Efektywność SMT solverów dla klasycznych problemów NP-trudnych

Effectiveness of SMT solvers for classical NP-hard problems

Tetiana Mossur

Promotor: *dr hab. Andrzej Zbrzezny prof. UJD*

Uniwersytet Jana Długosza w Częstochowie

10.04.2024

Cel i zakres pracy

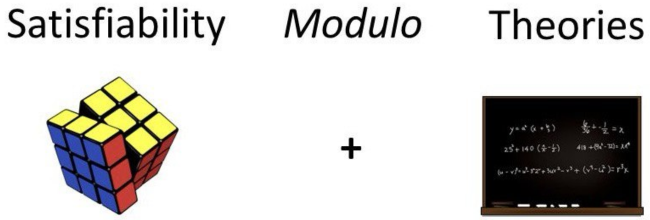
Celem niniejszej pracy było zbadanie skuteczności SMT solverów w rozwiązywaniu ośmiu klasycznych problemów NP-trudnych, a mianowicie: Ścieżka Hamiltona w grafie skierowanym oraz nieskierowanym, Pokrycie wierzchołkowe, Problem maksymalnej kliki, Problem maksymalnego zbioru niezależnego, Problem Komiwojażera, Kolorowanie grafu, oraz Problem sumy podzbioru.

Zakres pracy:

- zakodowanie problemów przy pomocy formuł logiki pierwszego rzędu
- analiza porównawcza działania trzech SMT solverów: Z3, Yices i cvc5

Motywacja

- **Kluczowa Rola w Informatyce:** W obszarach takich jak weryfikacja sprzętu i oprogramowania, wiele problemów można sprowadzić do sprawdzenia spełnialności formuł w logikach bardziej złożonych niż logika propozycjonalna, co wymaga zaawansowanych metod rozwiązania.
- **Rozwój Technologii Solverów SMT:** Ewolucja technologii SMT solverów umożliwiających rozwiązywanie problemów Satisfiability Modulo Theory (SMT) doprowadziła do szybkiego rozwoju tej dziedziny badawczej.
- **Zastosowania Praktyczne:** Rozwiązania SMT znalazły zastosowanie w różnych obszarach praktycznych, takich jak weryfikacja procesorów, analiza statyczna, generowanie przypadków testowych oraz optymalizacja, co sprawia, że badania w tej dziedzinie są kluczowe dla rozwoju nowych technologii i innowacji.



- SMT łączy problem spełnialności logicznej z różnymi teoriami matematycznymi.
- Zadaniem jest stwierdzenie, czy istnieją wartości zmiennych spełniające zarówno ograniczenia logiczne, jak i dodatkowe z teorii matematycznej.

Zastosowanie SMT w praktyce

- Weryfikacja Mikroprocesorów
- Sprawdzanie Równoważności Mikrokodu
- Testowanie Biało-Skrzynkowe
- Eksploracja Przestrzeni Projektowej
- Synteza Konfiguracji
- Odkrywanie Materiałów Kombinatorycznych
- Problemy NP-Trudne

Problemy NP-trudne

Definicja: Problemy NP-trudne to klasy problemów, które są co najmniej tak trudne jak najtrudniejsze problemy w klasie NP.

Klasyfikacja:

- *Klasa P*: Problemy decyzyjne rozwiązywalne w czasie wielomianowym przez algorytmy deterministyczne.
- *Klasa NP*: Problemy, których rozwiązania mogą być zweryfikowane w czasie wielomianowym przez algorytmy niedeterministyczne.

Cechy:

- Możliwość zredukowania każdego problemu w klasie NP do problemu NP-trudnego w czasie wielomianowym.
- Wymagają dużego nakładu obliczeniowego do rozwiązania i weryfikacji, nawet dla relatywnie niewielkich instancji problemów.

SMT solvery

- Z3
- Yices
- cvc5

Kodowanie problemów

- Każdy problem analizowany w pracy został zamodelowany w postaci bezkwantyfikatorowych formuł logiki pierwszego rzędu jako zestaw ograniczeń logicznych, definiujących jego specyfikację i warunki rozwiązania.
- W celu przeprowadzenia eksperymentów, problemy zostały zaimplementowane w języku Python, korzystając z wyspecjalizowanej biblioteki Z3. Kodowanie problemów odbyło się zgodnie z formułami.
- Biblioteka Z3 została wykorzystana do manipulacji oraz rozwiązywania ograniczeń logicznych, co umożliwiło skuteczną implementację algorytmów i procedur rozwiązujących problemy z zakresu kwantyfikacyjnej logiki boolowskiej.

