1951505 姜星宇

人工智能原理第 4 次作业

一、作业内容

基于消解原理中化为子句的相关技术, 编程实现并解决一个知识推理问题, 或者应用于具体项目中解决某个小问题.

二、实验原理

这次实验根据消解法以及 DPLL 和 WalkSAT 解决扫雷游戏中确定地雷的位置问题. 扫雷即为一个 16 × 30 的区域, 根据翻开的代表周围地雷数量的数字确定地雷的位置.

这个问题本质上是一个命题逻辑推理问题, 可以归结为 SAT(Boolean satisfiable problem, 布尔可满足性问题). 设定 480 个命题变项 x_i, 代表每一个格子是否为地雷, 目标就是用已知条件能否推理得到 x_i 与$\neg x_i$. 前提条件可以根据以翻开的数字与周围的未翻开格子的数量获得, 例如数字为 2, 并且周围未翻开格子的数量为 3, 则将这三个格子为地雷设为 y_1, y_2, y_3, 可以得到真值表

$y_1y_2y_3$	000	001	010	011	100	101	110	111
f	0	0	0	1	0	1	1	0

其中公式 f 为"三个中有两个地雷"的等值式. 不难看出真值表中 1 的数量不为格子中的数字时 f 取 0. 也就是说合取范式可以表达为这些情况下对应的文字. 例如上面的 f 可以表示为

$$(y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee y_2 \vee \neg y_3) \wedge (y_1 \vee \neg y_2 \vee y_3) \wedge (\neg y_1 \vee y_2 \vee y_3) \wedge (\neg y_1 \vee \neg y_2 \vee \neg y_3).$$

这样一来, 可以现根据 0 到 8 中的每一种情况生成这些 01 串, 再视为二进制数存储再数组中以便使用. 这样对于每一个格子, 可以获得一系列简单析取式即子句, 根据归谬法推理, 可以归结为 $KB \wedge \neg A$ 的可满足性问题即 SAT.

为解决 SAT, 可以有多种算法, 其中最简单的就是消解法. 消解法基于消解原理, 即假言三段论推理规则: $(A \rightarrow B) \wedge (B \rightarrow C) \vDash (A \rightarrow C)$. 也就是$(\neg A \vee B) \wedge (\neg B \vee C) \vDash (\neg A \vee C)$. 当$\neg A$ 与 C 为简单析取式 D_1 与 D_2, B 为某变元 x 时, 也就能得到$(D_1 \vee x) \wedge (\neg x \vee D_2) \vDash (D_1 \vee D_2)$. 这样只要每次循环对于任意两个变量判断是否能够消解, 即存在互补的文字. 而如果两个子句中存在两个互补文字, 最终

消解的新子句由于存在另一个互补文字的析取, 为永真式对于推理没有意义. 因此算法中还是要寻找有且仅有一个互补文字的两个子句. 但是因为消解的方法有很多种, 任意两个存在互补文字的子句都可以消解得到新的前提, 甚至没有互补文字的都可以进行简单的附加规则, 如何尽可能地判断出是否有空子句很难找到一个确定的方法. 而想要将所有可以达到的子句全部找出也是几乎难以做到的, 随着子句集规模的扩大, 每次都要对 n^2 数量级的子句对尝试消解, 大概也可以看到对于不同的子句数量与符号数量, 最坏的情况下复杂度将以 $2^{n \times n}$ 增长.

由于这里有大约上百条子句与 480 个符号需要去消解, 直接通过尝试生成空子句的方法很难行通, 因此还是要通过归谬法判断 $KB \wedge \neg A$ 的可满足性. 最简单的还是直接通过对于真值表的穷举, 即对于每一个符号赋值来寻找是否有解, 具体可以用递归来实现. 但对于存在 m 个符号的 CNF, 复杂度为 2^m 仍然难以接受.

