bufora do maszyny oraz ładowania z maszyny na wózek.

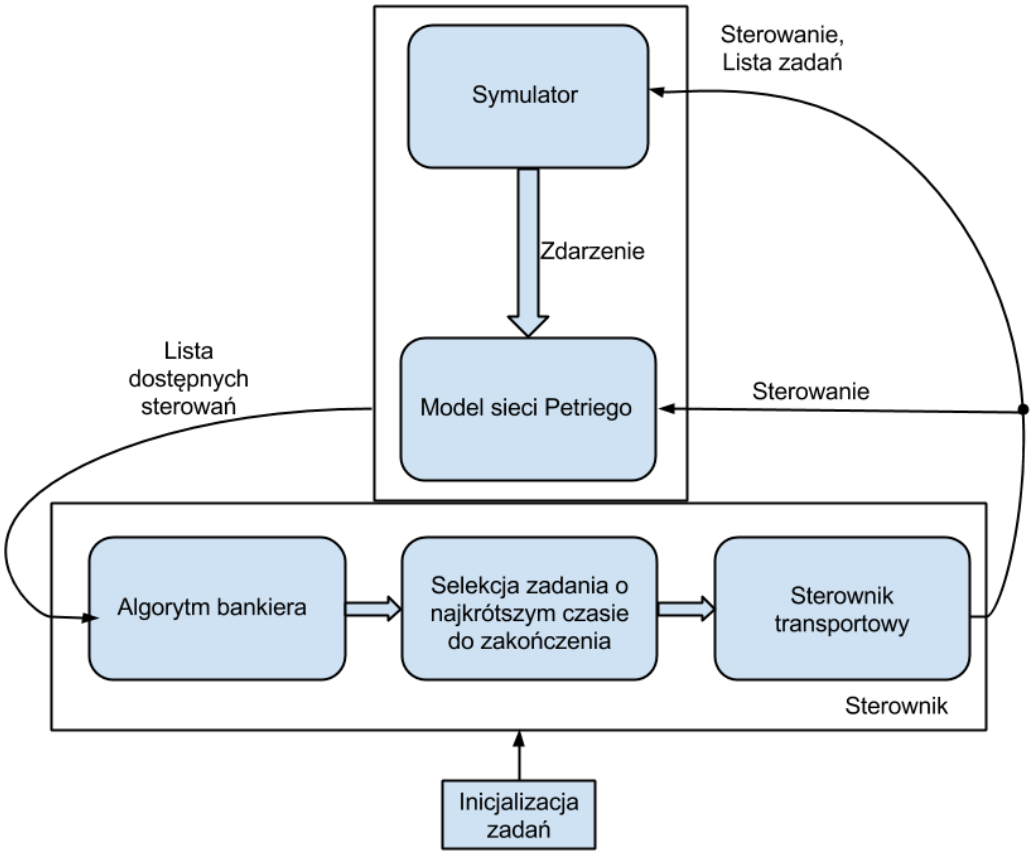
Pierwszy model sterownika działa na zasadzie wykonywania zadań jako pierwszych, których suma czasów wykonywania poszczególnych etapów jest najmniejsza. Posiada on wiedzę o zadaniach, jakie mogą być wykonane w danym kroku czasowym oraz o maszynach, na których będą wykonywane poszczególne etapy zadania jaki o czasach wykonania poszczególnych etapów zadania.

6 Unikanie blokad

W celu uniknięcia blokad stosujemy pojemność buforów wejściowych i wyjściowych maszyn większą niż 1. Zastosowaliśmy również w sterowniku głównym algorytm bankiera, który służy do sprawdzenia, czy dany stan jest bezpieczny, jeżeli tak to sterownik pozwala na realizację danego zadania, które doprowadza sieć do tego stanu bezpiecznego. Algorytm ten polega na skonstruowaniu ciągu bezpiecznego, tzn. takiego ciągu procesów $P(1), P(2), ..., P(n)$, w którym proces $P(i)$ żąda zasobów wolnych w danym stanie lub zajętych przez procesy począwszy od $P(1)$ do $P(i-1)$.

7 Opis sterowania

Rysunek nr 4 przedstawia działanie modelu zdarzeniowego.



Rysunek 4: Architektura sterownika

7.1 Inicjalizacja zadań

Zadanie to struktura wchodząca w skład listy zadań składająca się z

- numerów maszyn, na których są wykonywane poszczególne etapy zadań
  Lista_zadan(i).maszyny = [M_j, ..., M_n],

- czasów obróbki jednego detalu na maszynie
  Lista_zadan(i).czasy = [t_j, ..., t_n],
- liczby detali oczekujących na obróbkę na maszynie
  Lista_zadan(i).ilosc = [k, ..., 0].

