UNIWERSYTET ZIELONOGÓRSKI

**Wydział Informatyki,
Elektrotechniki i Automatyki**

ROZPRAWA DOKTORSKA

mgr inż. Łukasz Stefanowicz

Dekompozycja współbieżnych systemów sterowania opisanych sieciami Petriego

Promotor
dr hab. inż. Andrzej Karatkiewicz, prof. AGH

Promotor pomocniczy
dr hab. inż. Remigiusz Wiśniewski, prof. UZ

Zielona Góra 2023

Słowa kluczowe: *sieć Petriego, współbieżny system sterowania, przestrzeń stanów, hipergraf, algorytm, hipergraf transwersal dokładnych.*

Keywords: *Petri Net, concurrent control system, space state, hypergraph, algorithm, exact transversal hypergraph.*

Spis treści

Spis rysunków

Spis tablic

Rozdział 1

Wprowadzenie

Informatyka składa się z wielu nurtów naukowych. Jednym z nich jest dziedzina nazywana techniką cyfrową. Ostatnie lata przynoszą jej dynamiczny rozwój, a co za tym idzie rozmiar i stopień skomplikowania systemów współbieżnego sterowania ulega szybkiemu zwiększeniu [1–8]. Zaobserwować można trend związany z wpływem teorii grafów, a także hipergrafów na rozwój techniki cyfrowej. Z uwagi na tempo rozwojowe nie jest dzisiaj możliwe szybkie i łatwe zaprojektowanie układu bez użycia narzędzi oraz programów wspierających cały proces. Bardzo istotna staje się dekompozycja systemu sterowania, ponieważ pozwala na podzielenie dużego układu na automaty sekwencyjne, które mogą być realizowane niezależnie [9–11]. Każdy z nich można oczywiście zaprojektować jako osobny automat skończony (ang. Finite State Machine). Znane i powszechnie stosowane metody dekompozycji oparte są o wykorzystanie algebry liniowej oraz teorii grafów [1, 12–20]. Tempo rozwoju techniki cyfrowej, a także projektowanie coraz większych układów powoduje, że dotychczasowe metody co prawda pozwalają na dalszą implementację, jednak czas wyznaczania rozwiązania powoli staje się nieakceptowalny, z uwagi na wykładniczy czas metod dokładnych. Dlatego też, z reguły stosowane są przybliżone metody wielomianowe, które nie gwarantują optymalnego rezultatu. Sytuacja ta wymusza modyfikację istniejących już algorytmów w celu otrzymania akceptowalnych czasów uzyskiwania rozwiązania zadanego problemu projektowego. Z przeglądu literaturowego nie wynika jasno tendencja rozwoju. Zauważono podejście polegające na optymalizacji istniejących algorytmów poprzez wprowadzanie modyfikacji, jak i na opracowaniu nowych technik pozwalających na uzyskanie wyniku w założonym czasie na akceptowalnym poziomie dokładności [1, 9–21]. W rozprawie skupiono się na systemach współbieżnego sterowania. Przeanalizowano wybrane metody pozwalające na wspomaganie procesu projektowania układów cyfrowych. Autor zoptymalizował proces dekompozycji systemów logicznych z uwzględnieniem procesu selekcji uzyskanych komponentów. Opracowano nowe algorytmy bazujące na teorii hipergrafów oraz wykorzystujące sieci Petriego. Pozwalają one na rozwiązanie problemów dekompozycji oraz

selekcji sprawnie i skutecznie. Pod pojęciem sprawności w tym przypadku rozumiany jest czas szukania rozwiązania, natomiast skuteczność należy rozumieć jako generowanie optymalnych rezultatów (system zostaje zdekomponowany na najmniejszą możliwą liczbę składowych). Na potrzeby niniejszej rozprawy rozbudowana została biblioteka Hippo [22], której metody pozwalają na weryfikację eksperymentalną opracowanych metod. Autor jest współtwórcą internetowego systemu Hippo, udostępniającego funkcjonalności pomocne w dekompozycji układów logicznych.

1.1 Aktualny stan wiedzy oraz motywacja podjęcia tematu

Głównym tematem niniejszej rozprawy doktorskiej są systemy tranzycyjne opisane sieciami Petriego [23–34]. Są one dobrym narzędziem do modelowania układów współbieżnych, składają się z miejsc, tranzycji oraz skierowanych łuków. Sieci Petriego są bardzo powszechnie używane do modelowania nie tylko problemów sterowania, ale także do procesów rzeczywistego świata [11,12,15,24–27,35–63]. Rozpatrywana przestrzeń systemu składa się ze stanów lokalnych oraz stanów globalnych. Można stwierdzić, że miejsca sieci Petriego w takim systemie odpowiadają stanom lokalnym, natomiast globalnym znakowanie sieci Petriego.

Obecnie układy cyfrowe są dużo bardziej skomplikowane w porównaniu do projektowanych jeszcze kilka lat temu. Nie jest możliwe szybkie i łatwe wykonanie projektu bez zastosowania specjalistycznych metod oraz narzędzi. Popularnym podejściem jest dzielenie dużego układu na mniejsze bloki funkcjonalne, proces ten nazywany jest dekompozycją. Wynikiem dekompozycji są komponenty sekwencyjne, nazywane podsieciami automatowymi. Istotnym problemem jest również selekcja, która pozwala na wybranie najmniejszej liczby komponentów do pokrycia sieci. Najbardziej popularne techniki podziału systemu na składowe automatowe bazują na zastosowaniu teorii grafów, teorii hipergrafów oraz algebry liniowej. Pierwsza z nich zakłada wykorzystanie teorii grafów [1, 22, 64]. Algorytm może zostać opisany w pięciu krokach. Najpierw system sterowania jest specyfikowany za pomocą sieci Petriego. Na tej podstawie wyznaczana jest sieć nazywana makrosiecią. Jej charakterystyczną częścią są makromiejsca, które odpowiadają zbiorowi jednego bądź większej liczby miejsc pierwotnej sieci Petriego. Następnie wyznaczany jest graf, który zawiera informacje o relacji współbieżności strukturalnej pomiędzy miejscami sieci. Graf ten jest następnie kolorowany, a makromiejsca zastępowane podzbiorami miejsc sieci wejściowej. W ten sposób uzyskuje się rozwiązanie w postaci poszczególnych podsieci automatowych. Głównym problemem przedstawionej metody jest operacja kolorowania, ponieważ wyznaczenie rozwiązania minimalnego z wykorzystaniem algorytmu dokładnego realizowane jest w najgorszym przypadku w czasie wykładniczym [1, 19]. Oznacza to, że dla dużych sieci rozwiązanie może nie zostać znalezione w akceptowalnym czasie. Warto

zwrócić uwagę, iż algorytm dokładny przeszukuje wszystkie możliwe rozwiązania i zwraca najlepsze. Aby uzyskać rezultat nierzadko stosuje się algorytmy przybliżone, jak chociażby metody zachłanne, w których wyznaczane jest optimum lokalne. Tak wyznaczone rozwiązanie jednak niekoniecznie musi być optimum globalnym [22]. Warto w tym miejscu zaznaczyć, że istnieje możliwość zastosowania teorii grafów doskonałych. Jeśli graf współbieżności strukturalnej dla danej sieci jest grafem porównywalności, wówczas optymalne rozwiązanie może zostać znalezione w czasie wielomianowym [65].

Zastosowanie modelu hipergrafowego w dekompozycji systemów dyskretnych zostało przedstawione w pracach [10,22] oraz [66] i składa się z sześciu etapów. W pierwszej kolejności dana jest sieć Petriego. Następnie tworzona jest makrosieć na podstawie sieci wejściowej. Krok kolejny polega na wyznaczeniu hipegrafu współbieżności. Etap następny wymaga wyznaczenia hipergrafu sekwencyjności. Następnie wyznaczany jest hipergraf dualny, zastosowany zostaje szybki algorytm redukcji. W rezultacie tworzony jest hipergraf selekcji. Ostatni krok polega na wyznaczeniu transwersali dokładnej, która jest rozwiązaniem. Istotnym rdzeniem metody jest wykorzystanie hipergrafu c-dokładnego.

Trzecia metoda oparta o algebrę liniową wykorzystuje zastosowanie niezmienników miejsc (tzw. p-inwariantów). Polega na wyznaczeniu niezmienników miejsc pokrywających sieć, a następnie podsieci automatowych. Następnie, jeśli jest wymagana selekcja, wybierane są tylko niezbędne do uzyskania rozwiązania. Wykorzystanie tej metody jest zauważalnym trendem światowym [52, 67–75]. Wyznaczanie p-inwariantów jest realizowane w wykładniczym czasie. Jako najpopularniejszą metodę wyznaczania wskazać można algorytm Martineza-Silvy, który operuje na przekształceniach macierzowych [76].
Głównym problemem badawczym poruszanym w niniejszej rozprawie doktorskiej jest zagadnienie dekompozycji, w tym także selekcji, która należy do klasy problemów NP-trudnych [1, 19]. Oznacza to, iż nie jest znany wielomianowy algorytm dokładny pozwalający na uzyskanie rozwiązania. Selekcja podsieci automatowych w procesie dekompozycji jest o tyle istotna, iż pozwala na wybranie jedynie tych niezbędnych do pokrycia sieci. Autor zdecydował się na opracowanie metody, która po spełnieniu określonego warunku pozwala na uzyskanie rozwiązania minimalnego w wielomianowym czasie. Opracowana została także kompletna metoda dekompozycji współbieżnej systemów sterowania z wykorzystaniem niezmienników miejsc i algebry liniowej. Głównym celem tego algorytmu jest ukazanie innego podejścia do zagadnienia dekompozycji, a także wykonanie rzetelnego określenia jakości rozwiązań oferowanych przez wymienione wyżej metody.
Dekompozycja systemów sterowania opisanych sieciami Petriego i ich problematyka, w tym analiza, została przedstawiona w wielu publikacjach [1, 10, 12–20, 77]. Prace te w głównej mierze zakładają wykorzystanie teorii grafów, a co za tym idzie występują ogólnie znane ograniczenia danych algorytmów wynikające z wykładniczego czasu ich

działania. Interesującym problemem badawczym jest temat pokrycia dokładnego (ang. exact cover) [78, 79]. Kompletną pozycją poruszającą zagadnienia analizy przestrzeni stanów systemów sterowania jest monografia [64]. Koncepcja wykorzystania teorii hipergrafów do reprezentowania przestrzeni stanów sieci Petriego została zaproponowana przez prof. Mariana Adamskiego w ośrodku zielonogórskim , a następnie rozwinięta szczegółowo w pracach [10, 22]. Niniejsza rozprawa doktorska jest kontynuacją wielu tych prac. Należy podkreślić, że w hipergrafie krawędź może być incydentna do wielu wierzchołków, natomiast klasyczne grafy nieskierowane modelują zależności pomiędzy dwoma wierzchołkami (krawędź może być incydentna do dwóch wierzchołków). Przykładowo przedstawiają one relacje zarówno pomiędzy stanami lokalnymi, jak i globalnymi danego systemu dyskretnego. Z uwagi na ten fakt możliwa jest bardziej wnikliwa analiza, ponieważ dotyczy ona obu typów stanów. Wykorzystanie teorii hipergrafów ukazuje możliwość innego podejścia do zagadnienia dekompozycji, w porównaniu do metody opartej o grafy. Warto wskazać pewien specyficzny typ hipergrafu, hipergraf transwersal dokładnych ((w literaturze nazywanym xt-hipergrafem lub hipergrafem należącym do klasy xt [80]), który posiada interesujące własności.

1.2 Teza, cele oraz zadania pracy

Na podstawie dyskusji przedstawionej w poprzednim rozdziale, sformułowano następującą tezę pracy doktorskiej:
Dekompozycja współbieżnych systemów sterowania opisanych sieciami Petriego może zostać w sprawny i skuteczny sposób zrealizowana z zastosowaniem metod algebry liniowej oraz teorii hipergrafów.
W celu uszczegółowienia tezy głównej, sformułowano dwie tezy pomocnicze:

- podział współbieżnej sieci Petriego na podsieci typu automatowego może zostać zrealizowany z zastosowaniem algebry liniowej (metoda wyznaczania inwariantów),
- selekcja uzyskanych podsieci automatowych w wybranych przypadkach może zostać zrealizowana z zastosowaniem hipergrafu transwersal dokładnych (xt-hipergraf).

Należy także zaznaczyć, że przez sprawność rozumiane jest uzyskanie wyników w założonym czasie, natomiast skuteczność oznacza, że otrzymane wyniki są poprawne [81]. W niniejszej rozprawie kryterium sprawności proponowanych metod określana czas wykonania algorytmu, natomiast kryterium skuteczności stanowi najmniejsza możliwa liczba automatów (podsieci), na które może zostać zdekomponowany współbieżny system sterowania opisany siecią Petriego.

Nadrzędnym celem pracy doktorskiej jest opracowanie, a także algorytmizacja metod dekompozycji sieci Petriego na podsieci typu automatowego wraz z ich późniejszą selekcją.

Głównymi zadaniami pracy są:

- opracowanie algorytmów, które w sprawny i skuteczny sposób pozwolą na dekompozycję sieci Petriego, a także selekcję podsieci automatowych,
- algorytmizacja i implementacja zadania dekompozycji współbieżnych systemów sterowania opisanych sieciami Petriego,
- realizacja bibliotek w ramach systemu komputerowego wspomagającego procesy dekompozycji oraz selekcji.

1.3 Struktura rozprawy

Rozprawa doktorska złożona jest z siedmiu rozdziałów oraz czterech dodatków. Rozdział pierwszy zawiera krótkie wprowadzenie do tematyki poruszanej w rozprawie, sformułowana została teza, nakreślono cel, a także zadania pracy.

Rozdział drugi opisuje zagadnienia związane z podstawami teoretycznymi sieci Petriego. Omówiono najistotniejsze kwestie niezbędne z punktu widzenia zrozumienia rozprawy, w szczególności ich właściwości oraz klasy.

Trzeci rozdział przedstawia pojęcia z zakresu teorii hipergrafów. Opisane zostały podstawowe definicje, algorytmy. W niniejszej pracy wykorzystywane są także elementy, opracowane przez prof. Adamskiego, dr Monikę Wiśniewską, prof. Remigiusza Wiśniewskiego, które zostały zaimplementowane w bibliotece Hippo. Opisano je szczegółowo w dodatku A z naciskiem na nowe zagadnienia. Bibliotekę rozwinięto o niezbędne nowe metody ułatwiające proces eksperymentalnej weryfikacji opracowanych algorytmów.

Elementy nowatorskie zostały przedstawione w rozdziale czwartym. Autor przedstawił opracowane algorytmy dekompozycji współbieżnych systemów sterowania opisanych sieciami Petriego, a także selekcji podsieci automatowych w procesie dekompozycji. Metody oparto o teorię hipergrafów, a w szczególności hipergraf transwersal dokładnych. Całość uzupełnia nowy algorytm dekompozycji bazujący na:

- niezmiennikach miejsc,
- algorytmie selekcji.

W rozdziale piątym opisano eksperymentalną weryfikację zaproponowanych metod. Przedstawiona została metodologia badań, a także biblioteka sieci testowych, nazywanych benchmarkami. Opisany został także system Hippo pozwalający na przeprowadzenie badań eksperymentalnych. Skuteczność opracowanej metody selekcji została potwierdzona dowodem matematycznym pokazującym, że każda transwersala dokładna pozwala na uzyskanie rozwiązania dokładnego.

Rozdział szósty jest podsumowaniem rozprawy: omówiono szczegółowo uzyskane rezultaty, opracowaną metodę. Przedstawiono także jasno i klarownie elementy nowatorskie, a także zaproponowano kierunki dalszych prac związanych.

W skład rozprawy wchodzą cztery dodatki. Dodatek A zawiera opisaną bibliotekę bazową, w oparciu o którą zrealizowano badania eksperymentalne. Przedstawiono strukturę biblioteki poprzez opis poszczególnych klas. Wyszczególniono podział na trzy bloki funkcjonalne, dotyczące algorytmów:

- grafowych,
- hipergrafowych,
- związanych z sieciami Petriego.

Dodatek B opisuje bibliotekę modułów testowych.

Dodatek C prezentuje internetowy system Hippo, którego autor rozprawy jest jednym ze współautorów. Przedstawiono opis funkcjonalny systemu, graficzny interfejs użytkownika, możliwości systemu oraz przegląd wykorzystanych metod.

Dodatek D zawiera dowody matematyczne istotne z punktu widzenia rozważania niniejszej rozprawy doktorskiej.

Dodatek E opisuje szczegółowo uzyskane rezultaty badań eksperymentalnych. Podzielono go na dwie części dotyczące: dekompozycji oraz selekcji.

Rozdział 2

Podstawowe pojęcia

Rozdział zawiera podstawowe definicje pracy. Przedstawiono najważniejsze pojęcia związane ze współbieżnymi systemami sterowania opisanymi sieciami Petriego, a także sposoby ich reprezentacji. Opisane zostały także podstawowe pojęcia związane z teorią hipergrafów.

2.1 Sieci Petriego

2.1.1 Najważniejsze definicje

Definicja 2.1 *(Zwyczajną) Siecią Petriego jest dwudzielny, skierowany graf zbudowany z dwóch rodzajów wierzchołków: miejsc oraz tranzycji połączonych za pomocą skierowanego łuku [23]. Formalnie sieć Petriego definiuje czwórka:*

$$PN = (P, T, F, M_0) \tag{2.1}$$

gdzie:
$P = \{p_1, p_2, ..., p_m\}$ *to skończony oraz niepusty zbiór miejsc,*
$T = \{t_1, t_2, ..., t_n\}$ *to skończony oraz niepusty zbiór tranzycji,*
$P \cap T = \emptyset \wedge P \cup T \neq \emptyset$
$F \subseteq (P \times T) \cup (T \times P)$ *to skończony oraz niepusty zbiór łuków (relacja przepływu),*
$M_0 : M \subset P$ *to zbiór miejsc nazywany znakowaniem początkowym,*
p *jest miejscem wejściowym tranzycji* t*, jeśli* $(p, t) \in F$*,*
p *jest miejscem wyjściowym tranzycji* t*, jeśli* $(p, t) \in F$*.*

Znakowanie w literaturze z reguły jest definiowane jako funkcja przypisująca do miejsc liczby naturalne, które oznaczają liczbę tak zwanych *znaczników*. Z uwagi na fakt, że w niniejszej rozprawie skupiono się na sieciach bezpiecznych (sieci, w których miejsce może posiadać nie więcej niż 1 znacznik) przyjęto definicję określającą znakowanie jako zbiór.

W niniejszej rozprawie znakowanie również przedstawione jest jako wektor binarny, w którym elementy odpowiadają miejscom i dany element ma wartość 1, jeśli w danym znakowaniu odpowiednie miejsce ma znacznik oraz 0 w przeciwnym razie.

Definicja 2.2 *Stan sieci Petriego PN, nazywany znakowaniem, zdefiniowany jest jako podzbiór miejsc $M \subset P$. Znakowanie można też zdefiniować jako binarny wektor P przypisujący każdemu miejscu $p \in P$ liczbę znaczników (tokenów). $M(p) \in \{0, 1\}$ odpowiada liczbie tokenów miejsca p. Miejsce jest oznaczone, jeśli $M(p) = 1$. Znakowanie może zostać zmienione poprzez odpalenie tranzycji. Tranzycja może zostać odpalona jeśli każde z jej wejściowych miejsc zawiera znacznik. Odpalenie tranzycji usuwa znacznik z każdego miejsca wejściowego i dodaje znacznik do każdego miejsca wyjściowego.*

Sieci Petriego są szczególnym przypadkiem systemu tranzycyjnego i zostały opracowane do modelowania oraz analizy procesów sekwencyjnych i współbieżnych [23]. Za ich twórcę uważa się niemieckiego naukowca, Carla Adama Petriego. W roku 1962 sformułował rozprawie doktorskiej teoretyczne podstawy komunikacji pomiędzy asynchronicznymi częściami systemu komputerowego. Przedstawiona koncepcja dotyczyła zagadnień, które umożliwiały opis procesów współbieżnych w sposób graficzny za pomocą modelu matematycznego. Był to zalążek tematyki, która ewoluowała do znanych dziś sieci Petriego. Są one bardzo popularnym narzędziem do analizy oraz modelowania przede wszystkich systemów współbieżnych, równoległych oraz rozproszonych. Badanie sieci Petriego polega w pierwszej kolejności na określeniu własności: klasy sieci, żywotności, bezpieczeństwa, osiągalności. Bardzo często wyznaczany jest graf znakowań, który uzyskuje się na podstawie pierwotnej lub skondensowanej sieci Petriego. Proces ten polega na analizie zmian znakowania sieci podczas odpalania tranzycji. Macierz grafu ma następującą strukturę: wierzchołki stanowią zbiór miejsc znakowanych (w danym, określonym stanie), krawędzie z kolei odpowiadają odpalanym tranzycjom. Warto podkreślić, że graf znakowań jest systemem tranzycyjnym, w którym stany globalne zdefiniowane odpowiednio jako wierzchołki macierzy określają relacje współbieżności pomiędzy stanami lokalnymi.

2.1.2 Własności sieci Petriego

W bieżącej sekcji skupiono się na:

* żywotności,

* bezpieczeństwie,

* osiągalności.

Osiągalność w sieci Petriego

Definicja 2.3 *Znakowanie M_n jest osiągalne ze znakowania M_0 jeśli istnieje ciąg przejść $\alpha = t_1, t_2, .., t_n$, który prowadzi od znakowania M_0 do M_n [23].*

Żywotność sieci Petriego

Definicja 2.4 *Tranzycja t jest żywa, gdy osiągalne jest znakowanie w którym może zostać odpalona [64].*

Definicja 2.5 *Sieć Petriego PN jest żywa, gdy żywa jest każda tranzycja [23].*

Bezpieczeństwo sieci Petriego

Definicja 2.6 *Miejsce p jest bezpieczne, jeśli nie ma takiego osiągalnego znakowania M i tranzycji t, która w M może się odpalić, $M(p) > 0$, gdzie p nie jest miejscem wejściowym t i P jest wyjściowym miejscem t.*

Z powyższej definicji wynika, że dla dowolnego znakowania wszystkie miejsca sieci bezpiecznej zawierają maksymalnie jeden znacznik.

Definicja 2.7 *Sieć Petriego PN jest bezpieczna, gdy wszystkie jej miejsca są bezpieczne [23].*

Definicja 2.8 *Podsiecią PN_S sieci Petriego $PN = (P, T, F, M_0)$ jest:*

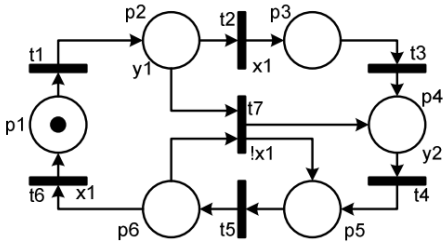
$$PN_S = (P_S, T_S, F_S, M_{S0}) \tag{2.2}$$

że:
$T_S \subset T$,
$P_S = \bullet T_S \cup T_S \bullet$,
$F_S = ((P_S \times T_S) \cup (T_S \times P_S)) \cap F$

Definicja 2.9 *Podsiecią automatową PN' sieci Petriego PN jest taka jej spójna podsieć:*

$$PN' = (P', T', F', M_0) \tag{2.3}$$

że:
$\forall t \in T' : |\bullet t| = |t \bullet| = 1$,
$P' = \bullet T' \cup T' \bullet$,
$F' = ((P' \times T') \cup (T' \times P')) \cap F$



Rysunek 2.1: Przykładowa interpretowana sieć Petriego

Definicja 2.10 *Graf znakowań (osiągalności) sieci Petriego jest skierowanym grafem, którego wierzchołkami są osiągalne znakowania danej sieci, natomiast łuki odpowiadają zmianom znakowań przez odpalenie pojedynczych tranzycji.*

Definicja 2.11 *Macierzą incydencji sieci Petriego* $PN = (P, T, F, M_0)$ *gdzie* $n = |P|$, $m = |T|$ *jest macierz* $C_{m \times n}$ *liczb całkowitych, w której element określony jest jako [23]:*

$$c_{ij} = \begin{cases} -1, (p_i, t_j) \in F \\ 1, (t_j, p_i) \in F \\ 0, \textit{w innym przypadku} \end{cases}$$

Definicja 2.12 *Niezmiennikiem miejsc (p-inwariantem) jest wektor* $\vec{y}$ *miejsc danej sieci Petriego* PN*, który spełnia równanie* $C * y = 0$ *[10]. Odpowiednio* C *jest macierzą incydencji opisującą sieć Petriego* PN*.*

2.1.3 Interpretowane sieci Petriego

Interpretowane sieci Petriego są dosyć powszechnie stosowane jako formalny zapis specyfikacji działania systemu sterowania. Generalnie miejsca, sygnały wejściowe oraz sygnały wyjściowe sieci Petriego odwzorowane są jako zmienne. Każde miejsce, sygnał wejściowy i wyjściowy traktowane są jako osobna zmienna. Istotą sterowania w sieciach interpretowanych jest określona reakcja systemu na zmiany parametrów wejściowych. Z uwagi na fakt, że parametry wejściowe można opisać alfabetem wejściowym, wyjściowe wyjściowym, pozwala to na interpretację miejsc analogicznie do stanów w przypadku automatu skończonego, natomiast tranzycji do akcji związanych ze zmianą stanu. Przykład interpretowanej sieci Petriego przedstawiono na rysunku 2.1 [82].

Interpretowaną sieć Petriego definiuje piątka [83]:

$$IPN = (P, T, F, \mathcal{X}, \mathcal{Y}) \tag{2.4}$$

gdzie:
$\mathcal{X}$ to skończony zbiór binarnych stanów wejściowych,

$\mathcal{Y}$ to skończony zbiór binarnych stanów wyjściowych,
$\mathcal{X} \cap \mathcal{Y} = \emptyset$

Sygnały wejściowe są przypisane do tranzycji. Zakłada się, że każda tranzycja, która może zostać odpalona ze względu na znaczniki w miejscach wchodzących, zostanie odpalona natychmiast kiedy (i jeśli) przypisane do niej sygnały będą miały wartość 1. Jeśli zbiór przypisanych sygnałów jest pusty, wówczas tranzycja zostaje odpalona natychmiast po tym jak wszystkie jej miejsca wchodzące uzyskują znaczniki. Wartości sygnałów wejściowych zależą od świata zewnętrznego (na przykład od sterowanego systemu). Natomiast sygnały wyjściowe są przypisane do miejsc i dany sygnał wyjściowy ma wartość 1 jeśli w danym znakowaniu istnieje miejsce zawierające znacznik, do którego on jest przypisany; inaczej ma on wartość 0.

2.2 Grafy i hipergrafy

Grafy pozwalają w klasyczny i dobrze znany sposób zamodelować systemy oraz relacje występujące pomiędzy poszczególnymi stanami [84–86]. Naukowcy używają ten model od wielu lat, wykorzystując jego zalety: przede wszystkim różnorodność oraz dostępność algorytmów pozwalających na wykonanie podstawowych czynności związanych z analizą. Hipergraf rozwija pojecie klasycznego grafu, ponieważ hipergrawędź może być incydentna z n-wierzchołkami. Można więc powiedzieć, że graf jest szczególnym przypadkiem hipergrafu, w którym każda hiperkrawędź jest incydentna z dwoma wierzchołkami. Zawieranie się wielu wierzchołków w jednej hiperkrawędzi istotnie wpływa na zakres modelowanych informacji, ponieważ struktura ta przekazuje informacje na temat stanów lokalnych oraz globalnych systemu. Teoria hipergrafów stosunkowo niedawno pojawiła się w algorytmach dekompozycji, która polega na podzieleniu układu na mniejsze jednostki funkcjonalne. System sterowania opisany jest za pomocą hipergrafu, którego wierzchołki odzwierciedlają dane moduły, natomiast hiperkrawędzie połączenia pomiędzy nimi.

2.2.1 Grafy

Definicja 2.13 *Nieskierowany graf G definiuje dwójka [87]:*

$$G = (V, E) \tag{2.5}$$

gdzie:
$V = \{v_1, v_2, ..., v_m\}$ *to skończony oraz niepusty zbiór wierzchołków,*
$E = \{E_1, E_2, ..., E_n\}$ *to skończony zbiór nieuporządkowanych par wierzchołków, zwanych*

krawędziami grafu.

Definicja 2.14 *Skierowany graf G_S definiuje dwójka [87]:*

$$G = (V, F) \tag{2.6}$$

gdzie:
$V = \{v_1, v_2, ..., v_m\}$ *to skończony oraz niepusty zbiór wierzchołków,*
$F = \{F_1, E_2, ..., F_n\}$ *to skończony zbiór uporządkowanych par wierzchołków, zwanych łukami grafu.*

2.2.2 Hipergrafy

Definicja 2.15 *Hipergraf H jest dwójką [87]:*

$$H = (V, E) \tag{2.7}$$

gdzie:
$V = \{v_1, v_2, ..., v_m\}$ *to skończony oraz niepusty zbiór wierzchołków,*
$E = \{E_1, E_2, ..., E_n\}$ *to zbiór niepustych podzbiorów V, nazywanych hiperkrawędziami.*

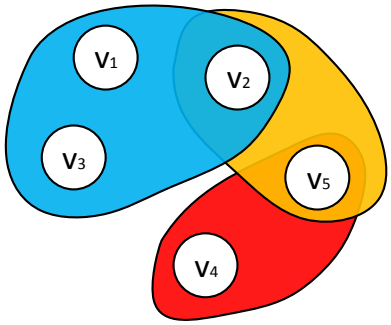
Definicja 2.16 *Wierzchołek v_i jest incydentny do hiperkrawędzi E_j, gdy $v_i \in E_j$*

Definicja 2.17 *Macierzą incydencji A hipergrafu o zbiorze wierzchołków:*
$V = \{v_1, v_2, ..., v_m\}$, *, krawędzi:* $E = \{E_1, E_2, ..., E_n\}$,
nazywamy macierz $A = a_{ij}$, w której wiersze odpowiadają krawędziom, kolumny wierzchołkom, tak, że:

$$a_{ij} = \begin{cases} 1 & \text{jeśli } v_i \text{ jest incydentny do hiperkrawędzi } E_j, \\ 0 & \text{jeśli } v_i \text{ oraz } E_j \text{ nie są incydentne.} \end{cases}$$

Rysunek 2.2 przedstawia przykładowy hipergraf (definicja 2.15).

