

**课程设计报告**

**题目：基于SAT的二进制数独游戏求解程序**

**课程名称： 程序设计综合课程设计**

**专业班级： CS1802**

**学 号： U201814531**

**姓 名： 李响**

**指导教师： 卢萍**

**报告日期： 2020.3**

**计算机科学与技术学院**

# 任 务 书

**设计内容**

SAT问题即命题逻辑公式的可满足性问题（satisfiability problem），是计算机科学与人工智能基本问题，是一个典型的NP完全问题，可广泛应用于许多实际问题如硬件设计、安全协议验证等，具有重要理论意义与应用价值。本设计要求基于DPLL算法实现一个完备SAT求解器，对输入的CNF范式算例文件，解析并建立其内部表示；精心设计问题中变元、文字、子句、公式等有效的物理存储结构以及一定的分支变元处理策略，使求解器具有优化的执行性能；对一定规模的算例能有效求解，输出与文件保存求解结果，统计求解时间。

**设计要求**

要求具有如下功能：

1. **输入输出功能：**包括程序执行参数的输入，SAT算例cnf文件的读取，执行结果的输出与文件保存等。(15%)
2. **公式解析与验证：**读取cnf算例文件，解析文件，基于一定的物理结构，建立公式的内部表示；并实现对解析正确性的验证功能，即遍历内部结构逐行输出与显示每个子句，与输入算例对比可人工判断解析功能的正确性。数据结构的设计可参考文献[1-3]。(15%)
3. **DPLL过程：**基于DPLL算法框架，实现SAT算例的求解。(35%)
4. **时间性能的测量：**基于相应的时间处理函数（参考time.h），记录DPLL过程执行时间（以毫秒为单位），并作为输出信息的一部分。(5%)
5. **程序优化：**对基本DPLL的实现进行存储结构、分支变元选取策略[1-3]等某一方面进行优化设计与实现，提供较明确的性能优化率结果。优化率的计算公式为：[(t-to)/t]\*100%,其中t 为未对DPLL优化时求解基准算例的执行时间，to则为优化DPLL实现时求解同一算例的执行时间。(15%)
6. **SAT应用：**将二进制数独游戏[5，6]问题转化为SAT问题[6]，并集成到上面的求解器进行问题求解，游戏可玩，具有一定的/简单的交互性。应用问题归约为SAT问题的具体方法可参考文献[3]与[6-9]。(15%)

**参考文献**

[1] 张健著. 逻辑公式的可满足性判定—方法、工具及应用. 科学出版社，2000

[2]TanbirAhmed.An Implementation of the DPLL Algorithm.Masterthesis,

Concordia University,Canada,2009

[3] 陈稳.基于DPLL的SAT算法的研究与应用.硕士学位论文，电子科技大学，2011

[4]CarstenSinz.Visualizing SAT Instances and Runsof the DPLL Algorithm.JAutom Reasoning (2007) 39:219–243

[5] Binary Puzzle：<http://www.binarypuzzle.com/>

[6] Putranto H. Utomo and Rusydi H. Makarim. Solving a Binary Puzzle. Mathematics in Computer Science,(2017) 11:515–526

[7] Tjark Weber. A sat-based sudoku solver. In 12th International Conference on Logic forProgramming, Artificial Intelligence and Reasoning, LPAR 2005, pages 11–15, 2005.

[8]InsLynce and JolOuaknine. Sudoku as a sat problem.In Proceedings of the 9th InternationalSymposium on Artificial Intelligence and Mathematics, AIMATH 2006, Fort Lauderdale.Springer,2006.

[9] Uwe Pfeiffer, Tomas Karnagel and Guido Scheffler.A Sudoku-Solver for Large Puzzles using SAT. LPAR-17-short (EPiC Series, vol. 13), 52–57

[10] Sudoku Puzzles Generating: from Easy to Evil.

http://zhangroup.aporc.org/images/files/Paper\_3485.pdf

**目 录**

[任 务 书 I](#_Toc24047)

[1 引言 1](#_Toc7893)

[1.1 课题背景与意义 1](#_Toc912)

[1.2 国内外研究现状 1](#_Toc23944)

[1.2.1 SAT问题简介 1](#_Toc31701)

[1.2.2 DPLL算法的提出与发展 1](#_Toc3651)