根据 CNF 的特点, DPLL 算法使用几种方法进行剪枝. 首先因为析取式只要有一个为真即可为真, 而合取式只要有一个为假即为假, 因此在没有完全赋值的情况下, 根据当前已经存在的赋值有可能判断出为真或假(early termination). 接下来是检查是否存在某一个符号只有一种文字出现(pure symbol), 即只有单独的 $\neg x$ 与 x 出现. 将其提出后得到$(x \vee A) \wedge B$, 如果说 $x = 0$ 可以满足, 那么 $x = 1$ 也一定可以满足. 即 $x = 1$ 对结果没有影响, 可以直接将其赋值, 结果就是这些存在 x 的子句全部为真, 可以大大简化运算时间. 下一步就是判断是否存在只有一个文字的子句(unit clause), 那么一定要求这个符号只能为某个值, 这样也可以直接赋值. 如果这些都不存在, 再对于某一个尚未赋值的符号赋真或假两种情况来递归寻找.

DPLL 核心就是提早将能判断的都确定下来, 根据 CNF 特点某一个变元的赋值唯一确定, 就不需要在这个变元处分支, 也极大地提高了效率. 当然对于最坏的情况, 它还是指数级的复杂度. 但是对于这次的实验已经够用了.

还有一种算法 WalkSAT 是另一种不同的处理方式. 它用随机化的赋值方式, 每次根据一个概率将某些变量翻转, 这些变量要么是随机的, 要么是让子句为真最多的赋值方法. 之所以要用随机化可能与模拟退火法中的从某一个峰值处下降达到另一个更高的峰值有着类似之处. 不过这个方法最坏的情况也是可能要找遍 2^n 的情况, 只是过程更加随机. 由于最终结果准确度并不是很高, 时间也并

不比 DPLL 快多少，可能是由于不知道哪里的一些 bug，再加上 DPLL 已经可以做到比较不错的效果，这次采取的方式就是 DPLL 了.

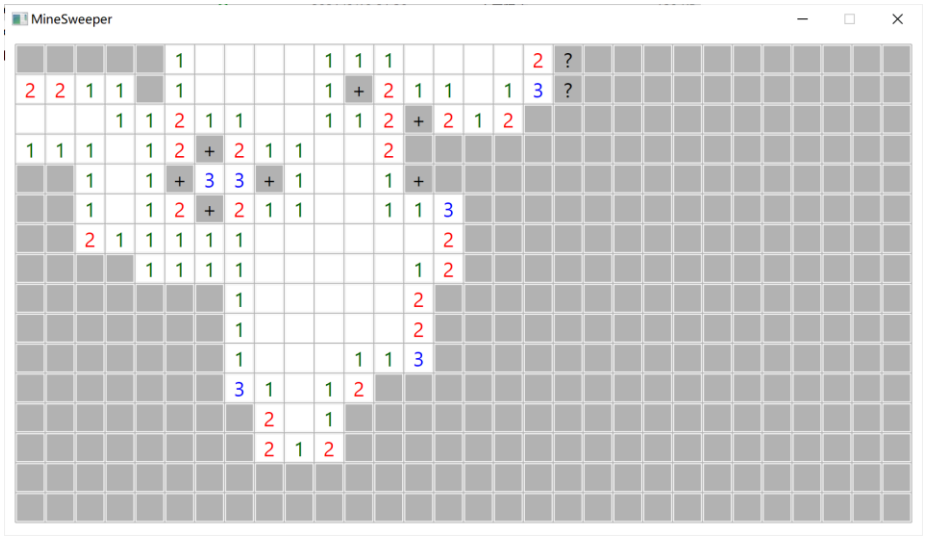
逻辑推理中转化的 SAT 问题也是计算理论中比较重要的内容，涉及到 NP 完全问题，很多设计枚举的问题也都可以像命题逻辑推理一样转化为布尔可满足性问题，对于它的研究的意义是的十分重要的.

参考资料:

Russell, S. J., & Norvig, P. (2010). Artificial Intelligence-A Modern Approach, Third Edition.

