data

logo

szkoła
x
x

Sprawozdanie

Metody Programowania

Rok akademicki x
Semestr x
Autor: x
Kierunek: x
Grupa x
Laboratorium x
Data zajęć: x
Temat: Algorytmy wyszukiwania lokalnego i z listą tabu
Prowadzący: x

Ocena

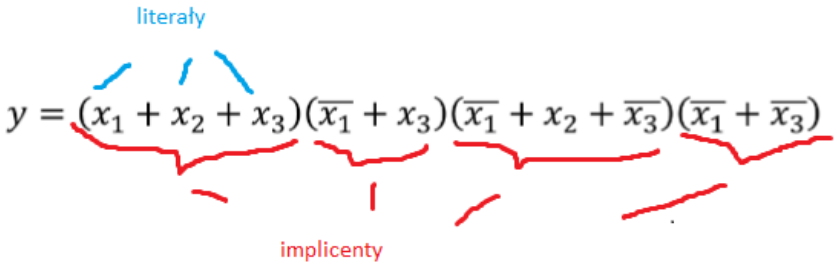
Spis treści

1) Wprowadzenie teoretyczne

Wyszukiwanie lokalne – jest techniką stosowaną przy rozwiązywaniu problemów optymalizacyjnych. Przestrzeń rozwiązań problemu jest przeszukiwana iteracyjnie. W każdym kroku przechodzimy od najlepszego znalezionego dotąd rozwiązania do kolejnego tymczasowego optimum, wyznaczonego w jego sąsiedztwie. Otoczenie (sąsiedztwo) bieżącego rozwiązania musi być odpowiednio zdefiniowane. Algorytm wyszukiwania lokalnego nie gwarantuje osiągnięcia optimum globalnego, najczęściej uzyskane rozwiazanie jest optymalne lokalnie

Problem spełnialności (SAT) - zagadnienie rachunku zdań, określające czy dla danej formuły logicznej istnieje takie podstawienie (wartościowanie) zmiennych zdaniowych, żeby formuła była prawdziwa. Jest problemem NP.-zupełnym.

Funkcje boolowskie można zapisać za pomocą implicentów połączonych znakiem koniunkcji, a implicenty składają się z literałów połączonych znakiem alternatywy (postać koniunkcyjna normalna).



Algorytmy wyszukiwania lokalnego dla problemu spełnialności (różnią się wyborem zmiennej do negacji)

a) GSAT – wybiera zmienną, której negacja minimalizuje liczbę implicentów o wartości “0”. W dalszej części sprawozdania algorytm zostanie zmodyfikowany o dodanie listy tabu.
b) WalkSAT - wybiera losowo implicent o wartości “0” przy aktualnym przyporządkowaniu a następnie neguje w tym implicencie losową zmienną, zmieniając wartość tego implicentu na “1”.

Algorytmy mają zastosowanie do formuł boolowskich w postaci konjunkcyjnej normalnej. Kontynuują swoje postępowanie aż cała formuła przyjmie wartość „1”.

Wyszukiwanie z listą tabu (tabu search) – algorytm stosowany do rozwiązywania problemów optymalizacyjnych. Podstawową ideą algorytmu jest przeszukiwanie przestrzeni, stworzonej ze wszystkich możliwych rozwiązań, za pomocą sekwencji ruchów. W sekwencji ruchów istnieją ruchy niedozwolone, ruchy tabu. Algorytm unika oscylacji wokół optimum lokalnego dzięki przechowywaniu informacji o sprawdzonych już rozwiązaniach w postaci listy tabu (TL).

2) Problemy do rozwiązania

Należy napisać program implementujący algorytm WalkSAT oraz GSAT z listą tabu.
Dane wejściowe / wyjściowe powinny zostać wprowadzone / wyprowadzone z pliku lub z konsoli.
Implicenty można reprezentować w postaci wektorowej (np.$x_1 + \overline{x_2} + x_4$ można zapisać w postaci: 10-11, gdzie 1 oznacza zwykłą zmienną, 0 zanegowaną, -1 brak) co umożliwi dalsze pisanie algorytmu.
Implicenty nie powinny się powtarzać.
Jeśli wynik nie zostanie znaleziony po określonej liczbie iteracji program należy zakończyć z informacją „stop_bez_rozwiazania”.
Program powinien mieć możliwość generowania instancji problemu losowo.
Należy sprawdzić poprawność napisanych programów.
W celu porównania algorytmów należy wykonać ich zestawienie w postaci tabeli.

