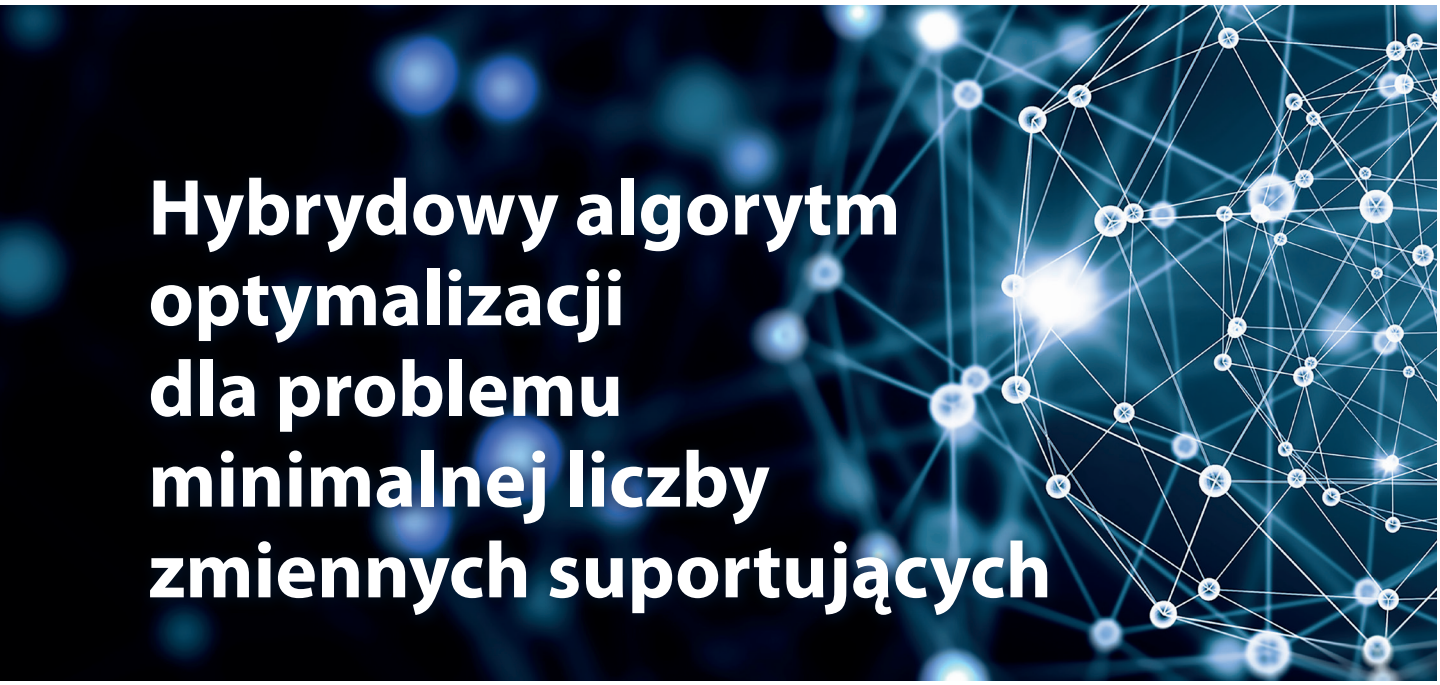


Hybrydowy algorytm optymalizacji dla problemu minimalnej liczby zmiennych suportujących

Przewodnik po nauczaniu informatyki kwantowej cz. 5.

Marek Perkowski

absolwent Wydziału Elektroniki Politechniki Warszawskiej, tu również zdobył tytuł doktora automatyki. Od 1983 r. pracuje na Wydziale Inżynierii Elektrycznej i Komputerowej w Portland State University, gdzie jest profesorem zwyczajnym i dyrektorem Laboratorium Robotów Inteligentnych.
Jeden ze współautorów WARP – pierwszego kompilatora języka VHDL dla układów FPGA. Twórca Diagramów Decyzyjnych Kroneckera, struktury krat logicznych i koncepcji robotów kwantowych. Przyczynił się do powstania oprogramowania dla syntezy logicznej, używanego w przemyśle USA. Pracował jako profesor wizytujący w Holandii, Francji, Japonii, Korei Południowej i Ludowej Republice Chin. W latach 2002–2004 był profesorem zwyczajnym w KAIST – Korean Advanced Institute of Science and Technology, gdzie zajmował się robotyką humanoidalną i komputerami kwantowymi. Kierował Komitetem Logiki Wielowartościowej IEEE w latach 2003–2005 i grupą roboczą Towarzystwa Inteligencji Obliczeniowej IEEE dla Inżynierii Kwantowej w latach 2006–2007. Autor ponad 515 publikacji o automatycznym projektowaniu, syntezie logicznej, logice wielowartościowej, logice odwracalnej, uczeniu maszynowym, robotyce i informatyce kwantowej.