[1.3 课程设计的主要研究工作 2](#_Toc31018)

[2 系统需求分析与总体设计 4](#_Toc21329)

[2.1 系统需求分析 4](#_Toc12283)

[2.2 系统总体设计 4](#_Toc26001)

[2.2.1 主控、交互与显示模块设计 5](#_Toc31551)

[2.2.2 核心DPLL模块、CNF解析模块以及二进制数独模块设计 6](#_Toc2895)

[3 系统详细设计 7](#_Toc19077)

[3.1 有关数据结构的定义 7](#_Toc22179)

[3.2 主要算法设计 7](#_Toc19767)

[3.2.1 CNF解析模块算法设计 7](#_Toc28702)

[3.2.2 核心DPLL模块算法设计 9](#_Toc11570)

[3.2.3 二进制数独模块算法设计 11](#_Toc6029)

[4 系统实现与测试 15](#_Toc31461)

[4.1 系统实现 15](#_Toc5140)

[4.1.1 系统功能 15](#_Toc16272)

[4.1.2 编程环境与运行环境描述 15](#_Toc22007)

[4.1.3 头文件、预定义常量以及数据结构定义说明 15](#_Toc3677)

[4.1.4 函数声明以及各模块函数调用说明 17](#_Toc24518)

[4.2系统测试 21](#_Toc29548)

[5 总结与展望 29](#_Toc31174)

[5.1 全文总结 29](#_Toc22534)

[5.1 工作展望 29](#_Toc2510)

[6 体会 31](#_Toc28045)

[参考文献 34](#_Toc31481)

[附录 35](#_Toc31984)

# 1 引言

## 1.1 课题背景与意义

二进制数独游戏（Binary Puzzle）是一个放置难题，与普通数独游戏（Sudoku）相似，所填入的数字需要满足一系列约束条件。同时二进制数独游戏也是运用纸和笔进行演算和推理的逻辑游戏，从发展至今，由于它具有开发智力，锻炼逻辑思维的优点，被大家所认识以及喜爱。

二进制数独游戏在逻辑上可以归约为一个SAT问题，而SAT问题在人工智能领域的重要地位，使得许多学者都在SAT问题求解领域做了大量的研究，可满足性问题进而也成为了国内外研究的热点问题，并在算法研究和技术实现上取得了较大的突破，这也推动了形式验证和人工智能等领域的发展。在SAT求解器被越来越多地应用到各种实际问题领域的今天，探寻解决SAT问题的高效算法仍然是一个吸引人并且极具挑战性的研究方向。

## 1.2 国内外研究现状

### 1.2.1 SAT问题简介

命题逻辑公式的可满足性问题（satisfiability problem，SAT），是数理逻辑、自动推理、计算机科学、集成电路的设计验证和人工智能等领域中的基本问题，是一个典型的NP完全问题，可广泛应用于许多实际问题如硬件设计、安全协议验证等，具有重要理论意义与应用价值。同时，SAT问题也是各类程序设计与竞赛的经典问题。

### 1.2.2 DPLL算法的提出与发展

最经典的求解SAT问题的完备算法是DPLL算法，它是由Davis和Putnam等人在1960年提出，其它的完备算法大都是在DPLL算法的基础上衍生出来的， 是对DPLL算法的改进。由于SAT问题本身的特性使得其最坏情况下的时间复杂度是指数级别，最初这使得许多的研究者望而却步。而后，S.A.Cook 在1971年证明了SAT问题是NP完全问题，这更加削弱了许多学者研究SAT问题的兴趣，从而导致了SAT问题在很长的一段时间里都没有得到较好的重视，发展非常缓慢，研究成果较少。

在1997年和2003年，Bart Selman和Henry Kautz分别于在人工智能第五届国际合作会议上提出了SAT问题面临的十大挑战性问题，并在2001年和2007年先后对当时的可满足性问题现状进行了全面的阐述和总结。这十大挑战性问题的提出对于SAT基准问题的理论研究和算法改进都起到了强有力的推动作用。这使得越来越多的人开始关注并研究 SAT 问题，所以这段时间也涌现出了众多新的高效的 SAT 算法如 MINISAT、SATO、CHAFF、POSIT和 GRASP等，SAT算法的研究成果显著，求解算法也越来越多地应用到了实际问题领域。