3) Program implementujący algorytm WalkSAT

a) Pseudokod

wejście: n – ilość zmienncyh N – ilość (różnych) implicentów implicentsVector – tablica 2d zawierająca implicenty

wyjście: logicalVars – tablica zawierająca wyznaczone wartości x

Opis działania:

1. Przyporządkuj losowe wartości logiczne wszystkim zmiennym i zapisz do tablicy implicentsVars
2. Wykonuj w nieskończoność
    a. Sprawdź wartości implicentów
    b. Jeśli wszystkie implicenty przyjmują wartości 1
        i. Wypisz zawartość tablicy implicentsVars
        ii. Zakończ program
    c. Wybierz losowy implicent o wartości 0 i zaneguj w nim losową zmienną
    d. Jeśli liczba iteracji przekroczy 10000
        i. Wypisz „stop_bez_rozwiazania”
        ii. Zakończ program

b) C++

Program został napisany w języku C++, obok linijek z kodem (lub nad nimi) po sekwencji „//” znajdują się komentarze do danego fragmentu.

Program został skompilowany oraz uruchomiony na sprzęcie/oprogramowaniu:

procesor CPU: Intel i5-4210H

system operacyjny: Windows 10 Home wersja: 20H2

kompilator: g++ wersja 9.2.0

Kod źródłowy znajduje się w pliku *main.cpp*

c) Przykładowe wejścia / wyjścia z programu

1.

Wejście:

n = 8 N = 100

Implicenty

-x0-x1-x4-x5-x6-x7	x1-x3-x5-x6	x1-x3-x4-x5-x7	x0x1x2-x3-x4x5x6x7
x2x3-x4x5x7	x0x1x2-x5x7	x1x3x4-x6	-x0-x2-x3x4x5x6
-x3-x4-x6	-x0x2-x3-x4-x6x7	x2x3-x4x5-x7	x0x1x2-x3
-x0-x1-x5x6	x1-x5x6x7	-x0-x1-x3x4x5x7	x1x2x3x4x6-x7
-x0x2x3x4-x7	-x0x1x3x5-x6	x1-x2x3x5x6	-x0x1x2x3x6-x7
-x2-x3x5-x6	-x0-x2x4x5-x7	-x0-x1x2x3x4x5	x0-x2x3x4x6
x0-x2-x4-x6-x7	-x0-x1x2-x3x4-x7	-x0-x1x2-x4-x6	x2x4x6-x7
x1-x2-x5-x6	-x0-x2x3-x5	x0x1-x2-x3-x4x5x7	-x1x2x3x4-x5
-x0-x1-x4x6	-x0-x2x4-x5-x7	-x0x1x5x6-x7	x0x2-x4-x5x6-x7
-x3-x4x5-x6-x7	x0-x1-x4-x5x6	x0x1-x2x3x5x7	x1x2x3-x5-x6
x3x4	x0x2-x3x4-x5x6	x1-x2x3x4-x6-x7	x0-x1x2x5
x0-x1x3-x5x7	-x1x3x4x6-x7	-x1x4x5x6	x0x1x2-x3x4-x5-x6
x1x2x3x4x6x7	x0-x1x2-x4-x5x6-x7	x0x1x2-x3-x4x5-x6	-x0-x2x3x4x6
x0x2-x3-x5x7	-x0-x1x2x3-x4-x5-x6x7	x0x1x2-x3x6x7	-x1x3x7
x3x4-x5x7	-x1x2-x5-x6	-x0-x1-x2-x3-x6x7	-x0-x3-x4x6-x7
x0-x4x5x6x7	x0-x1x2-x4x5-x6x7	x1-x2-x4x5-x6x7	x0-x1x4-x5-x6x7
x0x1x2-x3-x4-x6	-x0-x3x4x6-x7	x0-x1-x2-x4-x5x6x7	x0x1-x3x7
x0x1-x3-x4x5-x7	x1x3-x4x5	-x0x1x2x3-x4x5x6x7	x0x1x2x3x4x5x6-x7
-x0-x2-x3-x4x5-x7	x0x1x2x3x4-x6	x1x3-x4x5x7	-x0-x1x3-x4x7
x1-x3-x4x5x7	-x1x3-x4-x5x7	x0x1x3x4x5-x6x7	x0-x1-x4-x5x6-x7
x0-x2-x4x5x6-x7	-x0x2-x3x5-x6	x0-x1x3x4-x5-x6-x7	x0x1x3-x4x5-x7
x0-x1-x3-x4x5-x6-x7	x1x2-x3x4-x5-x6	-x0x1-x2x4-x5	-x1-x2x3x4-x5-x7
x2-x3-x4-x5-x6x7	x1-x2x4-x5x7	-x0x1-x3-x4x5x6	-x0x1x2-x5-x7
-x1x3-x4x5x6	-x0x1x2x3-x4x6x7	-x0-x1x3-x6x7	x0x2-x4x5
x0-x2x3x5-x6-x7	x1-x2x4x5x6-x7	-x1x2x3	-x0-x1-x2-x7