Źródło: GetReal-WordPress.com

„Przewodnik po nauczaniu informatyki kwantowej" przedstawia metodologię rozwiązywania decyzyjnych Problemów ze Spełnianiem Ograniczeń (PSO) i problemów optymalizacyjnych z wykorzystaniem hybrydowego systemu komputera klasycznego i komputera kwantowego z algorytmem Grovera. Po wprowadzeniu układów odwracalnych jako rozszerzenia układów boolowskich pokazujemy superpozycję i splątanie kwantowe w sposób prosty, ale ścisły. Następnie przedstawiamy podstawowe dla wielu algorytmów kwantowych pojęcie wyroczni. Omawiamy, w jaki sposób wyrocznie są stosowane do rozwiązywania problemów decyzyjnych i optymalizacyjnych. Przykład znalezienia wszystkich „Optymalnych Zbiorów Suportujących" dla funkcji boolowskiej, który znajduje zastosowania w uczeniu maszynowym, dokładnie ilustruje proponowaną metodologię. Na koniec wyjaśniamy, jak działa algorytm Grovera. Po przeczytaniu tego cyklu uważny Czytelnik powinien być w stanie tworzyć podobne systemy kwantowe dla nowych, podobnych do przedstawionych, problemów.

Przykładem systematycznej syntezy bottom-up wyroczni dla algorytmu Grovera (użytego jako część algorytmu hybrydowego dla problemu optymalizacji) jest znalezienie minimalnego zbioru zmiennych dla dowolnej funkcji boolowskiej opisanej zbiorem mintermów (samples). Pierwsza część tego algorytmu jest obliczana na klasycznym komputerze, a następnie „konwersja POS→APN dla funkcji monotonicznej" jest realizowana w algorytmie Grovera, co przynosi kwadratowe przyspieszenie. Funkcję monotoniczną opisuje równanie, w którym nie ma zanegowanych zmiennych.

Zajmiemy się teraz szczegółową syntezą układu wyroczni. Nasz algorytm może być prosto zaadaptowany do rozwiązania szeregu innych ważnych problemów, takich jak znajdowanie prostych implikantów funkcji boolowskiej czy rozwiązanie problemu pokrycia.

Istnieje wiele publikacji na temat minimalnej liczby zmiennych suportujących (Minimum Set of Support), ponieważ ma on ważne zastosowania przy minimalizacji funkcji boolowskich (APN, KPN, PLA, FPGA etc.), dekompozycji Ashenhursta-Curtisa i innych dekompozycjach funkcjonalnych dla funkcji binarnych i wielowartościowych, w syntezie systemów informacyjnych, kryptografii, pozyskiwaniu danych i uczeniu maszynowym, dużych bazach danych, systemach reguł i systemach ekspertowych, funkcjach generujących indeksy, komplementacji funkcji boolowskich, zbiorach przybliżonych, minimalizacji funkcji Petricka (problem pokrycia dla APN) i innych. Problem ten jest też znany jako *attribute reduction problem* i *unate covering problem*.

Sformułowanie problemu do rozwiązania

Dana jest funkcja boolowska f: $\{0,1\}^n \rightarrow \{0,1,x\}$ posiadająca n zmiennych. Symbol x oznacza wartość nieokreśloną. Funkcja ta może być kompletnie określona, ale najczęściej jest niezupełnie określona, czyli jest relacją w sensie matematycznym, co występuje zwłaszcza w uczeniu maszynowym. Problem minimalizacji liczby argumentów polega na znalezieniu wszystkich najmniejszych zbiorów argumentów, od których zależy ta funkcja. Chcemy zapisać tę funkcję lub odpowiadający jej zbiór reguł w systemie ekspertowym przy użyciu minimalnej liczby argumentów. Metoda ta jest przydatna, gdy chcemy maksymalnie zredukować pewne reprezentacje opisane początkowo mintermami w sposób daleki od minimalnego. Funkcja opisana jest początkowo poprzez zbiór ON mintermów prawdy i zbiór OFF mintermów fałszu na n zmiennych.

