Dokumentasi Tugas 1 Logika Komputasional

Immanuel Rhesa

1106087534 – Victoria Anugrah Lestari

# 1. Cara kerja MiniSat dan SAT Solver pada umumnya

## 1.1 Boolean Constraint Propagation

Boolean Constraint Propagation adalah proses untuk meng-*assign* semua unit literal dengan nilai *true.*

## 1.2 VSIDS Decision Heuristic

VSIDS (Variable State Independent Decaying Sum) digunakan untuk menentukan *literal* selanjutnya yang akan di *assign* nilainya. Bobot untuk setiap *literal* ditentukan sesuai dengan banyaknya kemunculan *literal* tersebut dari bentuk CNF yang diberikan. *Literal* dengan bobot tertinggi adalah *literal* yang akan di *assign* selanjutnya. Pada setiap periode, bobot dari setiap *literal* akan dibagi dengan suatu bilangan *constant.* Karena itu, *literal* pada *Clauses* yang baru ditambahkan akan mempunyai bobot / *weight* yang lebih tinggi.

## 1.3 Conflict Clauses Analysis

## 1.4 Cara kerja secara keseluruhan

Secara umum, langkah kerja dari algoritma yang dilakukan MiniSat adalah sebagai berikut:

1. Literal yang mempunyai bobot tertinggi akan di-*assign* dengan nilai tertentu
2. Lakukan Boolean Constraint Propagation untuk setiap unit literal.
3. Bila belum ada Conflict Clause (Clause yang nilai semua literalnya adalah false), ulangi langkah 1 dan 2. Apabila terjadi Conflict Clause,

# 2. Problem yang diangkat untuk dipecahkan dengan MiniSat

## 2.1 Map Coloring Problem

Map Coloring Problem adalah permasalahan mewarnai peta di mana daerah yang berbatasan tidak boleh diwarnai dengan warna yang sama. Map Coloring Problem biasa diangkat sebagai contoh dalam materi Constraint Satisfaction Problem (CSP). Akan tetapi, Map Coloring Problem juga dapat diterjemahkan menjadi suatu Boolean Satisfiability Problem sehingga dapat dipecahkan dengan MiniSat.

### 2.1.1 Four Color Theorem

Four Color Theorem menyebutkan bahwa “pada suatu bidang planar yang dipisah-pisahkan sehingga menghasilkan sebuah peta, tidak lebih dari empat warna dibutuhkan untuk mewarnai daerah-daerah pada peta sehingga dua daerah yang bersebelahan tidak berwarna sama.” Syaratnya, sebuah daerah haruslah *contiguous* (tidak boleh terpisah).

### 2.1.2 Encoding Warna

Berdasarkan Four Color Theorem, peta yang dihasilkan pada program oleh user diasumsikan cukup diwarnai menggunakan empat warna saja. Untuk memudahkan proses penyelesaian permasalahan, setiap warna direpresentasikan oleh 2 bit, dengan kemungkinan nilai 0 (false) dan 1 (true):

* 00 – merah
* 01 – kuning
* 10 – hijau
* 11 – biru

### 2.1.3 Representasi Permasalahan

Setiap daerah diberikan 2 buah variabel dan dengan:

* adalah nomor daerah
* adalah bit ke-0 daerah
* adalah bit ke-1 daerah

Daerah yang saling terhubung diilustrasikan oleh sebuah *constraint graph*. Contohnya adalah gambar berikut:

1

2

3

**Gambar 1. Daerah (kiri) dan constraint graph yang berpadanan (kanan)**

Pada contoh di atas, ketiga daerah tidak boleh memiliki warna yang sama. Maka salah satu solusi yang mungkin adalah 1 – merah, 2 – kuning, 3 – biru.

Representasi bitnya adalah , , , , , .

Encoding untuk permasalahan di atas adalah:

|  |
| --- |
|  |

Dengan kata lain, tidak boleh ada kasus di mana dan , serta dan , serta dan .

# 3. Rumusan solusi dari problem yang diangkat, menggunakan notasi/ bahasa MiniSat

## 3.1 Bentuk CNF

Untuk menyederhanakan bentuk encoding yang diperoleh pada nomor sebelumnya, diambil sebuah *case* dimana daerah pertama tidak boleh memiliki warna yang sama dengan daerah ketiga yaitu

|  |
| --- |
|  |

Langkah penyederhanaannya adalah sebagai berikut :

