Project milestone2 實作報告

學號:0556017 姓名:林怡秀

一、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

2. **Branching Heuristics: Variable State Independent Decaying Sum**

   heuristic(): 每次新學到一個 clause，將新 clause 中每個 literal 的分數(act_score)加 1，其餘 literal 的分數乘上 0.95[註 1]。

3. **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、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
new_watch	將新學習到的 clause 初始 pw, nw, watch，並放入 clause 中(clause size 大於 50 就不會學習)	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	vector<int> (conflict clause)

	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。	

4. Davis-Putnam-Logemann-Loveland Algorithm(DPLL): Non-Chronological Backtracking + Conflict analysis and learning

函式名稱	功能	回傳值
DPLL	Iterative DPLL: (先猜 1, 再猜 0) 如果遇到 conflict，回傳 conflict_clause → firstUIP(取得 learned_clause → backlevel(取得該回去的層數) → 回到該層原來的 variable 改猜 1，然後做 BCP →重復動作 沒有遇到 conflict: 從 heap 中拿取一個分數最高的 variable 做 BCP → 重複	void
firstUIP	取得 conflict_clause 重複做 resolve 得到 firstUIP →回傳 learned_clause	vector<int> (learned_clause)
backlevel	當 learned_clause 的 size 大於 50 或為 1 時皆重回第一層，其餘根據 learned_clause 重回除了此層剩餘的最大層數(最深層數)	int

二、測試結果

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

測資(variable/clause 數量)	執行時間(second)
aim-50-1_6-yes1-1.cnf (50/80)	0.001090
aim-50-1_6-no-1.cnf (50/80)	0.000713
aim-100-1_6-yes1-1.cnf (100/160)	0.004594
aim-100-1_6-no-1.cnf (100/160)	0.002042
aim-200-1_6-yes1-1.cnf (200/320)	0.005376
aim-200-1_6-no-1.cnf (200/320)	0.000862
par8-1-c.cnf (64/254)	0.001758
par8-1.cnf (350/1149)	0.009886
par16-1-c.cnf (317/1264)	6.091117
par16-1.cnf (1015/3310)	5.122076
par32-1-c.cnf (1315/5254)	∞
par32-1.cnf (3176/10277)	∞
jnh1.cnf (100/850)	0.014337
jnh10.cnf (100/850)	0.004824
jnh11.cnf (100/850)	0.011082
dubois20.cnf (60/160)	0.008079
dubois100.cnf (300/800)	0.114122
ii8a1.cnf (66/186)	0.000882
ii16a1.cnf (1650/19368)	0.256097
ii32a1.cnf (459/9212)	0.315830

三、困難

1. 程式進入無限迴圈：

後來察覺當從 conflict clause 一直做 resolve 到 learned clause 中如果遇到有 variable 是第 0 層就被做決定的(也就是固定值)，可以直接捨棄那個 variable，才解決了無限迴圈問題

2. 程式速度太慢：

(1) 將原來 for 迴圈還原之前層設定的變數改成 map 直接取得(空間換時間)

(2) 拿掉 2_literal_watching 中不必要的 loop，效果顯著

(3) 後來察覺 2_literal_watching 會將設過的 variable 重新處理，故加了 map 做檢查，防止反覆處理同個變數

(4) heuristic 中的 decay factor 其實對程式速度影響很大，0.95 的測試結果比起其他結果好

(5) 誤打誤撞的本來想把 clause size 大於 50 的 clause 捨棄，因為看錯行，把程式變成只要遇到 clause size 大於 50 的 learned clause 就重回第一層(讓原來跑了幾小時都跑不出來的測資(par16-1.cnf)變成 5 秒)

[註 1] In mVSIDS, the variables' activities are decayed by multiplying with a constant decay factor, typically 0.95.　Reference：Jia Hui Liang, Vijay Ganesh, Ed Zulkoski, Understanding VSIDS Branching Heuristics in Conflict-Driven Clause-Learning SAT Solvers (2015)