7.2 Sterownik

Sterownik na podstawie listy dostępnych sterowań wygenerowanych przez model sieci Petriego, listy zadań oraz wielkości buforów wyjściowych i wejściowych maszyn generuje **Sterowanie**. **Sterowanie** to struktura zawierająca w sobie

- typ zadania (typy zadań zostały opisane w sekcji 7.4.1),
- numer maszyny - określa nr maszyny, przy której zdarzenie wystąpiło,
- numer - to numer zadania z listy zadań,
- etap - to numer etapu zadania.

Sterowanie generowane jest poprzez algorytm bankiera, następnie z pośród możliwych zadań wybierane jest to, które ma najkrótszy czas do zakończenia. Algorytm bankiera jest zaimplementowany dla maszyn z zadania. Agent przyjmuje jako parametry

- listę dostępnych sterowań wygenerowaną przez model sieci Petriego, zawierającą możliwe **Sterowania**,
- listę zadań.

Do sterowania przejazdem wózków wydzielamy osobną warstwę w sterowniku - Sterownik transportu. Sterownik nadrzędny po wybraniu następnego zadania przekazuje je do warstwy sterowania transportem. Następnie sterownik od transportu bierze aktualny etap zadania (w celu ustalenia skąd ma wziąć detal) i następny etap zadania, aby ustalić dokąd ma trafić detal. Jeżeli detal znajduje się przy maszynie 1-ej i chcemy żeby trafił do maszyny 4-tej to począwszy od miejsca 1-tego w przeciwnym kierunku niż ruch wózków Sterownik transportowy szuka pierwszego, wolnego wózka. Jeżeli okazało by się, że przy miejscu 4-tym nie ma wolnych wózków to przesuwamy go o jeden do przodu (oczywiście przy większej liczbie powstanie rekurencja).

7.3 Symulator

W symulatorze można rozróżnić symulator maszyn i symulator wózków. Główne zadania symulatora to

- przeniesienie detalu z bufora wyjściowego maszyny na wózek i zasymulowanie ruchu tego wózka,
- zatrzymanie wózka przy maszynie i przeniesienie detalu do bufora wejściowego maszyny,
- przeniesienie detalu z bufora wyjściowego do gniazda obróbczego,
- przeniesienie detalu z gniazda obróbczego do bufora wyjściowego,

Każde z tych zdarzeń trwa określoną jednostkę czasu.

7.4 Model sieci Petriego - opis stanu sieci

Stan sieci to macierz rozmiaru 13x5. Stan początkowy wygląda następująco

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{1}$$

gdzie b_{we}, b_{wy} to odpowiednio rozmiar bufora wejściowego i wyjściowego, a

- 1. wiersz sieci to transport do maszyny 1,
- 2. wiersz sieci to transport do maszyny 2,
- 3. wiersz sieci to transport do maszyny 3,
- 4. wiersz sieci to transport do maszyny 4,
- 5. wiersz sieci to transport do LU,
- 6. wiersz sieci to oznaczenie miejsca z pustym wózkiem (domyślnie jeden pusty wózek przy LU),
- 7. wiersz sieci to ilość detali w buforach wejściowych (domyślnie zero dla każdej maszyny),
- 8. wiersz sieci to liczba pustych miejsc w buforach wejściowych dla każdej z maszyn,
- 9. wiersz sieci to liczba pustych miejsc w buforach wyjściowych dla każdej z maszyn,
- 10. wiersz sieci to dostępność maszyn (domyślnie wszystkie maszyny są dostępne),
- 11. wiersz sieci to obróbka na maszynie,
- 12. wiersz sieci to wózek w ruchu miedzy M1 a M2, M2 a M3, M3 a M4, M4 a LU, LU a M1 (domyślnie żaden wózek nie jest w ruchu),
- 13. wierz sieci to gotowy detal do odbioru znajdujący się w buforze wyjściowym maszyny (domyślnie nie ma gotowych detali).

Przykładowo transport do maszyny nr 4 z maszyny nr 3 to S[4][3]=1 oraz S[12][3]=1 (wózek w ruchu). Po wykonaniu tego transportu mamy S[4][3]=0, S[4][4]=1 oraz S[12][3]=0.