三、实验内容

1. 源文件构成: (编译器 G++, MinGW-w64)
   - main.cpp: 生成计算部分.
   - winmain.cpp: 生成演示部分.
2. 实现功能:
   - 扫雷基本功能实现: 左键点击方格挖开，右键第一次标记雷(加号)，第二次标记问号，第三次取消标记.



   - 扫雷与搜索程序通讯功能实现: 通过写入文件与读取文件来实现. GUI 程序写入 board.txt，下划线代表待确定的方格，数字对应周围雷数. 由于可以通过推理得到地雷的位置，因此不特别标注地雷的位置. 读取的文件为 mines.txt，第一行分别为当前计算出的空白格的数量与地雷格的数量，随后为每一个格子的下标(从 0 开始). 载入程序后，地图会自动更新. 具体为按上键载入 mines.txt，下键导出

board.txt，回车键先载入再导出.

board.txt

```
_____1000011100002_____
2211_100001_211013_____
0001121100112_212_____
111012_211002_____
__101_33_1001_____
__1012_21100113_____
__2111110000002_____
____11110000012_____
_______1000002_____
_______1000002_____
_______1000113_____
_______31012_____
________201_____
________212_____
_____
_____
```

mines.txt

```
23 28

0 2
0 3
0 4
2 18
3 13
4 0
4 14
5 1
6 1
7 2
8 3
8 4
8 5
8 15
9 6
10 6
10 14
11 13
12 12
13 11
14 7
14 8
14 9

0 0
0 1
0 18
1 4
1 11
1 18
2 13
```

- 计算程序的实现：计算程序从 board.txt 中读取数据，计算得出可以确定的地雷与非地雷的位置，输出到 mines.txt 中，并在标准输入输出中同时输出结果及用时．并不停循环等待输入回车继续读入并运算后输出.

```
Enter "q" or press "Ctrl+C" to quit
Press Enter to continue...
Elapsed time: 0.097
Total time: 0.097
Mines: 28
Blanks: 23
Press Enter to continue...
```

3. 主要数据结构:
   - `clause` 结构体：其中 `unsigned char seq[480]`的存储子句，值为宏 `POS` 代表 x，值为 `NEG` 代表$\neg x$，值为 `NONE` 代表不存在该符号.
   - `mat` 全局变量: `unsigned char` 二维数组存储地雷区域，值为宏 `BLANK` 代表待求方格，值为数字代表周围地雷个数.
   - `mdl` 全局变量: `vector<int>`二维数组. `mdl[i][j]`代表第 `i` 个格子中存在 `j` 个地雷的合取范式集合，二进制数代表一个简单析取式.
   - `null` 全局变量: `clause` 类型，代表空子句.

4. 主要函数:

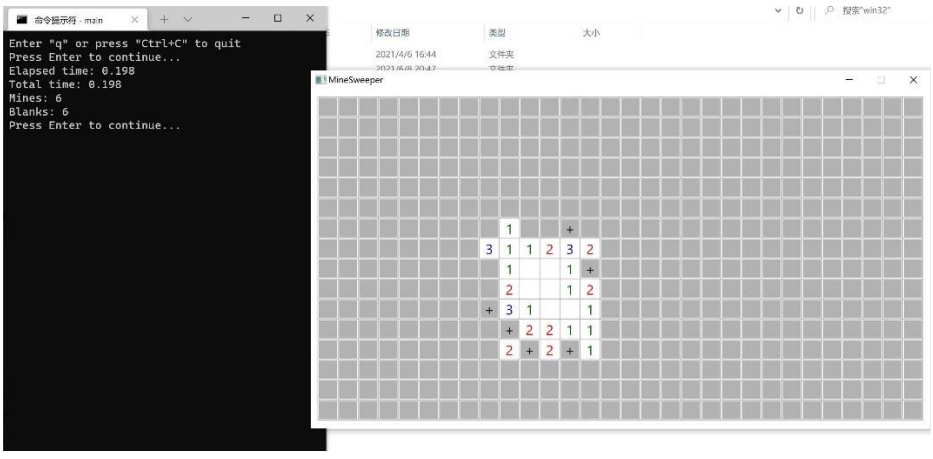
- `void run(vector<int> &mines, vector<int> &blanks, bool method(vector<clause>, clause &), int lower_bound = 0, int upper_bound = 30)`: 运行测试程序, 规定每一个待确定的点选取周围生成 CNF 的数字范围半径的下界与上界, 范围不断增加, 生成子句集与结论的否定引入可根据 `method` 函数计算结果. 如果找到空白格即输出坐标到 `mines` 与 `blanks` 两个集合中.
- `bool DPLL(vector<clause> vc, clause &model)`: DPLL 算法, 参数为子句集, 返回一个布尔值代表此 CNF 是否可满足. 引用 `model` 为赋值情况, `POS` 代表赋值为 1, `NEG` 代表赋值为 0, `NONE` 代表不赋值.

