Prof. dr hab. Włodzisław Duch
Katedra Informatyki Stosowanej,
Laboratorium Neurokognitywne
Uniwersytet Mikołaja Kopernika, Toruń

Recenzja rozprawy doktorskiej napisanej przez mgr inż. Adama Górskiego .
„Kosynteza oraz przydział nieprzewidzianych zadań w procesie projektowania systemów wbudowanych".

Mgr Górski przygotował rozprawę doktorską na Wydziale Informatyki, Elektroniki i Telekomunikacji AGH, pod opieką promotora prof. Macieja Ogorzałka i promotor pomocniczą dr Katarzyną Grzesiak-Kopeć. Praca dotyczy rozproszonych systemów wbudowanych. W celu zwiększenia wydajności takie systemy projektowane są używając technik kosyntezy, czyli jednoczesnej optymalizacji architektury sprzętowej i oprogramowania, przydzielającego poszczególne zadania jednostkom obliczeniowym tak by zminimalizować koszty. Ponieważ powstają coraz bardziej złożone systemy wbudowane, wykonujące wiele różnych zadań, optymalizacja ich działania jest ważnym zagadnieniem. Podobnie jak dla wielu problemów optymalizacyjnych jest to w ogólnym przypadku problem NP-trudny i trudno jest tu znaleźć globalnie optymalne rozwiązanie. Pierwsze publikacje na temat kosyntezy rozproszonych systemów wbudowanych metodami programowania genetycznego mgr Górski przedstawiał na konferencjach już w 2008 roku. Od 2014 roku swoje rezultaty przedstawiał w artykułach pisanych wspólnie z promotorem na dziewięciu międzynarodowych konferencjach, trzech w Polsce i sześciu za granicą.

Badania nad sprzętowo-programową kosyntezą zapoczątkowano na Uniwersytecie Stanforda około 1990 roku. R.K. Gupta obronił tam w 1993 roku doktorat i opublikował go w formie książkowej, ale nie wspomina w nim algorytmów genetycznych. Zaproponowano wiele metod konstrukcyjnych i rafinacyjnych dla problemu kosyntezy, jednak metody te mają swoje ograniczenia. Motywacja i tezy pracy są sformułowane w jednoznaczny sposób. Zastosowanie rozwojowego programowania genetycznego powinno pozwolić na odkrycie quasi-optymalnych rozwiązań problemu kosyntezy, jak też modyfikację algorytmu w przypadku pojawienia się zadań nieprzewidzianych na etapie projektowania.

Rozdział trzeci zawiera formalne definicje: systemu wbudowanego, reprezentacji w postaci acyklicznego skierowanego grafu zadań, ścieżki krytycznej i poprawności systemu wbudowanego, oraz bazy zasobów, która zawiera czas i koszty wykonania wszystkich zadań. Takie podejście nie obejmuje więc sytuacji, w której system wbudowany realizuje całkiem nowe zadanie, w którym czas wykonania poszczególnych etapów nie jest z góry znany. Dotyczy to również nieprzewidzianych zadań, które muszą być scharakteryzowane w taki sam sposób jak zadania uwzględniane przy projektowaniu. W wielu zastosowaniach jest to istotne ograniczenie, które nie zostało przez autora rozprawy wspomniane. Specyfikacja formalna może zawierać nie tylko pełny opis systemu wbudowanego i zadań, które ma on wykonywać, ale może również zawierać zestaw ograniczeń w postaci zbioru nierówności. To znacznie rozszerza możliwości optymalizacji w przypadku nieprzewidzianych zadań. Jednakże w modelu przyjętym w tej rozprawie, opartym na reprezentacji grafowej, nie widać możliwości uwzględniania ograniczeń w postaci nierówności na różnych etapach, poza czasem lub kosztami wykonywania wszystkich zadań.