这些新兴的算法大都是基于DPLL算法的改进算法，改进的方面包括：采用新的数据结构、新的变量决策策略或者新的快速的算法实现方案。国内也涌现出了许多高效的求解算法，如1998年作者梁东敏提出了改进的子句加权WSAT算法，2000年金人超和黄文奇提出的并行Solar算法，2002年作者张德富在文献中，提出模拟退火算法。

尽管SAT算法已经取得了举足轻重的改进，但是仍有一些问题没有得到高效的解决，已经解决的问题可能还存在更好的求解算法，因此研究并实现高效率的 求解算法仍是当前要解决的中心问题之一。

## 1.3 课程设计的主要研究工作

本实验主要学习命题逻辑可满足性问题的相关理论知识，基于DPLL的求解器所采用的关键技术及其算法框架进行了研究与实现。要求精心设计问题中变元、文字、子句、公式等有效的物理存储结构，并基于DPLL过程实现一个高效SAT求解器，对于给定的中小规模算例进行求解，输出求解结果，统计求解时间，具体研究工作如下：

1. 了解学习SAT问题的研究背景、意义及研究现状，学习并掌握命题逻辑可满足性问题的基本理论知识。

2. 学习并掌握基于DPLL算法的SAT完备型求解算法，使用C/C++语言实现DPLL算法，并且深入研究DPLL算法的改进策略（包括选词策略、数据结构形式以及算法流程等），对该算法中所使用到的数据结构和一些关键技术给予总结和分析。

3. 实现基于DPLL算法实现的SAT求解器，并对系统的算法以及数据结构进行优化，并给出求解对应的SAT问题所计算出来的优化率。

4. 学习并掌握挖洞法生成唯一解数独棋盘的基本理论知识，并利用挖洞法实现生成具有唯一解的二进制数独棋盘。

5. 使用改进后的DPLL算法解决二进制数独游戏（Binary Puzzle）问题。通过一定形式的编码方式将二进制数独转换为CNF公式，并调用DPLL求解器进行求解。

2 系统需求分析与总体设计

## 2.1 系统需求分析

基于二进制数独问题转化为SAT问题的思路，本次课程设计实验需要设计并实现一个高效的SAT求解器，用于实现二进制数独游戏生成与求解。系统同时具有两个不同的功能，第一，需要能够从外部读入CNF算例并求解；第二，需要能够随机生成有唯一解的二进制数独棋盘（或者从外界读入二进制数独棋盘格局），并对生成的二进制数独棋盘进行求解，最终实现的二进制数独游戏需要具有一定可玩性并且具有简单的交互性。

## 2.2 系统总体设计

本次课程设计实验所设计的系统主要分为4个模块，分别是主控、交互与显示模块，CNF解析模块，核心DPLL模块以及二进制数独模块（系统总体结构以及模块之间的调用关系详见图2-1）。