Wyjście

x0=0 x1=1 x2=1 x3=1 x4=1 x5=1 x6=1 x7=0

Wykonano iteracji: 8

Czas wykonania: 4,35s (uwzględniono wypisywanie pośrednich wyników każdej iteracji)

2.

Wejście

n = 8 N = 100

Implicenty:

x1x2-x3-x4x5-x7	-x1x3-x5	-x0-x1x4x5-x6-x7	-x0-x1-x2x4-x5x6
x0x1-x3-x4	x3-x5-x7	-x1-x2-x4-x6x7	-x1x2x3-x4x6-x7
x2	-x3x4x5-x7	x0-x3-x4-x6x7	x0-x1-x3x4x5x6x7
-x1x2x4x6	x0-x1x2-x3x4	-x0-x1-x2-x3x7	x2x3x4-x6-x7
x2-x4x5x6x7	x0-x1-x2-x3-x4-x5x6-x7	-x0x2x3-x5-x6	-x0-x2-x3-x4x5-x6x7
-x0x1x2-x4-x5-x6	-x0-x2-x4-x5-x6	-x0-x1x4-x5x6	x0-x2-x5-x7
x0-x1x2x4-x5x6-x7	-x2x3-x4x5-x7	x1x2-x3x4x6	x0x1x2-x3x6x7
-x0-x1-x2-x3x4x5x6x7	-x1-x4-x5-x6-x7	-x1-x3-x4-x7	x0x1x2x3-x4x5-x7
-x0x1x3x4x6	-x2x4-x5x6-x7	x0x1x2x3x5x6	x0x1x2-x4-x5-x7
-x0-x1-x3-x4-x6	-x0x1-x4	-x0-x1-x2x3-x4x6-x7	x0-x2x3x4-x5x7
-x0-x1-x3-x6	x0-x1x3x5x7	-x0x1x2x3-x6-x7	x0x2x4-x5x6
x1x2x5x6-x7	x1x3x4x5x7	x1-x2-x4-x5-x6x7	-x0-x1-x2
-x0x1-x3-x6-x7	-x1x3x4x5-x6	-x0-x2x3-x5-x6x7	x1x3x4x5x7
-x1x2-x3-x4x5x6-x7	x1x2x3x4-x5-x6	x4x5x6	x0-x1-x3x4-x5x6-x7
x2x3x4x5	x0-x1-x3x4x5	-x0x3x4-x5-x7	x0-x1-x4-x5-x6
-x0-x1-x2x3x4-x6	x0-x1x2-x3-x6x7	x2-x3x4-x6x7	-x1x2x5
-x0-x1-x2x5x7	-x1-x2x4x7	-x0-x1-x4x5-x6	x0-x3-x5x6-x7
-x0-x1x2x5x6-x7	-x0-x1-x3x4-x5-x6	x0x1x3x4x5-x7	-x0x1x2x5x6
x3-x4x7	x0x7	-x0x1x3-x4x5x6x7	x3x4x6x7
x0-x4x6-x7	-x2x3-x4x6	-x3x4x5x7	x1-x2-x3x4x7
x0-x1x2	-x0x1x2x3x4x5-x6-x7	x1x2x3-x6x7	x0x1-x2-x3x4-x5x7
x1x2x4x5-x6-x7	x1-x4-x7	-x0-x1-x2-x3x6	-x0-x1x2-x5
x0-x1-x2x3x4x5-x6x7	x0-x1-x2x3x7	-x0-x1x2x3x5x6-x7	x0x1-x3-x4-x6x7
x0x2-x4x6	-x0-x1-x2-x3-x4x6	-x0-x1x2x3-x4-x5x6x7	-x0-x3x4-x5-x6-x7
-x1-x4-x5x6x7	x0x1-x3x4x6x7	x0x1x2x3x4x5	x0x3-x4-x6-x7