Ścieżka Hamiltona w grafie skierowanym

Definicja

- Dany graf skierowany $G = (V, E)$, gdzie V to zbiór wierzchołków, a E to zbiór krawędzi.
- Zadaniem jest stwierdzenie, czy graf G zawiera ścieżkę Hamiltona.
- Ścieżka Hamiltona to sekwencja wierzchołków $s_0, s_1, ..., s_{n-1}$ przechodzącą przez każdy wierzchołek dokładnie raz.

Zastosowania

- Istotny w informatyce, telekomunikacji i bioinformatyce.
- Kluczowy dla analizy sieci i trasowania w systemach komunikacyjnych.

Używamy n zmiennych $v_0, ..., v_{n-1}$, gdzie n to liczba wierzchołków.

Formuły uwzględniające warunki dla zmiennych:

- Zakres zmiennych v_j: $proper\_numbers(n) = \left(\bigwedge_{j=0}^{n-1}(v_j \geq 0 \wedge v_j < n)\right)$.
- Unikalność zmiennych: $distinct\_vs(n) = \left(\bigwedge_{i=0}^{n-1}\bigwedge_{j=i+1}^{n}(v_i \neq v_j)\right)$.
- Sprawdzenie krawędzi grafu:
  $dir\_edges(n, E) = \left(\bigwedge_{i=0}^{n-1}\bigvee_{(s,t)\in E}(v_i = s \wedge v_{i+1} = t)\right)$.

Cała Formuła:

$$HamPath(n, E) = proper\_numbers(n) \wedge distinct\_vs(n) \wedge dir\_edges(n, E)$$

Ścieżka Hamiltona w grafie nieskierowanym

Warunki dla zmiennych:

- Zakres zmiennych v_j: $proper\_numbers(n)$.
- Unikalność zmiennych: $distinct\_vs(n)$.
- Możliwość przechodzenia przez krawędź w obie strony:
  $edges(n, E) = \left(\bigwedge_{i=0}^{n-1} \bigvee_{\{s,t\} \in E} (v_i = s \wedge v_{i+1} = t) \vee (v_i = t \wedge v_{i+1} = s) \right)$.

Formuła:

$$UHamPath(n, E) = proper\_numbers(n) \wedge distinct\_vs(n) \wedge edges(n, E)$$

Problem maksymalnej kliki w grafie nieskierowanym

Definicja

- Dany graf nieskierowany $G = (V, E)$, gdzie V to zbiór wierzchołków, E to zbiór krawędzi.
- Klika to pełny podgraf, gdzie każde dwa wierzchołki są połączone krawędzią.
- Maksymalna klika $C \subseteq V$ to taka, która zawiera największą możliwą liczbę wierzchołków.

Zastosowania

- Istotny w teorii grafów, sieciach społecznościowych, analizie sieci.
- Wykorzystywany w problemach planowania tras, optymalizacji sieci, analizie zależności między obiektami.

Problem maksymalnej kliki w grafie nieskierowanym: Kodowanie

Używamy k zmiennych $v_0, ..., v_{k-1}$, gdzie $0 < k \leq n$.

Warunki dla zmiennych:

- Zakres zmiennych v_j: $proper\_numbers(n)$.
- Unikalność zmiennych: $distinct\_vs(n)$.
- Każda para zmiennych v_i i v_j jest połączona krawędzią.

Formuła:

$$MaxClique(n, E, k) = proper\_numbers(k) \wedge distinct\_vs(k) \wedge$$
$$\left(\bigwedge_{i=0}^{k-1} \bigwedge_{j=i+1}^{k} \bigvee_{\{s,t\} \in E} ((v_i = s \wedge v_j = t) \vee (v_j = s \wedge v_i = t)) \right)$$

Problem maksymalnego zbioru niezależnego w grafie nieskierowanym

Definicja

- Dany graf nieskierowany $G = (V, E)$, gdzie V to zbiór wierzchołków, E to zbiór krawędzi.
- Niezależny zbiór to podzbiór wierzchołków, gdzie żadne dwa nie sąsiadują ze sobą.
- Problem maksymalnego zbioru niezależnego polega na znalezieniu największego takiego zbioru w grafie.

Zastosowania

- Istotny w teorii grafów i algorytmice.
- Ma zastosowanie w wielu problemach optymalizacyjnych i analizie sieci.