Macierz incydencji odpowiadająca hipergrafowi przedstawionemu na rysunku 2.2 wygląda następująco:



Rysunek 2.2: Przykładowy hipergraf H

$$A_2 = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} E_1 \\ E_2 \\ E_3 \end{matrix}$$

Definicja 2.18 *Transwersalą T hipergrafu H jest zbiór [87]:*

$$T \subseteq V \tag{2.8}$$

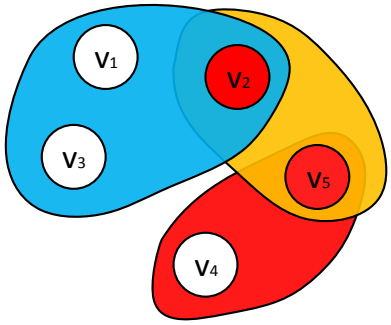
którego elementami są wierzchołki incydentne do każdej krawędzi hipergrafu.

Rysunek 2.3 przedstawia przykładową transwersalę hipergrafu H_1. W tym przypadku $T_1(H_1) = \{v_2, v_5\}$

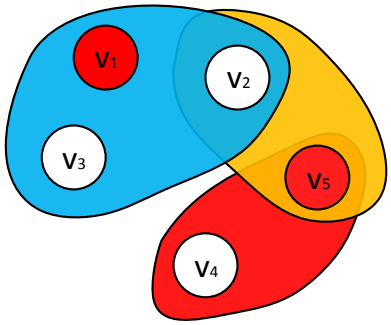
Definicja 2.19 *Transwersala jest minimalna wtedy, gdy nie zawiera żadnej innej transwersali hipergrafu H [87].*

Definicja 2.20 *Transwersalą dokładną D hipergrafu H jest zbiór [80]:*

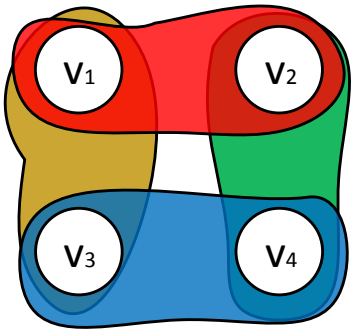
$$D \subseteq V \tag{2.9}$$



Rysunek 2.3: Rysunek przedstawiający transwersalę T_1 hipergrafu H_1



Rysunek 2.4: Rysunek przedstawiający transwersalę dokładną d_1 hipergrafu H



Rysunek 2.5: Rysunek przedstawiający hipergraf xt H_{XT}

którego elementami są wierzchołki incydentne do każdej krawędzi hipergrafu, jednakże każda krawędź jest incydentna z dokładnie jednym wierzchołkiem zbioru D tworzącego transwersalę dokładną.

Rysunek 2.4 przedstawia transwersalę dokładną (definicja 2.20). W tym przypadku $D_1(H_1) = \{v_1, v_5\}$

Definicja 2.21 *Hipergrafem transwersal dokładnych H_{XT} (nazywanym dalej xt-hipergrafem lub hipergrafem xt) jest taki hipergraf, w którym wszystkie minimalne transwersale są transwersalami dokładnymi.*

Rysunek 2.5 przedstawia przykładowy hipergraf xt.

Definicja 2.22 *Hipergrafem współbieżności $H_C = (P, M)$ jest hipergraf, którego wierzchołki odpowiadają miejscom sieci Petriego, natomiast hiperkrawędzie odpowiadają osiągalnym znakowaniom sieci: $M = M_0, ..., M_{k-1}$, gdzie M_0 jest znakowaniem początkowym, natomiast k liczbą wszystkich znakowań osiągalnych danej sieci Petriego.*

Definicja 2.23 *Hipergrafem sekwencyjności $H_{SEQ} = (P, PN')$ jest hipergraf, którego wierzchołki odpowiadają miejscom sieci Petriego, natomiast hipekrawędzie podsieciom automatowym $PN' = \{P'_1, ...P'_l\}$, gdzie l jest liczbą wszystkich podsieci automatowych danej sieci Petriego.*

Zgodnie z definicją 2.18 transwersala to pokrycie wierzchołkowe hipergrafu. Wyznaczanie transwersali jest w ogólnym przypadku realizowane w czasie wykładniczym.

Najpopularniejsze metody pozwalające uzyskać pokrycie wierzchołkowe hipergrafu to:

- szybki algorytm redukcji [88],
- algorytm z nawrotami [89, 90],
- algorytm zachłanny [89].

2.3 Definicje dotyczące złożoności obliczeniowej

Definicja 2.24 *Złożoność wielomianowa algorytmu B oznacza, że całkowity czas działania B jest ograniczony funkcją wielomianową rozmiaru danych wejściowych [91].*

Definicja 2.25 *Złożoność wykładnicza algorytmu B oznacza, że całkowity czas działania B jest ograniczony funkcją wykładniczą rozmiaru danych wejściowych [91].*

$$\exists k > 1, T(n) = \mathcal{O}(k^n) \tag{2.10}$$

gdzie:
n to rozmiar danych wejściowych,
k to pewna stała niezależna od rozmiaru danych wejściowych.

Definicja 2.26 *Wielomianowe opóźnienie algorytmu B oznacza, że B generuje wyniki w taki sposób, że czas między kolejnymi wyjściami jest ograniczony funkcją wielomianową rozmiaru danych wejściowych. Mówi się, że kolejne wyniki są generowane (wyznaczane, obliczane) w czasie wielomianowym [91].*

$$T(n) = \mathcal{O}(n^k) \tag{2.11}$$

gdzie:
n to rozmiar danych wejściowych,
k to pewna stała niezależna od rozmiaru danych wejściowych.

Rozdział 3

Algorytmy zaadaptowane na potrzeby pracy

Niniejszy rozdział zawiera opis najistotniejszych, z punktu widzenia bieżącej pracy, użytych algorytmów. Metoda wyznaczania niezmienników miejsc (nazywana metodą Martineza-Silvy) znajduje zastosowanie w proponowanym algorytmie dekompozycji. Algorytm XTREC pozwalający na określenie, czy zadany hipergraf należy do klasy xt (jest hipergrafem transwersal dokładnych), jest z kolei istotny pod kątem oszacowania, czy dana ścieżka dekompozycji pozwoli na uzyskanie rozwiązania dokładnego w czasie wielomianowym. Aby zwiększyć szansę na uzyskanie hipergrafu klasy xt, stosuje się szybki algorytm redukcji. Klasyczny algorytm pozwala na uzyskanie rozwiązania w oparciu o metodę z nawrotami, która także została zaprezentowana. Z uwagi na jej ograniczenia (sprawdzanie wszystkich rozwiązań w czasie wykładniczym), stosuje się także algorytm zachłanny, który w większości przypadków pozwala na uzyskanie wyniku przybliżonego. Proponowane podejście zakłada znalezienie transwersali (dokładnej), co z kolei umożliwia algorytm DLX (and Dancing Links). Zaprezentowane zostały także pokrótce istniejące podejścia do tematu dekompozycji systemów dyskretnych, wyróżniając metody oparte o: grafy, hipergrafy i inwarianty. Każda z nich ma swoje zalety, jak i wady.

3.1 Klasyczny algorytm wyznaczania niezmienników miejsc w sieci Petriego

Poniższa metoda (zwana także algorytmem Martineza-Silvy, od nazwisk autorów) jest znanym algorytmem pozwalającym na wyznaczenie niezmienników miejsc, jednak jej złożoność jest wykładnicza.

Dane wejściowe: macierz incydencji sieci Petriego C,
Dane wyjściowe: macierz zawierająca inwarianty w postaci wektorów miejsc.

1. *Utwórz macierz startową [B | C]*. Zbudowana jest ona z dwóch pomniejszych, a mianowicie *B* jest jednostkową macierzą miejsc, natomiast *C* jest macierzą incydencji sieci Petriego: -1 oznacza łuk wchodzący, 1 wychodzący.

2. *Dla każdej tranzycji t_m powtórz kroki:*

   (a) Zsumuj każdą parę wierszy, których elementy w *m-tej* kolumnie macierzy *C* mają wartości różne od 0 i w sumie dadzą 0.

   (b) Usuń wszystkie wiersze, których elementy w *m-tej* kolumnie macierzy *C* są różne od 0.

   (c) Usuń nieminimalne inwarianty (zawierające inne inwarianty) poprzez redukcję nadmiarowych wierszy (wierszy, które binarnie pokrywają inne).

3. Po wykonaniu pętli otrzymana macierz składa się z wektorów odpowiadających p-niezmiennikom danej sieci.

Przykład

Dana jest przykładowa sieć Petriego P_{ms}, której macierz [*B* | *C*] jest następująca:

$$[B|C] = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{cccccccc} p_1 & p_2 & p_3 & p_4 & t_1 & t_2 & t_3 \end{array} \\ \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & -1 \end{array}\right] \end{array}$$

Następnie przedstawiono zbiorczą tabelę z przebiegu wykonywania algorytmu.

[p]	p_1	p_2	p_3	p_4	t_1	t_2	t_3	
p_1	1	0	0	0	-1	0	1	
p_2	0	1	0	0	1	-1	0	Redukcja tranzycji pierwszej:
p_3	0	0	1	0	1	-1	0	$p_1 + p_2, p_1 + p_3$.
p_4	0	0	0	1	0	1	-1	

p_{1+2}	1	1	0	0	0	-1	1	Redukcja tranzycji drugiej: $p_{1+2}+p_4, p_{1+2}+p_4$.
p_{1+3}	1	0	1	0	0	-1	1	
p_4	0	0	0	1	0	1	-1	
p_{1+2+4}	1	1	0	1	0	0	0	Brak elementów niezerowych, koniec algorytmu.
p_{1+3+4}	1	0	1	1	0	0	0	

Tablica 3.1: Tabela przedstawiająca poszczególne etapy algorytmu

Pierwszy etap polega na znalezieniu wierszy, które w kolumnie t_1 mają elementy niezerowe. Są to odpowiednio: p_1, p_2 oraz p_3. Redukcja tranzycji pierwszej jest przeprowadzana w następujący sposób: do wiersza p_1 jest dodawany p_2 oraz do p_1 wierwsz p_3. W ten sposób kolumna t_1 zostaje wyzerowana. Redukcja tranzycji drugiej przebiega w następujący sposób: do wiersza p_{1+2} jest dodawany wiersz p_4, natomiast do wiersza p_{1+2} dodawany jest wierwsz p_4. Po przeprowadzeniu tych czynności kolumny t_1, t_2 oraz t_3 są zerowe, co skutkuje zakończeniem algorytmu. W ten sposób wyznaczone zostały dwa niezmienniki miejsc:

1. y_1 = [1101]

2. y_2 = [1011]

Pierwszy niezmiennik dotyczy miejsc p_1, p_2, p_4, natomiast drugi p_1, p_3, p_4.

3.2 Weryfikacja klasy xt hipegrafu (algorytm XTREC)

Sprawdzenie klasy hipergrafów jest istotne z punktu widzenia opracowanych metod, ponieważ pozwala na oszacowanie, czy w danym przypadku jest możliwe uzyskanie rozwiązania dokładnego. Hipergraf należący do klasy xt z definicji pozwala na szybkie uzyskanie minimalnego rozwiązania, ponieważ każda jego transwersala dokładna jest jednocześnie transwersalą minimalną. Sprawdzić przynależność hipergrafu do klasy xt pozwala algorytm XTREC, który został wprowadzony w pracy [80].

Dane wejściowe: macierz incydencji hipergrafu H,
Dane wyjściowe: wartość logiczna: prawda / fałsz.

1. Wyznacz wierzchołki, które są incydentne do przynajmiej jednej krawędzi. Następnie dla każdego wierzchołka:

2. Wyznacz zbiór wierzchołków gwiazdy. Gwiazdą nazywamy hipergraf powstały wskutek redukcji hipergrawędzi, które nie są incydentne do danego wierzchołka. Dla każdej krawędzi gwiazdy wyznacz część wspólną $F * V_e(min(R))$, gdzie $R = X - H$, $X = H - st(v_n)$

3. Jeśli $F * V_e(min(R))$!= 0 zwróć fałsz

Algorytm XTREC zwraca wartość "prawda"w przypadku, kiedy sprawdzany hipergraf jest hipergrafem transwersal doładnych (należy do klasy xt). W przeciwnym przypadku zwracana jest wartość "fałsz". Metoda ta pozwala na weryfikację klasy hipergrafu w czasie wielomianowym, a mianowicie $\mathcal{O}(n^2m^3)$, gdzie $n = |V|$, $m = |H|$. Autor niniejszej rozprawy wykorzystuje tę właściwość do oszacowania, czy rozwiązanie problemu selekcji jest możliwe w stosunkowo krótkim i osiągalnym czasie.

3.2.1 Przykład algorytmu XTREC dla hipergrafu, który należy do klasy xt

Dany jest hipergraf H_{xt}, którego macierz incydencji określona następująco:

$$I_{xtrec} = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{matrix}$$

gdzie:

$$V = \{v_1, v_2, v_3, v_4\} \tag{3.1}$$

$$V_e = \{v_1, v_2, v_3, v_4\} \tag{3.2}$$

$$E = \{E_1, E_2, E_3, E_4\} \tag{3.3}$$

V_e definiuje zbiór wierzchołków, ktore należą do przynajmniej jednej krawędzi.
Dla wszystkich wierzchołków $v \in V_e$:

1. v=v_1

   (a) *Wyznacz gwiazdę wierzchołka v_1 oraz F.*

$$St(v_1, H) = (V, \{E_1, E_3\}) \tag{3.4}$$

Gwiazdą wierzchołka v_1 jest zbiór krawędzi incydentnych.

$$F := \{v_1, v_2, v_3\} \tag{3.5}$$

Funkcję F określa zbiór wierzchołków incydentnych do krawędzi gwiazdy.
W dalszej kolejnosci dla każdej krawędzi gwiazdy wyznaczane są kolejno:

i. $H = E_1 = \{v_1, v_3\}$
   A. X=$\{E_2, E_4\}$=$\{\{v_2, v_4\}\}$,$\{\{v_3, v_4\}\}$
      Zbiór X jest dopełnieniem gwiazdy.
   B. R=X-H=$\{\{v_2,v_4\},\{v_4\}\}$
   C. min(R)=$\{v_4\}$
   D. F $\cap V_e$(min(R))=$\{v_1,v_2,v_3\}\cap\{v_4\}$=0
      Część wspólna zbioru F oraz V_e(min(R)).

ii. $H = E_3 = \{v_1,v_2\}$
   iii. X=$\{E_2, E_4\}$=$\{\{v_2,v_4\},\{v_3,v_4\}\}$
   iv. R=X-H=$\{\{v_4\},\{v_3,v_4\}\}$
   v. min(R)=$\{v_4\}$
   vi. F$\cap V_e$(min(R))=$\{v_1,v_2,v_3\}\cap\{v_4\}$=0

v=v_2

1. *Wyznacz gwiazdę wierzchołka v_2 oraz F.*

$$St(v_2, H) = (V, \{E_2, E_3\}) \tag{3.6}$$

$$F := \{v_1, v_2, v_4\} \tag{3.7}$$

Dla każdej krawędzi gwiazdy:

(a) $H = E_2 = \{v_2, v_4\}$
   i. X=$\{E_1, E_4\}$=$\{\{v_1,v_3\},\{v_3,v_4\}\}$
   ii. R=X-H=$\{\{v_1,v_3\},\{v_3\}\}$
   iii. min(R)=$\{v_3\}$
   iv. F*V_e(min(R))=$\{v_1,v_2,v_4\}$*$\{v_3\}$=0

(b) $H = E_3 = \{v_1, v_2\}$
   i. X=$\{E_1, E_4\}$=$\{\{v_1,v_3\},\{v_3,v_4\}\}$
   ii. R=X-H=$\{\{v_3\},\{v_3,v_4\}\}$
   iii. min(R)=$\{v_3\}$

iv. F*V_e(min(R))={v_1,v_2,v_4}*{v_3}=0

v=v_3

1. *Wyznacz gwiazdę wierzchołka v_3 oraz F.*

$$St(v_3, H) = (V, \{E_1, E_4\}) \tag{3.8}$$

$$F := \{v_1, v_3, v_4\} \tag{3.9}$$

Dla każdej krawędzi gwiazdy:

(a) $H = E_1 = \{v_1, v_3\}$

i. X={E_2, E_3}={{v_2,v_4},{v_1,v_2}}

ii. R=X-H={{v_2,v_4},{v_2}}

iii. min(R)={v_2}

iv. F*V_e(min(R))={v_1,v_3,v_4}*{v_2}=0

(b) $H = E_4 = \{v_3, v_4\}$

i. X=(E_2, E_3)={{v_2,v_4},{v_1,v_2}}

ii. R=X-H={{v_2},{v_1,v_2}}

iii. min(R)={v_2}

iv. F*V_e(min(R))={v_1,v_3,v_4}*{v_2}=0

v=v_4

1. Analogicznie F*V_e(min(R))=0

Z uwagi na fakt, że w każdym rozpatrywanym przypadku $F * V_e(min(R)) = 0$, hipergraf H_{xtrec} należy do klasy xt. Na potrzeby badań algorytm *xtrec* został zaimplementowany zgodnie ze wskazówkami autora oraz dodany do biblioteki *Hippo* jako natywny składnik pozwalający sprawdzać, czy dany hipergraf jest xt.

3.2.2 Przykład algorytmu XTREC dla hipergrafu, który nie należy do klasy xt

Dany jest hipergraf $mathcal H_{notxt}$ przedstawiony na rysunku 2.4, którego macierz incydencji określona następująco:

$$I_{xtrec} = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} E_1 \\ E_2 \\ E_3 \end{matrix}$$

gdzie:

$$V = \{v_1, v_2, v_3, v_4, v_5\} \tag{3.10}$$

$$V_e = \{v_1, v_2, v_3, v_4, v_5\} \tag{3.11}$$

$$E = \{E_1, E_2, E_3\} \tag{3.12}$$

V_e definiuje zbiór wierzchołków, ktore należą do przynajmniej jednej krawędzi.
Dla wszystkich wierzchołków $v \in V_e$:

1. v=v_1

   (a) *Wyznacz gwiazdę wierzchołka v_1 oraz F.*

   $$St(v_1, H) = (V, \{E_1\}) \tag{3.13}$$

   Gwiazdą wierzchołka v_1 jest zbiór krawędzi incydentnych.

   $$F := \{v_1, v_2, v_3\} \tag{3.14}$$

   Funkcję F określa zbiór wierzchołków incydentnych do krawędzi gwiazdy.
   W dalszej kolejnosci dla każdej krawędzi gwiazdy wyznaczane są kolejno:

   i. $H = E_1 = \{v_1, v_2, v_3\}$

      A. X=$\{E_2, E_3\}$=$\{\{v_2, v_5\}\}$,$\{\{v_4, v_5\}\}$
         Zbiór X jest dopełnieniem gwiazdy.

      B. R=X-H=$\{\{v_5\}$,$\{v_4, v_5\}\}$

      C. min(R)=$\{v_5\}$

      D. F*V_e(min(R))=$\{v_1$,v_2,$v_3\}$*$\{v_5\}$=0
         Część wspólna zbioru F oraz V_e(min(R)).

2. v=v_2

   (a) *Wyznacz gwiazdę wierzchołka v_2 oraz F.*

   $$St(v_2, H) = (V, \{E_1, E_2\}) \tag{3.15}$$

   $$F := \{v_1, v_2, v_3, v_5\} \tag{3.16}$$

   Dla każdej krawędzi gwiazdy:

   i. $H = E_1 = \{v_1, v_2, v_3\}$

A. X=$\{E_3\}$=$\{\{\{v_4, v_5\}\}$
   Zbiór X jest dopełnieniem gwiazdy.

B. R=X-H=$\{\{v_4, v_5\}\}$

C. min(R)=$\{v_4, v_5\}$
   Minimum zbioru R.

D. F*V_e(min(R))=$\{v_1$,v_2,v_3,$v_5\}$*$\{v_4, v_5\}$=$\{v_5\}$
   Algorytm zwraca FALSE i kończy swoje działanie.

ii. $H = E_2 = \{v_2, v_5\}$

A. X=$\{E_3\}$=$\{v_4$,$v_5\}$

B. R=X-H=$\{\{v_4\}\}$

C. min(R)=$\{v_4\}$

D. F*V_e(min(R))=$\{v_1$,v_2,v_3,$v_5\}$*$\{v_4\}$=0

3. v=v_3

(a) *Wyznacz gwiazdę wierzchołka v_3 oraz F.*

$$St(v_3, H) = (V, \{E_1\}) \tag{3.17}$$

$$F := \{v_1, v_2, v_3\} \tag{3.18}$$

Dla każdej krawędzi gwiazdy:

i. $H = E_1 = \{v_1, v_2, v_3\}$

A. X=$\{E_2, E_3\}$=$\{\{v_2, v_5\}\}$,$\{\{v_4, v_5\}\}$

B. R=X-H=$\{\{v_5\}$,$\{v_4, v_5\}\}$

C. min(R)=$\{v_5\}$

D. F*V_e(min(R))=$\{v_1$,v_2,$v_3\}$*$\{v_5\}$=0

4. v=v_4

(a) *Wyznacz gwiazdę wierzchołka v_4 oraz F.*

$$St(v_4, H) = (V, \{E_3\}) \tag{3.19}$$

$$F := \{v_4, v_5\} \tag{3.20}$$

Dla każdej krawędzi gwiazdy:

i. $H = E_3 = \{v_4, v_5\}$

A. X=$\{E_1, E_2\}$=$\{\{v_1, v_2, v_3\}\}$,$\{\{v_2, v_5\}\}$

B. R=X-H={{v_1,v_2,v_3},{v_2}}

C. min(R)={v_2}

D. F*V_e(min(R))={v_4,v_5}*{v_2}=0

5. v=v_5

(a) *Wyznacz gwiazdę wierzchołka v_5 oraz F.*

$$St(v_5, H) = (V, \{E_2, E_3\}) \tag{3.21}$$

$$F := \{v_2, v_4, v_5\} \tag{3.22}$$

Dla każdej krawędzi gwiazdy:

i. $H = E_2 = \{v_2, v_5\}$

A. X={E_1}={v_1, v_2, v_3}

B. R=X-H={{v_1,v_3}

C. min(R)={v_1, v_3}

D. F*V_e(min(R))={v_2,v_4,v_5}*{v_1,v$_3$}=0

ii. $H = E_3 = \{v_4, v_5\}$

A. X={E_1}={v_1, v_2, v_3}

B. R=X-H={{v_1,v_2,v_3}

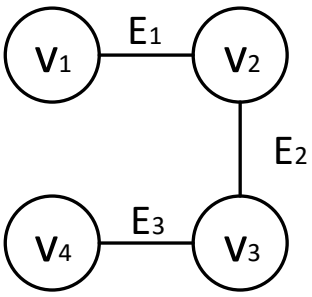
C. min(R)={v_1, v_2, v_3}

D. F*V_e(min(R))={v_2,v_4,v_5}*{v_1, v_2, v_3}={v_2} Ponownie FALSE, co świadczy o poprawności przykładu.

Z uwagi na fakt, że wierzchołki v_2, v_5 tworzą transwersalę minimalną, która nie jest dokładna, prezentowany hipergraf nie należy do klasy xt (co potwierdza algorytm XTREC, zwracając false dla wierzchołków v_2 i v_5.

3.3 Algorytm z nawrotami wyznaczający pokrycie wierzchołkowe w grafie

Algorytm z nawrotami cechuje się wykładniczym czasem wykonywania, jednak w rezultacie zwraca rozwiązanie dokładne. Metoda jest dobrze znana i przeszukuje wszystkie możliwe kombinacje w celu znalezienia najlepszego rozwiązania [92–94]. W pierwszej kolejności wybierany i redukowany jest wierzchołek oznaczony najmniejszą wartością, zgodnie z porządkiem leksykograficznym [22]. Gdy pozostały zbiór wierzchołków w dalszym ciągu



Rysunek 3.1: Przykładowy graf G_1

pozostaje pokryciem hipergrafu, a uzyskany rezultat jest lepszy od dotychczasowego, staje się on najlepszym rozwiązaniem. W dalszym kroku algorytm rekurencyjnie przeprowadza proces poszukiwania najlepszego rozwiązania dla wszystkich wierzchołków.

Dane wejściowe: macierz incydencji grafu G,
Dane wyjściowe: zbiór wierzchołków będących pokryciem wierzchołkowym.

1. *Ustal pokrycie jako zbiór wszystkich wierzchołków.*
2. *Wybierz wierzchołek oznaczony najmniejszą wartością, zgodnie z porządkiem leksykograficznym.*
3. *Dla wybranego oraz dla każdego kolejnego wierzchołka powtórz kroki:*
   (a) *Sprawdź, czy po usunięciu wierzchołka ze zbioru, istniejące dadzą pokrycie.*
   (b) *Jeśli bieżące pokrycie jest lepsze od dotychczasowego, ustaw je jako najlepsze*
4. *Po wykonaniu pętli otrzymany zbiór wierzchołków jest szukanym pokryciem wierzchołkowym.*

Przykład

Przykład działania algorytmu z nawrotami zostanie zaprezentowany w oparciu o graf G_1. Graf ten składa się z czterech wierzchołków $V = \{v_1, ..., v_4\}$. W pierwszej kolejności do analizy zostaje wybrany wierzchołek v_1. Po wykonaniu redukcji graf składa się z trzech wierzchołków $V^1 = \{v_2, v_3, v_4\}$. Stanowi on pokrycie, dlatego też zbiór ten staje się najlepszym rozwiązaniem. Następnie redukowany jest wierzchołek v_2, natomiast zbiór ma postać $V^2 = \{v_3, v_4\}$. Nie stanowi on pokrycia, ponieważ krawędź E_1 nie jest incydentna do jakiegokolwiek wierzchołka z tego zbioru. Algorytm powraca do

poprzedniego rozwiązania (V^1) oraz redukuje wierzchołek v_3. Zbiór $V^3 = \{v_2, v_4\}$ stanowi pokrycie, dlatego też staje się najlepszym rozwiązaniem. W dalszym kroku redukowany jest wierzchołek v_4. Ponieważ $V^4 = \{v_2\}$ nie stanowi pokrycia, algorytm powraca do poprzedniego rozwiązania (V^3). Analizie poddano wszystkie wierzchołki, dlatego też rozwiązanie $V^3 = \{v_2, v_4\}$ jest szukanym zbiorem pokrywającym graf G_1.

3.4 Algorytm zachłanny wyznaczający pokrycie wierzchołkowe w grafie

Algorytm zachłanny poszukuje rozwiązania na podstawie decyzji lokalnie optymalnej. Niewątpliwą zaletą algorytmu jest jego wielomianowy czas. Niestety nie jest możliwe określenie, czy znalezione rozwiązanie jest optymalne globalnie [92, 93, 95]. W algorytmie występuje pojęcie istotnego wierzchołka. Istotny wierzchołek to jedyny wierzchołek incydentny z daną hiperkrawędzią. Innymi słowy jeśli dla danego wierzchołka istnieje wiersz z pojedynczą jedynką oznacza to, że tylko ten wierzchołek należy do danej hiperkrawędzi i musi być częścią każdego pokrycia.

Dane wejściowe: macierz incydencji grafu G,
Dane wyjściowe: zbiór pokrycia V'

(a) *Zbiór pokrycia $V' = \emptyset$.*

(b) *Wybierz istotny wierzchołek, jeśli nie istnieje - o największym stopniu.*

(c) *Dodaj wierzchołek do zbioru pokrycia oraz usuń go z grafu wraz z incydentnymi krawędziami.*

(d) *Jeśli zbiór V' nie tworzy pokrycia, wróć do kroku 2. Jeśli tworzy, algorytm kończy działanie.*

Przykład

Przykład działania algorytmu zachłannego zostanie zaprezentowany w oparciu o graf G_1. Zgodnie z powyższym schematem algorytmu, w pierwszej kolejności ustalany jest zbiór pokrycia $V' = \emptyset$. W zadanym grafie nie istnieje wierzchołek istotny, dlatego też wybierany jest wierzchołek o największym stopniu. Istnieją dwa takie wierzchołki: v_2 oraz v_3. Zgodnie z porządkiem leksykograficznym wybierany jest wierzchołek v_2. Zostaje on usunięty z grafu wraz z krawędziami incydentnymi E_1 oraz E_2 oraz dodany do zbioru pokrycia $V' = \{v_2\}$. Zbiór ten nie stanowi pokrycia grafu G_1, dlatego też powtarzany jest krok drugi algorytmu. Wybierany jest wierzchołek v_3, usuwany z grafu

bieżącego wraz z incydentną krawędzią E_3 oraz dodawany do V'. Zbiór ten stanowi pokrycie, dlatego też szukanym rozwiązaniem jest zbiór $V' = \{v_2, v_3\}$.

3.5 Szybki algorytm redukcji

Szybki algorytm redukcji wykorzystuje zależności oraz powiązania, jakie przechowuje w swojej strukturze hipergraf. Redukcja dotyczy zmiejszenia rozmiaru jego macierzy incydencji, co ułatwia i przyśpiesza znalezienie pokrycia. W bieżącej pracy została zastosowana zmodyfikowana metoda przedstawiona w pracy [88]. Czas wykonywania algorytmu jest wielomianowy. W metodzie występuje pojęcie kolumny dominującej. Kolumna dominuje nad inną wtedy i tylko wtedy, gdy jej elementy są większe, badź równe od zdominowanej. Analogicznie kolumna jest zdominowana przez inną wtedy i tylko wtedy, gdy jej elementy są mniejsze od dominującej.

Dane wejściowe: macierz incydencji hipergrafu H,
Dane wyjściowe: macierz incydencji hipergrafu zredukowanego H'.

