Kamil MIELCAREK
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

WYBRANE ASPEKTY ZASTOSOWAŃ GRAFÓW DOSKONAŁYCH W AUTOMATYCZNYM PRZETWARZANIU UKŁADÓW CYFROWYCH

mgr inż. Kamil MIELCAREK

Kamil Mielcarek jest pracownikiem Instytutu Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zajmuje się zastosowaniem grafów doskonałych w procesach automatycznej syntezy i optymalizacji układów cyfrowych. Ponadto intensywnie zajmuje się zagadnieniami serwerowych systemów sieciowych opartych na systemie OpenBSD jak i ich bezpieczeństwem, oprogramowaniem oraz rozwiązaniami sieciowymi. Prowadzi prace związane z technologiami przeznaczonymi do budowy rozwiązań stron www. Zajmował się też systemami czasu rzeczywistego opartymi o systemy BSD i Linux.

e-mail: K.Mielcarek@iie.uz.zgora.pl

Streszczenie

Złożoność układów przerastająca możliwości ludzi wymusza zastosowanie specjalizowanych systemów komputerowych do realizacji rosnącego zapotrzebowania. Najczęstsze realizacje wykorzystują grafy, gdzie w ogólności występują problemy NP-zupełne. Obecnie stosowane algorytmy heurystyczne dające przybliżone wyniki można zastąpić algorytmami dającymi wyniki dokładne w czasie wielomianowym. Warunkiem jest zastosowanie grafów doskonałych w procesie analizy i syntezy układów. Obserwuje się, że większość grafów powstających w tych procesach należy go podklasy grafów doskonałych. Istnieje też potencjalna możliwość użycia grafów doskonałych do automatycznej weryfikacji układów.
Potencjalne miejsca zastosowania grafów doskonałych i uzyskanie korzyści płynących ze stosowania algorytmów o zaniżonej złożoności uzależnione są od stopnia wykorzystania grafów przynależących do podklasy grafów doskonałych. Praktycznie każde miejsce gdzie występują problemy takie jak np. kolorowanie grafów czy pokrycia oznaczają potencjalny zysk prędkości uzyskania wyników optymalnych.

Abstract

Potential strength of perfect graphs algorithms - especially algorithms witch use perfect graphs - for automated synthesis of digital circuits is huge. Typically known problems are NP-complete, however using perfect graphs, complexity is decreasing to polynomial. Studies in this matter show that plenty of dependencies describing real-life sequential circuits can be described using perfect graphs. For example analysis of compatibility graph derived from Petri Net or SFC description of digital automata shows that they belong to perfect graphs subset. In this case it is subset of interval graphs which shows similarity to graphical representation of simple Petri Net/SFC diagrams, reduced using macro-places. Interval graphs could be used to achieve decomposition to component FSMs. Perfect graphs also could be used to verify correctness whole project.

Słowa kluczowe: grafy doskonałe, algorytmy rozpoznawania grafów doskonałych, teoria grafów, automatyczna synteza sterowników.

Keywords: prefect graphs, prefect graphs recognizing algorithms, graph theory, automatic synthesis of digital controllers.