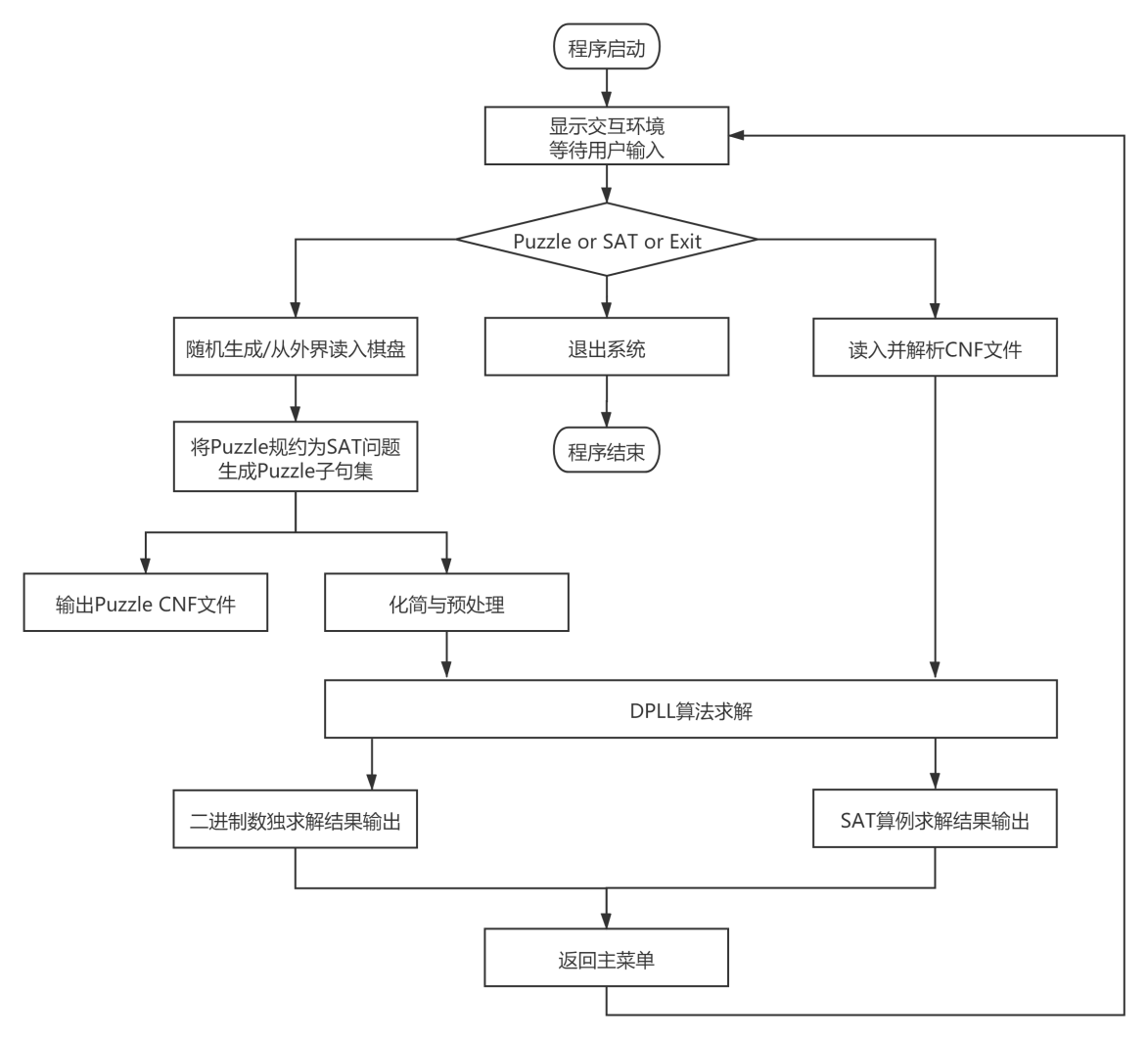


图2-1 实验程序流程图

### 2.2.1 主控、交互与显示模块设计

本次课程设计实验的主控、交互与现实模块的实现，借鉴了上学期的数据结构实验的主控交互框架结构。通过分析本次实验需求，将本次实验的主控交互模块设计成两级菜单的形式，主菜单的主要功能是通过选择不同的数字进入不同的二级菜单，二级菜单的主要功能是求解SAT算例或者解决二进制数独问题。

1. 在主菜单界面中有3个选项，分别可以实现退出当前系统，进入求解SAT算例菜单界面以及进入求解二进制数独游戏问题菜单界面（交互系统主菜单截图如图2-2所示）。

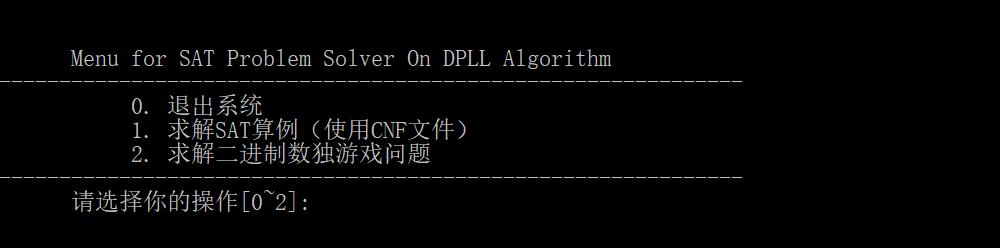


图2-2 系统主菜单截图

2. 交互系统具有两个不同的二级菜单界面，分别可以实现求解SAT算例的功能以及求解二进制数独游戏问题的功能。

求解SAT算例菜单界面有5个选项可供选择，分别实现了读取CNF文件（ReadFile），遍历输出CNF文件信息（TraverseCNF），使用未改进DPLL算法求解CNF算例（NotImproveDPLL），使用改进DPLL算法求解CNF算例（NotImproveDPLL），以及返回主菜单（Return）共5个功能（求解SAT算例二级菜单截图如图2-3-1所示）。