Problem maksymalnego zbioru niezależnego w grafie nieskierowanym: Kodowanie

Używamy n zmiennych $v_0, ..., v_{n-1}$, gdzie n to liczba wierzchołków w grafie.

Warunki dla zmiennych:

- Zakres zmiennych v_j: $proper\_numbers(n)$.
- Unikalność zmiennych: $distinct\_vs(n)$.
- Brak krawędzi między wierzchołkami.

Formuła:

$$MaxIndSet(n, E, k) = proper\_numbers(n) \land distinct\_vs(n) \land$$
$$\left(\bigwedge_{i=0}^{k-1} \bigwedge_{j=i+1}^{k} \bigvee_{\{s,t\} \in E} \neg((v_i = s \land v_j = t) \lor (v_j = s \land v_i = t)) \right)$$

Problem pokrycia wierzchólkowego

Definicja

- Pokrycie wierzchołkowe grafu nieskierowanego $G = (V, E)$ to podzbiór $V' \subseteq V$, gdzie każda krawędź ma przynajmniej jeden koniec w V'.
- Rozmiar pokrycia to liczba wierzchołków w nim zawartych.
- Problem polega na znalezieniu minimalnego pokrycia wierzchołkowego w danym grafie.

Zastosowania

- Istotny w optymalizacji grafów oraz problemach planowania tras.
- Stosowany w analizie sieci, zarządzaniu zasobami oraz problemach pokrycia.

Problem pokrycia wierzchółkowego: Kodowanie

Używamy k zmiennych $v_0, \ldots, v_{k-1}$, gdzie $0 < k \leq n$.

Warunki dla zmiennych:

- Zakres zmiennych v_j: $proper\_numbers(k)$.
- Unikalność zmiennych: $distinct\_vs(k)$.
- Dla każdej krawędzi w E co najmniej jeden z jej końców należy do V'.

Formuła:

$$VertexCover(n, E, k) = proper\_numbers(k) \wedge distinct\_vs(k) \wedge$$
$$\left(\bigwedge_{\{s,t\} \in E} \left(\bigvee_{j=0}^{k-1} (v_j = s \vee v_j = t) \right) \right)$$

Definicja

- Problem polega na określeniu minimalnej liczby kolorów potrzebnych do pokolorowania grafu nieskierowanego $G = (V, E)$.
- Dwa sąsiednie wierzchołki nie mogą mieć tego samego koloru.

Zastosowania

- Modelowanie problemu kolorowania map za pomocą grafu, w którym każdy wierzchołek reprezentuje kraj, i wierzchołki, których kraje mają wspólną granicę, ze sobą sąsiadują.
- Istotny w planowaniu tras, optymalizacji sieci oraz problemach planowania.

Używamy n zmiennych $c_0, ..., c_{n-1}$,gdzie n to liczba wierzchołków w grafie.

Warunki dla zmiennych:

- Każda zmienna c_j reprezentująca kolor wierzchołka j przyjmuje wartość z przedziału od 1 do k, gdzie k to liczba kolorów.
- Dla każdej krawędzi s, t w E, kolor wierzchołka s jest różny od koloru wierzchołka t.

Formuła:

$$GraphColoring(n, E) = \left(\bigwedge_{j=1}^{n} (c_j \geq 1 \wedge c_j \leq k) \right) \wedge \left(\bigwedge_{\{s,t\} \in E} (c_s \neq c_t) \right)$$

Definicja

- Problem polega na znalezieniu trasy, która minimalizuje sumaryczny koszt podróży, odwiedzając każde miasto dokładnie raz i kończąc w mieście początkowym.
- Dany graf nieskierowany $G = (V, E)$, gdzie wierzchołki reprezentują miasta, a krawędzie odpowiadają możliwym połączeniom między nimi. Koszt podróży z miasta i do j jest reprezentowany przez wagę $c(i, j)$.
- Celem jest znalezienie trasy o minimalnym koszcie, która odwiedza każde miasto dokładnie raz.

Zastosowania

- Logistyka, planowanie tras, optymalizacja sieci komunikacyjnych.

Używamy n zmiennych $v_0, ..., v_{n-1}$, gdzie n to liczba miast.

Warunki dla zmiennych:

- Właściwą wartość i unikalność.
- Istnienie krawędzi między kolejnymi wierzchołkami w trasie oraz zamknięcie trasy.
- Ograniczenie sumy wag krawędzi na trasie do wartości k.