Wyjście

x0=1 x1=1 x2=1 x3=0 x4=0 x5=1 x6=0 x7=1

Wykonano iteracji: 24

Czas wykonania: 8,05s (uwzględniono wypisywanie pośrednich wyników każdej iteracji)

d) Testowanie poprawności

W celu sprawdzenia poprawności programu sprawdzę czy dla wyznaczonych wartości x problem SAT jest spełniony

Test 1

Dane wejściowe (wczytane z pliku)

3

5

-x0

-x0-x2

x0-x1

-x0x2

-x0x1

Wyjście programu

x0=0 x1=0 x2=1

Ocena poprawności: Działa poprawnie, podana sekwencja spełnia problem SAT

Test 2

Dane wejściowe

1

1

x0

-x0

Wyjście programu

stop_bez_rozwiazania

Ocena poprawności: Działa poprawnie, brak rozwiązania dla podanych implicentów

Test 3

Dane wejściowe (wygenerowane losowo)

n=5

N=10

Implicent nr 0 x0x3

Implicent nr 1 -x0-x1x2

Implicent nr 2 x0-x1-x2-x4

Implicent nr 3 -x2x3-x4

Implicent nr 4 -x1x2x3x4

Implicent nr 5 x0-x1-x2x3

Implicent nr 6 -x0x1x2

Implicent nr 7 -x0x1x2

Implicent nr 8 -x0-x1-x3x4

Implicent nr 9 -x0x1-x2x3x4

Wyjście programu

x0=0 x1=0 x2=0 x3=1 x4=0

Ocena poprawności: Działa poprawnie

Test 4

Dane wejściowe (wygenerowane losowo)

n=4

N=7

Implicent nr 0 -x0-x1-x2x3

Implicent nr 1 x0x1x2-x3

Implicent nr 2 -x0x1x2-x3

Implicent nr 3 -x0x2

Implicent nr 4 x1-x2

Implicent nr 5 -x3

Implicent nr 6 -x2

Wyjście programu

x0=0 x1=1 x2=0 x3=0

Ocena poprawności: Działa poprawnie

Wniosek:

Program działa poprawnie

4) Program implementujący algorytm GSAT z listą tabu

a)Pseudokod

wejście: n – ilość zmienncyh N – ilość (różnych) implicentów implicentsVector – tablica 2d zawierająca implicenty t – rozmiar tablicy tabu

wyjście: logicalVars – tablica zawierająca wyznaczone wartości x

Opis działania:

1. Przyporządkuj losowe wartości logiczne wszystkim zmiennym i zapisz do tablicy implicentsVars
2. Wykonuj w nieskończoność
    a. Sprawdź wartości implicentów
    b. Jeśli wszystkie implicenty przyjmują wartości 1
        i. Wypisz zawartość tablicy implicentsVars
        ii. Zakończ program
    c. Dla i=0,1,…, n
        i. Zaneguj zmienną o indeksie i
        ii. Sprawdź wartości implicentów i policz ile jest implicentów o wartości 0, wynik zapisz do pamięci
        iii. cofnij zanegowanie
    d. Wyznacz zmienną której negacja minimalizuje liczbę implicentów o wartości 0
    e. Jeśli zmienna nie znajduje się na liście tabu to zaneguj
    f. Jeśli się znajduje porównaj koszt danej zmiennej z poziomem aspiracji
        i. jeśli koszt < poziomu aspiracji, to zaneguj znalezioną wcześniej zmienną
        ii. inaczej wyznacz inną zmienną do zanegowania i wróć do kroku e
    g. Jeśli liczba iteracji przekroczy 10000
        i. Wypisz „stop_bez_rozwiazania”
        ii. Zakończ program

Koszt wyboru danej zmiennej definiuję jako ilość implicentów o wartości 0 po negacji danej zmiennej.
Poziom aspiracji definiuję jako najlepszy koszt zmiennej, która nie znajduje się na liście tabu

b)C++

Program został napisany w języku C++, obok linijek z kodem (lub nad nimi) po sekwencji „//” znajdują się komentarze do danego fragmentu.