Zaproponowano wiele metod, bo rozwiązanie tego problemu poprawia zarówno metody syntezy logicznej, jak i uczenia maszynowego i systemów ekspertowych. Znalezienie wszystkich zbiorów zmiennych (atrybutów, cech), od których funkcja zależy, pozwala poprawić wiele innych cząstkowych metod w pełnym procesie syntezy. Takie opisy są łatwiejsze do zrozumienia dla człowieka, a także pozwalają unikać nadmiernego dopasowania (*overfitting*), co jest ważne w uczeniu maszynowym.

Zacznijmy od klasycznego podejścia do rozwiązania tego problemu dla funkcji binarnej (naszą metodę można stosunkowo łatwo uogólnić na zmienne wielowartościowe).

1. Skonstruuj tablicę, w której wszystkie wiersze odpowiadają mintermom ze zbioru OFF a kolumny odpowiadają mintermom ze zbioru ON.

2. Oblicz bit-po-bicie operację EXOR na wszystkich parach wektorów binarnych mintermów z OFF z mintermami z ON.

3. Napisz wynikowe wektory binarne jako operacje LUB odpowiednich zmiennych. Na każdym przecięciu wiersza i kolumny umieść sumę logiczną tych zmiennych, które oddzielają dany minterm ON z kolumny od odpowiedniego mintermu OFF z wiersza.

4. Po wypełnieniu całej tablicy utwórz formułę KPN będącą funkcją logiczną I wszystkich termów LUB z kratek tablicy.

5. Przetransformuj powyższą formułę KPN do równoważnej minimalnej formuły APN. Klasycznie i teoretycznie może to być zrobione na przykład poprzez stopniowe wymnażanie boolowskie razem z absorpcją produktów tworzonych w każdym kroku. Można też zastosować program poszukiwania. W tak utworzonej formule APN każdy produkt odpowiada minimalnemu zbiorowi zmiennych. Ponieważ ten ostatni krok algorytmu jest NP-zupełny[1], rozwiązujemy go z przyspieszeniem algorytmu Grovera. Nasz klasyczny procesor hybrydowego systemu wywoła wielokrotnie algorytm Grovera ulokowany na komputerze kwantowym.

Znanych jest kilka metod konwersji wyrażenia KPN do wyrażenia APN. Wariant podany tutaj jest prostszy, gdyż nasza formuła KPN jest funkcją monotoniczną. Wszystkie te metody konwersji KPN do APN mogą być użyte do konstrukcji algorytmu kwantowego do realizacji minimalnej formuły APN, klasycznego problemu w syntezie logicznej. Tu jednak rozwiązujemy uproszczony problem funkcji monotonicznej. Uważny czytelnik zauważy tu także związki ze słynnym problemem spełnialności SAT.

1 Konieczny, P., Jóźwiak, L.: *Minimal input support problem and algorithms to solve it*. Eindhoven University of Technology Report E. Fac. of Electrical Engineering, Eindhoven, 1995, Vol. 95-E-289.

Komentarz

Użytkownik może zweryfikować poprawność rezultatów tego algorytmu poprzez zwijanie tablicy Karnaugha ze względu na wszystkie możliwe kombinacje zmiennych, które nie występują w każdym minimalnym zbiorze zmiennych. Na przykład dla funkcji pięciu zmiennych f(a,b,c,d,e) poprawność minimalnego zbioru {a,b} weryfikujemy zwijając (*folding*) tablicę Karnaugha ze względu na zmienne c, d i e, nie spotykając sprzeczności i w rezultacie tworząc nową funkcję $f_1(a,b)$, taką że $f_1(a,b) = f(a,b,c,d,e)$.