Formuła:

$$TSP(n, c, E, k) = proper\_numbers(n) \wedge distinct\_vs(n) \wedge edges(n, E)$$
$$\wedge \left(\bigvee_{\{s,t\} \in E} ((v_{n-1} = s \wedge v_0 = t) \vee (v_0 = s \wedge v_{n-1} = t)) \right) \wedge \left(\bigwedge_{\{s,t\} \in E} \sum c(s, t) \leq k \right)$$

Definicja

- Problem sumy podzbioru polega na znalezieniu podzbioru $S' \subseteq S$, którego elementy sumują się dokładnie do wartości docelowej t.

Zastosowania

- Wykorzystywany w optymalizacji kombinatorycznej, analizie danych oraz w algorytmach szukających rozwiązań problemów optymalizacyjnych.
- Stosowany w bioinformatyce do analizy sekwencji genetycznych.
- W praktyce wykorzystywany w planowaniu finansowym, analizie portfela inwestycyjnego, czy w układaniu harmonogramów.

Używamy n zmiennych $x_0, ..., x_{n-1}$, gdzie n to liczba elementów w S.

Warunki dla zmiennych:

- każda zmienna x_j przyjmuje wartość logiczną 0 lub 1.
- suma iloczynów x_i i s_i dla wszystkich k elementów ze zbioru S jest równa wartości t.

Formuła:

$$SubsetSum(n, t) = \left(\bigwedge_{j=0}^{n-1} (x_j = 0 \lor x_j = 1) \right) \land \left(\sum_{i=0}^{n} (x_i * s_i) = t \right)$$

Generowanie danych wejściowych

- Do eksperymentów użyto laptopa Dell z procesorem Intel Core i5-1135G7 z częstotliwością 2.40GHz i 16 GB RAMu.
- Generowanie danych wejściowych, takich jak grafy, wykonano przy użyciu biblioteki igraph, która zapewnia wszechstronne możliwości manipulacji i analizy grafów.
- Zdecydowano się generować dwa rodzaje grafów - Barabasi-Alberta i Erdos-Rényi'ego - aby umożliwić badanie efektywności SMT-solverów w różnych warunkach.
- Do eksperymentów nad problemem SubsetSum wygenerowano zestawy losowych liczb całkowitych w określonym zakresie.

Identyfikacja czynników wpływających na efektywność

- Rozmiar Instancji Problemu: Wraz z rosnącym rozmiarem problemu, np. liczbą wierzchołków w grafie lub elementów w zbiorze, czas rozwiązania znacząco wzrasta.
- Struktura Grafu: Sposób generowania grafu oraz jego złożoność mogą wpływać na wydajność obliczeniową solverów.
- Rodzaj Problemu: Różnice w wydajności solverów zależą od rodzaju problemu NP-trudnego, np. SubsetSum vs. problemy grafowe.
- Zużycie Zasobów: Wartość czasowa i pamięciowa solverów różni się w zależności od problemu oraz zastosowanej strategii rozwiązania.
- Trudność Spełnialności vs. Niespełnialności: Szukanie niespełnialności może być trudniejsze niż spełnialności, co wpływa na czasochłonność rozwiązania problemu.

Wnioski

Po przeprowadzeniu eksperymentów oraz analizie czynników wpływających na efektywność solverów w rozwiązywaniu problemów NP-trudnych, można wyciągnąć kilka istotnych wniosków:

- Solver Z3 wykazał się jako najbardziej uniwersalny i efektywny w rozwiązywaniu różnorodnych problemów NP-trudnych.
- Przy rozwiązywaniu problemów NP-trudnych ważne jest efektywne zarządzanie zasobami pamięciowymi.

Wnioski te mogą być przydatne dla praktyków oraz badaczy zajmujących się problemami NP-trudnymi, pomagając im wybrać odpowiedni solver oraz zoptymalizować proces rozwiązywania problemów w praktyce.

Podsumowanie

- Dzięki analizie eksperymentalnej zyskałam głębsze zrozumienie wydajności solverów SMT w różnych scenariuszach, co jest istotne zarówno dla praktyki, jak i badań naukowych.
- Wyzwaniem naukowym jest rozwój kodowania szerszej gamy problemów NP-trudnych i przeprowadzenie eksperymentów na bardziej zróżnicowanych danych wejściowych. To pozwoliłoby na pełniejsze zrozumienie możliwości i ograniczeń solverów SMT oraz ich praktyczne zastosowanie.
- Wszystkie pliki, kod źródłowy, eksperymenty i inne zasoby związane z niniejszą pracą znajdują się na platformie GitHub.