Kolejne wyrocznie dla algorytmu Grovera

Przewodnik po nauczaniu informatyki kwantowej cz. 6.

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.

W problemach minimalizacji funkcji kosztu sterujący komputer klasyczny startuje z najniższego progu (*bound*) potencjalnego kosztu rozwiązania i stopniowo powiększa go. W naszym konkretnym problemie – nie mając żadnej informacji – zakładamy, że istnieje jedna zmienna, od której zależy funkcja, a więc jedna zmienna, która spełnia KPN (KPN to koniunkcyjna postać normalna). Zatem wartość stałej progowej *Threshold* jest początkowo ustawiana na 1 i wszystkie rozwiązania z jedną zmienną są sprawdzane przez liczniki i komparator. W zależności od problemu różne strategie zmian wartości *Threshold* mogą być podawane w kolejnych wywołaniach algorytmu Grovera. W ogólności dla problemów minimalizacji czy maksymalizacji funkcji kosztu możemy tworzyć różne strategie zmieniania wartości *Threshold*, wartości te mogą być rosnące lub malejące.

Przedyskutujmy dokładnie pełny proces komputera hybrydowego dla pewnego małego problemu (minimalizacja kosztu).

1. Wartość zmiennej *Threshold* jest ustawiona na 1 ponieważ poszukujemy najpierw tylko rozwiązań składających się ze zbiorów o jednym elemencie. Pierwsze wywołanie algorytmu Grovera znajduje rozwiązanie {a}.

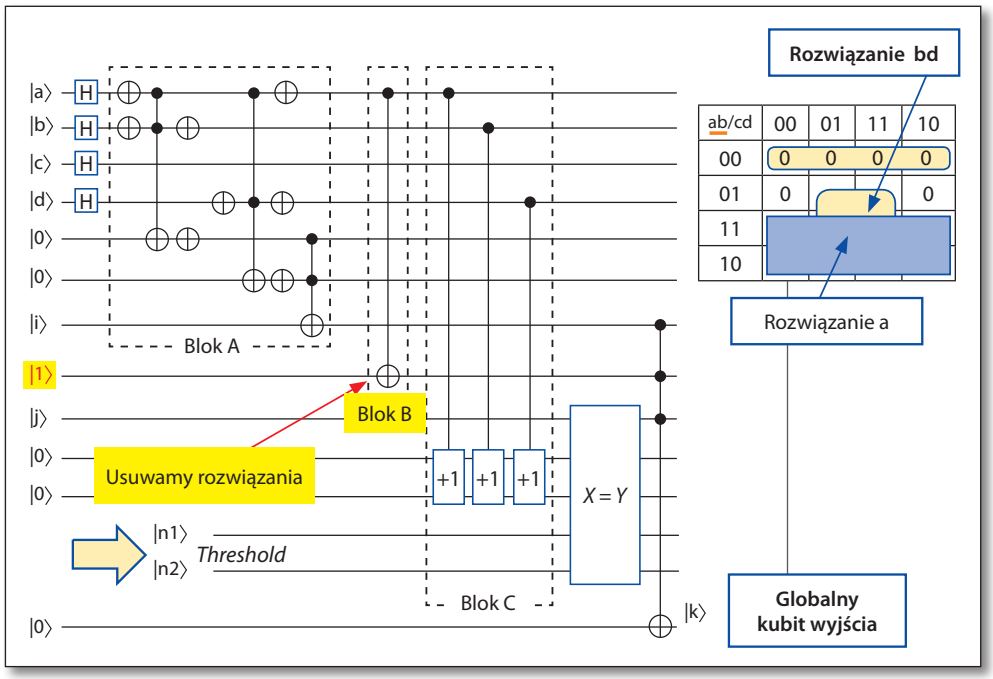
2. W celu znalezienia innych rozwiązań o koszcie 1, wyrocznia jest modyfikowana tak, jak na rys. 1. Ta modyfikacja wyklucza rozwiązanie {a}, a także wszystkie rozwiązania zawarte w {a}, to znaczy {a,b}, {a,c}, {a,b,c} itd. Jest tak dlatego, bo na przykład iloczyn *ab* jest zawarty w iloczynie *a*.