(a) *Sprawdź, czy macierz zawiera istotny wierzchołek. Takie wierzchołki muszą być częścią każdego pokrycia.*

(b) *Dla każdego wiersza i kolumny:*

   i. *Zredukuj zdominowane kolumny,*

   ii. *Zredukuj dominujące wiersze.*

(c) *Pozostała macierz jest szukanym rozwiązaniem.*

Przykład

Dana jest macierz incydencji A_1 hipergrafu H_1.

$$A_1 = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \end{matrix}$$

W macierzy incydencji A_1 hipegrafu H_1 istnieje istotna kolumna reprezentująca wierzchołek v_6, ponieważ jest on jako jedyny incydentny z krawędzią E_4. W takiej

sytuacji v_6 musi być częścią każdej transwersali hipergrafu. Wierzchołek ten jest dodawany do rozwiązania oraz usuwany z macierzy wraz z incydentnymi wierszami. W dalszej kolejności usuwane są zdominowane kolumny (reprezentujące wierzchołki v1, v2, v3, v5). Macierz A'_1 finalnie wygląda następująco:

$$A'_1 = \begin{array}{c} \begin{matrix} v_4 & v_6 \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \end{array} \begin{matrix} E_1 \\ E_5 \end{matrix}$$

Macierz A'_1 jest docelową macierzą uzyskaną po redukcji, ponieważ nie jest możliwe przeprowadzenie dalszych cykli. Szybki algorytm redukcji w przypadku ogólnym nie znajduje rozwiązania, a umożliwia jego znalezienie za pomocą innych algorytmów. W przypadku niektórych hipergrafów rezultat algorytmu pokrywa się z rozwiązaniem dokładnym.

3.6 Algorytm X

Algorytm X oraz jego efektywna implementacja zostały przedstawione w pracy [96]. Autor proponuje sposób rozwiązania problemu pokrycia dokładnego zbioru. Biorąc pod uwagę zbiór S podzbiorów zbioru X, pokrycie dokładne jest podzbiorem S' z S tak, że każdy element w X jest zawarty w dokładnie jednym podzbiorze w S'. Jeden pozbiór oznacza, że każdy element w X jest pokryty dokładnie jednym pozbiorem w S'. Generalnie metoda ta jest rekursywnym algorytmem backtackingowym przeszukującym w głąb, natomiast problem pokrycia jest reprezentowany za pomocą macierzy binarnej *A*.

Dane wejściowe: macierz incydencji A,
Dane wyjściowe: zredukowana macierz A'.

(a) Jeśli macierz *A* nie ma kolumn, bieżące rozwiązanie staje się rozwiązaniem końcowym. Zakończ algorytm.

(b) W innym wypadku wybierz kolumnę *c*.

(c) Wybierz wiersz *r*, taki aby $A_{r,c} = 1$ (niedeterministycznie[1]).

(d) Dodaj wiersz *r* do rozwiązania bieżącego.

[1]Wybór niedeterministyczny oznacza, że algorytm zasadniczo klonuje się w niezależne podalgorytmy, a każdy podalgorytm dziedziczy bieżącą macierz A, ale zmniejsza ją w odniesieniu do innego wiersza r. Jeśli kolumna c jest całkowicie zerowa, nie ma żadnych algorytmów podrzędnych, a proces kończy się niepowodzeniem.

(e) Dla każdej kolumny *j* takiej, że $A_{r,j} = 1$ wykonuj:

i. Dla każdego wiersza *i* takiego, że $A_{i,j} = 1$ wykonuj:

A. Usuń wiersz *i* z macierzy *A*.

ii. Usuń kolumnę *j* z macierzy *A*.

(f) Powtórz algorytm rekurencyjnie dla zredukowanej macierzy *A*.

Autor algorytmu X przedstawił efektywną implementację w formie listy czterokierunkowej. DLX (ang. Dancing Links) jest metodą przedstawienia danych i operacji nad nimi, wykorzystywanej w skutecznej implementacji algorytmu X. Idea opiera się na obserwacji takiej, że dla wartości cyklicznej listy kierunkowej (*x* jest węzłem listy):
$x.lewy.prawy \leftarrow x.prawy$,
$x.prawy.lewy \leftarrow x.lewy$
będą usuwać węzeł *x*, natomiast
$x.lewy.prawy \leftarrow x$
$x.prawy.lewy \leftarrow x$
przywróci pozycję *x*, zakładając powiązanie *x.prawy* oraz *x.lewy* niezmienione.

Biblioteka *Hippo*, która została rozwinięta przez autora rozprawy zawiera zaimplementowaną metodę DLX oraz wyszukiwania transwersali za pomocą algorytmu X.

3.7 Istniejące metody dekompozycji sieci Petriego

Projektowane systemy w dzisiejszych czasach charakteryzują się dużą złożonością. Projektanci z reguły spotykają się z sytuacją, w której rozmiar sterownika wykracza poza narzucone ramy prototypowanego układu. Jednym z możliwych rozwiązań jest odpowiedni podział systemu na osobne moduły, współpracujące ze sobą. Dekompozycja ułatwia w szczególności proces projektowania. W przypadku systemów współbieżnych opisanych sieciami Petriego powszechnie stosowaną metodą dekompozycji jest podział na sekwencyjne automaty składowe [10, 22, 97]. Poniżej omówiono najpopularniejsze metody dekompozycji sieci Petriego.

3.7.1 Metoda oparta o grafy

Pierwszym omawianym algorytmem jest metoda dekompozycji sieci Petriego wykorzystująca kolorowanie grafu współbieżności lub też wyszukiwania pokrycia

klikowego dopełnienia grafu współbieżności. [19, 22, 64]. Składa się z następujących etapów:

(a) Utworzenie makrosieci dla danej sieci Petriego.

(b) Wyznaczenie zbioru osiągalności.

(c) Wyznaczenie grafu współbieżności.

(d) Kolorowanie grafu współbieżności.

(e) Zastapienie makromiejsc odpowiednimi podzbiorami miejsc danej sieci Petriego.

Algorytm przebiega następująco: w pierwszej kolejności generowana jest makrosieć, która jest skondensowaną wersją wersją sieci wejściowej. Celem tego działania jest usunięcie elementów, które ostatecznie nie wpłynie na wynik analizy, natomiast zachowywana jest podstawowa struktura i właściwości sieci. Następnie analizowane są zmiany oznakowania sieci w miarę odpalania tranzycji. Zapisywane są wszystkie możliwe stany, które tworzą ostatecznie zbiór osiągalności. Krok kolejny polega na wyznaczeniu grafu współbieżności. Zawiera on informacje o relacjach pomiędzy miejscami dekomponowanej sieci Petriego: wierzchołki odpowiadają miejscom sieci Petriego, natomiast krawędzie relacje pomiędzy miejscami. Następnie należy pokolorować graf. Wyznaczone zbiory niezależne zawierają makromiejsca wchodzące w skład poszczególnych automatów sekwencyjnych.

Główną wadą metody jest przede wszystkim wykładnicza złożoność obliczeniowa. Drugim problemem jest to, że kolorowanie opiera się o zastosowanie grafu współbieżności, który nie zawsze odpowiada rzeczywistej relacji współbieżności. [10]

3.7.2 Metoda oparta o hipergrafy

W pracach [22, 66] zaporoponowano nowe podejście do tematu dekompozycji. Autorka przedstawia algorytm wykorzystujący teorię hipergrafów. Istotny jest fakt, że każda transwersala dokładna hipergrafu współbieżności odpowiada pojedynczemu automatowi sekwencyjnemu sieci Petriego. W metodzie można wyróżnić następujące etapy:

(a) Utworzenie makrosieci.

(b) Wyznaczenie hipergrafu współbieżności.

(c) Wyznaczenie hipergrafu sekwencyjności.

(d) Wyznaczenie hipergrafu dualnego do hipergrafu sekwencyjności i redukcja cykliczna.

(e) Wyznaczenie najmniejszej transwersali dokładnej hipergrafu uzyskanego w poprzednim punkcie.

W pierwszej kolejności tworzona jest makrosieć, a następnie wyznaczany jest hipergraf współbieżności. Reprezentuje on właściwą współbieżność pomiędzy miejscami sieci Petriego. Wierzchołki hipergrafu odpowiadają miejscom, natomiast hiperkrawędzie określają relacje współbieżności pomiędzy tymi miejscami. W dalszej kolejności określany jest hipergraf sekwencyjności. W tym celu wyznaczane są kolejne transwersale dokładne w hipergrafie współbieżności. Wierzchołki hipergrafu sekwencyjności odpowiadają wierzchołkom hipergrafu współbieżności, natomiast krawędzie określonym transwersalom dokładnym. Kolejny krok polega na wykonaniu operacji transpozycji, a następnie wykonanie redukcji za pomocą szybkiego algorytmu redukcji (ang. Fast Reduction Algorithm - FRA). Krok ostatni polega na wyznaczeniu najmniejszej transwersali dokładnej, która definiuje rozwiązanie. Istotną kwestią jest to, że nie każdy hipergraf posiada transwersalę dokładną, dlatego też rozwiązanie dokładne problemu może nie zostać uzyskane. W takiej sytuacji należy wyznaczyć najmniejszą transwersalę.

Główną wadą metody jest wykładnicza złożoność obliczeniowa w przypadku ogólnym. Zarówno hipergraf współbieżności, jak i sekwencyjności, a także selekcja są wykładnicze (liczba krawędzi może być wykłądnicza, choć kolejne krawędzie wyznaczane są wielomianowo).

3.7.3 Metoda oparta o inwarianty

Dekompozycja systemów dyskretnych może zostać wykonana także w oparciu o inwarianty miejsc. Głównym rdzeniem metody jest algorytm Martineza-Silvy pozwalający na uzyskanie niezmienników miejsc dla wejściowej sieci Petriego. Można wyróżnić następujące etapy:

- Wyznaczenie niezmienników miejsc dla danej sieci Petriego.
- Wyznaczenie poprawnych podsieci automatowych.
- Selekcja podsieci automatowych.
- Wyznaczenie rozwiązania.

W pierwszej kolejności dla danej sieci Petriego wyznaczane są niezmienniki miejsc za pomocą dobrze znanej metody Martineza-Silvy. Uzyskane niezmienniki miejsc niekoniecznie muszą odzwierciedlać poprawne podsieci automatowe, dlatego w dalszej kolejności niezbędne jest wybranie tylko tych właściwych. Krok kolejny polega na

selekcji podsieci automatych niezbędnych do pokrycia sieci właściwej i tym samym wyznaczenie rozwiązania.

Główną wadą metody jest jej wykładnicza złożoność obliczeniowa (liczba inwariantów rośnie wykładniczo). Uzyskane inwarianty nie zawsze odpowiadają poprawnym podsieciom automatowym i dlatego też niezbędne jest wykonanie dodatkowego sprawdzenia. Ostatnim etapem jest selekcja, która w przypadku ogólnym charakteryzuje się wykładniczą złożonością obliczeniową.

Rozdział 4

Zaproponowane metody

Rozdział bieżący dotyczy proponowanych metod. Dekompozycja współbieżnego systemu sterowania pozwala podzielić go na mniejsze elementy składowe, które można wykonywać równolegle. Częścią składową procesu dekompozycji jest selekcja podsieci automatowych. Problem selekcji nie jest trywialny i warto podkreślić, że nie jest znany wielomianowy algorytm pozwalający na uzyskanie rozwiązania minimalnego w przypadku ogólnym. Wykorzystywane są algorytmy przybliżone, które w rezultacie dostarczają rozwiązanie w akceptowalnym czasie.

Pierwsze dwie autorskie metody dotyczą kompletnej dekompozycji współbieżnego systemu sterowania w oparciu o niezmienniki miejsc oraz hipergrafy. Zaproponowano algorytm wykorzystujący metodę przedstawioną w punkcie 3.1, a także drugi zmodyfikowany o usprawnienia zorientowane na pokrycie sieci, a także opracowaną autorską metodę selekcji. Zgodnie z przeglądem literaturowym, dotychczas nie wykonywano dekompozycji wykorzystując inwarianty oraz hipergrafy. Autor ma także na celu ukazanie innego podejścia niż wyżej wymienione, które pozwala na usprawnienie procesu.

Dwie kolejne z przedstawianych metod wykorzystują hipergraf transwersal dokładnych, nazywany dalej xt-hipergrafem. Zaproponowane metody selekcji podsieci automatowych sieci Petriego pozwalają na uzyskanie rozwiązania minimalnego. Jest to możliwe z uwagi na wykorzystanie transwersali dokładnej xt-hipergrafu. Rozwiązanie dokładne uzyskiwane jest automatycznie, ponieważ każda transwersala dokładna xt-hipergrafu jest jednocześnie minimalna. Wynikiem algorytmu jest minimalne pokrycie sieci w przypadku, kiedy dany hipegraf należy do klasy xt.

4.1 Metody dekompozycji oparte o niezmienniki miejsc oraz hipergrafy

4.1.1 Sformułowanie problemu

Algorytm dekompozycji opisany w punktach 3.7.1, 3.7.2, 3.7.3 jest realizowany w najgorszym przypadku w czasie wykładniczym, dlatego też w niniejszym doktoracie opracowano i zaproponowano metodę wykorzystującą algebrę liniową oraz niezmienniki miejsc (p-inwarianty). Algorytm ten uzupełnia metodę wykorzystującą teorię hipergrafów, a także ukazuje inne podejście do problemu dekompozycji.

Wyznaczanie niezmienników miejsc w metodzie 1 odbywa się za pomocą klasycznego algorytmu Martineza-Silvy, natomiast w metodzie 2 za pomocą zmodyfikowanego algorytmu, którego pierwotną wersję przedstawiono w punkcie 3.1. Zmodyfikowany algorytm Martineza-Silvy zawiera następujące zmiany:

(a) Cykliczne sprawdzanie pokrycia sieci niezmiennikami miejsc, przerywanie algorytmu w momencie znalezienia pokrycia.

(b) Wybranie poprawnych inwariantów (podsieci automatowych).

Pierwsze usprawnienie pozwala na przerwanie algorytmu wyznaczania niezmienników miejsc w momencie, kiedy sieć jest już pokryta. Skutkuje to krótszym czasem wykonywania, a także odrzuca nadmiarowe niezmienniki. Rozwiązanie jednak może nie być optymalne.

Usprawnienie drugie pozwala na uzyskanie podsieci automatowych poprzez wybranie poprawnych inwariantów (zawierających jeden znacznik miejsca względem znakowania początkowego).

4.1.2 Idea proponowanej metody 1

Dane wejściowe: sieć Petriego przedstawiona za pomocą macierzy C.
Dane wyjściowe: składowe automatowe po dekompozycji sieci Petriego.

(a) *Wyznaczenie niezmienników miejsc za pomocą metody Martineza-Silvy.*

i. *Utworzenie macierzy startowej [B | C].* Zbudowana jest ona z dwóch pomniejszych, a mianowicie *B* jest jednostkową macierzą miejsc, natomiast *C* jest macierzą incydencji sieci Petriego: -1 oznacza łuk wchodzący, 1 wychodzący.

ii. *Dla każdej tranzycji t_m:*

A. Zsumowanie każdej pary wierszy, których elementy w *m-tej* kolumnie macierzy *C* mają wartości różne od 0 i w sumie dadzą 0.

B. Usunięcie wszystkich wierszy, których elementy w *m-tej* kolumnie macierzy *C* są różne od 0.

C. Usunięcie nieminimalnych inwariantów (zawierających inne inwarianty) poprzez redukcję nadmiarowych wierszy (wierszy, które binarnie pokrywają inne).

iii. Po wykonaniu pętli otrzymana macierz składa się z wektorów odpowiadających p-niezmiennikom danej sieci.

(b) *Wyznaczenie podsieci automatowych poprzez wybranie poprawnych niezmienników.* Krok ten polega na wyznaczeniu podsieci automatowych z inwariantów. Idea postępowania jest następująca: wybierane są tylko inwarianty poprawne czyli takie, które mają jeden znacznik.

(c) *Selekcja podsieci automatowych z wykorzystaniem metody transwersal dokładnych 1.* Następnie wykonywana jest selekcja podsieci automatowych z wykorzystaniem algorytmu transwersal dokładnych (1) opartego o xt-hipergraf opisanego w rozdziale 4.2. Macierz traktowana jest jako macierz incydencji hipergrafu. W pierwszej kolejności wyznaczany jest hipergraf dualny, a następnie wykonana redukcja za pomocą szybkiego algorytmu redukcji. Wyznaczane są wierzchołki istotne, a także redukowane dominujące wiersze i zdominowane kolumny. Następnie dokonywana jest klasyfikacja hipergrafu selekcji za pomocą algorytmu xtrec. Jeśli należy do klasy xt, rozwiązanie problemu dekompozycji wyznacza transwersala dokładna (algorytm X). Jeśli nie, rozwiązanie wyznaczane jest za pomocą algorytmu z nawrotami.

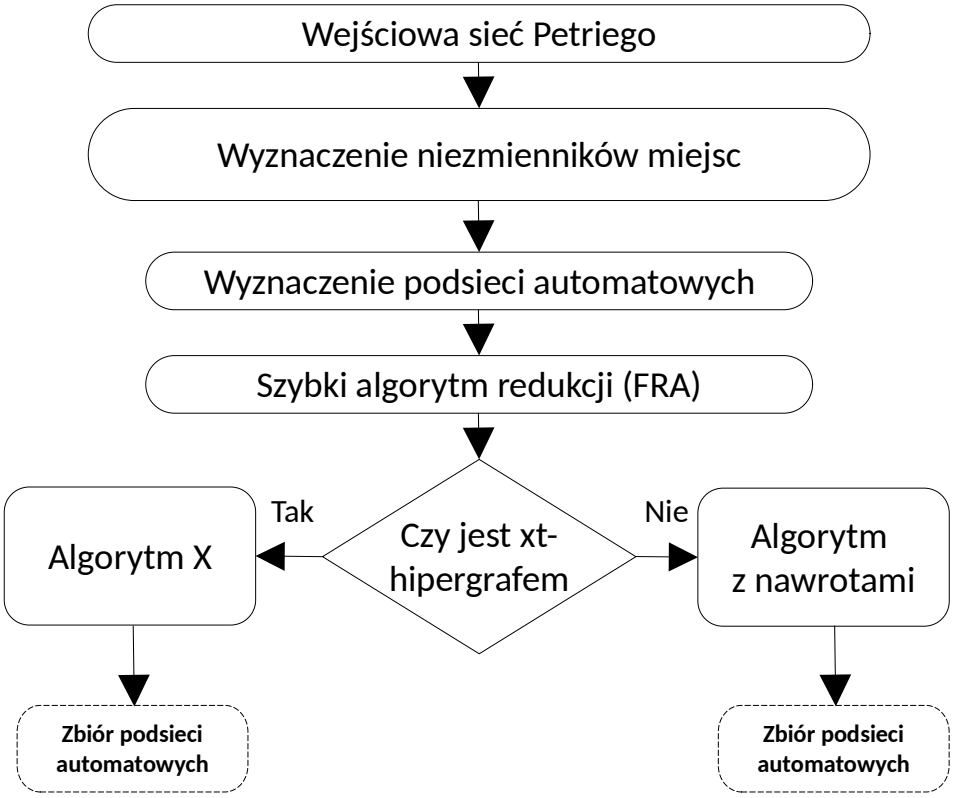
Ideę metody zaprezentowano na rysunku 4.1.

4.1.3 Idea proponowanej metody 2

Dane wejściowe: sieć Petriego przedstawiona za pomocą macierzy C.
Dane wyjściowe: składowe automatowe po dekompozycji sieci Petriego.

(a) *Wyznaczenie niezmienników miejsc za pomocą modyfikowanej metody Martineza-Silvy.*

i. *Utworzenie macierzy startowej [B | C].* Zbudowana jest ona z dwóch pomniejszych, a mianowicie *B* jest jednostkową macierzą miejsc, natomiast



Rysunek 4.1: Rysunek przedstawiający ideę metody dekompozycji (1) współbieżnych systemów sterowania za pomocą p-inwariantów

C jest macierzą incydencji sieci Petriego: -1 oznacza łuk wchodzący, 1 wychodzący.

ii. *Dla każdej tranzycji* t_m*:*

A. Zsumowanie każdej pary wierszy, których elementy w *m-tej* kolumnie macierzy *C* mają wartości różne od 0 i w sumie dadzą 0.

B. Usunięcie wszystkich wierszy, których elementy w *m-tej* kolumnie macierzy *C* są różne od 0.

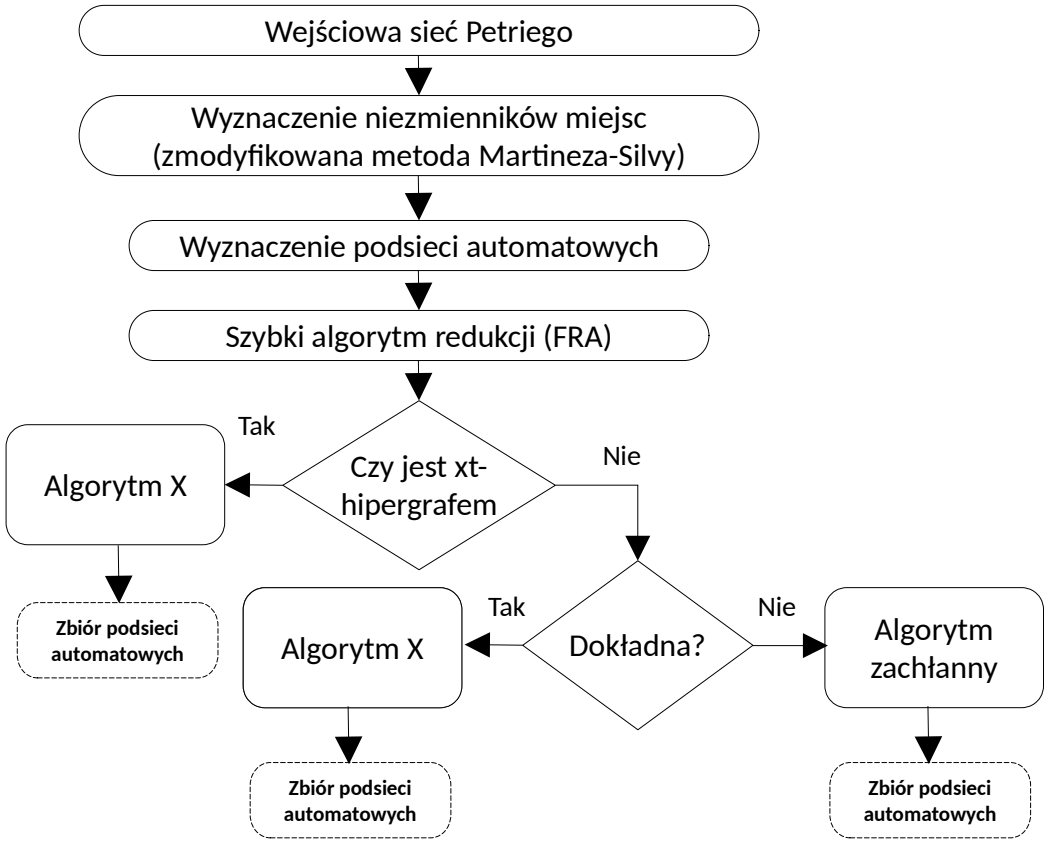
C. Redukcja tranzycji za pomocą Szybkiego Algorytmu Redukcji.

D. Określenie poprawnych składowych automatowych (składowa automatowa jest poprawna, jeśli zawiera jeden znacznik miejsca względem znakowania początkowego).

E. Sprawdzenie, czy sieć jest pokryta. Jeśli tak, przerwanie pętli.

iii. Po wykonaniu pętli otrzymana macierz składa się z wektorów odpowiadających p-niezmiennikom danej sieci.

(b) *Wyznaczenie podsieci automatowych poprzez wybranie poprawnych niezmienników.* Krok ten polega na wyznaczeniu podsieci automatowych z inwariantów. Idea

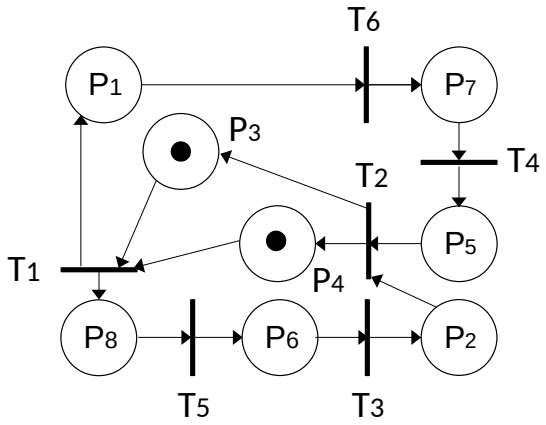


Rysunek 4.2: Rysunek przedstawiający ideę metody dekompozycji (2) współbieżnych systemów sterowania za pomocą p-inwariantów

postępowania jest następująca: wybierane są tylko inwarianty poprawne czyli takie, które mają jeden znacznik.

(c) *Selekcja podsieci automatowych z wykorzystaniem metody transwersal dokładnych 2.* Następnie wykonywana jest selekcja podsieci automatowych z wykorzystaniem algorytmu transwersal dokładnych (2) opartego o xt-hipergraf opisanego w rozdziale 4.2. Macierz traktowana jest jako macierz incydencji hipergrafu. W pierwszej kolejności wyznaczany jest hipergraf dualny, a następnie wykonana redukcja za pomocą szybkiego algorytmu redukcji. Wyznaczane są wierzchołki istotne, a także redukowane dominujące wiersze i zdominowane kolumny. Następnie dokonywana jest klasyfikacja hipergrafu selekcji za pomocą algorytmu xtrec. Jeśli należy do klasy xt, rozwiązanie problemu dekompozycji wyznacza transwersala dokładna (algorytm X). Jeśli nie, następuje próba wyznaczenia transwersali dokładnej. Jeśli istnieje, staje się rozwiązaniem, jeśli nie, rozwiązanie wyznaczane jest za pomocą algorytmu zachłannego.

Ideę metody zaprezentowano na rysunku 4.2.



Rysunek 4.3: Rysunek przedstawiający przykładową sieć P_{bridge}

4.1.4 Przykład metody - hipergraf należy do klasy xt

Dana jest sieć Petriego P_{bridge}, która została przedstawiona na rysunku 4.3.

Macierz incydencji sieci P_{bridge} jest następująca:

$$\mathrm{P}_{bridge}= \begin{array}{c} \begin{matrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \end{matrix} \\ \begin{bmatrix} 1 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{matrix}$$

Znakowanie początkowe jest następujące: $M_0 = [00110000]$.

Zgodnie z zaproponowanym algorytmem krok pierwszy polega na wyznaczeniu niezmienników miejsc na podstawie wejściowej sieci Petriego za pomocą usprawionego algorytmu Martineza-Silvy. Usprawnienie polegające na przerwaniu algorytmu w momencie, kiedy sieć jest pokryta niezmiennikami możliwe jest uzyskanie krótszego czasu wyszukiwania inwariantów. Zostanie to szerzej omówione w rozdziale dotyczącym badań eksperymentalnych. Finalnie zostają wyznaczone następujące inwarianty:

(a) $I_1 = [01100101]$

(b) $I_2 = [01010101]$

(c) $I_3 = [10101010]$

(d) $I_4 = [10011010]$

Krok kolejny polega na wybraniu poprawnych inwariantów. W ten sposób uzyskiwane są podsieci automatowe. Ponieważ wszystkie wyznaczone inwarianty są poprawne, stają się one odpowiednio kolejnymi podsieciami automatowymi.
Następnie przygotowywana jest macierz selekcji, która zawiera wszystkie wyznaczone podsieci automatowe. Jest ona traktowana jak hipergraf oraz poddawana redukcji. Zredukowana macierz H_S jest przedstawiona poniżej. Wiersze hipergrafu selekcji odpowiadają miejscom, natomiast kolumny podsieciom automatowym.

$$\mathrm{H}_S = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Zredukowany hipergraf selekcji H_S należy do klasy xt i posiada dwie transwersale dokładne D_1 oraz D_2.

(a) $D_1 = \{S_1, S_4\}$

(b) $D_2 = \{S_2, S_3\}$

Zgodnie z algorytmem, wyznaczana jest transwersala dokładna (powyżej dwie transwersale zostały przedstawione w celu zwiększenia czytelności przykładu), tak więc rozwiązaniem jest $D_1 = \{S_1, S_4\}$, a więc dwie podsieci automatowe.

(a) $S_1 = \{P_2, P_3, P_6, P_8\}$

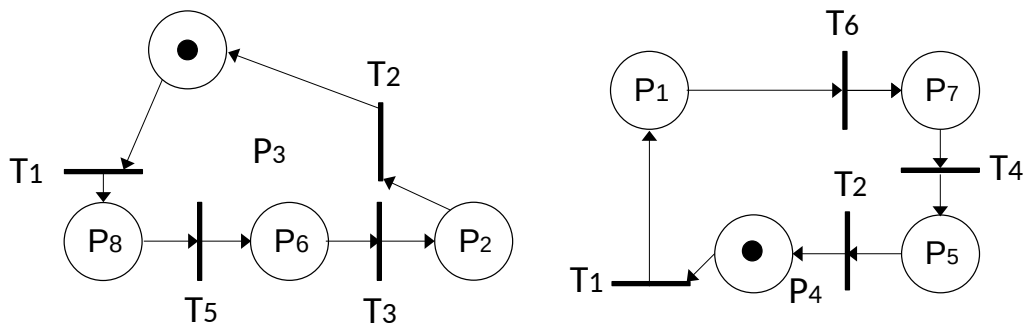
(b) $S_4 = \{P_1, P_4, P_5, P_7\}$

Zdekomponowana sieć została przedstawiona na rysunku 4.4.

4.2 Metody selekcji podsieci automatowych oparte o hipergraf transwersal dokładnych

Niniejszy podrozdział przedstawia dwa nowe algorytmy selekcji bazujące na hipergrafie transwersal dokładnych.

Przedstawiona metoda transwersal dokładnych wykorzystuje xt-hipergraf [74, 98–106], a wynikiem działania algorytmu 1 jest minimalne pokrycie programu systemu sterowania,



Rysunek 4.4: Rysunek przedstawiający sieć P_{bridge} zdekomponowaną z użyciem metody p-niezmienników

który został opisany siecią Petriego. Zaproponowana metoda selekcji polega na wyznaczeniu transwersali dokładnej. Z uwagi na specyficzną klasę hipergrafów, operacja ta jest realizowana w czasie wielomianowym.