Program został skompilowany oraz uruchomiony na sprzęcie/oprogramowaniu:

procesor CPU: Intel i5-4210H

system operacyjny: Windows 10 Home wersja: 20H2

kompilator: g++ wersja 9.2.0

Kod źródłowy znajduje się w pliku *main.cpp*

c)Przykładowe wejścia / wyjścia z programu

1.

Wejście:

n = 8 N = 100 t=1

Implicenty:

-x0-x1-x2x3x6-x7	-x0x2x3-x4	x0-x1-x2x3x4x5-x6-x7	-x2x3x6x7
x3x4-x6	x2x5-x7	-x0x1x2x3-x5-x6x7	x3x5
-x3x4x5-x6	x1x2x3-x4x5-x6-x7	-x1x2x3x4x5-x6-x7	x0-x1-x2-x3x4x6
x0x1x3-x4x5-x6	-x2x3-x4x5x6-x7	x2-x3-x4x5x6-x7	-x0x1-x3-x4
x3-x5	-x0-x1x2x3-x4x5x6x7	-x1x2-x3-x5x6	x0x2-x3x4-x5-x7
x0x1x2-x3-x4-x5-x7	x0x3-x4-x5-x7	x0x2-x4-x5x6x7	-x0-x1x2-x3x4-x5x6-x7
x1-x2x3-x4x6x7	x0x1x2x3x7	x0x1x2x3x6	-x1x3-x5-x6x7
-x2x3-x5x6	-x0x2-x3x4x5x7	-x0-x2x3-x4x7	x1-x2x3-x5-x7
-x0x1x2-x3-x5x6-x7	-x0-x1-x2x3-x4x5-x6	x0x1x2-x3x7	x0x1-x2x3x4x5-x6
-x0x1x3-x4x5-x6x7	-x2x3-x7	x1x2x3-x4-x5x6-x7	x0-x1x2-x3-x5x6
x0x2-x3-x4x5x7	-x1-x2-x3-x4x5-x6x7	-x0-x1-x4-x5x6-x7	x0-x1-x4-x5x6x7
-x0-x3-x4-x5-x7	x0x3x4x6	x0x1-x2x5x6-x7	-x0-x1x2-x3x4x5-x6
-x0-x1x3-x4x5-x6x7	x2-x5-x6	-x0x2-x5-x6	x0-x1x4x5-x7
-x0-x1-x2x3x5-x6x7	x1x2-x4x6-x7	-x1-x2x4x5-x6-x7	-x0-x1x5-x6-x7
-x0x1-x2-x3-x4x6	x0-x1-x5x6-x7	-x0x1x2x4-x6x7	-x0x1-x2-x6x7
x0-x1x2-x4-x6	-x0-x2x5-x6-x7	-x0x1-x2x3x5-x6x7	x1-x2-x4
x0-x1x3-x4-x5x7	-x1x3-x4-x5-x6-x7	x0x2-x3x4x5x6x7	x1x2-x3x5x7

-x0-x1-x4-x5x6x7	x0-x1-x2-x3x4-x5	x0-x1x2-x4x6	x2-x3-x4-x6
x1x2-x4-x5-x6x7	-x0-x1x2-x3x4-x5	x0-x3x4x5-x6x7	-x2x3x4-x5-x6-x7
x2x3-x5x6-x7	-x0x1-x2-x7	x0x3-x5-x6-x7	x0-x3x4-x6-x7
-x0x1x2x5x6	-x0-x2-x3-x4-x5-x6	-x0-x4-x5x6	-x0-x2-x3x4-x5x6
x0-x1x2-x3x4-x5-x6x7	x1x2-x3-x4x5-x6-x7	x1-x2x4	-x0-x1-x2-x4x5-x6-x7
x1-x2x3-x4x5x6x7	x0x1-x2x5x7	-x0-x1x2-x3-x4-x5-x6x7	-x0x2x4x5x6-x7
x0-x1-x2x3-x4x5-x6x7	x2-x3-x4x5x6	-x2-x3x4-x5x6x7	-x2x3-x4x5-x6-x7
-x0x1x2-x3x4x7	-x0-x2-x3-x4x5x7	-x1x4-x5x6-x7	-x1-x3x7

Wyjście

x0=1 x1=1 x2=1 x3=0 x4=0 x5=1 x6=0 x7=0

Wykonano iteracji: 9

Czas wykonania: 3,98s (uwzględniono wypisywanie pośrednich wyników każdej iteracji)

2.