3. Zatem blok B jest teraz dodany do wyroczni przez klasyczny komputer, który kompiluje nowy układ wyroczni, a następnie wykonuje go na komputerze kwantowym. Blok B zawiera też reprezentację pierwszego rozwiązania {a}, które jest usuwane (operator inhibicji X·Y' realizowany jako część dużej bramki Toffoliego po prawej). Dlatego wszystkie możliwe rozwiązania, które zawierają zmienną *a*, są wykluczone. Odpowiadają one wszystkim mintermom zawartym w iloczynie *a*.

4. Ponieważ nie wiadomo, czy istnieje inne rozwiązanie o koszcie 1, próg *Threshold* = 1 jest nadal utrzymywany przy następnym wywołaniu algorytmu Grovera poszukującego innych rozwiązań o koszcie 1. Jak widzimy na tablicy Karnaugha (prawy górny róg na rys. 1), takie rozwiązania nie istnieją. Zatem algorytm Grovera nie może znaleźć rozwiązania. Polega to na tym, że algorytm wygeneruje kilka losowych zbiorów, które nie mogą być zweryfikowane. Weryfikacja jest dokonywana poprzez ewaluacje tych zbiorów na wyroczni A użytej poza algorytmem Grovera.

5. Ponieważ nie ma więcej rozwiązań o koszcie 1, procesor klasyczny zmienia wartość stałej progowej *Threshold* na wartość 2, przygotowując procesor kwantowy do poszukiwania rozwiązań o koszcie 2 – patrz rys. 1.

Druga wyrocznia dla algorytmu Grovera



Rys. 1. Druga wyrocznia w sekwencji wywołań algorytmu Grovera – należy znaleźć wszystkie minimalne zbiory suportujące dla funkcji binarnej z rozwiązaniem {{a}, {b, d}}. KPN = (a+b) · (a+d) jest realizowana w bloku A. Ta wyrocznia weryfikuje, czy istnieje rozwiązanie niezawarte w rozwiązaniu {a} o koszcie 2, ponieważ próg Threshold = 2. Blok B wyklucza rozwiązanie {a}, które zostało znalezione wcześniej, oraz wszystkie rozwiązania logicznie w nim zawarte, takie jak {a,b}, {a,c}, {a,d}, {a,b,c}…, {a,b,c,d}. Układy lustrzane dla bloków A, B i C nie są pokazane.

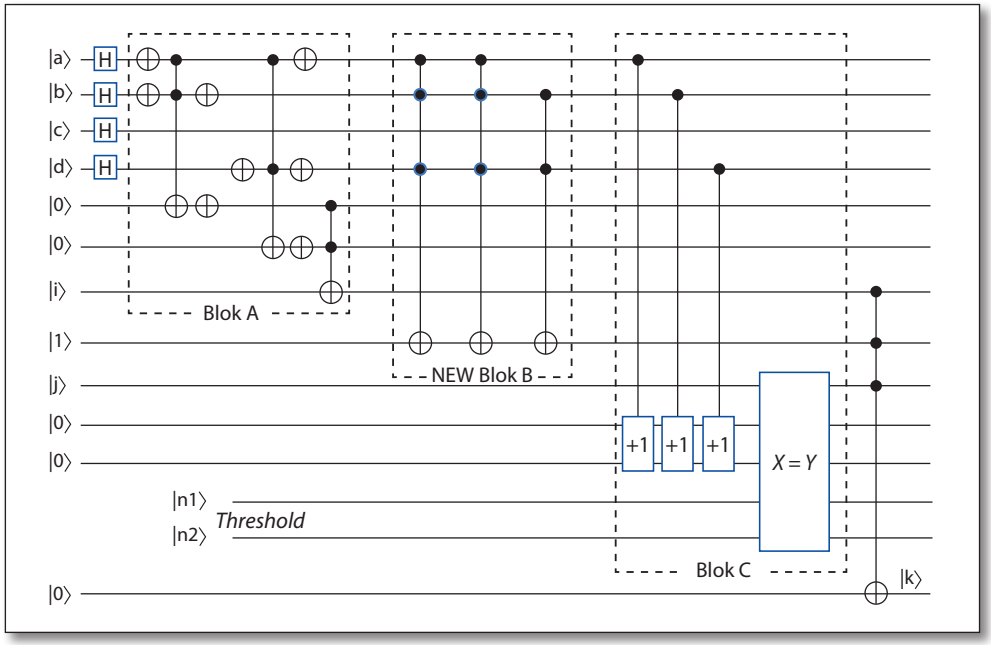
Proces jest kontynuowany:

1. Z progiem *Threshold* = 2 algorytm Grovera znajduje rozwiązanie {*b,d*}.

2. Iloczyn bd jest następnie dodany do wyrażenia Ekskluzywnej Postaci Normalnej – EPN (Exclusive-Or-Sum-of-Products – ESOP), realizowanego na wyjściu kubitu z bloku B z zaznaczeniem, że rozwiązanie {*b,d*} było już znalezione. Blok B zostaje więc opisany przez $a + bd = a \oplus bd \oplus abd$, jak pokazano na rys. 2 (ten EPN wynika z reguły boolowskiej $x+y = x \oplus y \oplus xy$). Taka metoda jest używana do wykluczania poprzednio znajdywanych rozwiązań i potencjalnych rozwiązań zawartych w następnych wywołaniach algorytmu

Grovera. Dlatego suma logiczna wszystkich poprzednich rozwiązań jest aktualizowana wraz z każdym znalezionym rozwiązaniem w bloku B.

Trzecia wyrocznia dla algorytmu Grovera



Rys.2. Trzecia wyrocznia W3. Zmodyfikowany blok B wyklucza wszystkie potencjalne rozwiązania zawarte w funkcji a + bd = a ⊕ bd ⊕ abd. Wyrocznia ta jest używana do poszukiwania rozwiązań, które nie są zawarte w {a} i nie są zawarte w {b,d}. Rozwiązań takich nie ma. Układy zwierciadlane dla bloków A, B i C nie są pokazane.

Proces jest kontynuowany:

1. Z wyrocznią jak na rys. 2 i progiem *Threshold* = 2 algorytm Grovera nie jest w stanie znaleźć więcej rozwiązań o koszcie 2. Ale ciagle nie jest jasne, czy istnieją rozwiązania z trzema czy więcej (w ogólności) zmiennymi. Program na komputerze sterującym może kontynuować powyższy proces, gdy chcemy znaleźć wszystkie rozwiązania, albo przerwać dalsze poszukiwania dla dowolnie założonej liczby zmiennych w rozwiązaniu.

2. W tym przypadku nie ma więcej rozwiązań. W ogólności procesor klasyczny zwiększa wartość progu, aby znaleźć wszystkie zbiory zmiennych suportujących. Aby potwierdzić, że nie ma więcej rozwiązań poza tymi, które juz zostały znalezione, wyrocznia jest użyta w wywołaniu algorytmu Grovera z blokami A i B, ale bez bloku C. Jeśli funkcja boolowska takiej nowej wyroczni W3 nie jest spełnialna, algorytm Grovera używa metody znanej w SAT, że nie ma rozwiązania. W tym akurat przypadku funkcja boolowska W3 = 0. System udowodnił, że znalezione poprzednio rozwiązania {*a*} i {*b,d*} to wszystkie rozwiązania optymalne problemu.

Zauważmy, że problem ten jest podobny do SAT i MaxSAT oraz prawie identyczny jak dwa znane problemy:

(a) Podproblem minimalizacji APN (alternatywnej postaci normalnej): znaleźć wszystkie proste implikanty funkcji boolowskiej zawierajace minterm Mi.

(b) Klasyczny problem pokrycia zbiorów (*unate covering*): dany jest zbiór Mi i zbiór PM podzbiorów zbioru Mi. Znaleźć wszystkie zbiory PMi z PM, które: (1) łącznie zawierają wszystkie elementy zbioru Mi, (2) każdy zbiór PMi nie zawiera innego zbioru PMj.

Istnieje wiele wariantów rozwiązania tego i podobnych problemów. Pamiętajmy, że każdy wektor binarny |a,b,c,d⟩ może być weryfikowany na bloku A użytym samym w sobie, poza algorytmem Grovera. W tym i podobnych problemach pomocne może być też policzenie (czasem przybliżonej) liczby potencjalnych rozwiązań, czyli liczby jedynek w naszej funkcji na dowolnym etapie. Na przykład w wyroczni W3 ta liczba jedynek wynosi/równa się zero. Liczenie jedynek odbywa się przy użyciu algorytmu liczenia kwantowego (*Quantum Counting Algorithm*)[1], który zlicza liczbę jedynek funkcji w tej części wyroczni, która opisuje sam problem, a nie w całej wyroczni. W naszym przypadku jest to część A wyroczni. Ta liczba jedynek, albo jest wręcz liczbą rozwiązań albo jest często związana z tą liczbą.