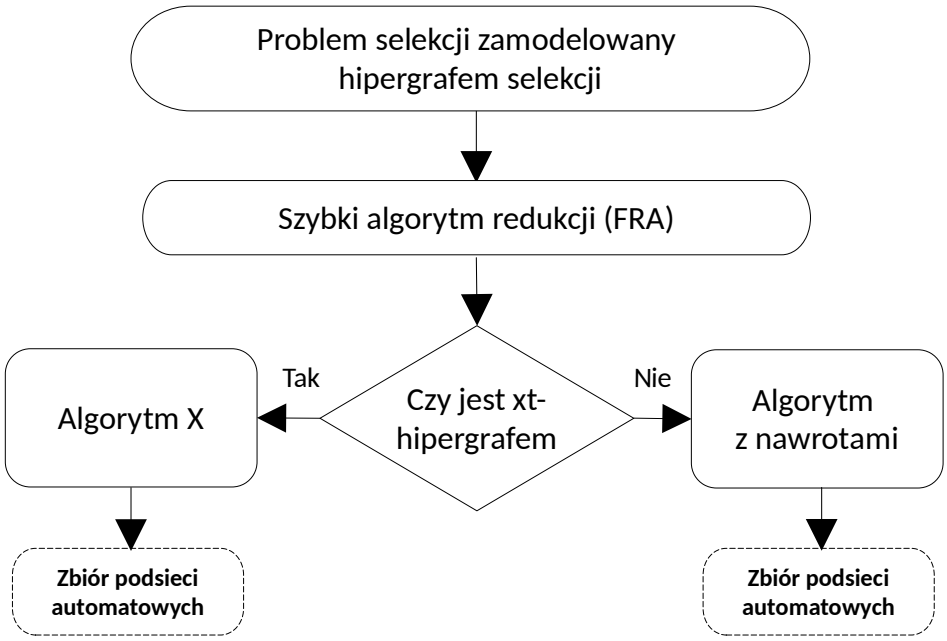
4.2.1 Sformułowanie problemu

Selekcja podsieci automatowych jest jednym z kroków procesu dekompozycji. Pozwala na uzyskanie mniejszego zbioru elementów, ostatecznie wpływając na uzyskanie rozwiązania.

W przypadku algorytmu opartego o hipergrafy [22] Autorka jako bazę hipergrafu selekcji wykorzystuje uprzednio zdefiniowany hipergraf c-dokładny, w którym każda para wierzchołków zgodnych (nie połączonych krawędzią) wchodzi w skład przynajmniej jednej transwersali dokładnej. Pomimo niewątpliwych zalet takiej struktury, nie został przedstawiony wielomianowy algorytm weryfikacji, czy hipergraf jest c-dokładny. Skutkuje to niemożliwością stwierdzenia, czy będzie możliwe wyznaczenie transwersali dokładnej, ponieważ nie każdy hipergraf ją posiada. W związku z tym w przedstawionym algorytmie dekompozycji rozwiązanie dokładne nie zawsze będzie możliwe do uzyskania. Z uwagi na ten fakt autor rozprawy proponuje ulepszenie metody.

4.2.2 Idea proponowanej metody 1

Proponowany algorytm dotyczy selekcji, która jest jednym z kroków dekompozycji opisanej w rozdziale 4.1. Wstępna koncepcja metody została opisana w [98] oraz [102].
Dane wejściowe: sieć Petriego przedstawiona za pomocą macierzy incydencji hipergrafu sekwencyjności.
Dane wyjściowe: wyselekcjonowane składowe automatowe.



Rysunek 4.5: Rysunek przedstawiający ideę selekcji za pomocą metody transwersal dokładnych (1)

(a) *Redukcja*. Redukcja hipergrafu sekwencyjności z wykorzystaniem szybkiego algorytmu redukcji (FRA) opisanego w sekcji 3.5.

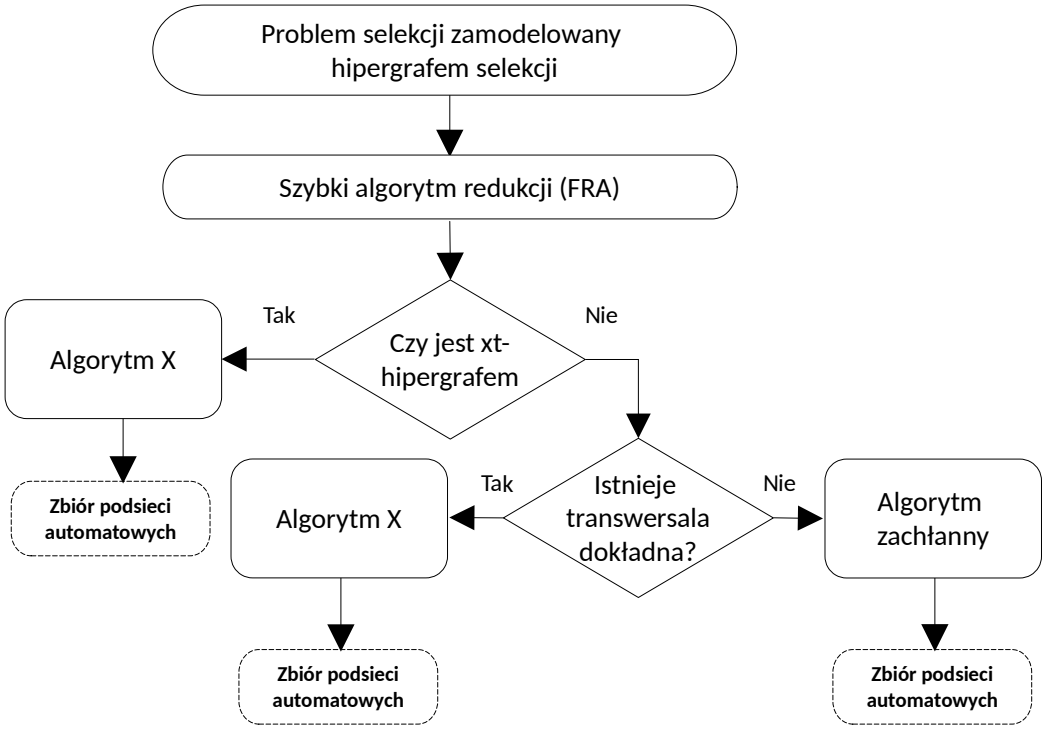
(b) *Klasyfikacja uzyskanego hipergrafu.* Krok ten polega na sprawdzeniu, czy hipergraf należy do klasy xt. W pracy [80] przedstawiony został algorytm xtrec realizujący to zadanie o wielomianowym czasie wykonywania.

(c) i. *Uzyskany hipergraf należy do klasy xt*. Jeśli hipergraf należy do klasy xt, wyznaczenie pierwszej transwersali dokładnej. Transwersala dokładna jest szukanym rozwiązaniem minimalnym zadanego problemu selekcji.

ii. *Uzyskany hipergraf nie należy do klasy xt*. Jeśli hipergraf nie należy do klasy xt, wyznaczenie rozwiązania metodą dokładną z nawrotami.

Opracowana metoda charakteryzuje się wielomianową złożonością obliczeniową w przypadku, kiedy hipergraf należy do klasy xt. Wynika to z następujących faktów:

- Redukcja (FRA) wykonywana jest w czasie wielomianowym [88],
- Sprawdzenie klasy xt hipergrafu odbywa się w czasie wielomianowym [80],
- Wyznaczenie pierwszej transwersali dokładnej w hipergrafie klasy xt wykonywane jest w czasie wielomianowym [80],



Rysunek 4.6: Rysunek przedstawiający ideę selekcji za pomocą metody transwersal dokładnych (2)

4.2.3 Idea proponowanej metody 2

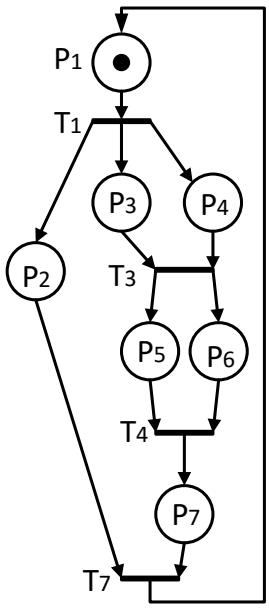
Proponowany algorytm dotyczy selekcji, która jest jednym z kroków dekompozycji opisanej w rozdziale 4.1. Wstępna koncepcja metody została opisana w [98] oraz [102].
Dane wejściowe: sieć Petriego przedstawiona za pomocą macierzy incydencji hipergrafu sekwencyjności.
Dane wyjściowe: wyselekcjonowane składowe automatowe.

(a) *Redukcja*. Redukcja hipergrafu sekwencyjności z wykorzystaniem szybkiego algorytmu redukcji (FRA) opisanego w sekcji 3.5.

(b) *Klasyfikacja uzyskanego hipergrafu.* Krok ten polega na sprawdzeniu, czy hipergraf należy do klasy xt. W pracy [80] przedstawiony został algorytm xtrec realizujący to zadanie o wielomianowym czasie wykonywania.

(c) i. *Uzyskany hipergraf należy do klasy xt*. Jeśli hipergraf należy do klasy xt, wyznaczenie pierwszej transwersali dokładnej. Transwersala dokładna jest szukanym rozwiązaniem minimalnym zadanego problemu selekcji.

ii. *Uzyskany hipergraf nie należy do klasy xt*. Jeśli hipergraf nie należy do klasy xt, próba wyznaczenia transwersali dokładnej. Jeśli istnieje, jest szukanym rozwiązaniem. Jeśli nie istnieje, wyznaczenie rozwiązania metodą zachłanną.



Rysunek 4.7: Rysunek przedstawiający przykładową sieć P_{xt1}

4.2.4 Przykład metody - hipergraf należy do klasy xt

Metoda zostanie zaprezentowana na przykładzie sieci zawartej w [22], P_{xt1} opisującej system sterowania. Rysunek 4.7 przedstawia P_{xt1}.

Znakowanie początkowe jest następujące:

$$M_0 = [1000000] \tag{4.1}$$

Macierz incydencji sieci jest następująca:

$$\mathbf{C}_{xt1}= \begin{array}{c} \begin{matrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 \end{matrix} \\ \begin{bmatrix} -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \end{array} \begin{matrix} T_1 \\ T_3 \\ T_4 \\ T_7 \end{matrix}$$

Przykładowa sieć Petriego posiada pięć podsieci automatowych:

(a) $PN'_1 = \{P_1, P_2\}$

(b) $PN'_2 = \{P_1, P_3, P_5, P_7\}$

(c) $PN'_3 = \{P_1, P_3, P_6, P_7\}$

(d) $PN'_4 = \{P_1, P_4, P_5, P_7\}$

(e) $PN'_5 = \{P_1, P_4, P_6, P_7\}$

Hipergraf sekwencyjności H_{SEQ} jest następujący:

$$\mathrm{H}_{SEQ}= \begin{array}{c} \begin{matrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 \end{matrix} \\ \left[\begin{matrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{matrix}\right] \end{array} \begin{matrix} PN'_1 \\ PN'_2 \\ PN'_3 \\ PN'_4 \\ PN'_5 \end{matrix}$$

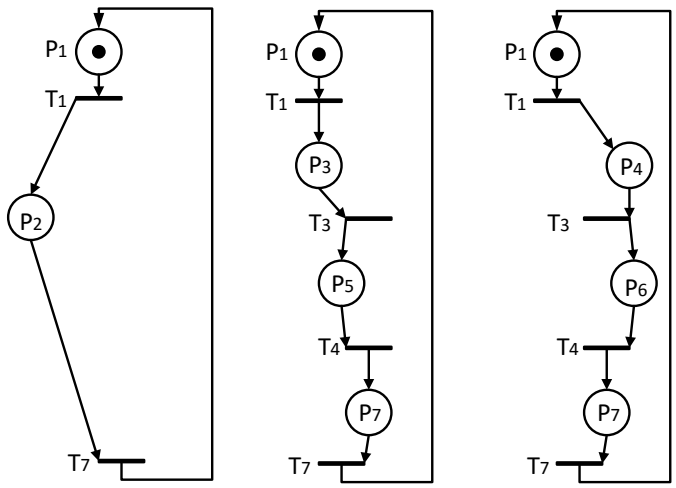
Kolejny krok polega na wyznaczeniu hipergrafu selekcji H_S. W związku z tym wyznaczany jest hipergraf dualny do hipergrafu H_{SEQ}. Operacja ta polega na wykonaniu transpozycji. Następnie po wykonaniu redukcji otrzymywany jest finalny hipergraf selekcji H_S.

$$\mathrm{H}_S= \begin{array}{c} \begin{matrix} PN'_1 & PN'_2 & PN'_3 & PN'_4 & PN'_5 \end{matrix} \\ \left[\begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{matrix}\right] \end{array} \begin{matrix} P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{matrix}$$

Następnie wykonywany jest test polegający na sprawdzeniu klasy hipergrafu. Hipergraf selekcji H_S należy do klasy xt i posiada dwie transwersale dokładne:

(a) $D_1 = \{PN'_1, PN'_2, PN'_5\}$

(b) $D_2 = \{PN'_1, PN'_3, PN'_4\}$



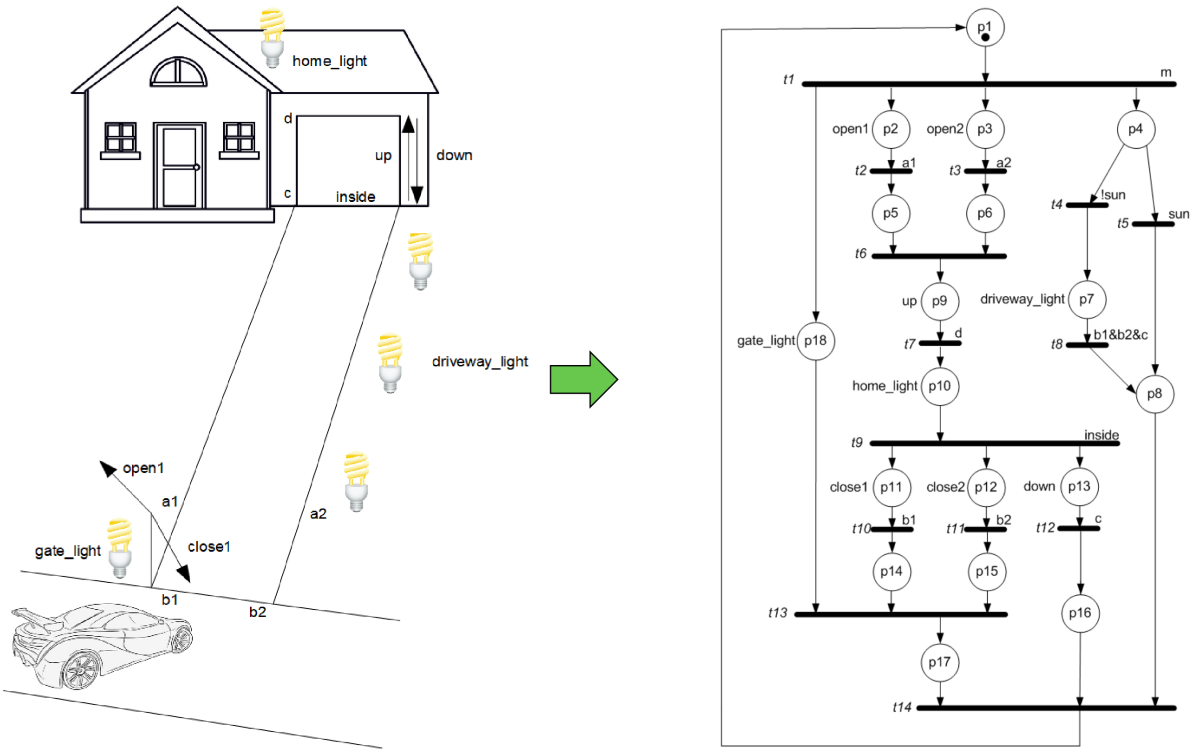
Rysunek 4.8: Rysunek przedstawia sieć P_{xt1} zdekomponowaną na trzy podsieci automatowe

Każda transwersala dokładna pozwala na uzyskanie tak samo dobrego rozwiązania, dlatego algorytm zakłada wyznaczenie dowolnej, tzw. "pierwsza". Finalne rozwiązanie w postaci zdekomponowanej sieci zostało przedstawione na rysunku 4.8.

Wąskim gardłem algorytmu jest przypadek, gdy hipergraf nie należy do klasy xt. W tej sytuacji transwersala wyznaczana jest metodą z nawrotami, która z definicji może być wykładnicza.

4.3 Przykład zastosowania opracowanych metod

Ostatnimi czasy inteligentne domy zyskują coraz to większą popularność. Wynika to z wielu rzeczy, m.in. możliwości automatyzacji pewnych mechanizmów gospodarstwa domowego. Rozważmy system sterowania przedstawiony na rysunku 4.9. Rysunek zawiera także odpowiadającą systemowi interpretowaną sieć Petriego $P_{smart-house}$ System ten wspiera kilka aspektów sterowania. Naciśnięcie przycisku na pilocie (t1) skutkuje załączeniem silnika otwarcia bramy (p2 i p3). W tym samym momencie uruchamiany jest przekaźnik załączenia światła bramy (p18), a także światła na podjeździe (p7, w zależności od pory dnia). Jeśli brama jest już otwarta (t6), brama garażowa rozpoczyna otwieranie (p9). W sytuacji, gdy brama garażowa jest otwarta (t7), światła wewnętrzne zostają zapalone. Przyjmuje się, ze światło garazowe zapalane jest zawsze, z uwagi na zaciemnione pomieszczenie obraz brak przeszkleń w garażu. Może zostać wyłączone przez mieszkańca, w sytuacji np. opuszczenia pomieszczenia. Następnie system czeka na zakończenie wjazdu do garażu przez samochód (t9).



Rysunek 4.9: Rysunek przedstawiający system obsługi inteligentnego domu wraz z odpowiadającą siecią $P_{smart-house}$

W dalszej kolejności brama garażowa (p13) i brama wjazdowa (p11, p12) są zamykane, a światła zewnętrzne wyłączane (p8, p16, p17). Ostatecznie proces został zakończony i może rozpocząć się ponownie (p1).

Omawiana interpretowana sieć Petriego jest żywa i bezpieczna, zawiera 18 miejsc. Proces rozpoczyna się po odpaleniu tranzycji t1, następnie cztery miejsca zawierać będą token (realizowane równolegle): p2, p3, p4 oraz p18. Proces oznaczony miejscem p18 dotyczy zapalenia się światła bramy, p4 jest czujnikiem zmierzchu i w razie potrzeby uruchamia światło podjazdu, natomiast p2 i p3 dotyczy otwierania się dwóch skrzydeł bramy wjazdowej. Po zakończeniu operacji open1 (przy aktywnym sygnale a1), jedno oskrzydło bramy jest już otwarte, co umożliwia odpalenie tranzycji t2. Następnie token jest przekazywany z p2 do p5, co powoduje otwarcie drugiego ramienia bramy. W momencie, gdy skrzydła bramy wjazdowej są otwarte, następuje otwarcie bramy garażowej, a następnie zapalenie światła wewnątrz. System czeka, aż samochód wjedzie do garażu, po czym równolegle wykonywane są trzy akcje: zamknięcie bramy garażowej, zamknięcie lewego i prawego skrzydła bramy wjazdowej. W tym momencie sterownik wraca do stanu początkowego i jest gotowy do dalszej pracy. Nalezy zwrócić uwagę, że zaprezentowany system jest odpowiedzialny za wjazd do garazu. Nic nie stoi na przeszkodzie, aby całość rozbudować o proces odwrotny, związany z wyjazdem. Etapem początkowym w tej sytuacji będzie stan, w którym auto znajduje się w garażu i następnie wykonywane będą kroki kolejne (otwarcie bramy, zapalenie światła na zewnątrz itp.).

$$\mathbf{C}_{smart-house}=$$

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}	
-1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	T_1
0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	T_2
0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	T_3
0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	T_4
0	0	0	-1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	T_5
0	0	0	0	-1	-1	0	0	1	0	0	0	0	0	0	0	0	0	T_6
0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	T_7
0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	T_8
0	0	0	0	0	0	0	0	0	-1	1	1	1	0	0	0	0	0	T_9
0	0	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	T_{10}
0	0	0	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	T_{11}
0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	T_{12}
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	0	1	-1	T_{13}
1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	-1	-1	0	T_{14}

Znakowanie początkowe jest następujące: $M_0 = [100000000000000000]$.

Zgodnie z zaproponowanym algorytmem krok pierwszy polega na wyznaczeniu niezmienników miejsc na podstawie wejściowej sieci Petriego za pomocą usprawnionego algorytmu Martineza-Silvy. Finalnie zostają wyznaczone następujące inwarianty:

(a) $I_1 = [100100110000000000]$

(b) $I_2 = [110010001100100100]$

(c) $I_3 = [101001001100100100]$

(d) $I_4 = [100000000000000011]$

(e) $I_5 = [110010001110010010]$

(f) $I_6 = [101001001110010010]$

(g) $I_7 = [110010001101001010]$

Metoda klasyczna Martineza-Silvy wyznacza z kolei następujące inwarianty:

(a) $I_1 = [100100110000000000]$

(b) $I_2 = [110010001100100100]$

(c) $I_3 = [101001001100100100]$

(d) $I_4 = [100000000000000011]$

(e) $I_5 = [110010001110010010]$

(f) $I_6 = [101001001110010010]$

(g) $I_7 = [110010001101001010]$

(h) $I_8 = [101001001101001010]$

Krok kolejny polega na uzyskaniu podsieci automatych, poprzez wybranie poprawnych inwariantów. Następnie przygotowywana jest macierz selekcji, która zawiera wszystkie wyznaczone podsieci automatowe. Jest ona traktowana jak hipergraf oraz poddawana redukcji. Zredukowana macierz H_{Smin} jest przedstawiona poniżej. Wiersze hipergrafu selekcji odpowiadają miejscom, natomiast kolumny podsieciom automatowym.

$$\mathrm{H}_{S_{min}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Dla klasycznej metody macierz ta wygląda następująco:

$$\mathrm{H}_{S_{max}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Zredukowany hipergraf selekcji H_{Smin} należy do klasy xt i posiada transwersalę dokładną D_1:

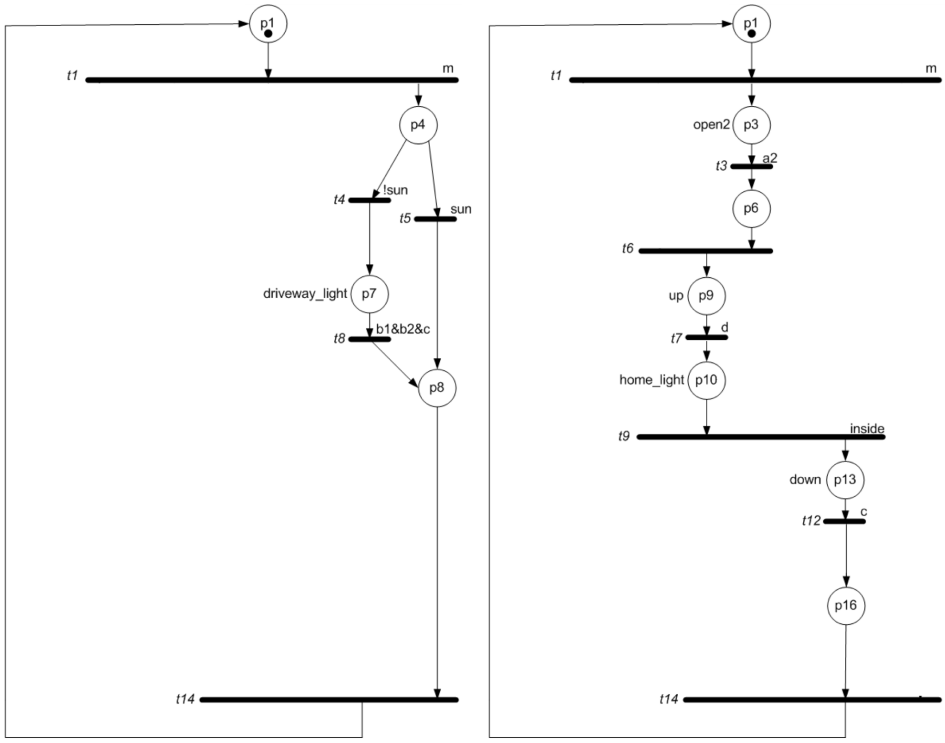
(a) $D_1 = \{S_1, S_3, S_4, S_6, S_7\}$

Zgodnie z algorytmem, wyznaczana jest transwersala dokładna, która zawiera 5 podsieci automatowych

(a) $S_1 = \{P_1, P_4, P_7, P_8\}$

(b) $S_3 = \{P_1, P_3, P_6, P_9, P_{10}, P_{13}, P_{16}\}$

(c) $S_4 = \{P_1, P_{17}, P_{18}\}$



(a) (b)

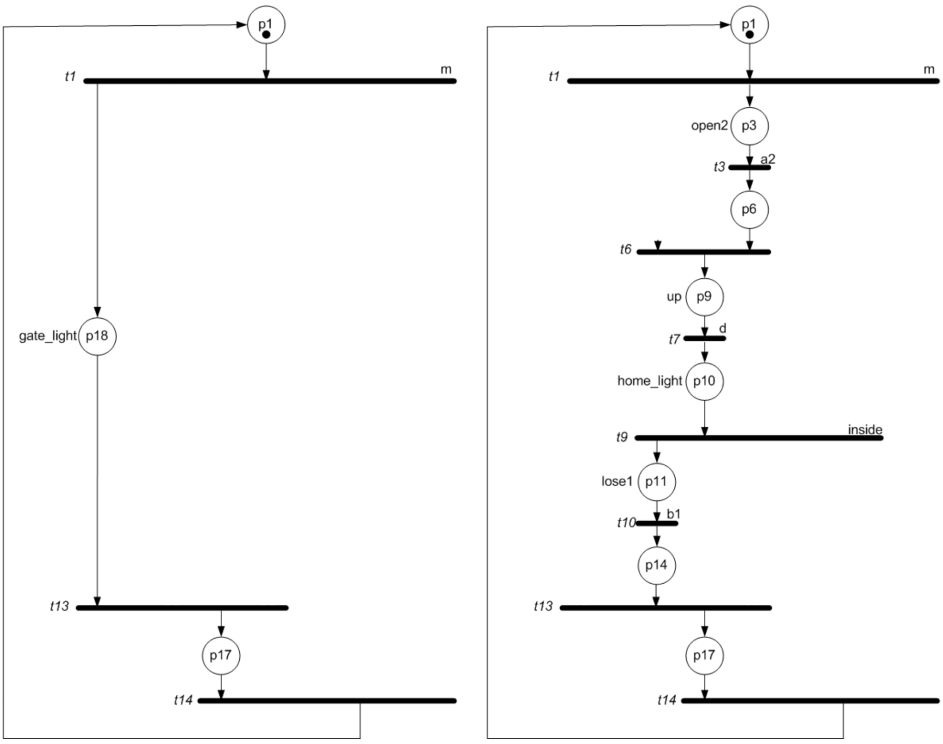
Rysunek 4.10: Podsieć 1 i 3 sieci $P_{smart-home}$

(d) $S_6 = \{P_1, P_3, P_6, P_9, P_{10}, P_{11}, P_{14}, P_{17}\}$

(e) $S_7 = \{P_1, P_2, P_5, P_9, P_{10}, P_{12}, P_{15}, P_{17}\}$

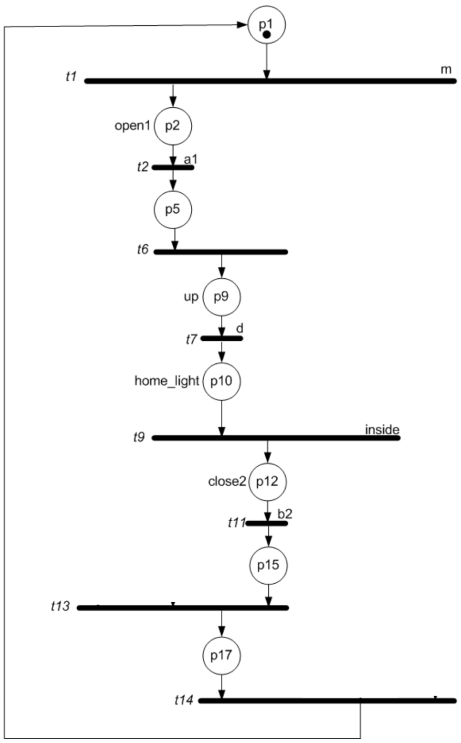
Zdekomponowana sieć została przedstawiona na rysunkach 4.10, 4.11, 4.12.

W wyniku dekompozycji otrzymano 5 podsieci, które mogą zostać zaimplementowane niezależnie np. z zastosowaniem mikroprocesorów (Arduino). Warto podkreślić, że w przypadku dekompozycji automatycznej w ogólnym przypadku nie zawsze można określić dokładne czynności realizowane przez daną wyznaczoną składową. Dla bieżącego przykładu jest to jednak możliwe. Podsieć S1 odpowiada za zapalenie światła przy wjeździe w przypadku, gdy załączy się czujnik zmierzchu. Podsieć S3 z kolei nadzoruje otwarcie skrzydła (a2), otwarcie bramy i zapalenie światła wewnątrz, a gdy pojad jest już zaparkowany, zamknięcie bramy garażowej. Podsieć S4 z kolei zapala światło na bramie. Ostatnie podsieci S6 i S7 uzupełniają S3, jednak finalnie zamykają oba skrzydła bramy wjazdowej.



(a) (b)

Rysunek 4.11: Podsieć 4, 6 sieci $P_{smart-home}$



Rysunek 4.12: Podsieć 7 sieci $P_{smart-home}$

Rozdział 5

Eksperymentalna weryfikacja zaproponowanych metod

Niniejszy rozdział zawiera szczegółowy opis przeprowadzonych badań, pokrótce omówiono rolę biblioteki benchmarków. Przedstawione zostały także główne badania dotyczące selekcji podsieci automatowych, w szczególności: metodologia badań, uzyskane rezultaty, a także ich analiza. Całość została podsumowana w ostatniej sekcji rozdziału.

5.1 Biblioteka modułów testowych

Biblioteka modułów testowych została opracowana w ramach systemu Hippo przez dr inż. M. Wiśniewską i dr hab. inż. R. Wiśniewskiego, prof. UZ pod kierownictwem prof. M. Adamskiego. Autor niniejszej rozprawy miał swój wkład w rozwój Hippo. Aktualnie zawiera łącznie ponad 240 benchmarków. Są one dostępne pod adresem *hippo.uz.zgora.pl*. Na potrzeby wygodnego przeglądania sieci, a także wykonywania podstawowych operacji utworzony został internetowy system Hippo. Jest on przydatnym narzędziem w analizie oraz dekompozycji współbieżnych systemów sterowania opisanych sieciami Petriego za pomocą teorii hipergrafów. System rozwijany jest również m.in. przez autora rozprawy. Benchmarkami są sieci Petriego, z których znaczna część modeluje systemy sterowania. Sieci pochodzą z trzech źródeł:

(a) Część została zaczerpnięta z literatury,

(b) Część opisuje realne systemy sterowania oraz programy sterowników,

(c) Część stanowią hipotetyczne systemy.