5. 可视化实现方法: 调用 WinAPI 与 Direct2D.

四、结果分析

1. DPLL 算法联合测试结果:

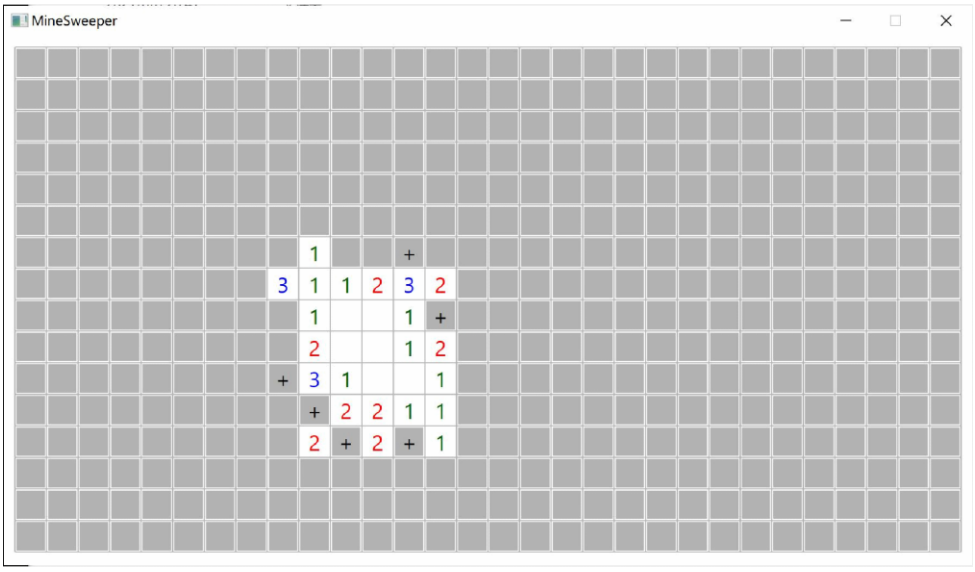
控制台中计算程序与窗体界面程序通过文件进行通讯过程如下, 先再左边完成计算后, 在右边更新结果并重新写入文件. 两个程序以此方式交替执行.



28 步运行总时间如下, 约 3 分 20 秒.