Wejście

n = 8 N = 100 t=3

Implicenty:

x1-x3-x5	x0x4x7	x1-x2-x4-x5x7	x1-x2-x3-x5x7
-x0-x1x2-x4x5x6	x0x1x2x3x5-x6	-x0x1-x2x4x7	x0x2-x4-x6-x7
x0-x1x4-x5x7	-x0-x1-x2x3x4x5x6x7	-x2-x4x6	x0-x1x2-x3-x4x5-x6
x0-x2x3-x4-x5x7	-x0x1-x4x5x7	x5x7	-x0-x1x3-x5x6-x7
-x0-x1x3x4x6x7	x1x2-x3x4x6	x1x3-x5-x6x7	-x0-x1-x2-x3-x4-x5-x7
-x0x2-x5	x0x1x2-x3x4-x6	-x0x1x2-x3x4x5x6	x1x2x3x4-x5-x6x7
-x0x1x2-x3x5-x6	-x1-x2-x3-x4-x5x6x7	x1-x2x3-x4x5-x6-x7	x0x2x3x4x6-x7
-x0x2x3x5x6x7	-x3x5-x7	x2-x3x5-x6x7	-x0x1-x2-x3-x5x6x7
-x2-x3x6	x5-x7	x0x1x4x6	-x0x1x2
x0-x2x4x5-x7	-x0x1-x2-x3-x4x5x6x7	-x0x1-x3x4x5-x6	-x1-x2-x3-x4x5x6x7
-x0x1-x2x3-x4x5x6	-x0-x1-x3-x5-x6x7	-x1-x2x3-x4-x5-x6x7	-x0x1x2-x6x7
x1x3-x5	x0x2-x3x6-x7	x0-x4x5	-x0-x3-x5-x6-x7
x1-x4-x5-x6	x0x2-x3-x4-x5	-x0-x1x2x5-x6-x7	x1-x3x4x5x7
-x0x1-x3-x5-x6x7	x0x1-x2-x3-x4x5x6-x7	x0-x1x2x3-x4x5x6	-x0x1-x2x4-x7
x0-x1-x2x3-x4-x5-x6x7	-x1x2-x3-x4	-x1x4-x5-x6-x7	x1x2x3x7
x0x3x4x5x6-x7	-x0x1-x2x3x4x6x7	-x0x2x4x6x7	-x0-x1x2x3x5x6x7
-x0x4x6x7	x0x2x3x4-x5x6	-x0-x1-x2-x3-x4x5-x6x7	x0-x2x3x4-x5
-x0x2-x3-x4-x5-x6-x7	x1x2-x3x5x6x7	-x0x4-x5x6	-x1x2x3x4x5x6x7
-x0-x2-x3-x5x6	-x0x1x3x6x7	-x0x2x3-x4-x5-x6-x7	-x2x3-x4x5x6
x1x2x3x4-x6-x7	x1x2-x3-x4x5x7	-x0-x1x2x3-x4x5-x6	-x0-x1-x2-x4x5x6
-x0-x2-x3-x4-x5x7	-x2-x3-x4-x6-x7	-x0-x2x3x4x5-x7	-x0-x2x4x5
-x0-x5x6x7	-x0x2-x3-x5x7	-x2x3-x5-x7	x2-x3x4-x6x7
-x0x1-x2x4x5x6-x7	-x1x2-x3x5-x7	-x1-x2-x3-x5x6-x7	x1x2-x3-x5-x6
-x0x1x2-x5	x0-x1x2-x7	x0x1x2x3-x4x5	x1x2x3-x4x5x6x7
x0-x1x2-x3-x4x5x6x7	-x5x6	-x0x3-x4x5x7	-x1-x2x4x5

Wyjście

x0=1 x1=0 x2=1 x3=0 x4=1 x5=1 x6=1 x7=1

Wykonano iteracji: 7

Czas wykonania: 3,17s (uwzględniono wypisywanie pośrednich wyników każdej iteracji)

d)Testowanie poprawności

W celu sprawdzenia poprawności programu sprawdzę czy dla wyznaczonych wartości x problem SAT jest spełniony