1. Wstęp

Gwałtowny rozwój elektroniki a w szczególności obwodów cyfrowych spowodował konieczność zastąpienia ręcznego opracowywania logiki zamkniętej wewnątrz układów scalonych. Dokonano tego przenosząc ciężar przetwarzania i sprawdzania poprawności na specjalizowane systemy komputerowe. Złożoność opisywanych układów jest na tyle duża, że potrzebny był kompromis między dokładnością działania i czasem wykonywania operacji. Do odwzorowania układu w pamięci komputera konieczne było opracowanie wewnętrznego modelu, opisującego zależności występujące w specyfikowanym układzie jak i opracować algorytmy potrafiące na nim operować. Często wybór padał na grafy, będące doskonałym matematycznym narzędziem odwzorowania, umożliwiającym opisywanie równoległej pracy układów. Problem pojawił się w algorytmach operujących na grafach - złożoność obliczeniowa. Potrzeby np. podziału układu w sposób umożliwiający zmieszczenie go do mniejszych rzeczywistych modułów urządzenia czy tez minimalizacji jak i weryfikacja opierała się głównie na operacjach, które dla grafów w ogólności należą do klasy problemów NP-zupełnych. Osiągnięcie dokładnych wyników w tym przypadku wiąże się z nieakceptowanie długim czasem obliczeń, co w czasach używania np. narzędzi do szybkiego prototypowania jest niedopuszczalne. W praktyce zastosowanie znalazły algorytmy heurystyczne, które choć szybsze dają wyniki zbliżone do optymalnych, czasem skrajnie dalekie.

Próbą rozwiązania powyższego problemu jest zastosowanie grafów doskonałych jako wewnętrznego modelu pośredniego do reprezentacji, przetwarzania jak i weryfikacji układów cyfrowych. Przemawia za tym zachowanie siły opisu rzeczywistości aparatu matematycznego, jakim są grafy i zredukowana złożoność obliczeniowa, która dla algorytmów operujących na grafach doskonałych jest rzędu wielomianowej.

Aby moc zastosować grafy doskonałe konieczne będzie wcześniejsze sprawdzenie czy powstający graf należy do podklasy grafów doskonałych ewentualnie poddając go przekształceniu, które zachowując informacje będzie nadal grafem doskonałym. Rozpoznawanie i przekształcanie jest bardo często ze sobą powiązane i jest procesem wstępnym do kolejnych kroków. Zazębianie się czynności redukuje narzut potrzebny do sprawdzenia przygotowanie reprezentacji w postaci grafów doskonałych.

Brak jednoznacznych dowodów na możliwość opisana układów wyłącznie w postaci grafów doskonałych zmusza do opracowania alternatywnej ścieżki postępowania w stosunku do obecnie używanej. Jednak prace prowadzone w tym temacie wskazują, że znacząca większość poprawnie zaprojektowanych układów daje się opisać grafami doskonałymi. Dlatego można postawić tezę, że poprawnie zaprojektowany układ jest opisany grafami należącymi do podklas grafów doskonałych. Jeśli jednak nie jest, to istnieje podejrzenie, ze układ może zawierać błędy. Tę zależność można by było włączyć jako element automatycznej weryfikacji układów scalonych.

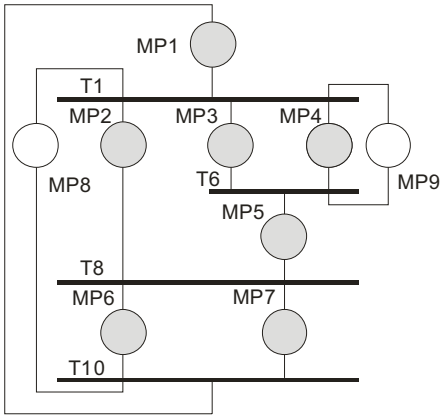
Wspomniana wcześniej konieczność podziału układu na mniejsze części wynika nie tylko z potrzeb podziału na kilka fizycznych elementów, ale z uwagi na możliwości sprzętu (szczególnie elementów wykonawczych) i ograniczeń modelu. Nie wprowadza ona jednak znaczącego problemu w zachowaniu przynależności do podklasy grafów doskonałych. Podgrafy grafu doskonałego należą do klasy grafów doskonałych z definicji, natomiast dokonując kompozycji z mniejszych grafów można uzyskać graf doskonały, jeśli grafy użyte do kompozycji będą należały do wybranej podklasy grafów doskonałych. Dzięki temu możliwe jest tworzenie i stosowanie bibliotek komponentów, wcześniej poddanych procesom weryfikacji przynależności do podklas grafów doskonałych.