5.1.1 Format PNH

Z uwagi na strukturę sieci Petriego wymagany był odpowiedni format pozwalający na opis sieci. Na potrzeby biblioteki testowej stosowany jest format PNH. Pliki zbudowane są następująco:

- Pierwszy wiersz pliku zawiera liczbę kolumn *p*.
- Drugi wiersz pliku zawiera informację o liczbie wierszy macierzy wejściowej łącznie ze znakowaniem początkowym.
- W dalszej kolejności jest zawartych *q* wierszy *p* elementowych, a więc sieć reprezentowana jest przez macierz o wymiarach *q* x *p*. Każdy wiersz opisuje jedną tranzycję, natomiast każda kolumna odnosi się do miejsca sieci Petriego. Miejsce wejściowe tranzycji oznaczane jest poprzez -1, natomiast wyjściowe poprzez 1. Miejsca nie będące w relacji z tranzycją są oznaczone jako *0*.
- Wiersz kolejny zawiera wektor znakowania początkowego.
- Wiersze następne zawierają informacje opcjonalne na temat sieci: nazwa benchmarka, autor, projektant, typ benchmarka, data utworzenia, link do źródła, klasa sieci.

5.1.2 Typy benchmarków

Biblioteka benchmarków testowych zawiera sieci, które należą do różnych klas. Warto podkreślić, że znaleźć można sieci żywe, bezpieczne, ograniczone. Część z nich reprezentują rzeczywiste systemy sterowania, część są to natomiast systemy hipotetyczne.

Ponadto różne jest przeznaczenie poszczególnych benchmarków. Wyróżnić można następujące typy:

- dekompozycja / selekcja,
- relacje współbieżności,
- relacje sekwencyjności.

Za przeznaczenie należy rozumieć łatwość wykonania określonych operacji.

Autorzy dołożyli starań, aby była ona zróżnicowana i zawierała benchmarki różnego typu przydatne przy przeprowadzaniu badań eksperymentalnych. Aby zwiększyć miarodajność analiz oraz eksperymentów, autor postanowił uzupełnić bibliotekę o nowe sieci.

5.2 Biblioteka Hippo

Biblioteka Hippo pozwala na wykorzystanie algorytmów i metod grafowych, hipergrafowych, a także technik bazujących na algebrze liniowej w kontekście dekompozycji oraz analizy współbieżnych systemów sterownaia. Zawiera metody wykorzystujące teorię hipergrafów. Autor rozprawy uzupełnił bibliotekę o klasy związane z selekcją oraz wykorzystaniem xt-hipergrafów. Dodatkowo wprowadził modyfikacje do metody Martineza-Silvy. Biblioteka szczegółowo jest opisana w dodatku A.

W dalszej części rozprawy omówiono pokrótce poszczególne autorskie elementy.

5.2.1 Klasa xtrec

Klasa ta implementuje algorytm xtrec pozwalający na klasyfikację hipergrafów transwersal dokładnych. Dla wskazanego hipergrafu w formie macierzowej wykonywane jest sprawdzenie oraz zwracany rezultat w postaci prawda lub fałsz. Algorytm jest wykonywany w wielomianowym czasie.

5.2.2 Konwerter XML do PNH

Konwerter pozwala na konwersję do formatu PNH zaprojektowanej w popularnych narzęciach sieci zapisanej do pliku XML. PNH jest formatem opracowanym na potrzeby biblioteki Hippo, z uwagi na co żaden edytor nie powiadał takiej możliwości. Jest to niezwykle istotne dla zewnętrzych modeli, które można w prosty sposób przekonwertować i przeprowadzić szereg analiz, sprawdzając interesujące kwestie.

5.2.3 Zmodyfikowana metoda Martineza-Silvy

Zaimplementowana została zmodyfikowana metoda Martineza-Silvy, zgodnie z usprawieniami wskazanymi w rozdziale 4.1.

5.3 System internetowy Hippo

Uzupełnieniem biblioteki hippo.h jest system internetowy Hippo dostępny pod adresem *hippo.uz.zgora.pl*. System ten ma za zadanie dostarczać podstawowych funkcjonalności pozwalających na analizę współbieżnych systemów sterowania opisanych sieciami

Petriego. Bardzo istotną funkcjonalnością jest możliwość szczegółowego przeglądania biblioteki benchmarków, czyli sieci Petriego wykorzystywanych w badaniach eksperymentalnych. Zaimplementowane metody dostarczają następujących funkcjonalności:

- wyznaczenie klasy sieci,
- podstawowa dekompozycja sieci,
- analiza p-niezmienników,
- wyznaczanie relacji współbieżności,
- wyznaczanie relacji sekwencyjności.

System ponadto pozwala na podgląd macierzy w formacie PNH oraz eksport wyrysowanej sieci do formatu graficznego PNG. Hippo udostępnia także mechanizmy pozwalające na analizę sieci za pomocą aparatu hipergrafowego. Funkcjonalność ta jest użytkowana przez zespół badaczy z Portugalii (Universidade Nova de Lisboa) - system GRES. Autor rozprawy aktywnie uczestniczył w tej kooperacji międzynarodowej.

System internetowy Hippo pozwala na swodobne przeglądanie testowych sieci Petriego. Zgromadzone w jednym miejscu benchmarki są wykorzystywane przez zespół badaczy z ośrodka zielonogórskiego do badań eksperymentalnych niezbędnych w pracach bieżących. Autor rozprawy wykorzystuje je celem weryfikacji eksperymentalnej opracowanych metod selekcji, a także dekompozycji. Przeglądarka w pierwszej chwili umożliwia podgląd struktury danej sieci oraz jej nazwę. Po wskazaniu danej sieci wyświetlane są informacje szczegółowe, a mianowicie:

- macierz incydencji,
- oznaczenie miejsc,
- oznaczenie tranzycji,
- oznaczenie znakowania początkowego,
- autor, projektant, źródło,
- klasa sieci Petriego.

Dana sieć wyświetlana jest także w większej rozdzielczości, aby móc bez problemu zlokalizować jej poszczególne miejsca i tranzycje.

Klasyfikacja sieci odbywa się zgodnie z definicjami poszczególnych klas i daje jednoznaczny wynik. Realizowana jest bardzo szybko za pomocą liniowego algorytmu. Docelowo wszystkie sieci w systemie internetowym Hippo posiadają w szczegółach pliku PNH dodaną informację na temat klasy.

W systemie została zaimplementowana także dekompozycja z użyciem grafów oraz kolorowania metodą LF. Złożoność obliczeniowa w najgorszym przypadku jest wykładnicza, dlatego dla dużych sieci rozwiązanie może nie zostać wyznaczone w akceptowalnym czasie.

Dostępna jest także analiza sieci za pomocą niezmienników miejsc. Metoda ta wykorzystuje ich wyznaczanie za pomocą algorytmu Martineza-Silvy, który w najgorszym przypadku jest wykładniczy.

System pozwala na wyznaczenie hipergrafu współbieżności. Z uwagi na fakt, iż liczba transwersal może wzrastać wykładniczo względem miejsc, algorytm w najgorszym przypadku ma złożoność wykładniczą.

Weryfikacja eksperymentalna metod zrealizowana została w oparciu o system Hippo, przedstawiony w [22]. W ramach bieżącej pracy biblioteka ta została rozwinięta o klasy związane z użyciem hipergrafów xt, a także kompletnym procesem selekcji. Przegląd rozwiązań na świecie wskazuje, iż nie istnieją stosowne biblioteki pozwalające na wykorzystanie hipergrafów pod kątem dekompozycji współbieżnych systemów sterowania opisanych sieciami Petriego a w szczególności selekcji, z uwagi na co autor zdecydował się kontynuować rozwój biblioteki Hippo. Dzięki stosownym dodatkom w postaci kolejnych klas (szczegółowy opis w dodatku A) pozwala ona rozwiązać poszczególne problemy z wykorzystaniem algorytmów alternatywnych. System Hippo znajduje praktyczne zastosowanie na Uniwersytecie Zielonogórskim oraz Universidade Nova de Lisboa (Portugalia) w analizie oraz dekompozycji systemów opisanych sieciami Petriego i jest wykorzystywany do realizacji większych projektów, które były pokazywane podczas Festiwali Nauki w Zielonej Górze, czy też dni otwartych Uczelni. Moduły biblioteki Hippo były podstawą przeprowadzonych eksperymentów. Sprawność zaproponowanych metod została zweryfikowana poprzez porównanie czasów wykonywania, natomiast skuteczność jako poprawność uzyskiwanych rezultatów. Punktem odniesienia był algorytm dokładny z nawrotami (ang. backtracking).

5.4 Badania eksperymentalne

Badania eksperymentalne stanowią ważny aspekt związany z potwierdzeniem głównych tez pracy. Zweryfikowano sprawność i skuteczność opracowanych metod opartych o hipergrafy i niezmienniki miejsc. Algorytmy te zostały zaprezentowane szczegółowo w rozdziale 4. Do badań wykorzystano sieci Petriego biblioteki testowej Hippo. Konfiguracja testowa złożona była z maszyny opartej o procesor Intel Core i5-6600 3.3

GHz, 8 GB RAM, dysk SSD 250 GB, działającej pod kontrolą systemu Windows 10 Pro 22H2.

5.4.1 Dekompozycja z wykorzystaniem niezmienników miejsc oraz hipergrafów

Tabela 5.1 przedstawia wybrane rezultaty dla metod opisanych w rozdziale 4.

	klasyczna	Metoda proponowana 1	Metoda proponowana 2
Sieć (miejsca / tranzycje)	t [ms] (SMCs)	t [ms] (SMCs)	t [ms] (SMCs)
cn crr7 (56/16)	-	-	23.19 (28)
cn crr10 (80/22)	-	-	45.77 (40)
cn crr15 (120/32)	-	-	124.84 (60)
cn crr25 (200/52)	-	-	750.11 (100)
crossroad SM (32/13)	$1.74*10^6(10)$	$1.74*10^6(10)$	18.19 (10)
esparza2 (15/13)	25.33 (4)	20.19 (4)	0.01 (4)
girault7 (11/7)	6.72 (4)	6.81 (4)	0.36 (4)
hard case (9/3)	333.75 (4)	0.44 (4)	0.01 (6)
inv exp 3 3 (9/3)	$1.21*10^6(3)$	0.67 (3)	0.21 (3)
zuberek3 (30/21)	63 453.31 (5)	3.15 (5)	0.01 (5)

Tablica 5.1: Wybrane rezultaty dekompozycji uzyskane za pomocą inwariantów miejsc

Pierwsza kolumna tabeli określa sieć testową z informacją dotyczącą liczby miejsc i tranzycji. Kolejna kolumna dotyczy wyników dla metody klasycznej - czas wykonywania oraz liczba podsieci automatowych. Następne dwie kolumny przedstawiają wyniki odpowiednio dla metody proponowanej 1 i metody proponowanej 2. Jak pokazują badania, metoda proponowana 2 pozwala na uzyskanie rezultatu w czasie krótszym od metody klasycznej, przy czym pozwala na uzyskanie rozwiązania tak samo dobrego. Przyjęto próg 30 minut, po których obliczenia były przerywane. Dla 8 sieci metoda klasyczna nie pozwala na znalezienie rozwiązania w przyjętym czasie, natomiast dla sieci 7 znajdywała rozwiązanie w czasie, który należy uznać za dalece odstający od metody proponowanej 2. Proponowana metoda 1 jest szybsza od metody

klasycznej o 50%, natomiast metoda 2 o 90%. Skuteczność metody 1 można zauważyc zwłaszcza w przypadku testów "hard case " oraz " zuberek3", dla których proponowana metoda znajduje rozwiązanie znacznie szybciej niż algorytm klasyczny. Otrzymane rezultaty pozwalaja stwierdzić dobrą sprawność proponowanej metody oraz przewagę nad rozwiązaniem klasycznym w sytuacji, gdy hipergraf selekcji należy do klasy xt. Należy podkreślić, że istnieją przypadki, kiedy metoda proponowana 1 jest wolniejsza od metody klasycznej. Dzieje się tak w przypadku, kiedy hipergraf selekcji nie należy do klasy xt i w tym podejściu wymagane jest wyznaczenie rozwiązania za pomocą algorytmu z nawrotami, wykorzystywanego klasycznie. Metoda 2 zawsze jest sprawniejsza od algorytmu klasycznego, natomiast nie zawsze równie skuteczna (hard case).

Pełne wyniki przedstawiono w dodatku E. Istotną kwestią jest to, że metoda proponowana 2 wykazuje większą sprawność dla sieci większych, bardziej złożonych, dla których metoda klasyczna zawodzi.

5.4.2 Selekcja podsieci automatowych sieci Petriego

Ważną częścią dekompozycji równoległej systemu sterowania jest selekcja podsieci automatowych. Niniejszy podrozdział przedstawia wyniki badań proponowanych metod. Pierwszym ważnym aspektem są badania dotyczące opracowanej metody selekcji wykorzystującej xt-hipergraf. Drugim aspektem jest uwypuklenie wpływu szybkiego algorytmu redukcji na klasę hipergrafu selekcji. Okazuje się, że redukcja w istotny sposób wpływa na podwyższenie skuteczności metody. Trzecim ważnym aspektem jest fakt, że każda transwersala dokładna daje rozwiązanie tak samo dobre, dlatego też w proponowanych metodach wyznaczana jest zawsze jedna. W dodatku D przedstawiony został dowód matematyczny, potwierdzający tę prawidłowość [107].

Metoda transwersal dokładnych i selekcja z wykorzystaniem xt-hipergrafu

Pierwsza część badań związana była z opracowaną metodą selekcji z wykorzystaniem xt-hipergrafu. Proponowany algorytm transwersal dokładnych (xt) 1 i 2 został porównany z klasyczną metodą z nawrotami.

Głównym celem tego etapu była eksperymentalna weryfikacja skuteczności i sprawności proponowanych metod. Uzyskana liczba podsieci automatowych w przypadku algorytmu z nawrotami oraz transwersal dokładnych xt-hipergrafu powinna być taka sama, ponieważ w założeniu są to metody dokładne i generują rozwiązania minimalne.

	Metoda klasyczna	Metoda proponowana 1	Metoda proponowana 2
Sieć (miejsca / tranzycje)	t [ms] (SMCs)	t [ms] (SMCs)	t [ms] (SMCs)
3carros (12/8)	0.372 (3)	0.015 (3)	0.002 (3)
beverage production (20/16)	0.075 (5)	0.001 (5)	0.001 (5)
crossroad SM (32/13)	7001 (10)	7200 (10)	5.312 (10)
girault7 (11/7)	0.034 (4)	0.003 (4)	0.003 (4)
hard case (9/3)	0.0897 (6)	0.001 (6)	0.001 (6)
hee2 (13/11)	-	0.003 (4)	0.004 (4)
lnet p2n2 (13/7)	0.016 (4)	0.001 (4)	0.002 (4)

Tablica 5.2: Rezultaty badań selekcji dla wybranych benchmarków

W tabeli 5.3 zestawiono wybrane wyniki badań spośród przetestowanych sieci. Pierwsza kolumna prezentuje nazwę sieci oraz liczbę miejsc / tranzycji. Kolejne przedstawia wynik dla algorytmu z nawrotami wyrażony w ms, a także wyselekcjonowaną liczbę podsieci automatowych. Następne dwie dotyczą odpowiednio metody proponowanej 1 i 2. Bazując na pełnych wynikach, xt-selekcja (2) jest średnio o 90% szybsza od algorytmu z nawrotami. Skuteczność proponowanej metody 1 można zauważyć np. w przypadku testu "hee2", dla którego wynik jest obliczany w krótkim czasie, natomiast metoda klasyczna nie jest w ogóle w stanie znaleźć rozwiązania. Istnieją także przypadki, dla których metoda proponowana jest nieco wolniejsza niż algorytm klasyczny (np. " crossroad SM"). Natomiast należy podkreślić, że otrzymane rezultaty pozwalają stwierdzić dobrą sprawność proponowanej metody selekcji oraz przewagę nad rozwiązaniem klasycznym w sytuacji, gdy hipergraf selekcji należy do klasy xt. Selekcja z wykorzystaniem transwersal dokładnych xt-hipergrafu pozwala na szybsze uzyskanie rezultatu niż z wykorzystaniem algorytmu z nawrotami. Badania eksperymentalne pokazują, że skuteczność metody klasycznej i proponowanej 1 jest identyczna, co także potwierdza poprawność proponowanej metody. Szczegółowe informacje zamieszczono w dodatku E.

Wpływ redukcji na klasę hipergrafu selekcji

Proponowany algorytm selekcji uwzględnia wykorzystanie szybkiego algorytmu redukcji, który redukuje wiersze i kolumny macierzy binarnej. Zostało wykonane

badanie polegające na weryfikacji klasy hipergrafu selekcji. Dokonano klasyfikacji hipergrafu selekcji przed wykonaniem redukcji oraz po.

Klasa hipergrafu	Przed redukcją	Po redukcji	Numer metody
xt	49%	80%	1
inna	51%	20%	1
xt	43%	93%	2
inna	57%	7%	2

Tablica 5.3: Rezultaty badań wpływu redukcji na klasę hipergrafu selekcji

Jak pokazują wyniki, po wykonaniu redukcji 93% sieci testowych posiadało hipergraf selekcji, który należał do klasy xt (hipergrafów transwersal dokładnych) w przypadku proponowanej 2 metody dekompozycji. Przed redukcją było to odpowiednio 43% sieci, co sumarycznie daje zysk wynoszący aż 50%. W przypadku proponowanej metody dekompozycji 1 przed redukcją 49% sieci należało do klasy xt, natomiast po redukcji już 80%.

Z uwagi na przynależność zredukowanych hipergrafów do klasy xt możliwa stała się selekcja w wielomianowym czasie wykonywania, ponieważ każda transwersala dokładna w hipergrafie xt może zostać wyznaczona wielomianowo. Jest to rozwiązanie minimalne, ponieważ każda transwersala dokładna hipergrafu xt jest jednocześnie minimalna [80].

Kolejne transwersale dokładne hipergrafu, a skuteczność rozwiązania

Każdy hipergraf może posiadać $n \geq 0$ transwersal dokładnych D. Zgodnie z proponowanym algorytmem selekcji w przypadku, kiedy hipergraf należy do klasy xt, rozwiązaniem jest transwersala dokładna. Interesującym aspektem jest skuteczność rozwiązania oferowana przez różne transwersale dokładne. Sytuacja nie jest tak klarowna w przypadku hipergrafu klasy nie należącej do xt, ponieważ ten może nie posiadać transwersal dokładnych. Z uwagi na ten fakt przeprowadzono badania eksperymentalne mające na celu weryfikację skuteczności rozwiązania uzyskiwanego w oparciu o kolejne transwersale dokładne.

Sieć testowa	min SMCs
bridge semaphore	3
cncrr001	3
img 280	3
lnet p6n1	3
TP5 I	2

Tablica 5.4: Wybrane rezultaty dla badania skuteczności rozwiązania uzyskanego za pomocą kolejnych transwersal dokładnych

Tabela 5.4 prezentuje wybrane rezultaty badań skuteczności rozwiązania uzyskanego za pomocą wyznaczonych kolejnych transwersal dokładnych D. Druga kolumna określa minimalną liczbę podsieci automatowych niezbędnych do pokrycia sieci. Jak pokazują badania, wszystkie uzyskane transwersale dokładne wyznaczały taką samą liczbę podsieci automatowych. Oznacza to, że w rezultacie każda transwersala dokładna umożliwia uzyskanie rozwiązaina tak samo dobrego, dlatego też proponowane algorytmy zakładają wyznaczenie transwersali dokładnej.

Dowody matematyczne przedstawiające istotę powyższych badań eksperymentalnych zostały przedstawione w dodatku D.

5.5 Podsumowanie badań

Badania zostały podzielone na poszczególne bloki, dotyczące danego zagadnienia. Przyjęto 30 minut czasu, po którym obliczenia danego przykładu były przerywane. Pierwszy z wymienionych bloków to dekompozycja z wykorzystaniem niezmienników miejsc oraz xt-selekcji, drugi to xt-selekcja podsieci automatowych sieci Petriego. W ramach xt-selekcji wykonano badania skuteczności metody xt w porównaniu do metody klasycznej wykorzystującej algorytm z nawrotami, wpływu szybkiego algorytmu redukcji na klasę hipergrafu selekcji, jakości rozwiązania problemu selekcji oferowana przez kolejne transwersale dokładne hipergrafu.

Badania proponowanej metody dekompozycji wykorzystującej niezmienniki miejsc oraz xt-selekcję związane były z weryfikacją sprawności oraz skuteczności. Sprawność metody klasycznej jest ograniczona z uwagi na charakter algorytmu. Słowa te znajdują odzwierciedlenie w eksperymentach, ponieważ dla 8 sieci metoda ta nie zwracała wyniku po 30 minutach obliczeń. Kolejne 7 sieci testowych było przeliczanych w czasie sporo dłuższym, niż metoda proponowana 2. W nawiązaniu do wyników badań, proponowana

metoda 1 jest o 50% szybsza od klasycznej. Występują także przypadki, kiedy metoda proponowana 1 jest wolniejsza od klasycznej i są to sytuacje, kiedy hipergraf selekcji nie należy do klasy xt i wymagane jest klasyczne podejście. Metoda proponowana 2 wykazuje bardzo dobrą sprawność i jest aż o 90% szybsza od metody klasycznej. Znajduje szybko rozwiązanie, które w sporej większości przypadków jest najmniejsze. Warto podkreślić, że dla benchmarka CrossroadSM metoda klasyczna potrzebuje ok. 29 minut na przeprowadzenie dekompozycji, natomiast metoda proponowana zwraca rezultat po ok. 18 ms. To, co jest istotne w tym przypadku to fakt, że oba obliczone rozwiązania są najmniejsze.

Xt-selekcja (1) podsieci automatowych sieci Petriego pozwala na uzyskanie rozwiązania o takiej samej jakości, jak w przypadku algorytmu z nawrotami. Metoda selekcji oparta o xt-hipergraf pozwala na uzyskanie rozwiązania w czasie wielomianowym, co jest ogromną zaletą. Wadą rozwiązania jest natomiast to, że nie każdy hipergraf należy do klasy xt, z uwagi na co będzie wymagane użycie klasycznej metody z nawrotami w celu uzyskania rozwiązania dokładnego. Istnieje wielomianowy algorytm weryfikujący przynależność hipergrafu do klasy xt, dlatego też stosunkowo szybko można uzyskać informację, czy możliwe będzie wykonanie selekcji oferującej rozwiązanie dokładne w czasie wielomianowym. Przeprowadzono także badania związane z redukcją hipergrafu selekcji, który nie jest xt, za pomocą szybkiego algorytmu redukcji. Jak się okazuje, istotnie wpływa on na klasę hipergrafu: o 31% więcej testowych sieci w przypadku metody 1 posiadało hipergraf selekcji, który należał do klasy xt i aż o 50% więcej w przypadku metody 2. Efektywność metody proponowanej 1 wzrosła z 49% do 80% testowych benchmarków i odpowiednio z 43% do 93% dla metody 2. Każda transwersala dokładna hipergrafu dla założonych sieci Petriego daje rozwiązanie problemu selekcji o tej samej jakości, co też potwierdzają badania, a także przedstawiony dowód formalny.

Głównym celem opracowanych metod była minimalizacja czasu uzyskania rozwiązania, a także ukazanie innego podejścia względem algorytmów już istniejących, rozwinięcie ich. Xt-selekcja jest rozwinięciem pracy [22], w której wprowadzono hipergraf c-dokładny. Proponowane metody dekompozycji oraz selekcji wskazuje możliwe alternatywne rozwiązanie problemu dekompozycji, w których wykorzystano niezmienniki miejsc, zmodyfikowany algorytm Martineza-Silvy oraz hipergraf transwersal dokładnych. Wprowadzone zmiany zostały opracowane pod kątem uzyskania pokrycia sieci. Rezultaty badań oraz weryfikacja eksperymentalna metod pozwala z optymizmem stwierdzić, iż opracowane metody stanowią dobrą alternatywę dla klasycznych algorytmów.

Rozdział 6

Podsumowanie rozprawy oraz wnioski

Informatyka jest jedną z najbardziej dynamicznie rozwijających się dyscyplin naukowych. Mechanizmy i algorytmy wykorzystywane dotychczas z biegiem czasu mogą okazać się niewystarczające, ponieważ każdego roku projektowane systemy sterowania są coraz większe. Wpływa to oczywiście na cały proces realizacji, ponieważ nie zawsze mogą zostać wykorzystane układy pozwalające na optymalną implementację. Wymagane jest ciągłe udoskonalanie metod oraz narzędzi umożliwiających na projektowanie złożonych systemów. Głównym celem autora rozprawy było zaproponowanie nowych metod dekompozycji współbieżnych systemów sterowania, w tym selekcji podsieci automatowych sieci Petriego, które sprawnie (w akceptowalnym czasie) oraz skutecznie (poprawne wyniki) umożliwią pozyskanie rezultatów. Istotne było także pokazanie innego podejścia dla procesu dekompozycji oraz problemu selekcji. Dekompozycja polega na podzieleniu programu danego systemu sterowania współbieżnego na automaty sekwencyjne, które mogą być realizowane równolegle. Jeżeli sieć zawiera nadmiarowe automaty, może zostać wykonana ich selekcja mająca na celu wybranie tylko tych niezbędnych. Tradycyjne metody oparte są głównie o wykorzystanie grafów nieskierowanych i do jeszcze niedawna były one wystarczające.

W rozprawie zaproponowano cztery metody: dwie dekompozycji za pomocą niezmienników miejsc oraz dwie selekcji podsieci automatowych. W metodzie dekompozycji (1) równoległej systemów sterowania opisanych sieciami Petriego wykorzystano teorię algebry liniowej i niezmienniki miejsc, a także autorski algorytm selekcji. Metoda ta jest nakierunkowana na skuteczność, a więc uzyskanie rozwiązania zwracającego najmniejszą liczbę podsieci automatowych, kosztem czasu obliczeń. Metoda dekompozycji (2) wprowadza istotną modyfikację metody Martineza-Silvy polegająca na generowaniu niepełnego zbioru inwariantów. Ta z kolei zakłada wysoką sprawność, a więc uzyskanie bardzo szybko rozwiązanie, które niekoniecznie musi

zawierać najmniejszą liczbę podsieci automatowych. Ważnym rdzeniem metody jest hipergraf transwersal dokładnych. Klasa xt hipergrafu jest o tyle specyficzna, że każda minimalna transwersala (nie zawiera się w innej) jest jednocześnie dokładna. Selekcja jest istotną częścią procesu dekompozycji, ponieważ pozwala na wybranie tylko tych podsieci automatowych, które są wymagane do pokrycia sieci właściwej.
Pierwsza proponowana metoda związana jest z dekompozycją za pomocą niezmienników miejsc i bazuje na dobrze znanej metodzie Martineza-Silvy. Następnie, po wyznaczeniu podsieci automatowych, dokonywana jest selekcja i wyznaczane właściwe rozwiązanie za pomocą algorytmu selekcji opartego o xt-hipergraf. Bazując na badaniach można jasno stwierdzić, że metoda proponowana 1 pozwala na uzyskanie rezultatu w większości przypadków w czasie o 50% krótszym od klasycznej metody z nawrotami. Druga proponowana metoda zawiera istotne usprawnienie - autor rozprawy zdecydował się na wprowdzenie modyfikacji polegającej na wyznaczaniu niepełnego zbioru podsieci automatowych i przerywaniu algorytmu w sytuacji, kiedy sieć bazowa jest już pokryta. Ponadto algorytm selekcji zorientowany jest na sprawność. Proponowana metoda 2 z kolei zwraca rozwiązanie o 90% szybciej, niż metoda klasyczna. Jak pokazują badania, jakość rozwiązań jest bardzo dobra w odniesieniu do porównywanych metod. Zaproponowany algorytm dekompozycji 2 z uwzględnieniem modyfikacji pozwala na zmniejszenie liczby przetwarzanych podsieci automatowych, co sumarycznie skutkuje obliczeniami na mniejszych macierzach proponowanego algorytmu selekcji.

Oprócz badań eksperymentalnych zaproponowano dowody formalne ukazujące słuszność założonych tez. Biorąc pod uwagę powyższe fakty można stwierdzić, iż proponowane metody przyspieszają znalezienie rozwiązania dokładnego dla procesu dekompozycji i selekcji. Przeprowadzone badania eksperymentalne na zbiorze bibliotek testowych pokazują, że opracowane algorytmy pozwalają na wsparcie i akcelerację wymienionych wcześniej procesów. Z uwagi na fakt, iż klasyczne metody dokładne charakteryzują się czasem wykładniczym, istotne jest uzyskiwanie rozwiązania dokładnego w krótszym czasie.

6.1 Potwierdzenie tezy

Teza rozprawy została potwierdzona eksperymentalnie. W pracy wykazano, że zastosowanie hipergrafów usprawnia proces dekompozycji oraz selekcji. Ponadto zastosowanie niezmienników miejsc w dekompozycji równoległej współbieżnych systemów sterowania pozwala na uzyskanie dobrych rezultatów, a metoda może być z powodzeniem używana obok metod tradycyjnych.