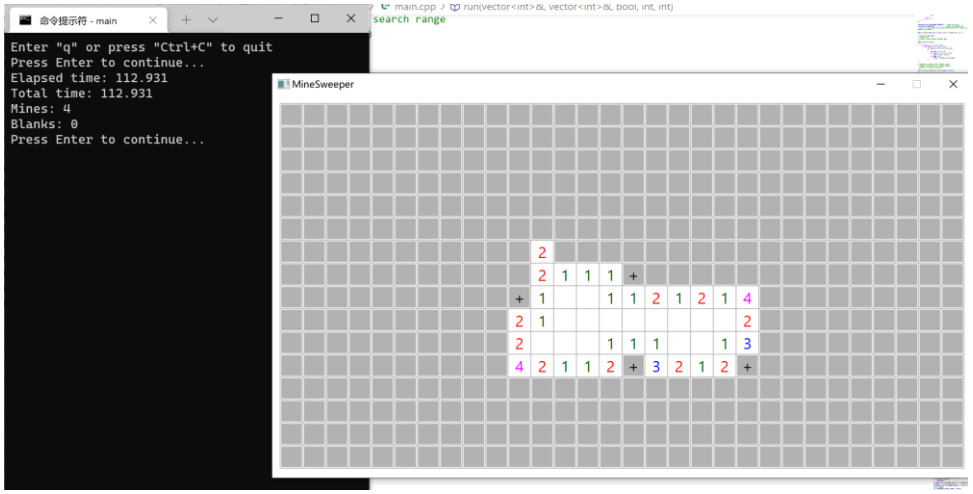
```
Elapsed time: 9.729
Total time: 197.908
Mines: 96
Blanks: 1
Press Enter to continue...
```

将每一步的过程制成 GIF 动画如下图. (详见图片附件)



2. 消解法联合测试结果:

将测试方法 `method` 参数改为 `resolve` 函数后, 将上下界都定为 1, 可以看见对于大约二十个左右子句数量就已经耗时接近两分钟, 并且只是搜索了每个点范围为 1 的区域, 无法将非雷区域找出. 可见这确实很难做到高效.



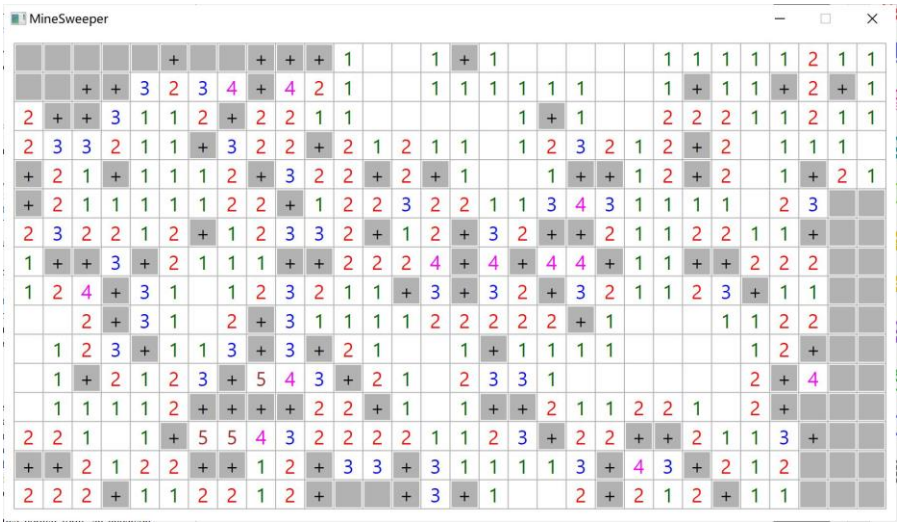
3. 结果分析:

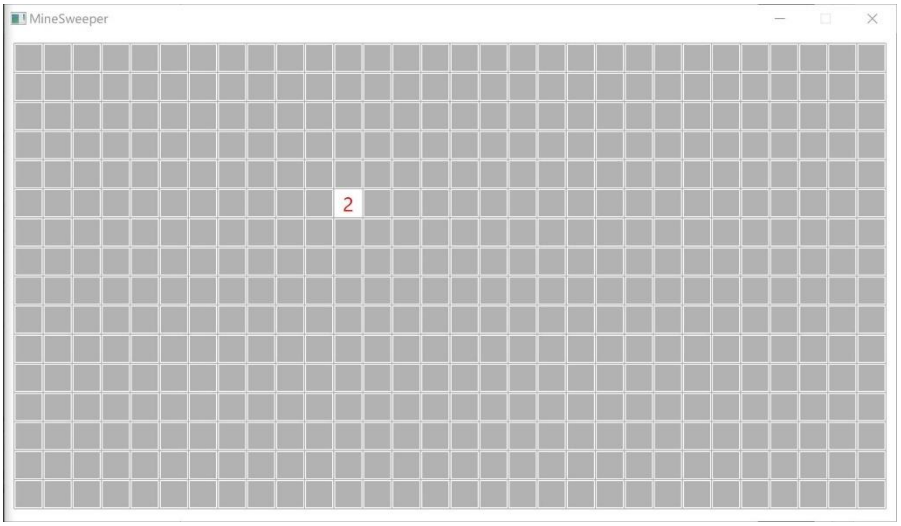
其实根据扫雷的世界纪录几十秒来说, 这个结果并没有体现出机器的优势. 但是因为扫雷并不是一个一定可解的问题, 因此存在一定的随机性, 记录的产生往往是靠直觉, 运气与手速, 而且是不考虑成功概率的前提下的. 因此与人类可比的还应当是推理的过程与准确性. 可以看见目前的效果上只要是算出来的结果, 就一定是正确的, 并且可以比较

成功地根据动态反馈从初始局面扩展到最终只能靠运气决定的局面. 从这一点推理正确性的能力上, 确实可以说产生了一定的智能. 针对时间的问题, 其实这个程序还有许多可以优化的地方, 譬如子句的存储方面, 480 个字节型数据中大部分符号都是 NONE, 也就是不存在, 而每次却需要遍历一遍寻找是否存在某个文字. 换一种存储方式应该可以提高一定的时间. 还有就是可以发现, 到后面时间会变长, 很可能是因为前面的所有地雷需要重新扫一遍. 如果输入输出能够把地雷的信息加上, 由于此前的地雷不需要重新计算, 并且增加了许多单元子句(unit clause)的信息, 应该可以使 DPLL 算法的速度得到进一步提升.

4. 问题的可解性:

在求解的过程中发现, 对于以下两种情况, 长时间计算得不到结果.





究其原因, 应该是由于在上一种情况下, 由于本身就无法确定能否

有解, 这样可能就不存在 pure symbol 与 unit clause, 就必须穷举每一种情况. 而下面的情形也是本身就找不到解, 找不到空白的格子只能不断扩大生成子句集格子的范围, 当这个范围很大时, 子句数量也就很大, 这样导致递归搜索树十分庞大.

实际上这两种情况属于正常的行为, 本身就是无法判断出结果的, 因此, 如果像获得一个比较完整的自动扫雷程序, 还需要限定时间或者生成子句集的格子范围, 并在无法搜得结果时随机选择一个. 当然进一步地说, 如果能够算出每个格子的大体概率那就更好了. 不过这些都不是这次作业的重点, 也就没有去体现.

5. 总结:

扫雷这个问题本质上可以用 DPLL 解的原因大约是 DPLL 考虑了人是如何去思考逻辑问题的. 也许对于这个问题思考的角度与具体方法可能不太相同, 毕竟这个程序还不能掌握一些计数的规律, 但是一些总的化简方案, 譬如这儿是地雷, 那么会怎么样等等, 对应的算法就是单一子句的确定. 书上也说了, 这里和前向链接类似, 如果子句都为 Horn 子句, 那么 DPLL 算法等价于前向链接算法. 而前向链接与人的思考是十分类似的. 也就是在穷举的基础上, 加上了一些人在逻辑推理中的思考方式, 因此 pure symbol 与 unit clause 都称为一种启发(heuristic). 本身作为 NP-Complete 问题, 穷举才能解决所有情形, 人可以推理必然也是因为问题的一些特殊性, 把这些知识转化为上面的启发, 逻辑智能体就可以做出较为准确的推断.

五、文件清单

1951505-姜星宇-第四次作业
|--实验报告.pdf
|--实验报告.pptx
|--pic(结果图片)
| |--(…)
|--main.cpp
|--main.exe (计算部分)
|--winmain.cpp
|--winmain.exe (GUI 部分)