Korzystając z wymienionych własności można podjąć próbę analizy przykładowej sieci Petriego opisującej sterownik logiczny.

Informacje przedstawiane w innych publikacjach autora oraz implementowane algorytmy i metody postępowania pozwalają na próbę analizy grafu zgodności układu w skończonych automatach stanów.

2. Zależności wewnętrzne

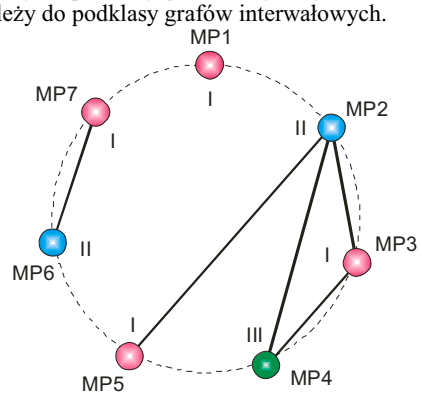
Niech dany jest opis układu w postaci grafu SFC. Za przykład posłuży sieć pochodząca z załącznika standardu IEC 1131-3 [14]. Oryginalny sposób interpretacji układowej sterownika rekonfigurowalnego przedstawiono w pracach [18] i [19]. Poniżej przedstawiono opis sterowania mieszalnikiem w postaci makro-sieci Petriego.



Rys. 1. Makrosieć Petriego odpowiadająca modelowi SFC [14]
Fig. 1. Petrie macro-net eqivalent to SFC model from [14]

Analiza dyskretna przestrzeni stanów lokalnych automatu współbieżnego, modelującego program sterownika logicznego, generuje grafy, które podczas analizy okazały się być grafami doskonałymi. Po wyznaczeniu relacji współbieżności między miejscami makrosieci, np. analizując graf znakowań makrosieci, uzyskano graf zgodności (współbieżności) między miejscami.

Na podstawie analizy stwierdzono, że graf ten jest
- złożony z trzech składowych spójności,
- każda z nich odpowiada składowej automatowej,
- składowe są grafami doskonałymi,
- przynależy do podklasy grafów trójkątnych,
- przynależy do podklasy grafów interwałowych.



Rys. 2. Graf (już pokolorowany) zgodności modelu z rys. 2
Fig. 2. Already colored compatibility graph of the model from fig. 2

Przynależność do podklasy grafów doskonałych pozwala na zastosowanie algorytmu kolorowania w z użyciem algorytmu Left Edge operującym na grafach interwałowych. Algorytm operuje na reprezentacji przedziałowej, która choć opisuje te same zależności, jest łatwiejsza w interpretacji. Szczególnie dla człowieka i jego sposobu odbioru ilustracji. Jednak wpływa on też na neutralizująco na komplikacje samego.

```
LEFT_EDGE (I)
{
    Sort elements of I in a list L in ascending order of l_i;
    c = 0;
    while (some interval has not been colored) do
    {
        S = Ø;
        r = 0;
        while ( exists s ϵ L such that l_s > r) do
        {
            s = First element in the list L with l_s > r;
            S = S ∪ {s};
            r = r_s;
            Delete s from L;
        }
        c = c + 1;
        Label elements of S with color c;
    }
}
```

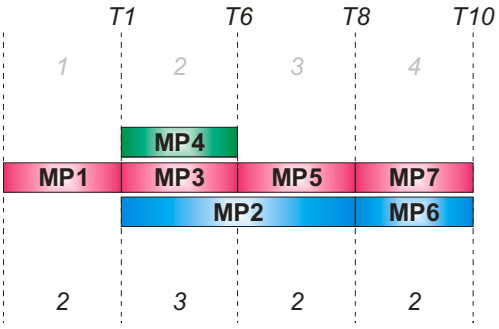
Rys. 3. Algorytm kolorowania Left Edge; [7]
Fig. 3. Left Edge - interval graph coloring algorithm; [7]

Wersja niniejszego algorytmu została zamieszczona w bibliotece procedur grafowych, która ma być podstawą dalszych badań autora.