Test 1

Dane wejściowe (wczytane z pliku)

3

5

1

-x0

-x0-x2

x0-x1

-x0x2

-x0x1

Wyjście programu

x0=0 x1=0 x2=1

Ocena poprawności: Działa poprawnie, podana sekwencja spełnia problem SAT

Test 2

Dane wejściowe

2

4

1

x0x1

x0-x1

-x0x1

-x0-x1

Wyjście programu

stop_bez_rozwiazania

Ocena poprawności: Działa poprawnie, brak rozwiązania dla podanych implicentów

Test 3

Dane wejściowe (wygenerowane losowo)

n=5

N=10

t=2

Implicent nr 0 -x0-x1x2x3x4

Implicent nr 1 x0x1x2x3x4

Implicent nr 2 x0-x1-x2x4

Implicent nr 3 x0-x1

Implicent nr 4 x0-x2x3

Implicent nr 5 -x2x3x4

Implicent nr 6 x0-x4

Implicent nr 7 -x0-x2x3x4

Implicent nr 8 x0x1-x4

Implicent nr 9 x0-x2-x3-x4

Wyjście programu

x0=1 x1=0 x2=0 x3=1 x4=0

Ocena poprawności: Działa poprawnie

Test 4

Dane wejściowe (wygenerowane losowo)

n=4

N=7

t=1

Implicent nr 0 x0

Implicent nr 1 -x1-x2

Implicent nr 2 x1x2-x3

Implicent nr 3 x1x3

Implicent nr 4 x0x2x3

Implicent nr 5 x1x2x3

Implicent nr 6 -x0-x1x2

Wyjście programu

x0=1 x1=0 x2=1 x3=1

Ocena poprawności: Działa poprawnie

Wniosek:

Program działa poprawnie

5) Zestawienie

W celu wykonania zestawienia wygeneruję kilka losowych instancji problemu, następnie uruchomię programy WalkSAT i GSAT (z listą tabu) z tym samym wygenerowanym problemem. Porównam czasy obliczeń, ilość iteracji, wyniki.
W celu zmniejszenia czasu wykonywania programu usunę wszystkie zbędne instrukcje takie jak wypisywanie wyników pośrednich każdej iteracji.

Nr prob.	n	N	t	Algorytm	Ilość iteracji	Czas wykonania [µs]	Wynik
1	10	500	-	WalkSAT LS	39	15589	x0=0 x1=1 x2=1 x3=0 x4=1 x5=1 x6=1 x7=0 x8=0 x9=0
1	10	500	1	GSAT TS	7	12544	x0=0 x1=1 x2=1 x3=0 x4=1 x5=1 x6=1 x7=0 x8=0 x9=0
1	10	500	2	GSAT TS	3	10565	x0=0 x1=1 x2=1 x3=0 x4=1 x5=1 x6=1 x7=0 x8=0 x9=0
1	10	500	3	GSAT TS	8	15555	x0=0 x1=1 x2=1 x3=1 x4=1 x5=1 x6=1 x7=0 x8=0 x9=0
1	10	500	4	GSAT TS	2	2910	x0=0 x1=1 x2=0 x3=1 x4=1 x5=1 x6=1 x7=0 x8=0 x9=0
1	10	500	5	GSAT TS	3	6509	x0=0 x1=1 x2=1 x3=0 x4=1 x5=1 x6=1 x7=0 x8=0 x9=0
1	10	500	6	GSAT TS	4	10655	x0=0 x1=1 x2=1 x3=0 x4=1 x5=1 x6=1 x7=0 x8=0 x9=0
1	10	500	7	GSAT TS	>100000	-	stop_bez_rozwiazania
2	10	1000	-	WalkSAT LS	>100000	-	stop_bez_rozwiazania
2	10	1000	1	GSAT TS	>100000	-	stop_bez_rozwiazania
2	10	1000	4	GSAT TS	>100001	-	stop_bez_rozwiazania
3	8	100	-	WalkSAT LS	11	15557	x0=1 x1=0 x2=1 x3=0 x4=1 x5=0 x6=0 x7=1
3	8	100	1	GSAT TS	4	4011	x0=1 x1=1 x2=1 x3=0 x4=1 x5=1 x6=0 x7=0
3	8	100	2	GSAT TS	4	15625	x0=1 x1=0 x2=0 x3=0 x4=1 x5=1 x6=0 x7=1
3	8	100	3	GSAT TS	2	3014	x0=1 x1=0 x2=1 x3=1 x4=0 x5=0 x6=0 x7=1
3	8	100	4	GSAT TS	3	15564	x0=0 x1=1 x2=1 x3=1 x4=0 x5=0 x6=1 x7=1
3	8	100	5	GSAT TS	5	20019	x0=0 x1=0 x2=0 x3=1 x4=1 x5=1 x6=1 x7=0