Uwagi o przedstawionej metodologii

Jako przykład stosowania zaproponowanej tu metodologii tworzenia algorytmów kwantowych podaliśmy algorytm hybrydowy, w którym przetwarzanie wstępne, to znaczy utworzenie formuły KPN, jest dokonywane w procesorze klasycznym. Jedynie rozwiązanie NP problemu, czyli *konwersja KPN→APN dla funkcji monotonicznej wraz ze znalezieniem wszystkich prostych implikantów*, jest realizowane przez algorytm hybrydowy, opierający się na sekwencji wywołań algorytmu Grovera ze zmodyfikowanymi wyroczniami.

1 Brassard, G., HØyer, P., Tapp, A.: *Quantum counting*. Automata, Languages and Programming, 1998, 820–831, DOI: 10.1007/BFb0055105

Pokazaliśmy szczegółowo konstrukcję wyroczni z typowych realizowalnych i często stosowanych bramek i bloków. Więcej szczegółów dotyczących takich bloków i ich syntezy logicznej oraz podobnych algorytmów hybrydowych można znaleźć w kilku publikacjach[2]. Ten i podobne przykłady pokazują pewne elementy proponowanej metodologii:

1. Wybór odpowiedniego kodowania problemu.

2. Wybór reprezentacji problemu dla tego kodowania.

3. Podjęcie decyzji co do konstrukcji układów realizujących poszczególne ograniczenia i sposobów ich łączenia.

4. Określenie funkcji kosztu dla problemu i zaprojektowanie metody jej realizacji w układzie kwantowym.

5. Projekt pełnego systemu hybrydowego i określenie roli klasycznego procesora, który nadzoruje komputer kwantowy i przygotowuje dla niego kolejne dane.

6. Określenie sposobu powtarzania algorytmu Grovera, wartości progów, sposobu modyfikacji wyroczni w celu efektywnego rozwiązania problemu optymalizacyjnego.

Przedstawiona metoda może być stosunkowo łatwo modyfikowana do algorytmu Grovera z wielowartościowymi kubitami, czyli kuditami. Ogólne podejście jest takie same, ale użytkownik musi nauczyć sie syntetyzować wielowartościowe układy kwantowe[3], na przykład trójwartościowe[4]. Główna koncepcja kwantowych układów niebinarnych pochodzi z pracy[5]. W modyfikacji algorytmu Grovera dla logiki trójwartościowej bramki Hadamarda są zastępowane bramkami Chrestensona[6] i tworzone są nowe bramki sterowane wartościami ternarnymi, które uogólniają binarne bramki sterowane. Na przykład w opracowaniu[7] bramki kwantowe w binarnym ciele Galois są uogólnione na ciało trójwartościowe, a w pracy[8] podano algorytm dekompozycji macierzy unitarnych na kwantowe bramki trójwartościowe. Praca[9] uogólnia metody syntezy binarnej na kwantową logikę trójwartościową z bramek sterowanych.

Podobne wyrocznie Grovera buduje się dla innych problemów EDA, teorii grafów, problemów sterowania czy optymalizacji, na przykład takich, jak komplementacja funkcji przełączającej, problemy pokrycia, znajdowanie maksymalnych klik w grafie, problemy podziału grafu, pokrycia grafu, kolorowania grafu, problem generacji prostych implikantów, problem znajdowania maksymalnych zbiorów niezależnych, minimalizacja układów typu APN, minimalizacja Ekskluzywnej Postaci Normalnej EPN, minimalizacja układów układów Rozdzielnej Alternatywnej Postaci Normalnej RAPN (Disjoint SOP – DSOP) i wiele innych. Problemy takie, jak gry i zagadki logiczne, mogą być redukowane do podobnych wyroczni.