Pożyteczną metodą wstępnego przetwarzania jest uproszczenie formuły początkowego KPN (punkt 2) przy wielokrotnym użyciu praw boolowskich; A.A = A; A. (A+B) = A i (A+B) · (A+B+C) = A+B. Zaczynamy zatem upraszczanie od najkrótszych termów LUB i usuwamy te, które je zawierają. Omawiany problem jest NP [34], zatem wszystkie optymalne algorytmy dla komputerów klasycznych mogą być stosowane jedynie do małych problemów, podczas kiedy obecnie zbiory danych ON, OFF mogą być olbrzymie. Dla większych problemów klasyczne algorytmy mogą być tylko heurystyczne, a potrzebują długiego czasu i dużej pamięci, aby znaleźć tylko niektóre minimalne lub w przybliżeniu minimalne zbiory zmiennych suportujących. Natomiast przedstawiony tutaj pełny algorytm znajduje wszystkie rozwiązania i to w sposób optymalny. Algorytm Grovera daje kwadratowe przyspieszenie dla kroku optymalnej konwersji KPN do APN dla funkcji monotonicznych.

Konkretny przykład rozwiązania

Przeanalizujmy krok po kroku nasz algorytm. Zakładamy, że funkcja boolowska reprezentowana jest przez tablicę Karnaugha o czterech zmiennych:

ab\cd	00	01	11	10
00	x	x	1	x
01	1	x	x	1
11	x	x	0	x
10	0	x	x	0

Rys. 1. Tablica Karnaugha dla niezupełnie określonej funkcji czterech zmiennych.

Krok 1. Określone wartości funkcji są 0 (OFF) oraz 1 (ON). Przystępujemy do porównania mintermów tworząc tablicę z rys. 2. Komórki w tablicy na przecięciu mintermów OFF i mintermów OFF tworzymy przy użyciu pobitowej operacji EXOR. Na przykład minterm OFF 1111 z mintermem ON 0011 tworzy wektor 1100, co zapisujemy przez *a+b*.

OFF\ON	0011	0100	0110
1111	a+b	a+c+d	a+d
1000	a+c+d	a+b	a+b+c
1010	a+d	a+b+c	a+b

Rys. 2. Ilustracja metody znajdowania wszystkich zmiennych separujących zmienne dla każdej pary mintermów ON i OFF dla funkcji z rys. 1. Minterm ON 1111 i minterm OFF 0011 są oddzielane od siebie zmienną a lub zmienną b. Odpowiada to formule a+b, która występuje w tablicy na przecięciu wiersza 1111 i kolumny 0011.

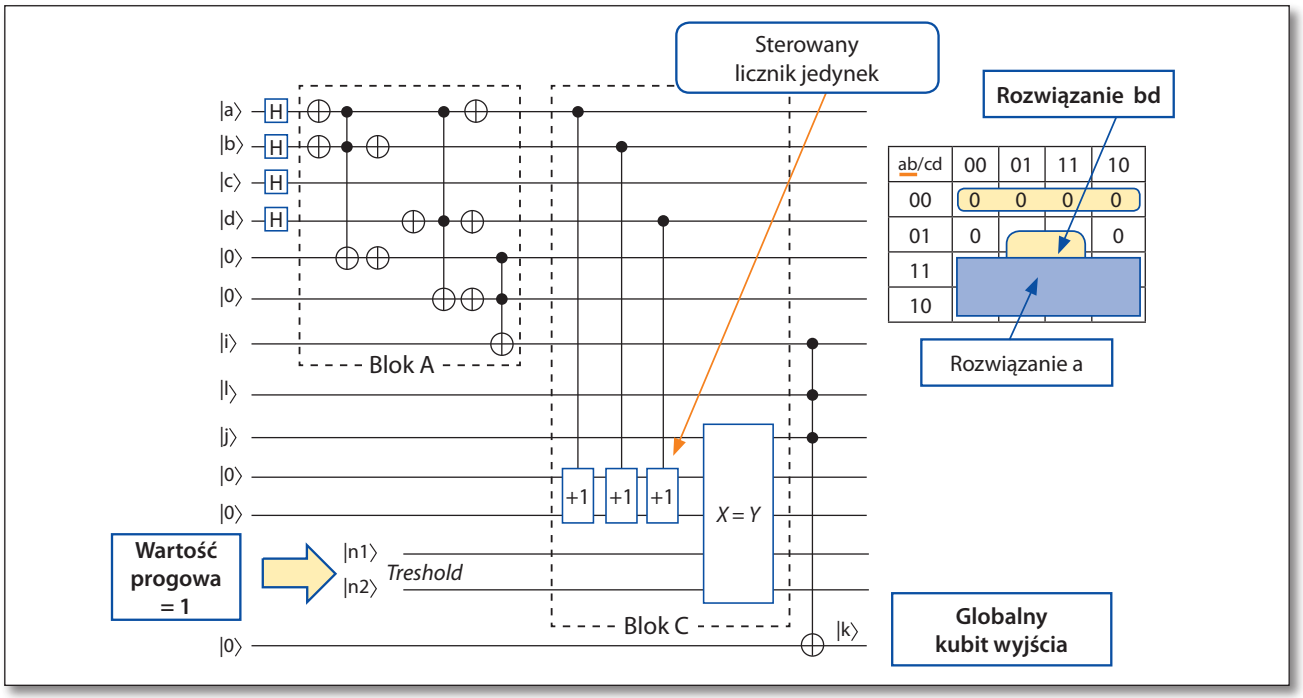
Krok 2. Formuła KPN jest utworzona jako iloczyn logiczny wszystkich termów z komórek tablicy z rys. 2: *(a+b) · (a+c+d) · (a+d) · (a+c+d) · (a+b) · (a+b+c) · (a+d) · (a+b+c) · (a+b).*

