Project milestone1 實作報告

學號:0556017 姓名:林怡秀

**ㄧ、程式實作：**

1. **Clauses database data Structure**

直接利用老師給的parser.cpp, parser.h讀入clauses，存入vector of vector

1. **Branching Heuristics: Jeroslaw-Wang Score**

heuristic(): 預先計算每個變數的分數，存入priority queue，取分數最大的先做猜測

1. **BCP mechanisms: 2-Literal Watching**

|  |  |  |
| --- | --- | --- |
| 函式名稱 | 功能 | 回傳值 |
| initial\_watch | 初始化watch(vector)存每個clause被監看的literals | void |
| initial\_var | 初始化vars(vector)存每個variable的狀態(包含ind: 變數索引值、value(0: false, 1: true, -1:unassign)、prevalue: 可被imply的值、pw: 存此變數被監看且為正的clause index、nw: 存此變數被監看且為負的clause index) | void |
| set\_UNIT | 預先將只有一個變數的clauses設為unit clause並給予變數true值 | void |
| BCPLiteralWatching | 將variable被設為1與0分開處理：  **當變數被設為1時需檢查nw裡所有clauses:**  **step1:** 先檢查是否有not-assign, not-watch的變數(有則設為新的watch literal，跳至下個clause)  **step2:** 檢查另一個watch literal的值：  為1:clause sat  為0:clause conflict  為-1:設為unit clause，同時設定imply value  **當變數被設為0時需檢查pw裡所有clauses，其餘同上述。** | bool |

1. **Davis-Putnam-Logemann-Loveland Algorithm(DPLL):**

dpllSimple(): Backtracking finding the solution ( call BCPLiteralWatching repeatly)，先猜0，後猜1

**二、測試**

**測試環境：**

硬體環境： CPU：2.7 GHz Intel Core i5 , RAM：8 GB 1867 MHz DDR3

軟體環境： GCC 4.2.1, Sublime Text, (C/C++)

作業系統： OS X 10.11.4

|  |  |  |
| --- | --- | --- |
| **variable/clause數量** | **執行時間(SAT/UNSAT)** | **資料筆數(SAT/UNSAT)** |
| sanity (from teacher) | 3.72x10-4/8.01x10-4 | 2/2 |
| tiny (from teacher) | 3.3x10-4/3.38x10-4 | 2/2 |
| 50/218 | 2.544x10-3 / 3.838x10-3 | 10/10 |
| 100/430 | 3.02x10-2 / 6.97x10-2 | 10/10 |
| 150/645 | 6.95x10-1 / 1.48 | 10/10 |
| 200/860 | 28.349 / 62.866 | 10/10 |
| 250/1065 | 485.65 / 1296.15 | 5/1 |

p.s. clauses 皆為3個變數

測資來源：Uniform Random-3-SAT : <http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html>

**三、困難**

在處理2-literal watching時，有很多條件需要考慮，使用很多導致程式碼很凌亂，同時還要maintain變數的狀態加上更新pw和nw(但更新的pw和nw不需要在backtracking時恢復到原本狀態)。原本將unit clause所imply的變數直接assign成true，後來發現在更新watching literal時會產生錯誤，只好多用一個preval來預存imply的值，之後再一個個更新。花了很多時間在debug，一直找到一些沒考慮到的情況。