W rezultacie kolorowania otrzyma się trzy składowe podsieci automatowe (patrz rys. 2):

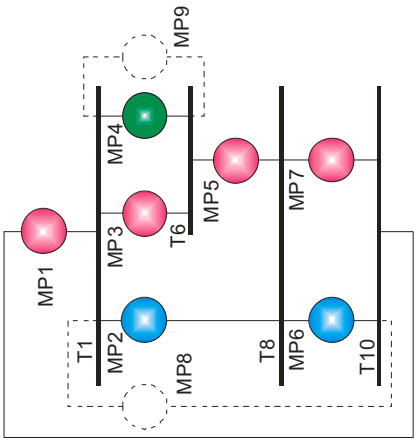
I: $\{MP_1, MP_3, MP_5, MP_7\} = \{P_1, P_5, P_6, P_7, P_9, P_{10}, P_{11}, P_{12}, P_{13}\}$
II: $\{MP_2, MP_6\} = \{P_2, P_3, P_4, P_{14}, P_{15}\}$
III: $\{MP_4\} = \{P_8, P_{16}\}$

Poniżej przedstawiono przedziałową reprezentację grafu zgodności po procesie kolorowania wierzchołków. U dołu każdego z przedziałów opisano dodatkowo rozmiar kliki mówiącego o minimalnej ilości bloków wykonawczych potrzebnych do wykonania równoległych procesów w sekwencji opisanej siecią Petriego.



Rys. 4. Graf zgodności jako graf interwałowy w zapisie przedziałowym (u dołu wielkości poszczególnych klik)
Fig. 4. Compatibility graph as a interval graph given as intervals (with clique size below)

Należy zauważyć, jak bardzo graf interwałowy powstały w tym przekształceniu odzwierciedla badaną strukturę makrosieci. Najlepiej można to zaobserwować, obracając makrosieć o 90º w lewo. Wielkość największej kliki maksymalnej odpowiada tu minimalnej ilości bloków wykonawczych, jakie były by w stanie obsłużyć poprawnie wykonywanie takiego układu.

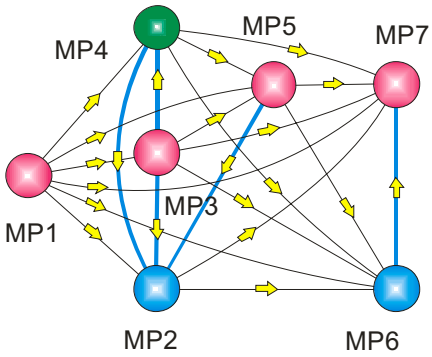


Rys. 5. Wizualizacja podobieństwa zapisu sieci petriego z reprezentacja przedziałową grafu zgodności
Fig. 5. Visualization of similarities between petri macro-net and interval version of compatibility graph

Ciekawa zależność występująca w postaci zbieżności występowania największych klik maksymalnych w grafie oraz dekompozycja metodą pokrycia grafu klikami, prowadzi na inna gałąź teorii grafów. Widać, że opierając swoje działania można dobrze opisać zależności przy użyciu hipergrafów, jednak te nie przynależą (a przynajmniej nie musza przynależeć) do podklas grafów doskonałych.

Oczywiście tworzące się grafy nie są wyłącznie ograniczone do grafów zgodności. Warto np. podjąć próbę przedstawienia zależności danych jak i zależności między wykonywaniem zadań. Konstrukcja grafu przepływu danych i sterowania, która nawiązuje do interwałowego przedstawienia grafu zgodności. Graf przepływu danych i sterowania powstaje przez operację domknięcia tranzytywnego grafu (dodane krawędzie koloru niebieskiego) powstałe-

go z grafu przepływu danych łączącego wierzchołki według następstwa przedziałów interwałowej reprezentacji grafu zgodności.