7.4.1 Symulacja zdarzeń i sterowań w modelu sieci Petriego

Tutaj zostało opisane działanie sieci Petriego wraz z obrazującym to działanie przykładem.
Istnieją 4 typy zadań

- Typ 3 - Załaduj wózek, Zgłaszamy, że detal nie jest już w gnieździe maszyny, maszyna ta staje się dostępna. Sprawdzamy czy mamy pusty wózek przy maszynie i czy jest miejsce w buforze wejściowym maszyny docelowej. Jeśli tak to rezerwujemy wózek. Zmniejszamy liczbę dostępnych miejsc w buforze wejściowym maszyny docelowej o 1. Zwiększamy miejsce w buforze wejściowym dla maszyny, która opuszcza detal. W wierszu o numerze maszyny docelowej, zaznaczamy transport do niej począwszy od maszyny aktualnej. W takim stanie możliwy jest transport. Do **Listy dostępnych zadań** dodawane jest **Sterowanie** - transport.

$$S_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$S_1[6][1] = 1$ jest pusty wózek przy maszynie nr 1. Żaden wózek nie jest w ruchu. $S_1[10][1] = 0$ maszyna M1 nie jest dostępna, ponieważ $S_1[11][1] = 1$ obróbka detalu trwa. $S_1[8][3] > 0$ bufor wejściowy maszyny docelowej M3 nie jest pełny. Symulator po czasie trwania obróbki detalu na

maszynie M1 zgłasza **Zdarzenie** do sieci o przeniesienie tego detalu do bufora wyjściowego tej maszyny.

$$S_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$S_2[13][1]$=1 - gotowy detal do odbioru w buforze wyjściowym maszyny M1. $S_2[11][1] = 0$ obróbka zakończona, $S_2[10][1] = 1$ - maszyna M1 znów dostępna. Po zgłoszeniu **Zdarzenia**- Załaduj wózek mamy

$$S_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} - 1 & b_{we} & \infty \\ b_{wy} + 1 & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$S_3[13][1]$=0 - detal został zabrany z bufora wyjściowego. $S_3[6][1] = 0$ rezerwujemy wózek. $S_3[3][1] = 1$ zaznaczamy transport z maszyny nr 1 na maszynę docelową (nr 3). $S_3[8][3] = b_{we} - 1$ zmniejszamy liczbę dostępnych miejsc w buforze wejściowym maszyny docelowej. $S_3[9][1] = b_{wy} + 1$ zwiększamy liczbę dostępnych miejsc w buforze wyjściowym maszyny nr 1. Możliwe jest **Sterowanie** - transport (dodawane do **Listy dostępnych zadań**).

- Typ 2 - Wykonaj transport

$$S_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

Symulator wystawia zdarzenie ruchu wózka $S_4[12][1]$ - wózek w ruchu między M1 a M2.

$$S_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$S_5[12][1]$=0 - koniec ruchu miedzy M1 a M2. Symulator wystawia **Zdarzenie** - $S_5[3][1] = 0$ został wykonany transport z maszyny nr 1 na maszynę nr 2, dalej wykonujemy **Zdarzenie** - transport z maszyny nr 2 na maszynę nr 3, $S_5[3][2] = 1$.

$$S_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

Symulator wystawia **Zdarzenie** - ruch wózka między M2 a M3 -$S_6[12][2] = 1$.

$$S_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

Po czasie jazdy wózka symulator informuje sieć o **Zdarzeniu** - koniec ruchu wózka $S_7[12][2] = 0$, $S_7[3][2] = 0$ - został wykonany transport z maszyny nr 2 na maszynę nr 3. Następnie sieć wykonuje

Zdarzenie - załadunek do bufora wejściowego maszyny nr 3 - $S_7[3][3] = 1$.

$$S_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$S_8[3][3] = 0$, $S_8[6][3] = 1$ mamy pusty wózek przy maszynie nr 3, $S_8[7][3] = 1$ mamy detal w buforze wejściowym maszyny nr 3. Zgłaszamy możliwość **Sterowania** - załadunek detalu na maszynę.

- Typ 1 - Załaduj detal na maszynę. Jeśli jest to ostatni etap zadania i maszyna docelowa to LU, to zadanie zostaje zakończone. Zgłaszamy pusty wózek. Zwiększamy liczbę miejsc w buforze wejściowym maszyny o 1. Zgłaszamy **Sterowanie** - możliwość obróbki (do **Listy dostępnych zadań**).

$$S_9 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we}+1 & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$S_9[8][3] = b_{we} + 1$ zwiększamy liczbę miejsc w buforze wejściowym maszyny o 1. $S_9[7][3] = 0$ - jest to **Zdarzenie** przeniesienie z bufora wejściowego maszyny nr 3 do jej gniazda obróbczego.