Badania eksperymentalne potwierdziły wysoką sprawność i skuteczność opracowanych metod. Selekcja z wykorzystaniem xt-hipergrafu umożliwia rozwiązanie zadanego problemu szybciej, niż metody klasyczne wykorzystujące algorytm z nawrotami. Średnia liczba podsieci automatowych niezbędnych do pokrycia sieci bazowej w przypadku algorytmu z nawrotami oraz proponowanej metody 1 była taka sama co weryfikuje skuteczność obu metod. Sprawność metody proponowanej natomiast jest dużo wyższa, ponieważ pozwala na uzyskanie rozwiązania w czasie o wiele krótszym. Szczegółowe wyniki zostały przedstawione w dodatku E.
Bardzo istotnym aspektem jest wpływ szybkiego algorytmu redukcji na klasę hipergrafu selekcji. Jak pokazują badania, po wykonaniu redukcji 31% więcej testowych sieci posiadało xt-hipergraf selekcji w przypadku pierwszej proponowanej metody dekompozycji i 50% więcej w przypadku metody 2. Biorąc pod uwagę ten fakt obszar zastosowania został powiększony do wartości sięgającej 93% benchmarków biblioteki testowej. Warto podkreślić, że każda transwersala dokładna hipergrafu umożliwia rozwiązanie problemu selekcji optymalnie, z uwagi na co autor postanowił wyznaczać pierwszą. Ma to ogromne znaczenie, ponieważ w przypadku ogólnym wyznaczanie transwersal dokładnych może się wiązać ze złożonością wykładniczą.

6.2 Elementy nowatorskie zawarte w rozprawie doktorskiej

Do najważniejszych nowatorskich aspektów pracy doktorskiej Autor zalicza:

- opracowanie metody dekompozycji (1) zorientowanej na skuteczność (w oparciu o niezmienniki miejsc oraz autorską metodę selekcji),
- opracowanie metody dekompozycji (2) zorientowanej na sprawność (w oparciu o zmodyfikowaną metodę wyznaczającą niezmienniki miejsc oraz autorską metodę selekcji),
- opracowanie metody selekcji (1) zorientowanej na skuteczność (najlepsze możliwe rozwiązanie kosztem czasu),
- opracowanie metody selekcji (2) zorientowanej na sprawność (rozwiązanie w najgorszym przypadku przybliżone, zawsze znalezione w krótkim czasie,
- opracowanie hipotez badawczych dotyczących teorii hipergrafów w procesie selekcji,
- modyfikacje biblioteki Hippo na potrzeby prac związanych z rozprawą,

- modyfikacje biblioteki benchmarków testowych na potrzeby prac związanych z rozprawą,
- wkład w rozwój systemu internetowego Hippo,
- weryfikacja efektywności opracowanych metod poprzez przeprowadzenie szeregu badań i eksperymentów.

6.3 Ograniczenia zaproponowanych metod oraz kierunki dalszych prac

Opracowane algorytmy posiadają szereg zalet, warto wspomnieć także o ich ograniczeniach. Złożoność obliczeniowa selekcji (1) w najgorszym przypadku może być wykładnicza, jednak jak pokazują badania takie przypadki są niezwykle rzadkie. Z uwagi na fakt oparcia selekcji o hipergraf xt, stosowalność metody w czasie wielomianowym ogranicza się do hipergrafów klasy xt. Wiekszość benchmarków testowych posiada odpowiednii hipergraf, co w rezultacie skutkuje potrzebą wykorzystania wykładniczej metody z nawrotami dla małej grupy przypadków.
Opracowany algorytm selekcji (2) jest skuteczny dla 93% przypadków z biblioteki testowej. Pozostałe albo posiadają transwersalę dokładną, co w gruncie rzeczy pozwala na uzyskanie najmniejszej liczby podsieci automatowych, albo rozwiązanie przybliżone.
Opracowane algorytm dekompozycji (1) oparty o inwarianty także jest metodą, która w najgorszym przypadku jest wykładnicza i nie zawsze zwróci rozwiązanie w akceptowalnym czasie. Z kolei nie wszystkie inwarianty są poprawnymi podsieciami automatowymi i z reguły wymagane jest dodatkowe sprawdzenie.

Opracowane metody mogą stanowić podstawę do prowadzenia dalszych badań. Selekcja oparta o hipergraf transwersal dokładnych (xt) jasno i klarownie pozwala na przeprowadzenie wyboru określonych podsieci w przypadku, kiedy dany hipergraf należy do klasy xt. Ważnym aspektem jest przypadek, kiedy hipergraf nie należy do klasy xt. Stwierdzono na podstawie badań eksperymentalnych, że jeśli posiada transwersalę dokładną, rozwiązanie to jest optymalne. Także każda kolejna transwersala dokładna daje w rezultacie rozwiązanie takie, jak każda inna. Dalszym kierunkiem prac może być weryfikacja struktury sieci oraz wykazanie kwestii braku transwersal dokładnych w przypadku niektórych hipergrafów modelujących dane sieci. Istotne jest to, czy pewne modyfikacje sieci mogą wpłynąć na to, że dany hipergraf będzie należał do klasy xt, czy też przynajmniej posiadał transwersalę dokładną. Warto tutaj zwrócić uwagę na szybki algorytm redukcji, dzięki któremu aż 93% hipergrafów selekcji w przypadku metody

dekompozycji (2) należy do klasy xt. Kolejnym ciekawym zagadnieniem może być próba utworzenia algorytmu pozwalającego na oszacowanie liczby transwersal dokładnych w xt-hipergrafie. Pozwoli to na oszacowanie chociażby liczby podsieci automatowych, badź też liczby możliwych rozwiazań danego problemu (np. selekcji).
Praca została częściowo zrealizowana ze środków grantu dla młodych naukowców i uczestników studiów doktoranckich z budżetu państwa (w latach 2013-2014 oraz 2015-2016).

6.4 Osiągnięcia naukowe uzyskane w trakcie realizacji rozprawy doktorskiej

W ramach prac związanych z realizacją niniejszej rozprawy doktorskiej zredagowano łącznie 35 pozycji literaturowych (m.in. [74, 98–112], w tym:

- 5 pozycji zawartych jako rozdziały w wydawnictwach zwartych,
- 8 pozycji w czasopismach, w tym 2 pozycje notowane na liście filadelfijskiej,
- 23 pozycje konferencyjne, w tym 15 pozycji notowanych w Web of Science.

Indeksy naukowe (wg Web of Science, na dzień 20.04.2023):

- cytowania artykułów: 124,
- indeks Hirscha: 7,
- indeks i10: 6.

Zrealizowane granty dla młodych naukowców oraz uczestników studiów doktoranckich:

- " Opracowanie oraz implementacja metod selekcji składowych automatowych w procesie dekompozycji systemów współbieżnych", 2013-2014, kierownik: prof. dr hab. inż. M. Adamski,
- "Wykorzystanie teorii hipergrafów w dekompozycji układów dyskretnych opisanych sieciami Petriego, a w szczególności opracowaniu i implementacji algorytmów hipergrafowych selekcji", 2015-2016, kierownik: dr hab. inż. R. Wiśniewski, prof UZ.

Dodatek A

Biblioteka Hippo - opis klas oraz możliwości

Biblioteka Hippo została napisana przez *M. i R. Wiśniewskich* w ośrodku zielonogórskim pod kierunkiem *prof. Mariana Adamskiego*. Wykorzystany został język C++ pozwalający na podejście obiektowe. W ramach bieżącej pracy biblioteka została rozwinięta o klasy związane z obsługą hipergrafów xt, a także procesem selekcji. Niniejszy dodatek opisuje poszczególne moduły biblioteki pozwalając na zapoznanie się z jej możliwościami pod kątem analizy i dekompozycji systemów dyskretnych opisanych sieciami Petriego, a w szczególności projektowania systemów logicznych.

A.1 Struktura biblioteki

Biblioteka została podzielona na poszczególne pliki nagłówkowe, które dotyczą danych zagadnień. Tabela A.1 zawiera wypisane wszystkie składowe z krótkim opisem syntetycznym.

Nazwa klasy	Opis syntetyczny
Hippo	Klasa bazowa implementująca typ hipergrafu oraz podstawowe mechanizmy pozwalające na manipulacje
HippoCliques	Klasa potomna, która pozwala na operacje związane z klikami hipergrafów
HippoColors	Klasa potomna, która implementuje algorytmy kolorowania

Nazwa klasy	Opis syntetyczny
HippoComparability	Klasa potomna, która dostarcza mechanizmy pozwalające na operacje związane z grafami porównywalności
HippoComplement	Klasa potomna dostarczająca mechanizmy pozwalające na wyznaczenie dopełnienia oraz operacje związane ze zbiorami niezależnymi
HippoCovers	Klasa potomna, która pozwala na wyznaczenie pokrycia oraz transwersal
*HippoMethods**	Autorska klasa potomna, która pozwala na selekcję z wykorzystaniem: algorytmu z nawrotami, zachłannego oraz transwersal dokładnych
*HippoSelect**	Autorska klasa potomna dostarczająca ogólne mechanizmy selekcji
HippoVectors	Klasa potomna, która pozwala na operacje oraz przekształcenia na wektorach oraz macierzach binarnych
PNets	Klasa bazowa implementująca aspekty związane z sieciami Petriego. Autor wprowadził modyfikacje do algorytmu Martineza-Silvy tworząc nową funkcję zwracającą rezultat
SMCs	Klasa pozwalająca na wyznaczanie podsieci automatowych
*xtrec**	Autorska klasa implementująca algorytm xtrec pozwalający na sprawdzenie, czy dany hipergraf jest xt
dlx	Klasa implementująca algorytm X w postaci metody tańczących powiązań

Tablica A.1: Tabela przedstawiająca klasy biblioteki hippo

Oznaczenie klas jest następujące:

- czcionka zwykła - klasa nie została zmodyfikowana przez autora rozprawy,
- kursywa - klasa zawiera modyfikacje wprowadzone przez autora rozprawy,
- kursywa oraz symbol * - klasa utworzona od podstaw przez autora rozprawy.

Jak można zauważyć, dostępne są kompletne rozwiązania pozwalające na operacje hipergrafowe oraz grafowe na systemach dyskretnych opisanych sieciami Petriego.

Ponadto zaimplementowany został algorytm X poprzez metodę tańczących powiązań (*ang. dancing links*).

A.1.1 Klasy składające się na bibliotekę Hippo

Hippo jest klasą bazową, po której dziedziczą kolejne klasy hipergrafowe. Zbiory metod Hippo można podzielić na cztery typy, a mianowicie:

- algorytmy związane z wczytywaniem danych,
- algorytmy związane z przetwarzaniem danych,
- algorytmy związane z prezentacją danych,
- algorytmy związane z eksportem danych.

Algorytmy związane z wczytywaniem danych pozwalają na pobranie danych z pliku tekstowego o ujednoliconym formacie PNH. Mowa o nim w rozdziale 5.1.1 na stronie 64. Plik PNH opisuje bazową sieć Petriego poprzez przedstawienie miejsc i tranzycji, a także znakowania początkowego. Algorytmy grafowe i hipergrafowe generalnie jako wejście przyjmują macierz incydencji hipergrafu.

Algorytmy przetwarzające dane pozwalają na uzyskanie dla zadanych danych wejściowych odpowiedniego wyniku. Mowa tutaj o wszelakiej maści algorytmach grafowych lub hipergrafowych, np. wyznaczenie transwersal.

Algorytmy prezentacji danych umożliwiają wyświetlenie zadanych czynności wynikających zarówno z przetwarzania danych, jak i wczytania plików wejściowych.

Algorytmy eksportujące pozwalają na zapis rezultatów do pliku, w szczególności eksport do formatu zgodnego z prezentacją wyników w programie Microsoft Excel.

A.1.2 Najważniejsze funkcjonalności

Biblioteka zawiera funkcjonalności przydatne w analizie, dekompozycji oraz projektowaniu systemów dyskretnych.

Najważniejsze metody klasy Hippo

Klasa Hippo jest klasą bazową i dostarcza podstawowych funkcjonalności manipulacji danymi. Konstruktory klasy pozwalają na inicjalizację za pomocą:

- macierzy incydencji,

- ścieżki do pliku,
- wektora danych,
- inicjalizacji pustej.

Hippo umożliwia na wyświetlenie wczytanych danych zarówno w formie sformatowanej, jak i bezpośredniej. Różnica polega na wyświetlaniu (lub nie) oznaczeń poszczególnych wierzchołków, krawędzi, separatorów elementów macierzy.

Zaimplementowano metody funkcjonalne. Przykładowe algorytmy, które zwracają określone dane:

- stopień wierzchołków i krawędzi,
- wierzchołek istotny,
- transpozycję macierzy,
- konwersję do grafu incydencji i sąsiedztw,
- dopełnienie grafu sąsiedztw,
- redukcja za pomocą szybkiego algorytmu redukcji,
- sprawdzenie pokrycia przez krawędzie.

Wyszczególnione zostały najistotniejsze pozycje. Lista wszystkich dostępnych metod jest obszerniejsza.

Najważniejsze metody klasy HippoCliques

Klasa HippoCliques dziedziczy po Hippo. Konstruktory klasy umożliwiają inicjalizację za pomocą: macierzy incydencji, scieżki do pliku macierzy, a także inicjalizacji pustej. Najistotniejszą metodą klasy jest funkcja pozwalająca na wyznaczenie wszystkich maksymalnych klik metodą:

- Bron-Kerbosh,
- Bron-Kerbosh pivot.

Najważniejsze metody klasy HippoColors

Klasa HippoColors dziedziczy po Hippo. Konstruktory klasy umożliwiają inicjalizację za pomocą: macierzy incydencji, hipergrafu. Zaimplementowane zostały najpopularniejsze metody kolorowania, a mianowicie:

- bez uporządkowania,

- LF,
- SL,
- z uporządkowaniem losowym,
- z nawrotami (*ang. Backtracking*).

Najważniejsze metody klasy HippoCompaparability

Klasa HippoCompaparability dziedziczy po Hippo. Konstruktory klasy umożliwiają inicjalizację za pomocą: macierzy incydencji, scieżki do pliku macierzy, a także inicjalizacji pustej. Klasa ta dostarcza metody związane z grafami porównywalności. Dostępne są następujące metody:

- określenie przynależności,
- największa klika,
- klika ważona,
- zbiór niezależny,
- kolorowanie,
- pokrycie klikowe.

Najważniejsze metody klasy HippoComplement

Klasa HippoComplement dziedziczy po Hippo. Konstruktory klasy umożliwiają inicjalizację za pomocą: macierzy incydencji, scieżki do pliku macierzy, a także inicjalizacji pustej. Klasa zawiera algorytmy dopełnienia dla grafów i hipergrafów oraz wyznaczania zbiorów niezależnych. Dostępne są następujące metody:

- wyszukiwania wszystkich zbiorów niezależnych metodą opracowaną przez: Johnson, Yannakakis, Papadimitriou,
- wyszukiwania wszystkich zbiorów niezależnych w oparciu o kliki,
- wyszukiwania dopełnienia grafu.

Najważniejsze metody klasy HippoCovers

Klasa HippoCovers dziedziczy po Hippo. Konstruktory klasy umożliwiają inicjalizację za pomocą: macierzy incydencji, scieżki do pliku macierzy, a także inicjalizacji pustej. Klasa zawiera algorytmy pokrycia hipergrafów. Dostępne są następujące metody:

- zachłanna,
- z nawrotami,
- szybki algorytm redukcji,
- szybki algorytm redukcji wraz z metodą zachłanną.

Najważniejsze metody klasy HippoMethods

Klasa HippoMethods dziedziczy po HippoCovers i została opracowana przez autora rozprawy. Konstruktory klasy umożliwiają inicjalizację za pomocą: macierzy incydencji, scieżki do pliku macierzy. Klasa zawiera algorytmy selekcji w kontekście dekompozycji systemów dyskretnych opisanych sieciami Petriego. Dostępne są następujące metody:

- zachłanna,
- z nawrotami,
- proponowana metoda transwersal dokładnych.

Najważniejsze metody klasy HippoSelect

Klasa HippoSelect dziedziczy po HippoCovers i została opracowana przez autora rozprawy. Konstruktory klasy umożliwiają inicjalizację za pomocą: macierzy incydencji, scieżki do pliku macierzy. Klasa zawiera algorytmy selekcji. Dostępne są następujące metody:

- zachłanna,
- z nawrotami,
- proponowana metoda transwersal dokładnych,
- oparta o algorytm DLX.

Najważniejsze metody klasy HippoVectors

Klasa HippoVectors jest klasą bazową. Pozwala na manipulacje logiczne na wektorach oraz macierzach. Dostępne są następujące metody:

- XOR oraz AND dwóch zadanych wektorów,
- iloczyn oraz suma dwóch zadanych macierzy,
- transpozycja macierzy,
- wyznaczanie macierzy jednostkowej,
- wyznaczanie macierzy zerowej.

Najważniejsze metody klasy pnets

Klasa pnets dziedziczy po Hippo oraz HippoVectors. Pozwala na manipulacje związane z sieciami Petriego. Dostępne są następujące metody:

- pobieranie wejść i wyjść tranzycji,
- pobieranie wejść i wyjść miejsc,
- wyznaczanie makrosieci,
- wyznaczanie hipergrafu współbieżności,
- wyznaczanie grafu współbieżności,
- wyznaczanie zbioru znakowań osiągalnych,
- wyznaczanie hipergrafu selekcji,
- wyznaczanie podsieci automatowych,
- wyznaczanie niezmienników miejsc metodą Martineza-Silvy,

Najważniejsze metody klasy SMCs

Klasa SMCs dziedziczy po Petri, HippoCover oraz HippoColor. Konstruktory klasy umożliwiają inicjalizację za pomocą scieżki do pliku macierzy. Klasa ta zawiera algorytmy związane z wyznaczaniem podsieci automatowych za pomocą metod wykorzystujących:

- hipergraf współbieżności,
- przeszukiwanie grafu,
- kolorowania grafu,
- niezmienniki miejsc,
- algorytm zachłanny,
- algorytm z nawrotami.

Najważniejsze metody klasy xtrec

Klasa SMCs dziedziczy po Hippo i pozwala na sprawdzenie, czy zadany hipergraf należy do klasy xt. Najważniejsze metody klasy są następujące:

- sprawdzenie, czy wektor A zawiera się w B,
- wyznaczenie gwiazdy,
- zwrócenie wartości algorytmu xtrec.

Dodatek B

Biblioteka modułów testowych

Biblioteka sieci testowych została utworzona w ośrodku zielonogórskim przez *M. i R. Wiśniewskich*. Autor niniejszej rozprawy uczestniczył w pracach mających na celu rozbudowanie tego zbioru. Finalnie na chwilę pisania niniejszej rozprawy biblioteka zawierała ponad 200 sieci testowych. Wartość tą można uznać za reprezentatywną na potrzeby badań związanych z eksperymentalną weryfikacją opracowanych metod: selekcji oraz dekompozycji. Część sieci pochodzi z literatury, część opisuje realne systemy dyskretne, natomiast część systemy hipotetyczne. Benchmarki cechuje spore zróżnicowanie co do klasy sieci oraz ich wielkości.

Benchmarki biblioteki Hippo zostały udostępnione w sieci internet w formie serwisu pozwalającego na wyszukiwanie, podgląd struktury, reprezentację macierzową, a także proste manipulacje. Zbiór dostępny jest pod adresem *hippo.uz.zgora.pl*

Sieci testowe są opisywane przez kilka istotnych atrybutów. Są one następujące:

- nazwa sieci, dzięki której możlwie jest wyszukanie benchmarka w bibliotece internetowej
- klasa sieci Petriego, która definiuje budowę sieci,
- liczba miejsc i tranzycji, pozwalające na poglądowe określenie wielkości benchmarka,
- czy rzeczywista, umozliwiający określenie źródła sieci.

Dodatek C

Internetowy system Hippo - prezentacja oraz szczegółowy opis możliwości

System internetowy Hippo dostępny pod adresem *hippo.uz.zgora.pl* jest uzupełnieniem biblioteki hippo.h. Został utworzony przez zespół badaczy z ośrodka zielonogórskiego, w tym autora niniejszej rozprawy. System ten ma za zadanie dostarczać podstawowych funkcjonalności pozwalających na analizę, a także wspomagać w dekompozycji i projektowaniu systemów dyskretnych opisanych sieciami Petriego. Integralną funkcjonalnością jest przeglądarka biblioteki benchmarków, czyli sieci Petriego wykorzystywanych w badaniach eksperymentalnych. System Hippo został utworzony zgodnie z najnowszymi trendami dotyczącymi aplikacji mobilnych. Pierwsza wersja systemu napisana została w języku python oraz frameworku django.

C.1 Możliwości systemu

System internetowy Hippo posiada zaimplementowane funkcjonalności, pozwalające na podstawowe analizy systemów dyskretnych opisanych sieciami Petriego. Pokrótce, ich lista wygląda następująco:

- klasyfikacja sieci w oparciu o klasyczne definicje klas,
- analiza p-niezmienników (niezmienników miejsc),
- dekompozycja sieci w oparciu o hipergrafy,
- dekompozycja sieci w oparciu o grafy porównywalności,
- dekompozycja sieci w oparciu o heurystyczne inwarianty,
- pokrycie sieci Petriego w oparciu o hipergrafy,

- pokrycie sieci Petriego w oparciu o heurystyczne inwarianty,
- analiza relacji współbieżności w oparciu o kliki grafów,
- analiza relacji sekwencyjności w oparciu o kliki grafów,
- eksport do kodu Verilog,
- eksport do formatu PIPE XML.

System ponadto pozwala na podgląd macierzy w formacie PNH oraz eksport wyrysowanej sieci do formatu graficznego PNG. Hippo udostępnia także mechanizmy systemowi GRES zespołowi badaczy z Portugalii (Universidade de Nova) pozwalające na analizę sieci za pomocą aparatu hipergrafowego.

C.2 Przegląd funkcjonalności

W bieżącym podrozdziale opisano pokrótce zaimplementowane funkcjonalności.

C.2.1 Klasyfikacja sieci

Algorytm klasyfikacji sieci dokonuje przyporządkowania klasy dla danej sieci w oparciu o klasyczne definicje klas występujące w pozycji [23]. Dopuszczalne opcje są następujące:

- Maszyna stanów (SM),
- Graf zamarkowany (MG),
- Sieć wolnego wyboru (FC),
- Sieć rozszerzonego wolnego wyboru (EFC),
- Sieć asymetrycznego wyboru (AC, znaną także jako sieć prostą SN)

W rezultacie metoda zwraca pełną lub skróconą nazwę klasy.

C.2.2 Dekompozycja sieci

W ramach systemu została dodana podstawowa dekompozycja sieci w oparciu o teorię grafów nieskierowanych i kolorowanie metodą LF grafu współbieżności. W rezultacie dla zadanej sieci Petriego wyświetlany jest wynik algorytmu kolorowania. Dostępne są także metody dekompozycji w oparciu o model hipergrafowy, grafowy oraz inwariantowy.

C.2.3 Pokrycie sieci Petriego

System internetowy Hippo umożliwia wygenerowanie pokrycia sieci Petriego za pomocą hipergrafów i inwariantów.

C.2.4 Analiza niezmienników miejsc

Metoda analizy w oparciu o niezmienniki miejsc została oparta o algorytm Martineza-Silvy, który w czasie wykładniczym pozwala na uzyskanie rezultatu w oparciu o daną sieć Petriego. Wyświetlana jest także informacja, czy sieć jest pokryta przez niezmienniki miejsc.

C.2.5 Analiza relacji współbieżności i sekwencyjności

System internetowy Hippo pozwala na przeprowdzenie podstawowej analizy relacji zarówno współbiezności, jak i sekwencyjności. Wyznaczane są miejsca współbieżne, a także sekwencyjne za pomocą modelu hipergrafowego.

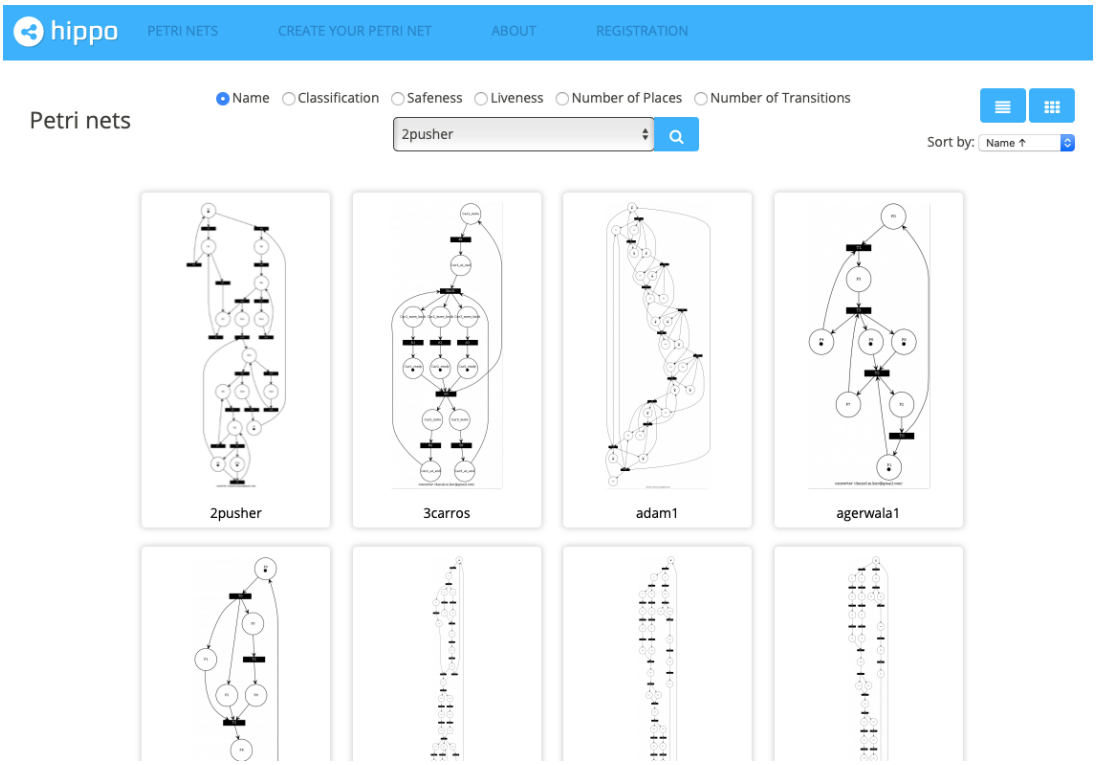
C.3 Graficzny interfejs użytkownika

System webowy Hippo posiada intuicyjny interfejs pozwalający na swobodną komunikację. Po wybraniu głównego adresu strony *hippo.uz.zgora.pl* ładowany jest moduł obsługujacy przeglądanie testowych sieci Petriego. Benchmarki przedstawiane są domyślnie w formie graficznej: miniaturka sieci oraz podpis w formie nazwy sieci. Rysunek C.1 przedstawia główny moduł systemu. Dopuszczalne jest wyszukiwanie sieci po jej nazwie a także zmiana sposobu wyświetlania listy.

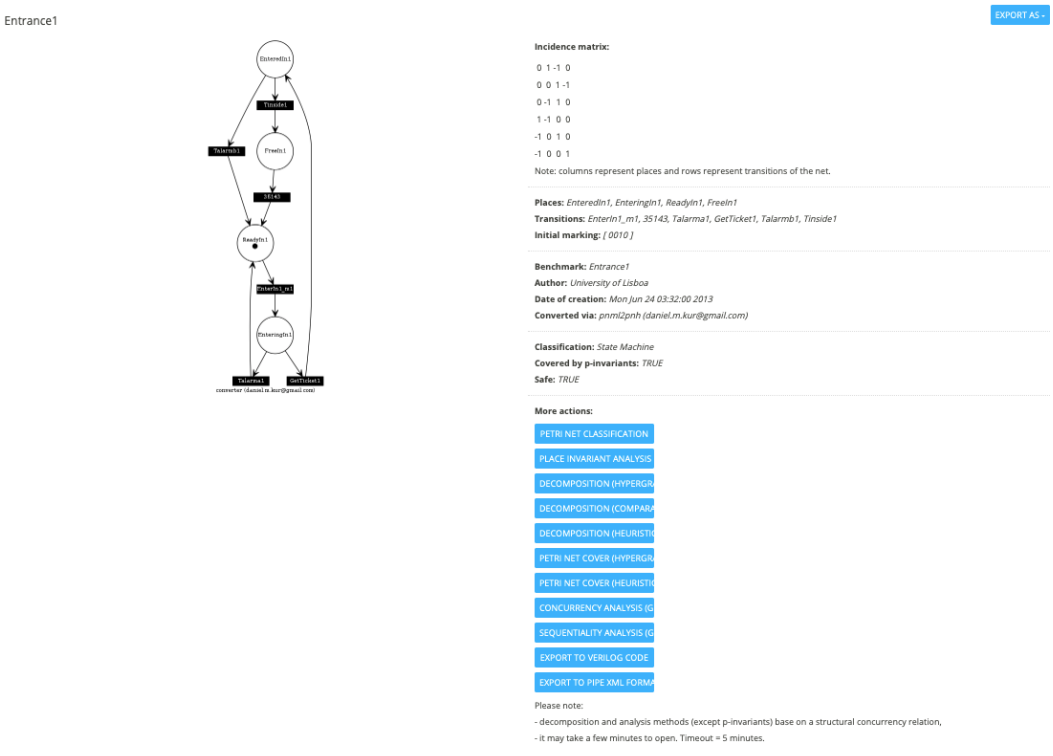
W górnej części karty znajduje się menu pozwalające na wyświetlenie informacji o autorach systemu oraz powrót do przeglądarki benchmarków. Po wybraniu danego benchmarka wyświetlana jest karta informacyjna. Przedstawiono to na rysunku C.2.