1. Hukum De Morgan

|  |
| --- |
|  |

1. Aturan Distribusi

|  |
| --- |
|  |

Sehingga diperoleh bentuk sederhana seperti berikut

|  |
| --- |
|  |

Dengan encoding yang diperoleh pada nomor sebelumnya, dapat disimpulkan bahwa berapapun banyaknya daerah yang berada pada constraint graph, kalimat logika yang dihasilkan tetap mengikuti pola yang sama. Bentuk kalimat logika untuk seluruh *case* di atas dapat diubah ke bentuk *conjunctive normal form* (CNF) berikut:

|  |
| --- |
|  |

Apabila dirumuskan dengan notasi MiniSat, maka representasinya menjadi:

|  |
| --- |
| p cnf 6 12  -10 -20 -11 -21 0  -10 -20 11 21 0  10 20 -11 -21 0  10 20 11 21 0  -10 -30 -11 -31 0  -10 -30 11 31 0  10 30 -11 -31 0  10 30 11 31 0  -30 -20 -31 -21 0  -30 -20 31 21 0  30 20 -31 -21 0  30 20 31 21 0 |

Jumlah variabel adalah 6 dan jumlah klausa adalah 12.

## 3.2 Generalisasi masalah

Diberikan suatu *graph* g, konjungsi klausa c merepresentasikan setiap *edge* e yang terdapat pada *graph* g. *Edge* e menghubungkan 2 buah daerah. Untuk setiap daerah i dan j yang terhubung dalam e, konjungsi klausa c akan berbentuk:

|  |
| --- |
|  |

Untuk setiap tambahan daerah yang terhubung (ada di edge), cukup menambahkan sebuah konjungsi klausa c.

# 4. Contoh eksekusi dan outputnya

## 4.1 Contoh eksekusi

Class utama pada program adalah Runner.java. Apabila dijalankan, program akan menampilkan sebuah *graphical user interface* berupa panel dan button. Program akan mensimulasikan *map coloring problem* dengan peta sebagai input dari pengguna. Setiap daerah diwakili oleh sebuah bentuk. Tersedia 2 pilihan bentuk, yaitu persegi dan persegi panjang. Pengguna dapat memilih bentuk dengan mengklik button yang bersesuaian.

Untuk membuat peta pada panel, pengguna dapat mengklik panel. Ketika pengguna mengklik pada bagian panel yang kosong, sebuah bentuk akan muncul pada panel. Untuk memindahkan bentuk, klik pada area di dalam bentuk tersebut. Dua kali klik (*double click*) pada bentuk akan menghapus bentuk tersebut.

Pada bagian atas panel terdapat 2 buah button, yaitu button “Ok” dan button “Restart”. Button “Ok” berfungsi sebagai pemberi solusi pada peta yang telah dibuat pengguna. Button “Restart” berfungsi menghapus semua bentuk yang ada pada panel sehingga pengguna dapat mengulang dari awal. Pada bagian bawah panel terdapat sebuah button “Show other solution”. Button ini berfungsi menunjukkan solusi lain (pewarnaan lain) pada peta.

Output dari program ini adalah daerah-daerah pada peta yang sudah diwarnai. Setiap daerah yang bersebelahan akan memiliki warna berbeda.

## 4.2 Penjelasan program

Program terdiri dari 8 class, yaitu 5 class untuk logic, 2 class untuk GUI, dan 1 class utama yaitu Runner.java.

1. Runner

Class utama pada program. Ketika dijalankan, program akan menampilkan JFrame berisi panel kosong dengan button-button.

1. MainGUI

Class yang bertanggung jawab membuat seluruh GUI program serta logic di belakang tampilan GUI tersebut. MainGUI membuat gambar persegi dan persegi panjang, menjalankan fungsi button, serta memanggil class-class logic untuk menjalankan MiniSat.

1. EncodedColor

Class yang menyimpan 4 buah warna: merah, kuning, hijau, biru, dalam 2 bit. EncodedColor berfungsi sebagai “penerjemah” bit representasi warna (2 buah integer) ke warna aslinya (class Color).

1. Graph

Class yang merupakan penyederhanaan dari graph sebenarnya. Graph ini hanya menyimpan edge list untuk menyimpan keterhubungan suatu daerah dengan daerah lainnya. (Daerah yang tidak terhubung tidak dimasukkan ke dalam graph.)

1. Edge
2. CNFMaker
3. Minisat
4. MinisatInputMaker

# Referensi

<http://minisat.se/downloads/escar05.pdf>

<http://minisat.se/downloads/synth_in_sat.pdf>

<http://cacm.acm.org/magazines/2009/8/34498-boolean-satisfiability-from-theoretical-hardness-to-practical-success/fulltext>

<http://mathworld.wolfram.com/Four-ColorTheorem.html>

<http://www.c3.lanl.gov/mega-math/gloss/math/4ct.html>

http://swtv.kaist.ac.kr/lab-orientation/Intro%20to%20MiniSat%20v1.14.pptx