2 Hou, W., Perkowski, M.: *Quantum-based algorithm and circuit design for bounded Knapsack optimization problem.* Quantum Information and Computation, 2020, 20(9–10), 766–786.

Lee, B., Perkowski, M.: *Quantum Machine Learning Based on Minimizing Kronecker-Reed-Muller Forms and Grover Search Algorithm with Hybrid Oracles.* Proc IEEE, 2016 EuroMicro Conference on Digital System Design (DSD).

Li, Y., Tsai, Y., Perkowski, M., Song, X.: *Grover-Based Ashenhurst-Curtis Decomposition using Quantum Language Quipper.* Quantum Information and Computation, 2019, 19(1–2), 35–66.

3 Bullock, S.S., O'Leary, D.P., Brennen, G.K.: *Asymptotically Optimal Quantum Circuits for d-Level Systems.* Physical Review Letters, 2005, 94(23), 230502. Also quant-ph/0410116.

4 Burlakov, A.V., Chekhova, M.V., Karabutova, O.V., Klyshko, D.N., Kulik, S.P.: *Polarization state of a biphoton: quantum ternary.* Physical Review A, 1999, 60(6), R4209.

Wang, Y., Perkowski, M.: *Improved Complexity of Quantum Oracles for Ternary Grover Algorithm for Graph Coloring.* Proc. ISMVL, 2011, 294–301, DOI: 10.1109/ISMVL.2011.42.

5 Muthukrishnan, A., Stroud, Jr. C.R: *Multivalued logic gates for quantum computation.* Physical Review A, 2000, 62(5), 1–8.

6 Lee, B., Perkowski, M.: *Quantum Machine Learning Based on Minimizing Kronecker-Reed-Muller Forms and Grover Search Algorithm with Hybrid Oracles.* Proc IEEE 2016 EuroMicro Conference.

7 Khan, M.H.A., Perkowski, M., Kerntopf, P.: *Multi-Output Galois Field Sum of Products Synthesis with New Quantum Cascades.* Proc. 33rd ISMVL, Tokyo, May 16–19, 2003, 146–153.

8 Khan, F., Perkowski, M.: *Synthesis of Ternary Quantum Logic Circuits by Decomposition.* Proceedings of 7th International Symposium on Representations and Methodology of Future Computing Technologies, RM 2005, University of Tokyo, September 5–6, 2005, 114–118.

9 Yang, G., Song, X., Perkowski, M., Wu, J.: *Realizing ternary quantum switching networks without ancilla bits.* Journal of Physics A: Mathematical and General, 2005.

Idee związane z używaniem algorytmu Grovera

Klasyczny komputer może w nieuporządkowanej liście znaleźć pewien obiekt w czasie zależnym liniowo od rozmiaru tej listy N. Komputer kwantowy może użyć algorytmu Grovera, który będzie wymagał jedynie pierwiastka tego czasu, czyli √N, o ile czas potrzebny do weryfikacji tego obiektu jest mały.

> *Algorytm Grovera jest podstawowym algorytmem poszukiwania kwantowego, wynalezionym przez Lova Grovera w 1996 r. Określenie to sugeruje, że jest to "rozwiązywacz" pewnej określonej klasy konkretnych problemów, mniemanie to jest jednak mylące. Lepiej jest myśleć o tym algorytmie jako o pętli programowej, niezależnej od konkretnego problemu i realizowanej na równoległym sprzęcie.*

Algorytm ten jest prawdopodobnie jednym z najbardziej przydatnych pomysłów w informatyce kwantowej. Z możliwie największym prawdopodobieństwem poszukuje on w „funkcji czarnego pudełka", czasem nazywanej „nieposortowaną bazą danych", takiego jej elementu, który spełnia wyrocznię[10]. Nazwa „nieposortowana baza danych" jest jednak myląca i należy ją traktować jedynie jako metaforę ślepego poszukiwania, które nie wykorzystuje żadnej informacji. Lepiej jest myśleć o tym algorytmie jako o równoległym poszukiwaniu w wyroczni, to znaczy obliczaniu funkcji boolowskiej F dla wszystkich jej mintermów na raz, aby znaleźć wszystkie te mintermy, dla których F=1. Algorytm kwantowy znajduje stan kwantowy odpowiadający zbiorowi wszystkich mintermów będących rozwiązaniami, ale pomiar daje tylko jeden z nich na raz. Znalezienie innych rozwiązań wymaga ponownego procesu poszukiwania. Pomiar jest tu więc wąskim gardłem, z czego musi sobie zdawać sprawę projektant pragnący efektywnie użyć algorytmu Grovera.