Rys. 6. Graf przepływu danych i sterowania powstały po domknięciu symetrycznym grafu zgodności z użyciem wierzchołków klik z przedziałów grafu interwałowego
Fig. 6. Control and Data Flow Graph derived by transitive closure of compatibility graph using vertices of clique in different interval graph sections.

Jak łatwo zauważyć dodane krawędzie nie są niczym innym jak grafem zgodności powstałym po analizie makrosieci Petriego. Jednocześnie wyznaczenie w/w krawędzi sprowadza się do odczytania, które największe kliki maksymalne w przedziałowej reprezentacji grafu zgodności zachodzą na siebie. W tym przypadku występują 4 kliki: $\{MP_1\}$, $\{MP_2, MP_3, MP_4\}$, $\{MP_2, MP_5\}$, $\{MP_6, MP_7\}$ dostępne jako wierzchołki leżące między przedziałami wyznaczonymi przez „pozostałości tranzycji". Jednocześnie wiadomo ze taki graf przez konstrukcję będzie grafem porównywalności. Na rys. 6 przedstawiono przykładową orientację tranzytywną grafu oraz wyróżniono krawędzie domknięcia tranzytywnego zgodne z założeniami grafu porównywalności:

$$F \cap F^{-1} = \varnothing \qquad (1)$$

$$F + F^{-1} = E \qquad (2)$$

$$F^2 \subseteq F, \text{ gdzie } F^2 = \{(a,c) \mid (a,b), (b,c) \in F\} \text{ dla pewnego wierzchołka b} \qquad (3)$$

3. Wnioski

Współczesne metody projektowania układów bardzo szeroko stosują formalne modele pośrednie, do wewnętrznej i zewnętrznej reprezentacji struktur układów logicznych. Powszechnie stosowane są rozwiązania na bazie grafów czy sieci Petriego. Analiza większej liczby sieci Petriego, opisującej rzeczywiste układy sterowania podsuwa przypuszczenie, że relacja uporządkowania przestrzeni stanów lokalnych takiej sieci jest opisana grafem doskonałym. Wniosek ten potwierdza pośrednio [6], w której przeprowadzono znaczną liczbę testów na grafach opisujących rzeczywiste układy cyfrowe. Duże prawdopodobieństwo całkowitej przynależności grafów zgodności do podklasy grafów doskonałych wskazuję i tu możliwość zastosowania tej drogi optymalizacji rozwiązań. Wyniki prac eksperymentalnych w tej dziedzinie wskazują, że znacząca większość rzeczywistych układów cyfrowych opisanych grafami należącymi do klasy grafów doskonałych daje się właściwie pokolorować w prosty sposób w czasie wielomianowym. Przegląd literatury wskazuje, że choć grafy doskonałe i ich specyficzne właściwości są znane to jednak brakuje ich szerokiego zastosowania. Brak opracowań wpływa na pomijanie tej części teorii grafów, mimo że fakt ich istnienia jest wymieniany w książkach związanych z syntezą układów logicznych.

Istnieją różne algorytmy mniej i bardziej złożone pozwalające na rozpoznanie przynależności grafów do podklas grafów doskonałych, pozwalających jednocześnie przeprowadzić wstępne działania. Jednak da się znaleźć algorytmy będące pewnym novum w tej dziedzinie. Są nimi metody rozpoznawania grafów doskonałych przez wyszukiwanie nieparzystych dziur. Istniejące rozwiązania sprowadzają problem do dekompozycji grafu na mniejsze części gdzie dokonywane są sprawdzenia własności, które są wykonywalne w czasie wielomianowym.