Weryfikacja systemów wbudowanych jest zagadnieniem trudnym, nie ma tu możliwości pełnego sprawdzenia zachowania systemu we wszystkich sytuacjach. W procesie kosyntezy dokonuje się alokacji zasobów, przydziału i szeregowania zadań. W rozdziale czwartym autor omawia algorytmy konstrukcyjne i rafinacyjne, oparte na metodach szukania optymalnych rozwiązań rozwiniętych w sztucznej inteligencji. Liczne publikacje stosowały w tym celu heurystyczne metody oparte na metodach rojowych, algorytmach genetycznych, symulowanym wyżarzaniu, metodzie optymalizacji tabu i wielu innych algorytmach. Programowanie genetyczne omówione zostało bardziej szczegółowo. Nowym problemem dla algorytmów kosyntezy, badanym od 2015 roku przez mgr Górskiego, jest przydział nieprzewidzianych zadań do realizacji przez wbudowane systemy rozproszone.

Genotyp systemu wbudowanego przedstawiony został w postaci drzewa, którego każdy węzeł reprezentuje gen odpowiedzialny za jakąś decyzję projektową. Drzewo to podlega ewolucji przez klonowanie, mutacje i krzyżowanie, zachowując stałą strukturę. Z każdym z tych trzech procesów związany jest parametr, dodatkowym parametrem jest wielkość pokolenia, warunek stopu i maksymalny czas wykonywania wszystkich zadań. Poza tym sposób alokacji jednostek obliczeniowych i magistral komunikacyjnych możliwy jest na kilka sposobów. W obu przypadkach wybrano 4 sposoby alokacji, przypisując im arbitralnie prawdopodobieństwa wyboru 0.5, 0.2, 0.2, 0.1. Rysunek 4 ilustruje sposób alokacji nowych zasobów

lub wykorzystania starych w algorytmie konstrukcyjnym ProGen. Jest na nim 18 arbitralnie wybranych prawdopodobieństw wyboru różnych opcji związanych z alokacją zasobów. Razem z 6 parametrami sterującymi ewolucją mamy 24 parametry. Klonowanie, mutacja i krzyżowanie można wykonać na kilka sposobów. Algorytm rafinacji genotypu jest nieco prostszy, jeśli nie liczyć samego procesu tworzenia wyjściowej architektury, czyli embrionu podlegającego modyfikacji. Każdy węzeł drzewa genotypu reprezentuje funkcję modyfikującą lokalnie architekturę systemu. W sumie otrzymujemy algorytm o dużej złożoności, ale większość parametrów jest w nim arbitralnie ustalona.

Rozdział 7 omawia przydział nieprzewidzianych zadań. Architektura systemu jest ustalona dla zadań planowanych, więc można jedynie programowo zmienić przydział zadań do procesorów. Przypadkiem szczególnym jest dodanie nowych zadań po zakończeniu wszystkich planowanych zadań, w przypadku ogólnym zadania mogą się pojawić w dowolnym momencie. Dla obu przypadków zaproponowano rafinacyjne algorytmy. Jest to najbardziej nowatorska część tej rozprawy. Ograniczeniem jest jednak konieczność pełnego opisu nieprzewidzianych zadań w bazie danych zasobów, co w praktyce nie zawsze jest możliwe. W niektórych zadaniach mamy jedynie oszacowania złożoności procesów. Tu do szeregowania zadań zastosowano iloczyn czasu i kosztów dla danego procesora. Zmiana procesora wymaga by osiągnięty został minimalny zysk. W ogólnym przypadku konieczna jest reorganizacja całego procesu z uwzględnieniem zadań nieprzewidzianych. Jeśli takie zadania pojawią się na skutek wykonania części zadań planowanych ich pełny opis może nie być znany.

Rozdział 8 jest najbardziej obszerny i przedstawia wyniki eksperymentów numerycznych z użyciem siedmiu grafów stosowanych w benchmarkach, od 10 do 110 wierzchołków. Ponieważ używane algorytmy są probabilistyczne uśredniono wyniki po 30 przebiegach, zbadano na grafie o 50 wierzchołkach wpływ różnych operatorów genetycznych, ustalonego ograniczenia czasowego, oraz wpływ wielkości pokolenia początkowego kontrolowanego przez parametr α, przyjmując 7 wartości tego parametru. Parametr α zmienia czas i koszty wykonania od kilku do kilkunastu procent, pomimo generacji ponad pół miliona rozwiązań. Największe różnice są dla ekstremalnych wartości α, ale zwiększanie tego parametru powyżej wartości 90-100 nie przynosi istotnych zysków. Te oceny pokazały niewielkie różnice czasu i kosztów wykonania dla różnych wielkości pokoleń początkowych, ale również pomiędzy najlepszymi i najgorszymi rozwiązaniami.

