E05 SAT Problem

Suixin Ou

School of Computer Science
Sun Yat-sen University

October 26, 2021

Task

Problem

- In logic and computer science, the Boolean satisfiability problem (SAT) is the problem of determining if there exists an interpretation that satisfies a given Boolean formula.
- In other words, it asks whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE.
- If this is the case, the formula is called satisfiable. On the other hand, if no such assignment exists, the function expressed by the formula is FALSE for all possible variable assignments and the formula is unsatisfiable.

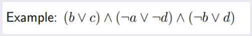

Example: $(b \vee c) \wedge (\neg a \vee \neg d) \wedge (\neg b \vee d)$

Figure 1: SAT Problem

Task

Input-output

- Input: a cnf in ”data.txt”.
  - The file ”data.txt” starts with a line, ”p cnf nbvar nbclauses”, that indicates that the instance is in CNF format; nbvar is the exact number of variables appearing in the file; nbclauses is the exact number of clauses contained in the file.
  - Then the clauses follow. Each clause is a sequence of distinct non-null numbers between -nbvar and nbvar ending with 0 on the same line; it cannot contain the opposite literals i and -i simultaneously.
  - Positive numbers denote the corresponding variables. Negative numbers denote the negations of the corresponding variables.
- Output: an interpretation that satisfies the given CNF formula.

Task

Submission

pack your report E05_YourNumber.pdf and source code into zip file E05_YourNumber.zip, then send it to ai_course2021@163.com.

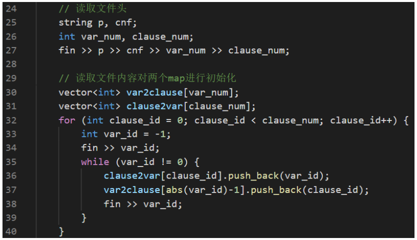
Solution

Algorithm

- Stochastic hill climbing : choose at random from among the uphill moves; the probability of selection can vary with the steepness of the uphill move.
- Random-restart (随机重启) hill climbing: "If at first you don't succeed, try, try again." Conduct a series of hill-climbing searches from randomly generated initial states, until a goal is found.
- Random walk (随机游走) hill climbing: With probability p, pick randomly a neighbor; with probability $1-p$, pick randomly a best neighbor.

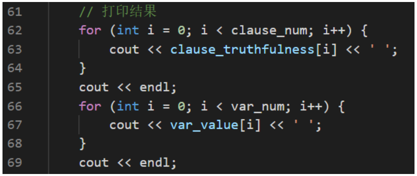
Solution

Read input



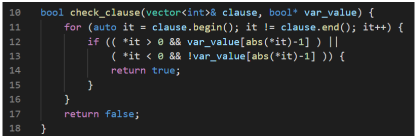
```cpp
    // 读取文件头
    string p, cnf;
    int var_num, clause_num;
    fin >> p >> cnf >> var_num >> clause_num;

    // 读取文件内容对两个map进行初始化
    vector<int> var2clause[var_num];
    vector<int> clause2var[clause_num];
    for (int clause_id = 0; clause_id < clause_num; clause_id++) {
        int var_id = -1;
        fin >> var_id;
        while (var_id != 0) {
            clause2var[clause_id].push_back(var_id);
            var2clause[abs(var_id)-1].push_back(clause_id);
            fin >> var_id;
        }
    }
```

Visualize output



```cpp
    // 打印结果
    for (int i = 0; i < clause_num; i++) {
        cout << clause_truthfulness[i] << ' ';
    }
    cout << endl;
    for (int i = 0; i < var_num; i++) {
        cout << var_value[i] << ' ';
    }
    cout << endl;
```

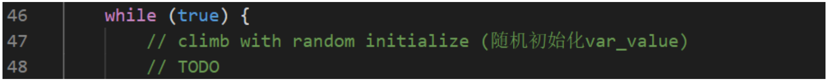
check whether a clause is correct



```cpp
bool check_clause(vector<int>& clause, bool* var_value) {
    for (auto it = clause.begin(); it != clause.end(); it++) {
        if (( *it > 0 && var_value[abs(*it)-1] ) ||
            ( *it < 0 && !var_value[abs(*it)-1] )) {
            return true;
        }
    }
    return false;
}
```

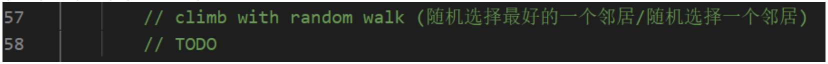
Solution

You should finish the hill climbing algorithm with random start,



```
    while (true) {
        // climb with random initialize (随机初始化var_value)
        // TODO
```

and random walk.



```
            // climb with random walk (随机选择最好的一个邻居/随机选择一个邻居)
            // TODO
```

The End