Krok 3. Używając praw boolowskich z algorytmu, formuła zostaje uproszczona do (a+b).(a+d). W tym prostym dla celów dydaktycznych przykładzie transformacja KPN –> APN jest trywialna na podstawie algebry Boole'a: $(a+b) \cdot (a+d) \cdot a + ab + ad + bd \cdot a + bd$. W ostatniej formule znaleźliśmy, że funkcja boolowska z rys. 1 zależy albo tylko od pojedyńczej zmiennej ze zbioru {*a*}, albo od dwóch zmiennych ze zbioru {b, d}. Funkcja ta ma zatem dwa minimalne zbiory suportujące – {*a*} i {*b,d*}. Dla dużych funkcji, które mogą mieć tysiące zmiennych, ta transformacja KPN do minimalnego APN jest jako NP problem nierealizowalna klasycznie, co powoduje, że ten krok wykonywany jest w algorytmie Grovera na komputerze kwantowym systemu hybrydowego.

Wyjaśnijmy teraz, jak w systemie hybrydowym algorytm Grovera jest wielokrotnie wywoływany ze zmodyfikowanymi wyroczniami w celu znalezienia wszystkich minimalnych rozwiązań problemu.

Pierwsza wyrocznia dla algorytmu Grovera

Aby znaleźć minimalne zbiory suportujące dla funkcji z rys. 1, tworzymy sekwencję wyroczni dla problemu KPN (*a+b*) · (*a+d*) do APN. Najpierw musimy zakodować nasz problem jako problem binarny. Dla naszej wyroczni przestrzeń rozwiązań to zbiór wszystkich podzbiorów zbioru zmiennych {a, b, c, d}. Niektóre z tych podzbiorów to poszukiwane przez nas rozwiązania problemu. Wszystkie rozwiązania naszego problemu mogą być reprezentowane jako funkcja boolowska na zmiennych *a, b, c, d* reprezentowana jako KPN. Kodujemy nasz problem kodem „one-hot", wybrana zmienna a jest kodowana jako 1000, wybrany zbiór {*a, b, c*} jest kodowany jako 1110 itd. Blok C zawiera „licznik kosztu" (liczbę zmiennych) oraz komparator, które razem służą do wybierania tylko tych potencjalnych rozwiązań o koszcie równym wartości *Threshold*. Klasyczny procesor w systemie hybrydowym kompiluje i kontroluje pierwszą wyrocznię (rys. 3). Zakłada się tu, że istnieje