- Typ 4 - Wykonaj obróbkę. Do odpowiedniej kolumny wiersza 11 jest wpisywana jedynka.

$$S_{10} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

Zdarzenie - $S_{10}[11][3] = 1$ trwa obróbka na maszynie nr 3. $S_{10}[10][3] = 0$ maszyna nr 3 nie jest

wolna.

$$S_{11} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b_{we} & b_{we} & b_{we} & b_{we} & \infty \\ b_{wy} & b_{wy} & b_{wy} & b_{wy} & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

Po upłynięciu czasu obróbki, symulator wystawia **Zdarzenie** - przeniesienie detalu z gniazda obróbczego do bufora wyjściowego maszyny. $S_{11}[13][3] = 1$ - gotowy detal w buforze wyjściowym maszyny M3. $S_{11}[11][3] = 0$ - koniec obróbki na maszynie M3, $S_{11}[10][3] = 1$ - maszyny M3 jest wolna. Zgłaszamy możliwość **Sterowania** - załadunek wózka (do **Listy dostępnych zadań**), po wykonanej obróbce.

8 Struktura programu

- AlgBankiera.m funkcja zawierająca algorytm bankiera,
- IleZadanNaLiscie.m funkcja liczy ile zadań pozostało do realizacji na danej *Licie Zada*,
- PetriNet.m klasa implementująca sieć Petriego,
- Sterowanie.m klasa implementuje pojedyncze zdarzenie sterowania,
- Zadanie.m klasa implementuje zadanie do zrealizowania przez system zdarzeniowy,
- agent.m funkcja implementuje agenta decyzyjnego sterownika zdarzeniowego,
- agvInit.m funkcja implementuje wózek AGV jak strukturę,
- loadOnInputBuf.m funkcja implementuje wrzucenie detalu na bufor wejściowy maszyny,
- loadOnMachineSocket.m funkcja implementuje wrzucenie detalu na gniazdo obróbcze maszyny,
- loadOnOutputBuf.m funkcja implementuje wrzucenie detalu na bufor wyjściowy maszyny,
- machineInit.m funkcja inicjuje maszyny jako struktury danych,
- main.m program emuluje działanie robotycznego systemu obróbczego w orientacji kołowej oraz implementuje sterownik zdarzeniowy zarządzający siecią,
- simInit.m inicjalizacja symulatora.

8.1 Główna pętla programu

Przykładowa inicjalizacja zadań jest następująca

Listing 1: Inicjalizacja zadań w pliku main.m

```
Lista_zadan(1).maszyny = [ 1, 2, 3];
Lista_zadan(1).czasy = [ 20, 8, 35];
Lista_zadan(1).ilosc = [ 5, 0, 0];
Lista_zadan(2).maszyny = [ 2, 1, 2, 3];
Lista_zadan(2).czasy = [ 7, 14, 5, 40];
Lista_zadan(2).ilosc = [ 5, 0, 0, 0];
IleZadanNaStart = IleZadanNaLiscie(Lista_zadan);
IleZadanW_U = 0;
```

Główna pętla programu z wyszczególnionymi symulatorem siecią Petriego oraz sterownikiem.

Listing 2: Część pliku main.m

```
while(IleZadanW_U < IleZadanNaStart)
% iteruj symulator
Zdarzenie = AktualizujSymulator( Sterowanie, Lista_zadan );
% jesli Sterowanie = [] to iteruj do nastepnego zdarzenia
% iteruj siec petriego
Lista_sterowan = PetriNet.AktulaizujOZdarzenie( Zdarzenie);
% selekcja zadan
Sterowanie = agent(Petri.listaDostepnychSterowan(),
/Lista_zadan, Wielkosc_buforow);
% aktualizowanie sieci w oparciu o Sterowanie
Petri.wykonajSterowanie(Sterowanie);
end
```

Literatura

[1] S. Samolej, B. Trybus, *Zastosowanie kolorowanych sieci Petriego w projektowaniu systemów czasu rzeczywistego*. Pomiary, Automatyka, Kontrola, 2005, tom R. 51, nr 1, 11-13.

[2] I. Grobelna, M. Grobelny, *Projektowanie sterowników logicznych z wykorzystaniem łuków zezwalających i zakazujących sieci Petriego*. Pomiary, Automatyka, Kontrola, 2012, tom R. 58, nr 7, 605-607.

[3] G. Bocewicz, W. Muszyński, Z. Banaszak, *Modele multimodalnych sieci i procesów transportowych*. Postępy robotyki pod redakcją K. Tchonia i C. Zielińskiego, Oficyna wydawnicza Politechniki Warszawskiej, Warszawa 2014, tom 2, s. 543-552.