Algorytmy rozpoznawania przynależności grafów do podklasy grafów trójkątnych, porównywalności i interwałowych zostały zrealizowane praktycznie w formie biblioteki procedur z wykorzystaniem języka ANSI C. Jednocześnie trwają prace nad implementacją algorytmów wielomianowych rozpoznawania grafów przez wyszukiwanie dziur nieparzystych [5], [16]. Planowane jest przeniesienie kodu biblioteki do systemów unixowych z rodziny BSD W tym przypadku do systemu OpenBSD, który pozbawiony jest ograniczeń natury prawnej związanej z późniejszym użyciem komercyjnym.

4. Literatura

[1] Conforti M., Cornuejols G., Kapoor A., Vuškovič K., „Even-hole-free-graphs, Part II: Recognition algorithm", Journal of Graph Theory 40 (2002) 238-266
[2] Cornuéjols G., „The Strong Perfect Graph Conjecture"
[3] Cornuéjols G., „The Strong Perfect Graph Theorem"
[4] Cornuéjols G., Cunnigham W.H., „Compositions for perfect graphs", Discrete Mathematics 55 (1985) 245-254
[5] Cornuéjols G., Xinming L., Vuškovič K., "A polynomial algorithm for recognizing perfect graphs", Proceeding 44th Annual IEEE Symposium, 2003
[6] Coudert O., „Exact Coloring of Real-Life Graphs is Easy", Synopsys, Inc. 700 East Middlefield Rd., Moutain View, CA 94043, Proc. of the 34th Design Automation Conference DAC,, Anaheim, CA, USA, June 1997, pp. 121-126
[7] Giovani De Micheli: „Synteza i optymalizacja układów cyfrowych", Wydawnictwa Naukowo-Techniczne, Warszawa 1998
[8] Golumbic M.C., „Algorithmic Graph Theory and Perfect Graphs", Courant Institute of Mathematical Science, New York University, Academic Press 1980
[9] Leuker G. S., "Interval graph algorithms. Ph. D. thesis", Princeton University
[10] Lovasz L., „A Characterization of Perfect Graphs", Journal of Combinatorial Theory (1972)
[11] Mielcarek K, „Grafy doskonałe jako alternatywa dla automatycznej syntezy i optymalizacji układów cyfrowych", Metody i systemy komputerowe w badaniach naukowych i projektowaniu inżynierskim : IV Krajowa Konferencja. Kraków, Polska, 2003, Oprogramowanie Naukowo-Techniczne, 2003
[12] Mielcarek Kamil, " Rozpoznawanie grafów doskonałych na potrzeby automatycznej syntezy układów cyfrowych", KNWS'06, Pomiary Automatyka Kontrola 6bis/2006
[13] Nikolopoulosy S. D., Paliosz L., "Hole and Antihole Detection in Graphs"
[14] International Electrotechnical Commission, "Programable controllers. Part 3 – Programing Languages IEC 1131-3", Genewa, 1993
[15] Chudnovsky M., Cornuéjols G., Xinming L., Seymour P., Vuškovič K., "Cleaning for Bergeness", 2002
[16] Chudnovsky M., Cornuéjols G., Xinming L., Seymour P., Vuškovič K., „ Recognizing Berge Graphs", 2002
[17] Cornuéjols G., Cunnigham W.H., "Compositions for perfect graphs", Discrete Mathematics 55 (1985) 245-254
[18] Adamski M., „Aplication Specific logic Controler for Safety Critical Systems", 1999 IFACTriennial congress, Beijing (Chiny) vol. 0, ss. 519-529
[19] Adamski M., „Metodologia projektowania reprogramowalnych sterowników logicznych z wykorzystaniem elementów CPLD i FPGA", Reprogramowalne Układy Cyfrowe RUC 98, Szczecin 12-13.03.1998, ss. 15-22, 1998
[20] D. L. Springer, D.E. Thomas: „Exploiting the Special Structure of Conflict and Compatibility Graphs in High-Level Synthesis"; IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, No. 7, lipiec 1994, pp. 843-856

Title: Selected aspects of perfect graphs applications to automated processes of digital circuits.

Artykuł recenzowany