Rys. 3. Pierwsza wyrocznia w sekwencji wyroczni dla rozwiązania problemu minimalnych zbiorów suportujących dla funkcji binarnej z rys. 1 z minimalnym zbiorem zmiennych suportujących {{a}, {b, d}}. KPN = (a+b) · (a+d) = 1⊕ (a'b')' · (a'd')' jest realizowana w bloku A. Bramki Hadamarda u góry z lewej nie należą do wyroczni. Bramki te zostały tu umieszczone dla celów dydaktycznych, ponieważ służą one jako generator wszystkich podzbiorów zbioru {a,b,c,d} poprzez stworzenie superpozycji wszystkich liczb od 0 do 15. Ta pierwsza wyrocznia weryfikuje, czy istnieje rozwiązanie z kosztem 1, ponieważ wartość ograniczenia Threshold została ustawiona na 1. Tablica Karnaugha pokazuje funkcję (a+b) · (a+d) opisaną przez 0 w kratkach tablicy. Grupa a, która jest pierwszym rozwiązaniem, ilustruje jak wszystkie rozwiązania zawarte w a są następnie zastępowane przez 0. Po tym zastąpieniu pozostaje jedynie iloczyn logiczny bd(a)'.

rozwiązanie o koszcie 1, ponieważ dwukubitowa liczba *Threshold* na wejściu do komparatora jest ustawiona na wartość 1. Nasza wyrocznia sprawdza, czy wartości zmiennych wejściowych spełniają następujące dwa ograniczenia: A – że KPN jest spełniony, C – że liczba zmiennych w rozwiązaniu jest równa wartości stałej *Threshold* podanej na wejściu.

Układ wyroczni z rys. 3 nie jest kompletny ze względu na ograniczenia rozmiaru tego artykułu. Brakuje lustrzanego odbicia układu odpowiadającego blokom A i C ze wszystkimi bramkami w odwrotnym porządku. Zadaniem układu lustrzanego jest przywrócenie początkowych wartości wszystkim kubitom różnym od kubitu wyjściowego wyroczni. Dotyczy to zarówno kubitów zmiennych, jak i kubitów dodatkowych (*ancilla*). Są one przywracane do ich oryginalnych wartości w wyroczni tak, by wyjście ostatniej pętli Grovera było mierzone razem ze stanem wejściowym kubitów odpowiadającym zmiennym problemu. Ogólnie „układ lustrzany" jest skomponowany z lustra bloku C, po nim z lustra bloku B (blok B i jego lustro nie istnieją w pierwszej wyroczni) i następnie z lustra bloku A. Kubit |k⟩ jest rozwiązaniem wyroczni. W ogólności ten kubit wyjściowy zaznacza, że wszystkie ograniczenia bloków A, B i C zostały spełnione.

Blok A jest boolowską funkcją SAT w formie KPN. Kubit |i⟩ jest inicjalizowany do |1⟩, aby zrealizować KPN *(a+b) · (a+d)* na podstawie twierdzenia DeMorgana z bramek Toffoliego i inwerterów. Ten kubit rozpoznaje każde rozwiązanie danego KPN z superpozycji wejść. Jest on podany jako jedno z trzech pomnożonych logicznie wejść do bramki Toffoliego po prawej, która tworzy rozwiązanie całej wyroczni. Blok B nie istnieje w pierwszym wywołaniu algorytmu Grovera i jest symulowany stałą 1, drugą od góry, w globalnej bramce iloczynu logicznego I po prawej (realizowanej przez bramkę Toffoliego). Blok ten będzie reprezentował modyfikacje ograniczeń wyroczni w następnych wywołaniach algorytmu Grovera. Jak już wspominaliśmy, taka modyfikacja wyroczni jest typowa dla problemów optymalizacyjnych.

Blok C zawiera trzy liczniki i komparator równości. Trzy liczniki zliczają liczba zmiennych wejściowych potrzebnych do spełnienia funkcji KPN. Każdy licznik dodaje 1, jeśli odpowiadająca mu zmienna wejściowa kontrolująca ma wartość |1⟩. Komparator X=Y porównuje wyjście ostatniego licznika z wartością progową *Threshold*, podaną jako stałe wartości binarne kubitów |n1⟩ i |n2⟩ na wejściu do algorytmu Grovera. Następnie bramka Toffoliego jest stosowana do wyjściowego kubitu |k⟩ kontrolowanego kubitami |i⟩, |1⟩ i |j⟩. Jeśli wszystkie ograniczenia są spełnione, kubit |k⟩ przyjmuje wartość |1⟩. Kubit ten zmienia fazę kwantową rozwiązania, tak że zaznaczane są wszystkie elementy przestrzeni rozwiazań, które spełniają wszystkie ograniczenia.