Karta została zaprojektowana pod kątem optymalnego rozmieszczenia niezbędnych danych. Lewą część ekranu zajmuje graficzny rysunek sieci. Wygenerowano go za pomocą oprogramowania *Graphviz*. Prawa część natomiast zajęta jest przez następujące kontrolki:

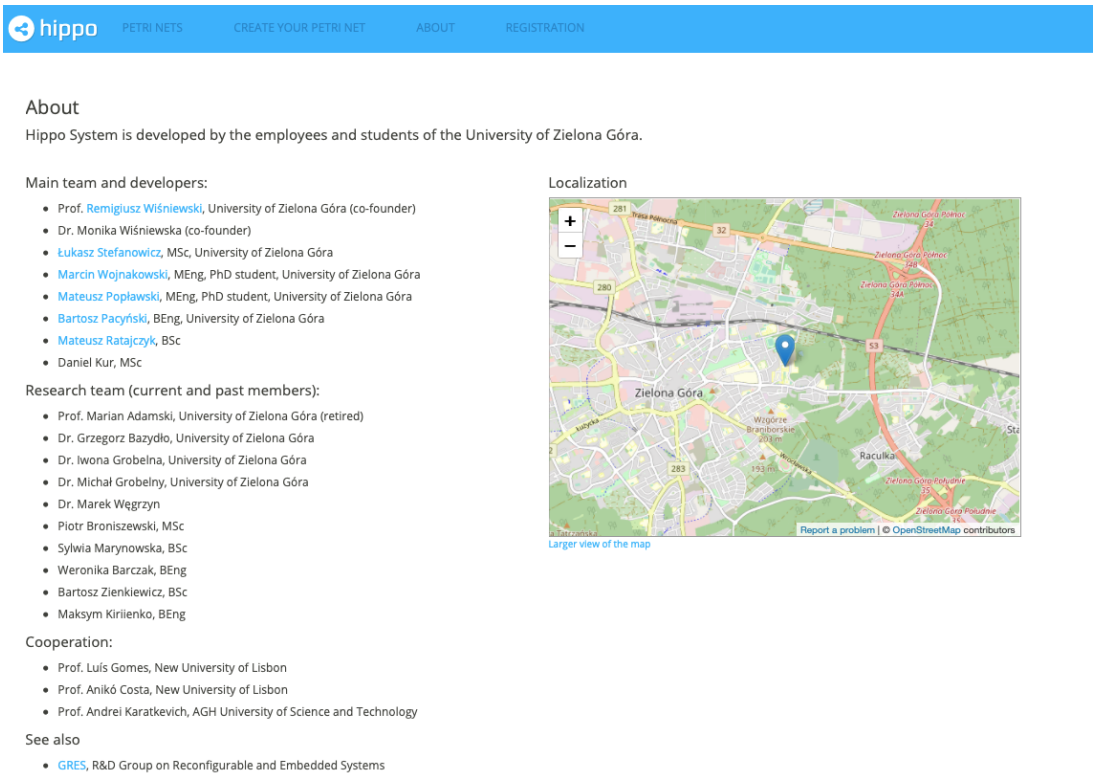
- macierz incydencji,
- oznaczenia miejsc, tranzycji,



Rysunek C.1: Rysunek przedstawiający główny moduł systemu



Rysunek C.2: Rysunek przedstawiający kartę informacyjną w systemie Hippo



Rysunek C.3: Rysunek przedstawiający kartę iinformacji o systemie i autorach Hippo

- znakowanie początkowe,
- dane szczegółowe benchmarka: nazwa, autor, klasyfikacja,
- akcje funkcyjne, które pozwalają na wywołanie danych algorytmów.

Integralną częścią systemu jest karta informacji o systemie i autorach. Przedstawia ją rysunek C.3.

Plansza została podzielona na dwie strony: pierwsza, lewa przedstawia następujące informacje dotyczące:

- twórców systemu,
- badaczy wykorzystujących lub współtworzących system Hippo w kontekście naukowym,
- naukowców współpracujących,
- systemów partnerskich.

, a także druga, prawa, zawierająca mapę lokalizacyjną.

Ważnym aspektem jest współpraca z naukowcami z Lizbony (system GRES), którzy zajmują się m.in. aspektami analizy systemów dyskretnych opisanych sieciami Petriego. System internetowy Hippo umożliwia przyjmowanie żądań ze strony systemu GRES oraz analizować przesyłane sieci.

Dodatek D

Dowody matematyczne

Niniejszy dodatek zawiera najważniejsze dowody matematyczne, wykorzystane w rozprawie. Całość twierdzeń i dowodów dostępna jest w [107]

Lemat D.1 *Niech Σ będzie siecią Petriego pokrywalną przez podsieci automatowe i $H_S = (S, P)$ jej dualnym hipergrafem sekwencyjności. Jeśli H_S posiada transwersale dokładne wtedy wszystkie posiadają tę samą liczbę elementów równą liczbie tokenów sieci Σ, która jest konserwatywna.*

Dowód. Niech $T \in Tr(H)$ będzie transwersalą dokładną H_S. Każda $s \in S$ określa podsieć automatową sieci Σ, stąd zgodnie z definicją transwersali dokładnej każde miejsce P jest pokryte przez dokładnie jedną $s \in T$. Niech M będzie znakowaniem osiągalnym sieci Σ. Załóżmy $|M| > |T|$. Wtedy istnieje $s \in T$ pokrywające przynajmniej dwa współbieżne miejsca, które zaprzeczają definicji podsieci automatowej. Załóżmy $|M| < |T|$. Wtedy istnieje $p \in M$ pokryte przez przynajmniej dwa elementy należące do T, które zaprzeczają twierdzeniu "każde miejsce P jest pokryte przez dokładnie jedno $s \in T$". Załóżmy $|M| = |T|$. Tak długo, jak powyższe rozumowanie można zastosować dla dowolnej transwersali dokładnej hipergrafu H_S oraz dowolnego znakowania osiągalnego sieci Σ, wtedy sieć Σ jest konserwatywna i każda transwersala dokładna hipergrafu H_S składa się z tej samej liczby elementów równej liczbie tokenów sieci Σ. □

Twierdzenie D.2 *Niech Σ będzie siecią Petriego pokrywalną przez podsieci automatowe i H_S jej dualnym hipergrafem sekwencyjności. Jeśli H_S posiada transwersalę dokładną $T \in Tr(H)$, wtedy określa ona minimalne pokrycie sieci Σ za pomocą podsieci automatowych.*

Dowód. Zgodnie z definicją dualnego hipergrafu sekwencyjności, jego transwersale określają pokrycie sieci Petriego za pomocą podsieci automatowych. Zgodnie z lematem

D.1, $|T| = |M|$, gdzie M jest dowolnym znakowaniem osiągalnym sieci Σ. Załóżmy, że istnieje kolejna transwersala T' z H_S taka, że $|T'| < |T|$. Wtedy istnieje $s \in T'$ pokrywająca przynajmniej dwa współbieżne miejsca, co zaprzecza definicji podsieci automatowej. Stąd wynika, że pokrycie określone przez T jest minimalne. $\square$

Dodatek E

Szczegółowe rezultaty badań eksperymentalnych

Niniejszy dodatek jest rozwinięciem ogólnie zaprezentowanych w rozprawie rezultatów badań. Zawiera wyniki dla wszystkich żywych sieci ze zbioru Hippo.

E.1 Dekompozycja za pomocą metody niezmienników miejsc

Poniżej przedstawiono nagłówki tabeli prezentującej wyniki.

- Sieć (miejsca/tranzycje) - kolumna określa nazwę sieci testowej oraz jej liczbę miejsc, tranzycji,
- Metoda klasyczna: t [ms] - czas dekompozycji, SMCs - liczba uzyskanych podsieci automatowych,

- Metoda proponowana 1: t [ms] - czas dekompozycji, SMCs - liczba uzyskanych podsieci automatowych,
- Metoda proponowana 2: t [ms] - czas dekompozycji, SMCs - liczba uzyskanych podsieci automatowych.

'-' oznacza, że wynik nie został uzyskany w czasie 30 minut.

	Metoda klasyczna		Metoda proponowana 1		Metoda proponowana 2	
Sieć (miejsca / tranzycje)	t [ms]	SMCs	t [ms]	SMCs	t [ms]	SMCs
3carros (12/8)	3	2.1906	3	0.3895	3	0.0079
adam1 (24/12)	12	14.95	12	14.231	12	0.0016
agerwala1 (8/4)	4	0.1707	4	0.168	4	0.0011
agostini1 (7/5)	3	0.112	3	0.1226	3	0.001
alfa net copy milling machnie subprocess synchr (32/27)	4	2.5321	4	1.0622	4	0.001
barkaoui1 (11/13)	2	0.1969	2	0.1739	2	0.001
barkaoui2 (10/8)	4	0.1916	4	0.2159	4	0.0011
barkaoui3 (18/13)	6	0.5481	6	0.5282	6	0.0027
basile1 (21/17)	5	0.7881	5	0.7741	5	0.0011
beverage (16/13)	4	0.2617	4	0.3339	4	0.0011
beverage production (20/16)	4	0.5125	4	0.5281	4	0.001

beverage production part2 (16/13)	4	0.2713	4	0.3015	4	0.0011
bouillard1 (12/12)	3	0.2323	3	0.2479	3	0.0026
bridge (8/6)	2	0.2162	2	0.2598	2	0.0009
bridge semaphore (9/6)	3	0.2947	3	0.3131	3	0.0023
campos1 (11/8)	4	0.2327	4	0.264	4	0.0011
campos2 (12/10)	5	0.2024	5	0.2158	5	0.0012
campos3 (15/11)	6	0.3008	6	0.329	6	0.0013
chiola1 (16/15)	5	0.2316	5	0.3131	5	0.0011
chrzastowski1 (10/10)	3	0.1478	3	0.1602	3	0.0011
chrzastowski2 (22/21)	2	0.2872	2	0.3351	2	0.0009
cncrr001 (7/4)	5	0.2074	3	0.1617	3	0.001
cncrr002 (11/7)	5	0.2333	5	0.2913	5	0.0011
CNC machine (7/5)	2	0.0994	2	0.1276	2	0.0015
cn crr7 (56/15)	-	-	-	-	28	23.1842
cn crr10 (80/21)	-	-	-	-	40	45.762
cn crr15 (120/31)	-	-	-	-	60	124.834
cn crr25 (200/51)	-	-	-	-	100	750.712

coloured (4/3)	2	0.1491	2	0.1216	2	0.0009
communications protocol (6/8)	1	0.1293	1	0.0734	1	0.0009
Consistent Example Message View (10/6)	4	0.2129	4	0.2541	4	0.001
cortadella1 (9/6)	4	0.17	4	0.212	4	0.001
credit procedure (10/8)	2	0.1322	2	0.1577	2	0.0009
crossroadSM FPGA (32/13)	10	1740000	10	1740000	10	18.1782
dataflow computation (11/7)	4	0.2028	4	0.2231	4	0.0015
desel1 (15/9)	7	0.2571	7	0.3281	7	0.0013
dingle1 (20/20)	4	0.3079	4	0.3993	4	0.0011
dongen1 (15/17)	2	0.182	2	0.2016	2	0.0009
elevator 2 (8/8)	1	0.1284	1	0.1173	1	0.0008
Entrance1 (4/6)	1	0.0475	1	0.0707	1	0.001
eshuis1 (17/15)	3	0.3836	3	0.5148	3	0.001
esparza1 (11/10)	4	0.1551	4	0.2329	4	0.001
esparza2 (15/13)	4	22.4599	4	20.7924	4	0.0023
esparza3 (13/15)	3	0.2463	3	0.2052	3	0.001

Exe5 split (8/6)	2	0.209	2	0.209	2	0.001
fernandez1 (13/11)	4	0.2109	4	0.2276	4	0.001
fernandez2 (11/10)	3	0.1587	3	0.2142	3	0.0009
fernandez3 (21/20)	4	1.5431	4	1.4284	4	0.001
fig311 01 (6/6)	2	0.0771	2	0.0787	2	0.0008
FMS main SIPN (7/4)	3	0.1783	3	0.1839	3	0.001
four philosophers (12/8)	8	0.2347	8	0.301	8	0.0014
frame manufact (13/10)	4	0.2582	4	0.2559	4	0.0011
franczok1 (10/14)	1	0.1276	1	0.1236	1	0.0009
gals-example (6/4)	3	0.1147	3	0.125	3	0.001
gaubert1 (16/8)	-	-	4	2.3257	4	0.0026
gaubert2 (8/4)	4	0.2795	4	0.1604	4	0.0012
girault1 (12/6)	5	3.6557	5	0.3931	5	0.1703
girault2 (16/10)	5	3.6484	5	0.5595	5	0.1992
girault3 (14/9)	7	0.4954	7	0.5532	7	0.0013
girault4 (10/7)	4	0.1459	4	0.1847	4	0.0011

girault7 (11/7)	4	6.7199	4	6.81	4	0.3557
girault8 (8/8)	4	0.1511	4	0.175	4	0.0011
hack1 (13/13)	4	0.2316	4	0.2903	4	0.0013
HAN (24/40)	6	0.4888	6	0.5415	6	0.0012
handling of incoming order (17/14)	5	0.4203	3	0.5908	3	0.001
hard case (9/3)	4	333.754	4	0.4365	6	0.0012
health care process (6/8)	1	0.0759	1	0.0861	1	0.0009
hee1 (14/12)	4	0.4279	4	0.3952	4	0.0022
hee2 (13/11)	4	0.3946	4	0.3369	4	0.0021
heiner1 (22/20)	7	0.5287	7	0.5151	7	0.0012
holliday1 (20/15)	10	0.419	10	0.4559	10	0.0015
holloway1 (13/10)	3	0.4217	3	0.4845	3	0.0023
hulgaard1 (19/12)	-	-	4	240.165	4	0.6104
IEC (15/12)	3	0.6009	3	0.4901	3	0.001
img 280 (8/6)	3	0.0996	3	0.1251	3	0.001
invariants exponent 3 2 (6/2)	3	1.9226	3	0.1635	3	0.0022

invariants exponent 3 3 (9/3)	3	1.20582e+006	3	0.6675	3	0.2043
invariants exponent 3 4 (12/4)	-	-	3	4.0633	3	0.8515
jeng1 (14/16)	2	0.3235	2	0.2445	2	0.001
jeng2 (10/9)	3	0.1383	3	0.1608	3	0.001
julvez 1 (11/8)	4	0.246	4	0.2228	4	0.001
julvez 2 (12/11)	3	0.1574	3	0.1796	3	0.0011
kavi1 (18/20)	3	0.239	3	0.2708	3	0.001
lab5 (8/6)	2	0.1352	2	0.1376	2	0.001
lasire1 (15/13)	2	1.4177	2	0.4222	2	0.001
li1 (17/17)	3	0.3122	3	0.3203	3	0.001
li2 (12/12)	2	0.1906	2	0.216	2	0.0009
li3 (11/14)	2	0.1475	2	0.1832	2	0.001
lnet p10n1 (12/9)	4	0.2202	4	0.206	4	0.001
lnet p1n1 (6/4)	3	0.0961	3	0.1678	3	0.001
lnet p1n2 (6/5)	3	0.1152	3	0.112	3	0.0009
lnet p1n4 (9/6)	4	0.1451	4	0.1541	4	0.001
lnet p2n2 (13/7)	4	143554	4	134614	4	0.0023
lnet p2n3 (9/7)	2	0.1582	2	0.2028	2	0.001
lnet p4n1 (37/41)	1	0.6238	1	0.7907	1	0.0009

lnet p5n1 (11/9)	3	0.1605	3	0.1794	3	0.001
lnet p5n2 (13/11)	3	0.1609	3	0.308	3	0.0009
lnet p5n3 (10/8)	3	0.1575	3	0.1595	3	0.001
lnet p6n1 (12/11)	3	0.2346	3	0.2052	3	0.001
lnet p6n3 (6/5)	3	0.1144	3	0.1542	3	0.0011
lnet p7n1 (41/32)	12	2.1087	12	2.3408	12	2.6906
lnet p8n2 (28/17)	4	0.4537	4	0.4248	4	0.0014
lnet p8n3 (21/17)	5	0.4009	5	0.5274	5	0.0011
lnet p8n4 (15/16)	3	0.2354	3	0.2819	3	0.0014
lnet p9n1 (14/16)	3	0.1983	3	0.2594	3	0.001
maruster1 (13/14)	3	0.1958	3	0.273	3	0.001
maruster2 (20/14)	6	10.3487	6	8.7171	6	0.0012
maruster3 (12/13)	2	0.1653	2	0.9412	2	0.0009
maruster4 (16/13)	4	0.2221	4	0.2934	4	0.0011
medeiros1 (9/13)	2	0.1439	2	0.1521	2	0.0009
medeiros2 (10/8)	4	0.2695	4	0.2286	4	0.0012
miczulski1 (9/8)	2	0.1964	2	0.2026	2	0.0011
miling machine (20/16)	4	1.1994	4	1.0865	4	0.0009
milling (20/16)	4	1.0753	4	1.0989	4	0.001

mixer (19/15)	4	0.4603	4	0.4779	4	0.001
mixer mod1 (20/16)	4	0.5073	4	0.4869	4	0.001
mixer mod2 (8/5)	4	0.1286	4	0.15	4	0.0011
mixer one cup (16/13)	4	0.2463	4	0.3175	4	0.0011
multi robot (9/6)	4	0.181	4	0.2007	4	0.0085
mutual exclusion (9/8)	3	0.1087	3	0.1214	3	0.001
net1 (6/5)	2	0.1067	2	0.1524	2	0.001
net3 (8/8)	2	0.1258	2	0.1213	2	0.0011
net4 (8/7)	2	0.178	2	0.1549	2	0.0009
np3 (6/3)	3	0.1178	3	0.1346	3	0.0009
np5 (10/5)	3	7.4468	3	0.4004	3	0.0022
oil separator cover alfa net (27/21)	4	172690	4	132840	4	0.002
oil separator cover s net (29/25)	4	177267	4	134041	4	0.0024
oil separator cover s net v2 (15/11)	4	143566	4	133994	4	0.0022

oil separator cover s net v3 (11/11)	2	0.2054	2	0.2508	2	0.0008
one Way Transmission System (9/6)	4	0.1633	4	0.1924	4	0.0011
parking lot (6/8)	2	0.0968	2	0.1348	2	0.001
pascal reach (10/9)	4	0.1566	4	0.205	4	0.0011
pcncp (6/4)	3	0.085	3	0.1173	3	0.001
philosophers 2 (14/10)	6	0.2546	6	0.2973	6	0.0012
philosophers 2 rev 2 (14/10)	6	0.3422	6	0.2872	6	0.001
philosophers 5 (20/15)	10	0.4652	10	0.4766	10	0.0015
ping1 (12/14)	2	0.1494	2	0.178	2	0.001
pnbrexpl 02 (13/10)	4	0.2672	4	0.2844	4	0.0011
pnbrexpl 03 (9/8)	2	0.1967	2	0.2096	2	0.001
pnbrexpl 04 (13/13)	3	0.1976	3	0.203	3	0.0009
pnbrexpl 05 (7/7)	2	0.0877	2	0.0984	2	0.0009

pnbrexpl 06 (11/9)	2	0.2174	2	0.264	2	0.001
pnbrexpl 07 (11/12)	3	0.1425	3	0.166	3	0.001
pnbrexpl 08 (13/11)	3	0.4527	3	0.5079	3	0.001
pnbrexpl 09 (16/14)	4	0.754	4	0.6949	4	0.3029
pnbrexpl 12 (14/13)	4	0.2657	4	0.2332	4	0.0011
pnbrexpl 15 (13/12)	3	0.1675	3	0.1973	3	0.001
pn campos silva2 (13/12)	4	0.1659	4	0.221	4	0.0011
pn campos silva7 (13/13)	3	0.1966	3	0.258	3	0.0013
pn desel 01 (7/8)	2	0.1236	2	0.1256	2	0.0009
pn desel 02 (8/5)	4	0.1325	4	0.1589	4	0.0146
pn desel 03 (6/7)	2	0.1558	2	0.124	2	0.001
pn fernandez3 (21/20)	4	1.7807	4	1.4246	4	0.0012
pn silva 01 (9/9)	2	1.2742	2	0.2699	2	0.0009
pn silva 02 (6/4)	2	0.122	2	0.1239	2	0.001
pn silva 03 (14/10)	5	0.6594	5	0.7242	5	0.0011

pn silva 05f (5/4)	3	0.0849	3	0.0822	3	0.0013
prio ex (2/3)	1	0.0298	1	0.0651	1	0.0009
prod cons (8/5	3	0.1128	3	0.1489	3	0.0011
PUMA loading (6/4)	2	0.0754	2	0.0863	2	0.001
PUMA unloading (5/3)	2	0.1024	2	0.1018	2	0.0009
PWM extended (49/31)	0	-	0	-	10	29.657
PWM patterns (19/12)	0	-	0	-	4	1.6649
reactor small (9/8)	2	0.2016	2	0.2645	2	0.0009
real-life system smarthome simplified (12/7)	5	2.7497	5	0.3684	5	0.0023
reiseg1 (13/9)	5	0.3546	5	0.2586	5	0.0011
rejers1 (14/11)	4	0.2398	4	0.3111	4	0.0011
rejers2 (11/10)	2	0.1496	2	0.1494	2	0.0009
return books (4/6)	1	0.0459	1	0.0566	1	0.0009
RHINO loading (6/4)	2	0.0737	2	0.0858	2	0.0009

RHINO unloading (6/4)	2	0.0888	2	0.1048	2	0.0009
semaphore (3/4)	1	0.0453	1	0.0484	1	0.0009
semaphore2 (9/6)	5	0.1527	5	0.173	5	0.0012
silva1 (7/7)	2	0.1496	2	0.1477	2	0.0009
silva14 (8/7)	3	0.1235	3	0.1351	3	0.001
silva4v2 (13/12)	4	0.2477	4	0.2247	4	0.0014
silva5 (16/8)	6	14.5896	6	13.6172	6	13.121
silva7 (13/13)	3	0.2061	3	0.2612	3	0.001
simple production system (9/6)	4	0.1545	4	0.1713	4	0.0011
sivaraman1 (10/10)	2	0.1332	2	0.143	2	0.0008
speedway (10/7)	3	0.2253	3	0.2256	3	0.0011
state space16 (16/16)	8	0.265	8	0.3154	8	0.0012
state space32 (32/32))	16	0.6721	16	0.9992	16	0.002
state space48 (48/48)	24	1.5755	24	2.4212	24	0.0028
sub-task of PLT and PMN count (8/5)	3	0.1171	3	0.1525	3	0.0008

s net copy milling machine subprocess (31/28)	4	2.7065	4	1.1937	4	0.0011
s net frame manufact quality v1 (17/16)	4	0.3566	4	0.4226	4	0.0011
s net frame manufact quality v2 mod (19/18)	4	0.3943	4	0.4353	4	0.0011
tank heating (4/4)	1	0.0626	1	0.0471	1	0.0009
TP5 I (8/6)	2	0.1828	2	1.6107	2	0.0008
traffic lights (6/5)	3	0.0885	3	0.1008	3	0.001
traffic light v1 (8/3)	4	0.5035	4	0.2738	4	0.0022
traffic light v2 (4/3)	2	0.0636	2	0.0695	2	0.0009
two traffic lights (7/7))	3	0.1016	3	0.1138	3	0.001
vanDerAalst1 (12/14)	2	0.161	2	0.1877	2	0.001
vanDerAalst3 (14/13)	2	0.1703	2	0.192	2	0.0009

vanDerAalst4 (18/16)	4	0.2867	4	0.323	4	0.0012
vanDerAalst5 (23/23)	4	0.4963	4	0.5976	4	0.001
vanDerAalst6 (9/13)	2	0.1576	2	0.1641	2	0.0009
vanDerAalst7 (14/13)	3	0.1927	3	0.2713	3	0.001
voorhoeve1 (13/12)	4	0.2155	4	0.3386	4	0.001
voorhoeve2 (14/12)	4	0.5235	4	0.5881	4	0.0011
voorhoeve3 (7/5)	3	0.125	3	0.1576	3	0.0012
weijters1 (12/14)	2	0.1843	2	0.217	2	0.001
well built alfa net copy milling machine subprocess (30/25)	4	2.6017	4	1.0626	4	0.0014
xie1 (24/26)	3	2.8209	3	2.5152	3	0.0009
zuberek1 (30/22)	5	470.941	5	303.95	5	0.0011
zuberek3 (29/21))	5	63453.3	5	3.1499	5	0.0011
zuberek4 (30/21)	-	-	-	-	6	0.0012

Tablica E.1: Tabela przedstawiająca szczegółowe rezultaty badań - dekompozycja za pomocą niezmienników miejsc)

E.2 Selekcja podsieci automatowych

Dane na potrzeby selekcji uzyskiwane są za pomocą zmodyfikowanej metody dekompozycji opartej o niezmienniki miejsc (Martinez-Silva). '-' oznacza, że wynik nie został uzyskany w czasie 30 minut.

- Sieć (miejsca/tranzycje) - kolumna określa nazwę sieci testowej oraz jej liczbę miejsc, tranzycji,
- Metoda klasyczna: t [ms] - czas selekcji, SMCs - liczba uzyskanych podsieci automatowych,
- Metoda proponowana 1: t [ms] - czas selekcji, SMCs - liczba uzyskanych podsieci automatowych,
- Metoda proponowana 2: t [ms] - czas selekcji, SMCs - liczba uzyskanych podsieci automatowych,

	Metoda klasyczna		Metoda proponowana 1		Metoda proponowana 2	
Sieć (miejsca / tranzycje)	t [ms]	SMCs	t [ms]	SMCs	t [ms]	SMCs
3carros (12/8)	3	0.3716	3	0.0152	3	0.0023
adam1 (24/12)	12	0.8004	12	0.0016	12	0.0016
agerwala1 (8/4)	4	0.0331	4	0.001	4	0.001
agostini1 (7/5)	3	0.0341	3	0.0016	3	0.0011

alfa net copy milling machnie subprocess synchr (32/27)	4	0.1914	4	0.0013	4	0.0011
barkaoui1 (11/13)	2	0.012	2	0.0009	2	0.0009
barkaoui2 (10/8)	4	0.0229	4	0.001	4	0.001
barkaoui3 (18/13)	6	0.2326	6	0.0022	6	0.0026
basile1 (21/17)	5	0.0587	5	0.0011	5	0.0012
beverage (16/13)	4	0.0228	4	0.001	4	0.0011
beverage production (20/16)	5	0.0753	4	0.001	4	0.0011
beverage production part2 (16/13)	4	0.023	4	0.001	4	0.0009
bouillard1 (12/12)	3	0.0293	3	0.0013	3	0.001
bridge (8/6)	2	0.0223	2	0.0011	2	0.001
bridge semaphore (9/6)	3	0.0764	3	0.0021	3	0.0022
campos1 (11/8)	4	0.0378	4	0.0011	4	0.001
campos2 (12/10)	5	0.0306	5	0.0011	5	0.0011

campos3 (15/11)	6	0.0361	6	0.0012	6	0.0013
chiola1 (16/15)	5	0.0294	5	0.0011	5	0.0012
chrzastowski1 (10/10)	3	0.0162	3	0.001	3	0.001
chrzastowski2 (22/21)	2	0.0148	2	0.0009	2	0.0009
cncrr001 (7/4)	3	0.0292	3	0.001	3	0.001
cncrr002 (11/7)	5	0.0294	5	0.0011	5	0.0011
CNC machine (7/5)	2	0.0103	2	0.001	2	0.001
cn crr10 (80/21)	-	-	-	-	40	75.2325
cn crr15 (120/31)	-	-	-	-	60	685.139
cn crr25 (200/51)	-	-	-	-	100	657.17
cn crr7 (56/15)	-	-	-	-	28	38.8418
coloured (4/3)	2	0.0117	2	0.0009	2	0.001
communications protocol (6/8)	1	740000	1	0.017	1	0.001
Consistent Example Message View (10/6)	4	0.0368	4	0.0011	4	0.0011
cortadella1 (9/6)	4	0.0451	4	0.0012	4	0.001
credit procedure (10/8)	2	0.0264	2	0.0009	2	0.001

crossroadSM FPGA (32/13)	10	7001	10	7200	10	5.312
dataflow computation (11/7)	4	0.0074	4	0.0008	4	0.0011
desel1 (15/9)	7	0.0066	7	0.0009	7	0.0012
dingle1 (20/20)	4	0.0369	4	0.001	4	0.0012
dongen1 (15/17)	2	0.0223	2	0.001	2	0.0009
elevator 2 (8/8)	1	0.0974	1	0.0021	1	0.0008
Entrance1 (4/6)	1	0.0175	1	0.001	1	0.0011
eshuis1 (17/15)	3	0.023	3	0.001	3	0.001
esparza1 (11/10)	4	0.0238	4	0.001	4	0.0012
esparza2 (15/13)	4	0.0166	4	0.0012	4	0.0022
esparza3 (13/15)	3	0.8049	3	0.0011	3	0.3702
Exe5 split (8/6)	2	0.0106	2	0.001	2	0.0009
fernandez1 (13/11)	4	0.0538	4	0.001	4	0.001
fernandez2 (11/10)	3	0.0543	3	0.0013	3	0.001
fernandez3 (21/20)	4	0.0358	4	0.0011	4	0.0012
fig311 01 (6/6)	2	0.0078	2	0.0139	2	0.001
FMS main SIPN (7/4)	3	0.0153	3	0.0011	3	0.0011

four philosophers (12/8)	-	-	8	0.0026	8	0.0135
frame manufact (13/10)	4	0.0338	4	0.001	4	0.001
franczok1 (10/14)	1	1.5724	1	0.0017	1	0.0008
gals-example (6/4)	3	1.6265	3	0.0153	3	0.0009
gaubert1 (16/8)	-	-	5	0.0015	5	0.0021
gaubert2 (8/4)	4	0.0225	4	0.0011	4	0.001
girault1 (12/6)	5	1.5358	5	0.2753	5	0.1718
girault2 (16/10)	5	0.0207	5	0.0011	5	0.1834
girault3 (14/9)	7	0.0218	7	0.0011	7	0.0013
girault4 (10/7)	4	0.0584	4	0.001	4	0.0011
girault7 (11/7)	4	0.0344	4	0.003	4	0.003
girault8 (8/8)	4	0.0694	4	0.0012	4	0.0014
hack1 (13/13)	4	0.007	4	0.001	4	0.0011
HAN (24/40)	6	0.099	6	0.0023	6	0.0011
handling of incoming order (17/14)	3	0.1148	3	0.002	3	0.001
hard case (9/3)	6	0.0897	6	0.0013	6	0.0011
health care process (6/8)	1	0.0864	1	0.0014	1	0.0008

hee1 (14/12)	4	0.0872	4	0.0022	4	0.0023
hee2 (13/11)	-	-	4	0.0027	4	0.0039
heiner1 (22/20)	7	0.1325	7	0.001	7	0.0015
holliday1 (20/15)	10	0.0158	10	0.001	10	0.0014
holloway1 (13/10)	3	0.3697	3	0.0024	3	0.0023
hulgaard1 (19/12)	4	3743.34	4	0.0021	4	0.5945
IEC (15/12)	-	-	3	0.0043	3	0.0009
img 280 (8/6)	3	0.0114	3	0.0011	3	0.001
invariants exponent 3 2 (6/2)	3	0.0169	3	0.001	3	0.0023
invariants exponent 3 3 (9/3)	3	0.0226	3	0.0011	3	0.2035
invariants exponent 3 4 (12/4)	3	0.0164	3	0.0011	3	0.8626
jeng1 (14/16)	2	0.0193	2	0.001	2	0.0009
jeng2 (10/9)	3	0.0231	3	0.001	3	0.001
julvez 1 (11/8)	4	0.282	4	0.001	4	0.0011
julvez 2 (12/11)	3	0.0188	3	0.0009	3	0.001
kavi1 (18/20)	3	0.0356	3	0.0009	3	0.001