Czytelnik wie już, jak go używać w problemach PSO i w zadaniach optymalizacji. Może on być jednak stosowany w wielu innych zagadnieniach, takich jak znajdowanie minimum, obliczenia arytmetyczne. Jak więc stosować ten algorytm w nowy sposób?

Konstrukcja wyroczni w algorytmie Grovera

Konstrukcję tę musi charakteryzować element celowy w całej przestrzeni poszukiwań opisanej przez transformatę Hadamarda czy inną transformatę inicjalizującą przestrzeń rozwiązań dla wszystkich kubitów odpowiadających zmiennym problemu. Wyrocznia jest budowana dla danej instancji problemu, jest więc zawsze zależna od konkretnego wymiaru problemu, podobnie jak jest to w klasycznych FPGA. Na przykład, algorytm kwantowy dla rozwiązania Sudoku 4*4 jest inny od algorytmu dla rozwiązania Sudoku 9*9. Projektant powinien rozważyć, co robić w przypadkach gdy $N \neq 2^n$, jak traktować binarne wektory ze zbioru $N - 2^n$? Wektory binarne o długości *n* (dla n zmiennych wejściowych) tworzone przez transformatę Hadamarda na wejściu naturalnie odpowiadają: wszystkim liczbom naturalnym 0,… 2^n–1, wszystkim podzbiorom zbioru *n* elementów, pewnym mapowaniom, odwzorowaniom itd. Jak reprezentować permutacje, kombinacje, listy, formuły, grafy, drzewa czy inne struktury danych? To nie jest trywialny problem i powinien być tematem badań.

Mimo że wielu autorów opisuje algorytm Grovera w terminach macierzy unitarnych i teoretycznych operatorów, w pracach naszego zespołu projektujemy wyrocznie z konkretnych realizowalnych bramek kwantowych, aby móc ocenić praktyczną realizowalność naszych wyroczni czy całych algorytmów hybrydowych na symulatorach kwantowych i w istniejących technologiach komputerów kwantowych. Synteza przeprowadzana jest tak, by zminimalizować „koszt kwantowy"[11] tego układu, czyli koszt bramek niezależny od konkretnej kwantowej technologii. Koszt ten jest jednak pewnym uśrednieniem, oceniającym złożoność zwłaszcza wielokubitowych bramek Toffoliego i jako taki jest bardzo użyteczny do oceny praktycznej złożoności sprzętowej dla wielu technologii kwantowych. Dla każdego typu bramki możemy obliczyć jej koszt, zaś cena całej wyroczni jest sumą tych kosztów dla wszystkich użytych w niej bramek. Bramki Toffoliego z dużą liczbą wejść są bardzo kosztowne w każdej technologii. Często warto zastąpić je bramkami Peresa[12], na przykład w licznikach kwantowych, co obniża koszt kwantowy. Program syntetyzujący może więc rozpatrywać różne warianty syntezy logicznej na teoretycznych bramkach kwantowych, a także layout kwantowy i wybierać najlepsze rozwiązania. Biorąc pod uwagę rozmiar wyroczni i to, jak używa ona bramek, synteza układu kwantowego bardzo wpływa na zachowanie a nawet realizowalność algorytmu Grovera dla konkretnego problemu ze względu na ograniczenia istniejących symulatorów kwantowych i komputerów kwantowych.

10 Cerf, N. J., Grover, L. K., Williams, C. P.: *Nested Quantum Search and NP-Hard Problems.* Appl. Algebra Eng. Commun. Comput., 2000, 10(4–5), 311–338.

11 Maslov, D., Dueck, G.: *Improved Quantum Cost for n-bit Toffoli Gates.* Faculty of Computer Science, University of New Brunswick, Fredericton, Canada, 2004.

12 Szyprowski, M., Kerntopf, P.: *Low Quantum Cost Realization of Generalized Peres and Toffoli Gates with Multiple-Control Signals.* Proc. 13th IEEE Intern. Conf. on Nanotechnology, 2013, 802–807.