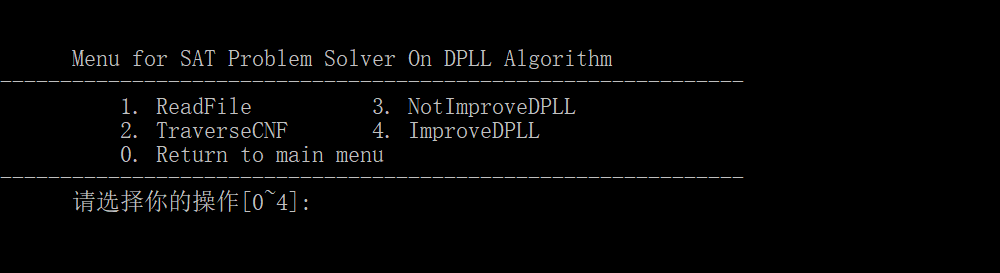


图2-3-1 系统求解SAT算例二级菜单截图

求解二进制数独问题菜单界面有5个选项可供选择，分别实现了生成二进制数独棋盘（GeneratePuzzle），从外界读取二进制数独棋盘信息（ReadPuzzle），保存二进制数独归约成的CNF文件（SaveCNFFile），求解二进制数独游戏（SolvePuzzle），以及返回主菜单（Return）共5个功能（求解二进制数独游戏二级菜单截图如图2-3-2所示）。