Wnioski:

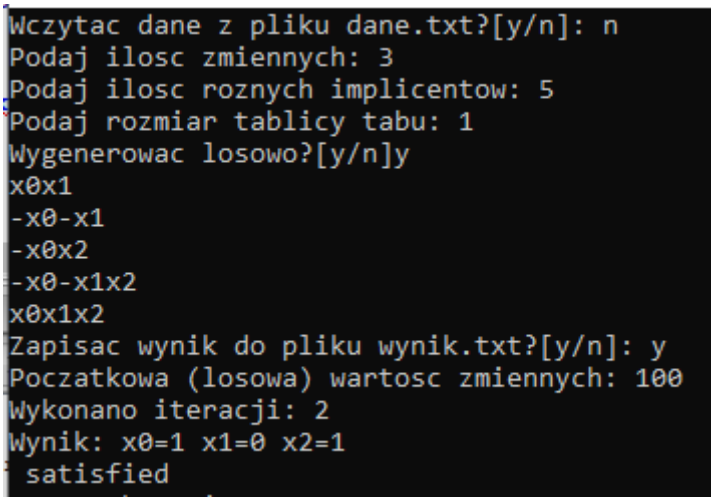
Rozmiar listy tabu wpływa na szybkość wykonania algorytmu. W przypadku testowanego problemu nr 1 najlepiej, aby lista tabu miała rozmiar 4. Po przekroczeniu rozmiaru 7 program wypisuje stop_bez_rozwiazania (tablica tabu jest za duża). W problemie 3 najoptymalniejszy rozmiar tablicy tabu to 3.
Otrzymane wyniki dla tej samej instancji problemu są różne, bo dla danej formuły boolowskiej może istnieć wiele rozwiązań problemu SAT.
Jeśli dla danej instancji problemu oba algorytmy dają wyjście stop_bez_rozwiązania prawdopodobnie rozwiązanie nie istnieje.
Wynik zależy również od losowych wartości początkowych zmiennych.

6) Obsługa programu

Po uruchomieniu programu bez argumentów wyświetla się informacja o programie oraz menu. Użytkownik zostaje poproszony o wybranie opcji z menu. Później należy wpisać odpowiednie parametry dla wybranego algorytmu. Wybranie opcji 3 kończy program. Program umożliwia wczytanie danych z pliku i zapisanie wyniku do pliku.

Menu programu:

Przykładowy zrzut ekranu z programu dla opcji 2:



Uruchomienie programu z argumentem -v wyświetla informacje o wersji programu

```
Version: Jun  2 2021 | 17:48:43
```

7) Wnioski

Program przetestowano i stwierdzono, że działa poprawnie. Im więcej zmiennych lub implicentów tym dłuższy czas wykonania programu. Jeśli rozwiązanie nie istnieje to wypisywana jest stosowna informacja.

WalkSAT i GSAT są algorytmami wyszukiwania lokalnego, nie gwarantuje osiągnięcia optimum globalnego, najczęściej uzyskane rozwiazanie jest optymalne lokalnie.

Algorytm GSAT z listą tabu jest szybszy od algorytmy WalkSAT bez listy tabu. Wykonuje się średnio w mniejszej ilości iteracji i krótszym czasie. Rozmiar listy tabu jest istotny w stosunku do efektywności programu.

8) Literatura

[1] https://pl.wikipedia.org/wiki/Problem_spe%C5%82nialno%C5%9Bci

(dostęp: 31.05.2021)

[2] https://pl.wikipedia.org/wiki/Przeszukiwanie_tabu (dostęp: 31.05.2021)

[3] https://en.wikipedia.org/wiki/Local_search_(optimization) (dostęp: 31.05.2021)

[4] https://en.wikipedia.org/wiki/WalkSAT (dostęp: 31.05.2021)