Parametr β kontrolujący proces klonowania miał wartości 0.1, 0.2, 0.3 i 0.4, i stosowany był tylko dla grafów o 30, 50 i 70 wierzchołkach z różnymi ograniczaniami czasowymi. Wraz ze wzrostem tego parametru generowana była coraz większa liczba pokoleń, czas się wydłużał, pomimo tego różnice pomiędzy najlepszymi i najgorszymi wynikami były niewielkie. W wyniku tych eksperymentów ustalono wartość β=0.2. Wpływ mutacji, kontrolowanych podobnie jak klonowanie, na czas wykonania był minimalny, a na koszty niewielki. Minimalna wartość parametru δ=0.1 okazuje się wystarczająca. Parametr γ kontrolujący proces krzyżowania zmieniano od 0.5 do 0.8, co 0.1. Również w tym przypadku różnice pomiędzy najlepszymi i najgorszymi wynikami w zależności od wartości tego parametru wynosiły od kilku procent do maksymalnie 11%. Po analizie wyników przyjęto γ=0.7. Parametr określający warunek stopu określa po ilu pokoleniach, w których nie było istotnej poprawy należy przerwać proces poszukiwania rozwiązań. Koszty rozwiązań maleją monotonicznie wraz ze wzrostem tego parametru od 2 do 5, ale też liczba generowanych rozwiązań dla grafu o 50 wierzchołkach zbliżała się do miliona już przy wartości 4. Zwiększanie tego parametru powyżej 5 jest obliczeniowo kosztowne, chociaż może nadal nieznacznie poprawić wyniki.

Podrozdział 8.6 zawiera porównanie wyników algorytmu ProGen z ustalonymi parametrami na grafach mających od 10 do 110 wierzchołków z wynikami algorytmów EWA oraz Yen-Wolf. Niestety nie ma wyjaśnienia jak te algorytmy działają, ani uwag na temat innych algorytmów nadających się do porównań. O ile czas wykonania wszystkich zadań jest dla wszystkich algorytmów zbliżony, to koszty różnią się znacznie, w większości przypadków na korzyść algorytmu ProGen, jak pokazano na Wykresie 3. Algorytm Yen-Wolf słabo sobie radził z większymi grafami. Tylko dla grafów o 60 i 80 wierzchołkach algorytm EWA znalazł rozwiązania o niższych kosztach. Nie jest jasne dlaczego algorytm ProGen dla tych przypadków nie znalazł lepszych rozwiązań. Być może ustalenie jednakowych parametrów dla wszystkich grafów nie jest najlepszą strategią, a być może wynika to z jakichś własności struktury tych grafów, które należałoby przeanalizować.

Podrozdział 8.7 zawiera opis eksperymentów z algorytmem RafGen. Wyniki pod względem czasów i kosztów są zbliżone do ProGen, przy mniejszej liczbie pokoleń. Podobnie jak poprzednio dla grafów zawierających 60 i 80 wierzchołków widać duży skok kosztów. Również na wykresach 5 i 6, obrazujących czas obliczeń widać, że dla 80 wierzchołków jest on wyraźnie dłuższy niż dla 90. Skok czasu obliczeń dla 110 wierzchołków sugeruje szybki wzrost złożoności obliczeń dla optymalizacji dużych systemów, ale na podstawie tych wykre-

sów nie można oszacować jak złożone problemy daje się rozwiązać za pomocą szybszych komputerów.