lab5 (8/6)	2	0.0117	2	0.0009	2	0.001
lasire1 (15/13)	2	0.0222	2	0.0011	2	0.001
li1 (17/17)	3	0.015	3	0.001	3	0.0012
li2 (12/12)	2	0.0154	2	0.0012	2	0.0009
li3 (11/14)	2	0.022	2	0.0013	2	0.0009
lnet p10n1 (12/9)	4	69.8384	4	0.0023	4	0.0011
lnet p1n1 (6/4)	3	0.0368	3	0.001	3	0.001
lnet p1n2 (6/5)	3	0.0247	3	0.0008	3	0.001
lnet p1n4 (9/6)	4	0.0171	4	0.001	4	0.0011
lnet p2n2 (13/7)	4	0.0164	4	0.001	4	0.0024
lnet p2n3 (9/7)	2	0.0299	2	0.001	2	0.0012
lnet p4n1 (37/41)	1	0.029	1	0.0009	1	0.0009
lnet p5n1 (11/9)	3	0.015	3	0.001	3	0.001
lnet p5n2 (13/11)	3	0.6799	3	0.5993	3	0.0011
lnet p5n3 (10/8)	3	0.1075	3	0.0015	3	0.001
lnet p6n1 (12/11)	3	0.0348	3	0.001	3	0.0012
lnet p6n3 (6/5)	3	0.0175	3	0.0011	3	0.0011
lnet p7n1 (41/32)	9	0.0178	9	0.001	9	3.279
lnet p8n2 (28/17)	9	0.0166	9	0.001	9	0.0113
lnet p8n3 (21/17)	5	0.0851	5	0.0012	5	0.0012
lnet p8n4 (15/16)	3	0.0349	3	0.0009	3	0.0014
lnet p9n1 (14/16)	3	0.023	3	0.001	3	0.0009

maruster1 (13/14)	3	0.0114	3	0.0012	3	0.001
maruster2 (20/14)	6	0.0299	6	0.0012	6	0.0012
maruster3 (12/13)	2	0.0242	2	0.0009	2	0.001
maruster4 (16/13)	4	0.376	4	0.001	4	0.0011
medeiros1 (9/13)	2	0.4608	2	0.0013	2	0.001
medeiros2 (10/8)	4	0.0445	4	0.001	4	0.001
miczulski1 (9/8)	3	0.0476	2	0.0011	2	0.001
miling machine (20/16)	4	0.0227	4	0.001	4	0.0011
milling (20/16)	4	0.0512	4	0.001	4	0.001
mixer (19/15)	4	0.0227	4	0.001	4	0.0011
mixer mod1 (20/16)	4	0.0197	4	0.001	4	0.0012
mixer mod2 (8/5)	4	0.0103	4	0.0011	4	0.0011
mixer one cup (16/13)	4	0.0107	4	0.0009	4	0.0011
multi robot (9/6)	4	0.0242	4	0.001	4	0.0011
mutual exclusion (9/8)	3	0.0156	3	0.001	3	0.0011
net1 (6/5)	2	0.4184	2	0.002	2	0.001

net3 (8/8)	2	88.8691	2	0.0023	2	0.001
net4 (8/7)	2	89.8012	2	0.0021	2	0.001
np3 (6/3)	3	69.4111	3	0.0021	3	0.0011
np5 (10/5)	3	0.0223	3	0.0008	3	0.0156
oil separator cover alfa net (27/21)	4	0.0354	4	0.001	4	0.0022
oil separator cover s net (29/25)	4	0.0105	4	0.001	4	0.0024
oil separator cover s net v2 (15/11)	4	0.0218	4	0.001	4	0.0021
oil separator cover s net v3 (11/11)	2	0.0152	2	0.0009	2	0.001
one Way Transmission System (9/6)	4	0.0361	4	0.0012	4	0.0011
parking lot (6/8)	2	0.0364	2	0.0012	2	0.0011
pascal reach (10/9)	4	0.086	4	0.0013	4	0.008
pcncp (6/4)	3	0.012	3	0.0009	3	0.0012

philosophers 2 (14/10)	6	0.0219	6	0.0011	6	0.0013
philosophers 2 rev 2 (14/10)	6	0.0232	6	0.0009	6	0.0011
philosophers 5 (20/15)	10	0.0172	10	0.0009	10	0.0015
ping1 (12/14)	2	0.0107	2	0.0009	2	0.0009
pnbrexpl 02 (13/10)	4	0.0219	4	0.001	4	0.0011
pnbrexpl 03 (9/8)	2	0.0167	2	0.001	2	0.0009
pnbrexpl 04 (13/13)	3	0.1052	3	0.001	3	0.001
pnbrexpl 05 (7/7)	2	0.3017	2	0.2487	2	0.001
pnbrexpl 06 (11/9)	2	0.0239	2	0.0011	2	0.0009
pnbrexpl 07 (11/12)	3	0.0175	3	0.0011	3	0.001
pnbrexpl 08 (13/11)	3	0.2157	3	0.0002	3	0.0011
pnbrexpl 09 (16/14)	7	0.0224	7	0.0011	4	0.3529
pnbrexpl 12 (14/13)	4	0.0164	4	0.001	4	0.0012
pnbrexpl 15 (13/12)	3	0.0244	3	0.001	3	0.0017

pn campos silva2 (13/12)	4	0.01	4	0.0009	0	0.0011
pn campos silva7 (13/13)	3	0.4259	3	0.001	4	0.0009
pn desel 01 (7/8)	2	0.1092	2	0.0009	3	0.0009
pn desel 02 (8/5)	4	0.02	4	0.0009	2	0.001
pn desel 03 (6/7)	2	0.0332	2	0.0011	4	0.0009
pn fernandez3 (21/20)	4	0.0144	4	0.0011	4	0.0011
pn silva 01 (9/9)	2	0.0061	2	0.0009	2	0.001
pn silva 02 (6/4)	2	0.0153	2	0.0009	2	0.001
pn silva 03 (14/10)	5	0.0103	5	0.0009	5	0.0011
pn silva 05f (5/4)	3	0.0108	3	0.0009	3	0.001
prio ex (2/3)	1	3584.16	1	3584.16	1	0.0009
prod cons (8/5)	3	0.0235	3	0.001	3	0.001
PUMA loading (6/4)	2	0.4726	2	0.0023	2	0.0009
PUMA unloading (5/3)	2	0.0562	2	0.0012	2	0.0011
PWM extended (49/31)	0	0	0	0	10	41.8946
PWM patterns (19/12)	4	0.012	4	0.0009	4	1.6233

reactor small (9/8)	2	0.0063	2	0.0009	2	0.001
real-life system smarthome simplified (12/7)	5	0.0106	5	0.0009	5	0.0026
reiseg1 (13/9)	5	0.0106	5	0.001	5	0.0012
rejers1 (14/11)	4	0.0065	4	0.0008	4	0.0011
rejers2 (11/10)	2	0.028	2	0.0011	2	0.0009
return books (4/6)	1	0.011	1	0.001	1	0.0009
RHINO loading (6/4)	2	0.0153	2	0.0012	2	0.0012
RHINO unloading (6/4)	2	0.0222	2	0.0011	2	0.001
semaphore (3/4)	1	0.0386	1	0.0011	1	0.001
semaphore2 (9/6)	5	0.0222	5	0.0011	5	0.0011
silva1 (7/7)	2	0.0116	2	0.0011	2	0.001
silva14 (8/7)	3	0.0281	3	0.0011	3	0.0011
silva4v2 (13/12)	4	0.0556	4	0.0014	4	0.0011
silva5 (16/8)	3	0.204	3	0.002	3	0.001
silva7 (13/13)	3	0.728	3	0.0032	3	0.0013
simple production system (9/6)	4	0.0154	4	0.001	4	0.0009

sivaraman1 (10/10)	2	0.2499	2	0.0011	2	0.0011
speedway (10/7)	3	0.0251	3	0.001	3	0.0013
state space16 (16/16)	8	0.0243	8	0.0012	8	0.002
state space32 (32/32))	16	0.0065	16	0.0009	16	0.0027
state space48 (48/48)	24	0.0225	24	0.001	24	0.0073
sub-task of PLT and PMN count (8/5)	3	0.015	3	0.0012	3	0.001
s net copy milling machine subprocess (31/28)	4	0.082	4	0.0022	4	0.001
s net frame manufact quality v1 (17/16)	4	0.0096	4	0.0009	4	0.0011
s net frame manufact quality v2 mod (19/18)	4	0.0151	4	0.001	4	0.0009
tank heating (4/4)	1	0.012	1	0.001	1	0.001
TP5 I (8/6)	3	0.0119	3	0.0009	2	0.0012

traffic lights (6/5)	4	0.0336	4	0.001	3	0.0022
traffic light v1 (8/3)	2	0.0257	2	0.0011	4	0.0011
traffic light v2 (4/3)	3	0.0117	3	0.0009	2	0.001
two traffic lights (7/7))	2	0.0179	2	0.001	3	0.001
vanDerAalst1 (12/14)	2	32.9219	2	32.9219	2	0.001
vanDerAalst3 (14/13)	4	0.0228	4	0.0013	2	0.0012
vanDerAalst4 (18/16)	4	0.3001	4	0.0013	4	0.0018
vanDerAalst5 (23/23)	2	0.0145	2	0.0009	4	0.001
vanDerAalst6 (9/13)	3	0.0118	3	0.0009	2	0.0011
vanDerAalst7 (14/13)	14	0.1861	14	0.001	3	2.9059
voorhoeve1 (13/12)	4	0.1882	4	0.001	4	0.001
voorhoeve2 (14/12)	4	0.4602	4	0.0012	4	0.0009
voorhoeve3 (7/5)	3	0.4328	3	0.0011	4	0.0011
weijters1 (12/14)	2	0.1861	2	0.0012	3	0.0013

well built alfa net copy milling machine subprocess (30/25)	4	2.601	4	0.001	4	0.0011
xie1 (24/26)	3	0.1882	3	0.001	3	0.0011
zuberek1 (30/22)	5	0.4602	5	0.012	5	0.0012
zuberek3 (29/21))	5	0.4328	5	0.011	5	0.0013
zuberek4 (30/21)	6	135148	6	0.012	6	0.0012

Tablica E.2: Tabela przedstawiająca szczegółowe rezultaty badań - selekcja podsieci automatowych

Bibliografia

[1] T. Łuba. *Synteza układów logicznych*. OWPW, 2005.

[2] R.W. Lewis. *Programming industrial control systems using IEC 1131-3*. IEE, London, 1998.

[3] A. Zakrevskij, Yu. Pottosin, and L. Cheremisinova. *Design of Logical Control Devices*. TUT Press, Tallinn, 2009.

[4] I. Grobelna, M. Grobelny, and M. Adamski. Model checking of uml activity diagrams in logic controllers design. In *Proc. of the 9th Int. Conf. on Dependability and Complex Systems DepCoS-RELCOMEX, Springer*, pages 233–242, 2014.

[5] I. Grobelna. *Formal verification of logic controller specification by means of model checking*. University of Zielona Gora Press, 2013.

[6] T. Kropf. *Introduction to Formal Hardware Verification: Methods and Tools for Designing Correct Circuits and Systems. 1 edn.* Springer-Verlag New York, Inc., 1999.

[7] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. New York, USA: Cambridge University Press, 2004.

[8] D. Kania and A. Milik. Logic synthesis based on decomposition for CPLDs. *Microprocessors and Microsystems - Embedded Hardware Design*, 34(1):25–38, 2010.

[9] F. Hsieh. Analysis of flexible assembly processes based on structural decomposition of Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics-Part A*, 37(5):792–803, 2007.

[10] R. Wiśniewski. *Prototyping of Concurrent Control Systems Implemented in FPGA devices*. Berlin-Heidelberg: Springer International Publishing, 2017.

[11] D. He, B. Strege, H Tolle, and A. Kusiak. Decomposition in automatic generation of Petri nets for manufacturing system control and scheduling. *Int. J. of Prod. Research*, 38(6):1437–1457, 2000.

[12] Z. Li Y. Chen and A. Al-Ahmari. Nonpure Petri Net Supervisors for Optimal Deadlock Control of Flexible Manufacturing Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(2):252–265, 2013.

[13] M. Blanchard. *Comprendre, maitriser et appliquer le Grafcet*. Automatisation Production, 1979.

[14] O. Coudert. Two-level logic minimization: An overview. *Integration*, Vol. 17(no 2):97–140, 1994.

[15] J. Paulo L. Gomes and A. Costa. *Structuring Mechanisms in Petri Net Models: From specification to FPGA based implementations*, pages 153–166. Springer, 2004.

[16] L. Gomes and A. Costa. From use cases to system implementation: Statechart based co-design. In *Proc. 1st ACM & IEEE Conf. Formal Methods and Prog. Models for Codesign*, pages 24–33, Mt. Saint-Michel, France, 2003. IEEE Comp. Soc. Press.

[17] L. Gomes, J. P. Barros, A. Costa, and R. Nunes. The input-output place-transition Petri net class and associated tools. In *2007 5th IEEE International Conference on Industrial Informatics*, volume 1, pages 509–514, June 2007.

[18] A. Costa and L. Gomes. Petri net partitioning using net splitting operation. In *2009 7th IEEE International Conference on Industrial Informatics*, pages 204–209, June 2009.

[19] Z. Banaszak, J. Kuś, and K. Piwakowski. Petri nets: modeling, control and discrete systems synthesis. *College of Engineering Publisher*, 1993. in Polish.

[20] M. Kubale, P. Obszarski, and K. Piwakowski. Hypergraphs coloring. *Scientific Papers of Silesian University of Technology*, 2006. in Polish.

[21] Saul-Alonso Nuno-Sanchez, Antonio Ramírez-Treviño, and Javier Ruiz-León. Structural sequence detectability in free choice interpreted Petri nets. *IEEE Transactions on Automatic Control*, 61(1):198–203, 2016.

[22] M. Wiśniewska. *Application of Hypergraphs in Decomposition of Discrete Systems*, volume 23 of *LNCCS*. University of Zielona Góra Press, Zielona Góra, 2012.

[23] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of IEEE*, 77:548–580, 1989.

[24] G. Callou, P. Maciel, D. Tutsch, J. Araújo, J. Ferreira, and R. Souza. A Petri net-based approach to the quantification of data center dependability. In *Petri Nets-Manufacturing and Comp. Science*, pages 213–336. InTech, 2012.

[25] M.C. Zhou and N.Q. Wu. *System modeling and control with resource-oriented Petri nets*, volume 35. CRC Press, 2009.

[26] N. Ran, H. Su, A. Giua, and C. Seatzu. Codiagnosability analysis of bounded Petri nets. *IEEE Transactions on Automatic Control*, 63(4):1192–1199, April 2018.

[27] N. Ran, S. Wang, and W. Wu. Event feedback supervision for a class of Petri nets with unobservable transitions. *IEEE Access*, 6:6920–6926, 2018.

[28] M. Szpyrka. *Petri nets in modeling and anaylsis of concurrent systems (in Polish)*. WNT, Warsaw, 2008.

[29] W. Chang, C. Tseng, and W. Chou. Petri net-based analysis on object assignment in distributed object-oriented systems. *Journal of Systems Architecture*, 44(12):955 – 970, 1998.

[30] E. Best, R. Devillers, and M. Koutny. *Petri net algebra*. Springer Science & Business Media, 2013.

[31] K. Jensen and G. Rozenberg. *High-level Petri Nets: Theory and Application*. Springer, 2012.

[32] Q. Zhu, M. Zhou, Y. Qiao, and N. Wu. Petri net modeling and scheduling of a close-down process for time-constrained single-arm cluster tools. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(3):389–400, March 2018.

[33] F. Basile, P. Chiacchio, and J. Coppola. Identification of time petri net models. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(9):2586–2600, September 2017.

[34] R. Yang, Z. Ding, M. Pan, C. Jiang, and M. Zhou. Liveness analysis of -independent petri nets based on new modified reachability trees. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–12, 2016.

[35] M. Hack. *Analysis of Production Schemata by Petri Nets.* MIT Project MAC TR-94, 1972. Corrections: Project MAC, Computation Structures Note 17 (1974).

[36] P. Starke. *Petri-Netze: Grundlagen, Anwendung, Theorie*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1980.

[37] J. Tkacz and M. Adamski. Structured mapping of petri net states and events for FPGA implementations. *International Journal of Electronics and Telecommunications*, Vol. 59(no. 4):331–339, 2013.

[38] D. Gilbert M. Heiner and R. Donaldson. Petri nets for systems and synthetic biology. In Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro, editors,

Formal Methods for Computational Systems Biology, volume 5016 of *Lecture Notes in Computer Science*, pages 215–264. Springer Berlin Heidelberg, 2008.

[39] P. Buchholz and P. Kemper. Hierarchical reachability graph generation for petri nets. *Formal Methods in System Design*, 21(3):281–315, 2002.

[40] A. Węgrzyn and M. Węgrzyn. Petri net-based specification, analysis and synthesis of logic controllers. In *Proceedings of the IEEE International Symposium on Industrial Electronics - ISIE 2000*, volume Vol. 1, pages 20–26, Udla, Mexico, 2000. Universidad de las Americas-Puebla, Piscataway.

[41] B. Steinbach and A. Zakrevskij. Parallel automaton basic model, properties and diagnostics. In *Boolean Problems, Proceedings of the 4th International Workshops on Boolean Problems*, pages 151–158, Freiberg, 2000. Freiberg University of Mining and Technology.

[42] Z. Li and Z. MengChu. Control of elementary and dependent siphons in petri nets and their application. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(1):133–148, Jan 2008.

[43] J. Kluska L. Gniewek. Hardware implementation of fuzzy Petri net as a controller. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(3):1315–1324, June 2004.

[44] M. Zhou Z. Li. Correction to ęlementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems". *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 34(2):289–289, March 2004.

[45] Z. Li H. Hesuan, M. Zhou. Liveness and ratio-enforcing supervision of automated manufacturing systems using petri nets. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 42(2):392–403, March 2012.

[46] Z. Li H. Hesuan, M. Zhou. Supervisor design to enforce production ratio and absence of deadlock in automated manufacturing systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(2):201–212, March 2011.

[47] I. Grobelna, R. Wiśniewski, M. Grobelny, and M. Wiśniewska. Design and verification of real-life processes with application of Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(11):2856–2869, Nov 2017.

[48] Z. Li, M. Zhou, and N. Wu. A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):173–188, March 2008.

[49] R. Wiśniewski. Dynamic partial reconfiguration of concurrent control systems specified by Petri nets and implemented in Xilinx FPGA devices. *IEEE Access*, 6:32376–32391, 2018.

[50] J. Ye, M. Zhou, Z. Li, and A. Al-Ahmari. Structural decomposition and decentralized control of Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[51] C. Girault and R. Valk. *Petri nets for systems engineering: a guide to modeling, verification, and applications*. Springer Science & Business Media, 2013.

[52] T. Zhang C. Lin. *An Algorithm for Computing S-invariants for High Level Petri Nets*. Fachberichte Informatik. Univ., 1991.

[53] M. Adamski R. Dylewski and J. Jablonski. Application of linear programming for analysis of petri net. In *Proc. of KNWS' 11*, pages 267–273, Karpacz, Poland, 2011.

[54] F. Basile, R. Cordone, and L. Piroddi. A branch and bound approach for the design of decentralized supervisors in Petri net models. *Automatica*, 52:322 – 333, 2015.

[55] L. Cortés, P. Eles, and Z. Peng. Modeling and formal verification of embedded systems based on a petri net representation. *Journal of Systems Architecture*, 49(12):571 – 598, 2003. Synthesis and Verification.

[56] L. Ni, J. Zhang, and J. Yu. Priced timed Petri nets based resource allocation strategy for fog computing. In *2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)*, pages 39–44, Oct 2016.

[57] T. Tapia-Flores, E. Lopez-Mellado, A. Estrada-Vargas, and J. Lesage. Discovering Petri Net Models of Discrete-Event Processes by Computing T-Invariants. *IEEE Transactions on Automation Science and Engineering*, 15(3):992–1003, jul 2018.

[58] R. Wiśniewski, A. Karatkevich, M. Adamski, A. Costa, and L. Gomes. Prototyping of concurrent control systems with application of Petri nets and comparability graphs. *IEEE Transactions on Control Systems Technology*, pages 1–12, 2017.

[59] M. Cabasino, A. Giua, A. Paoli, and C. Seatzu. Decentralized diagnosis of discrete-event systems using labeled petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(6):1477–1485, November 2013.

[60] H. Leroux, D. Andreu, and K. Godary-Dejean. Handling exceptions in petri net-based digital architecture: From formalism to implementation on FPGAs. *IEEE Transactions on Industrial Informatics*, 11(4):897–906, August 2015.

[61] Z. Ma, Y. Tong, Z. Li, and A. Giua. Basis marking representation of petri net reachability spaces and its application to the reachability problem. *IEEE Transactions on Automatic Control*, 62(3):1078–1093, March 2017.

[62] J. Ye, M. Zhou, Z. Li, and A. Al-Ahmari. Structural decomposition and decentralized control of petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(8):1360–1369, August 2018.

[63] F. L. Tiplea and I. Leahu. The reversible released form of petri nets and its applications to soundness of workflow nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(2):303–312, February 2016.

[64] A. Karatkevich. *Dynamic analysis of Petri net-based discrete systems*. Berlin: Springer, 2007.

[65] R. Wisniewski, A. Karatkevich, M. Adamski, A. Costa, and L. Gomes. Prototyping of concurrent control systems with application of petri nets and comparability graphs. *IEEE Transactions on Control Systems Technology*, 26(2):575–586, March 2018.

[66] R. Wisniewski, M. Wisniewska, and M. Jarnut. C-exact hypergraphs in concurrency and sequentiality analyses of cyber-physical systems specified by safe petri nets. *IEEE Access*, 7:13510–13522, 2019.

[67] S. Cayir and M. Ucer. An Algorithm to Compute a Basis of Petri Net Invariants.

[68] C. Amer-Yahia. A method based on eigenvalues for calculating minimal invariants in Petri nets.

[69] D. Marinescu, M. Beaven, and R. Stansifer. A Parallel Algorithm for Computing Invariants of Petri Net Models. *Proceedings of Petri Nets and Performance Models, IEEE Press.*, 1991.

[70] N. Kortbeek and R.J. Boucherie. P- and T-invariant characterization of product form and decomposition in stochastic Petri nets. Technical report, Department of Applied Mathematics, University of Twente, 3 2011. ISSN 1874-4850.

[71] M. Silva, E. Teruel, and J. Colom. Linear algebraic and linear programming techniques for the analysis of place/transition net systems. *Lectures on Petri Nets I: Basic Models*, 1491:309–373, 1998.

[72] J. Desel, K. Neuendorf, and M.-D. Radola. Proving nonreachability by modulo-invariants. *Theor. Comput. Sci.*, 153(1&2):49–64, 1996.

[73] A. Taguchi, S. Taoka, and T. Watanabe. An algorithm gmst for extracting minimal siphon-traps and its application to efficient computation of Petri net invariants.

In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volume 3, pages III–172–III–175 vol.3, May 2003.

[74] Ł. Stefanowicz. Conception of discrete systems decomposition algorithm using p-invariants and hypergraphs. In Ryszard S. Romaniuk, editor, *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2016*. SPIE, sep 2016.

[75] D. Zaitsev. Decomposition-based Calculation of Petri Net Invariants. *Proc. of the 25-th Int. Conf. on Application and Theory of Petri nets*, pages 79–83, June 21-25 2004.

[76] M. Silva J. Martinez. A simple and fast algorithm to obtain all invariants of a generalized petri net. In *Selected Papers from the European Workshop on App. and Theory of Petri Nets*, pages 301–310, London, UK, 1982. Springer-Verlag.

[77] A. Karatkevich. Petri nets in design of control algorithms. In *Design of Reconfigurable Logic Controllers*, pages 1–14. Springer International Publishing, December 2015.

[78] B. Steinbach and Ch. Postchoff. Improvements of the Construction of Exact Minimal Covers of Boolean Functions. *Lecture Notes in Computer Science*, pages 272–279, 2012.

[79] M. Lehmann, T.L. Mai, B. Wollschlaeger, and K. Kabitzsch. Design approach for component-based automation systems using exact cover. *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, pages 1–8, 2014.

[80] T. Eiter. Exact Transversal Hypergraphs and Application to Boolean μ-functions. *Journal of Symbolic Computation*, Vol.17(3):215–225, 1994.

[81] D.E. Knuth. *The Art of Computer Programming*, volume Volume 1 (3rd Ed.): Fundamental Algorithms. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1997.

[82] G. Andrzejewski. *Programowy model interpretowanej sieci Petriego dla potrzeb projektowania mikrosystemów cyfrowych*. PhD thesis, Uniwersytet Zielonogórski, 2003.

[83] R. Wisniewski, I. Grobelna, and A. Karatkevich. Determinism in cyber-physical systems specified by interpreted Petri Nets. *Sensors*, 20(19):5565, September 2020.

[84] M. Golumbic. Algorithmic graph theory and perfect graphs., 1980.

[85] R. Shamir. Advanced topics in graph algorithms, 1994.

[86] A. Kovalyov and J. Esparza. A polynomial algorithm to compute the concurrency relation of free-choice signal transition graphs. In *In Proc. of the International Workshop WODES*, pages 1–6, 1995.

[87] C. Berge. *Graphs and hypergraphs*. North-Holland Pub. Co.L American Elsevier Pub. Co., Amsterdam, New York, 1973.

[88] R. L. Rudell. *Logic Synthesis for VLSI Design*. PhD thesis, EECS Department, University of California, Berkeley, CA, USA, 1989.

[89] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[90] A. Zakrevskij, Yu. Pottosin, and L. Cheremisinova. *Combinatorial Algorithms of Discrete Mathematics*. TUT Press, Tallinn, 2008.

[91] C.H. Papadimitriou. *Computational Complexity*. MA: Addison-Wesley, Reading, 1994.

[92] G. DeMicheli. *Synthesis and Optimization of Digital Circuits*. PhD thesis, McGraw-Hill Higher Education, 1994.

[93] P. Sapiecha. *Algorytmy syntezy funkcji i relacji boolowskich w aspekcie metod reprezentacji i kompresji danych*. PhD thesis, Warszawa, 1999.

[94] J.S. Kowalik M.M. Sysło, N. Deo. *Algorytmy optymalizacji dyskretnej z programami w jezyku Pascal*. Wydawnictwo Naukowe PWN, 1999.

[95] T. Stutzle H.H. Hoos. Local search algorithms for sat: An empirical evaluation. *J. Autom. Reasoning, Vol. 24, No. 4*, pages 421–481, 2000.

[96] D. Knuth. Dancing links. *Millennial Perspectives in Computer Science*, 2000.

[97] C.A. Papachristou. A scheme for implementing microprogram addressing with programmable logic arrays. *Digital Processes*, 5(3-4):235–256, 1979.

[98] L. Stefanowicz and M. Adamski. Aspects of selection of SM components with the application of the theory of hypergraphs. In *2014 7th International Conference on Human System Interactions (HSI)*. IEEE, jun 2014.

[99] Ł. Stefanowicz and I. Grobelna. Application of modified Martinez-Silva algorithm in determination of net cover. In *AIP Conference Proceedings 1790*. AIP Publishing, 2016.

[100] R. Wiśniewski, Ł. Stefanowicz, M.Wiśniewska, and D. Kur. Exact cover of states in the discrete state-space system. In *AIP Conference Proceedings 1702*. AIP Publishing LLC, 2015.

[101] Ł. Stefanowicz and P. Mróz. State Machine Components selection based on minimal transversals. In *AIP Conference Proceedings 1702*. AIP Publishing LLC, 2015.

[102] Ł. Stefanowicz, M. Adamski, and R. Wisniewski. Application of an exact transversal hypergraph in selection of SM-components. In *IFIP Advances in Information and Communication Technology*, pages 250–257. Springer Berlin Heidelberg, 2013.

[103] Ł. Stefanowicz, M. Adamski, R. Wiśniewski, and J. Lipiński. Application of hypergraphs to SMCs selection. In *Technological Innovation for Collective Awareness Systems*, pages 249–256. Springer Berlin Heidelberg, 2014.

[104] R. Wiśniewski, Ł. Stefanowicz, A. Bukowiec, and J. Lipiński. Theoretical aspects of petri nets decomposition based on invariants and hypergraphs. In *Lecture Notes in Electrical Engineering*, pages 371–376. Springer Berlin Heidelberg, 2014.

[105] Ł. Stefanowicz. Zastosowanie hipergrafu transwersal dokładnych w selekcji podsieci automatowych sieci Petriego. In *Conference Archives PTETIS*, volume 31, pages 37–40, 2012.

[106] R. Wiśniewski, Ł. Stefanowicz, G. Bazydło, and M. Węgrzyn. Application of hypergraphs in the prime implicants selection process. *IFAC-PapersOnLine*, 48(4):302–305, 2015.

[107] A. Karatkevich and L. Stefanowicz. Minimization of SM-covers of petri net specifications of control systems. In *2019 MIXDES - 26th International Conference Mixed Design of Integrated Circuits and Systems*. IEEE, June 2019.

[108] Ł. Stefanowicz, R. Wiśniewski, and A. Karatkevich. Selection of state machine components for a petri net based on the computation of an exact transversal. AIP Publishing, 2018.

[109] G. Bazydlo, M. Adamski, and L. Stefanowicz. Translation UML diagrams into verilog. In *2014 7th International Conference on Human System Interactions (HSI)*. IEEE, jun 2014.

[110] R. Wiśniewski, I. Grobelna, and Ł. Stefanowicz. Partial reconfiguration of concurrent logic controllers implemented in FPGA devices. Melville: American Institute of Physics, 2016.

[111] I. Grobelna, Mi. Grobelny, and Ł. Stefanowicz. A rule-based approach to model checking of UML state machines. AIP Publishing, 2016.

[112] R. Wisniewski, M. Wojnakowski, and Ł. Stefanowicz. Safety analysis of Petri nets based on the SM-cover computed with the linear algebra technique. AIP Publishing, 2018.