Ostatnie dwa podrozdziały dotyczą nieprzewidzianych zadań w przypadku szczególnym i ogólnym. Eksperymenty wykonano zakładając dwa procesory jednego typu i dwa drugiego, oraz od 10 do 50 niespodziewanych zadań, które zostały zdefiniowane przez grafy zadań. Algorytm rafinacyjny dla przypadku szczegółowego radził sobie wyraźnie lepiej z kosztami, ale nieco gorzej z czasem wykonywanych zadań w stosunku do algorytmu zachłannego. Ponieważ Autor nic o algorytmie zachłannym nie napisał podejrzewam, że był on nastawiony na procesy szukania rozwiązań opartych tylko na czasie wykonania, a nie na kosztach.

Dla testowania algorytmu rafinacyjnego w ogólnym przypadku nieprzewidzianych zadań użyto tych samych grafów co dla przypadku szczególnego, dodając wszystkie przewidziane zadania. Grafy zdefiniowane są przez listy sąsiedztwa podane w Dodatku. Szkoda, że nie ma tam samych grafów, co można by łatwiej przeanalizować. Wyniki są podobne jak w szczegółowym przypadku. Algorytm zachłanny znalazł rozwiązania o krótszym czasie wykonania ale większych kosztach. Autor tego nie skomentował, ograniczając się do omówienia wyników dla poszczególnych grafów.

Rozprawę kończy krótki rozdział 9 zawierający dość oczywiste wnioski. Zastosowane algorytmy prowadząc stochastyczny proces szukania optymalnych rozwiązań potrafią unikać lokalnych minimów, ale nie zawsze dają najlepsze rezultaty. Podobnych wyników można się spodziewać używając wielu metod globalnej optymalizacji. Mgr Górski nie podaje niestety przykładów rzeczywistych zastosowań różnych metod kosyntezy, a byłoby rzeczą interesującą dowiedzieć się, jak sobie radzą duże firmy elektroniczne z tym problemem. Ograniczenia sprzętowe często powodują trudności w aktualizacji oprogramowania systemowego do najnowszych wersji, jak to obserwujemy nawet w przypadku stosunkowo nowych urządzeń mobilnych. W rozprawie nie określono również ograniczeń proponowanego podejścia w sytuacjach, w których nieprzewidziane zadania nie mają ustalonych kosztów, lub w trakcie wykonywania zadania pojawia się potrzeba dodatkowych obliczeń, co jest częstym przypadkiem w przypadków obliczeń sterowanych przez dane (data flow).

Pod względem redakcyjnym praca napisana jest poprawnie. Największym mankamentem jest obszerny spis literatury w formie nieuporządkowanej ani alfabetycznie ani numerycznie, tylko w kolejności cytowania. Są też drobne usterki, sklejone wyrazy i brak numerów stron. Znalazłem troch błędów stylistycznych, np. „wartości poszczególnych wartości” (s. 15), „za-

stawiane za kosztem” (s. 22), „ranking za kosztem” (s. 30), „pozostają nie zostaną przydzielone” (s. 40), „było znacznie tańszy” (s. 60), itp. Zdanie „Każdy węzeł odpowiada funkcjom konstrukcyjnym odzwierciedla strukturę grafu zadań – posiada tyle samo węzłów” pozbawione jest sensu.

Wyrażone powyżej uwagi krytyczne nie zmieniają faktu, że jest to praca dotycząca aktualnej i bardzo ważnej tematyki, a jej rezultaty opublikowane zostały w artykułach na liczących się konferencjach. Przeprowadzono liczne eksperymenty komputerowe z algorytmami rozwojowego programowania genetycznego pokazując, że takie algorytmy sprawdzają się dobrze w poszukiwaniu quasi-optymalnych rozwiązań problemu kosyntezy, a algorytm rafinacyjny pozwala na modyfikacje optymalizujące wykonanie zadań nieprzewidzianych na etapie projektowania. Cele pracy zostały więc osiągnięte, a ustawowe wymogi dotyczące prac doktorskich spełnione. Dlatego wnioskuję o dopuszczenie jej Autora do dalszych etapów przewodu doktorskiego.

Włodzisław Duch,

Toruń, 30 sierpień 2019