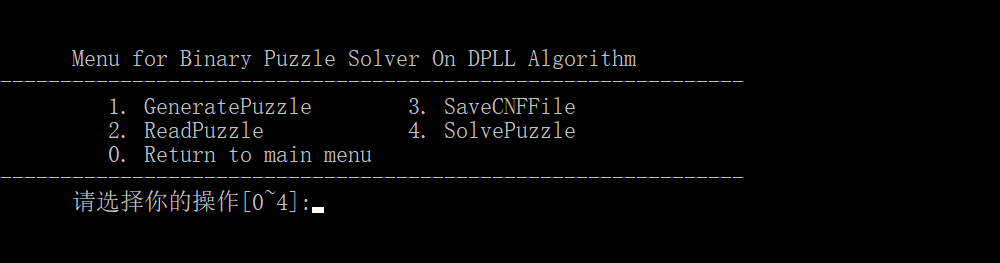


图2-3-2 系统求解二进制数独游戏二级菜单截图

### 2.2.2 核心DPLL模块、CNF解析模块以及二进制数独模块设计

对于CNF解析模块、核心DPLL模块以及二进制数独模块这三大模块的设计将在系统详细设计中进行叙述，在此仅描述各个模块之下的小模块或功能，具体描述如下：

1. 解析CNF模块：CNF文件的读取，DPLL算法辅助函数设计（包括子句生成函数、子句删除函数、判断单子句函数等等）。

2. 核心DPLL模块：DPLL算法，选词策略以及排序函数以及使用单子句规则简化算例函数等。

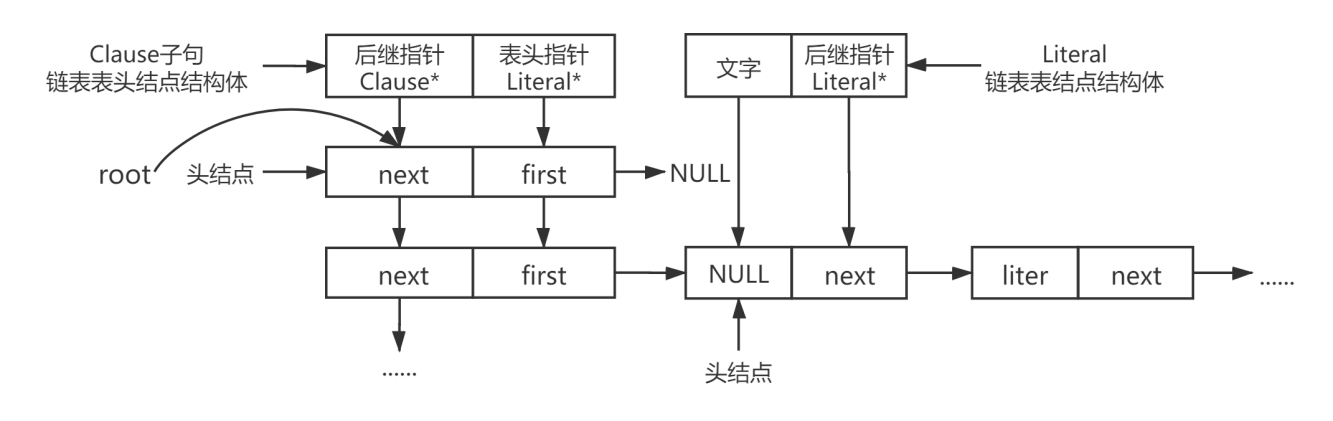
3. 二进制数独模块：生成二进制数独游戏格局，二进制数独归约为SAT化简和预处理，二进制数独游戏求解，输出二进制数独对应的CNF文件。

# 3 系统详细设计

## 3.1 有关数据结构的定义

在本次课程设计实验中，由于需要完成两项任务，分别需要解析并处理CNF文件以及求解二进制数独游戏，故需要处理的数据对象也分成两类，第一类是与处理CNF文件（DPLL算法）相关的变元或文字、子句、公式等；第二类是与二进制数独相关的棋盘数组，但是二进制数独棋盘仅需要设计一个二维数组即可解决，故在此不再赘述，在此将重点描述与处理CNF文件相关的数据结构定义。

由于每个CNF公式变元与子句数可能不同，同一个算例中子句长度也可能不等，因此本次实验采用将子句表示为由文字构成的链表，整个公式则是由子句构成的链表的形式，为使得DPLL算法处理方便，将公式和每个子句都设置为带头结点的链表（具体结构如图3-1所示）。

图3-1 CNF公式存储结构图

此外，为方便有效地实现回溯功能，在DPLL算法中，每次进行选词处理的时候都会复制一个CNF公式进行操作，以免影响回溯，当开始回溯时，则会删除当前复制的CNF公式，以防止内存溢出。

## 3.2 主要算法设计

### 3.2.1 CNF解析模块算法设计

CNF解析模块的主要功能是从文件中读取完整的CNF公式，并对其进行解析，生成可操作的CNF公式数据结构，除此之外，CNF解析模块还需要为其他模块的操作提供基础性的辅助功能，故还需要定义一些基础性的操作以供调用，具体功能及其函数算法设计如下。

（1）ReadFile(Clause\* root,int &num\_of\_liter,int &num\_of\_clause,char filename[])；

算法功能：读取CNF文件并生成可操作的CNF公式。

算法设计：从缓冲区读入文件名称，使用fopen函数打开文件，使用fscanf函数读入注释内容和CNF公式数据，反复调用创建子句（createClause）、子句增加（addClause）等函数，进行CNF公式的创建和生成。

（2）Clause\* createClause(int n,int lit[])；

算法功能：创建CNF公式子句（带头结点）。

算法设计：使用malloc函数分配空间，依次将lit数组中的CNF文字数据转化为链表的表结点，创建成功后返回子句的地址，否则返回NULL。

（3）status destroyClause(Clause\* p)；

算法功能：删除CNF公式子句。

算法设计：使用遍历指针遍历CNF公式子句，依次删除所有子句的结点，最后删除子句结点，删除成功返回OK，否则返回ERROR。

（4）status addClause(Clause\* root,Clause\* pNew)；

算法功能：将新生成的CNF子句加入CNF公式中。

算法设计：将CNF子句使用首插法插入CNF公式，插入成功返回OK，否则返回ERROR。

（5）status isUnitClause(Clause\* p)；

算法功能：判断当前CNF子句是否为单子句。

算法设计：如果当前子句不存在或者存在多与1个的表结点（不包括头结点），则表示不是单子句，返回FALSE，否则返回TRUE。

（6）status printClause(Clause\* root)；

算法功能：打印CNF公式。

算法设计：使用遍历指针遍历链表，按规定格式输出整个CNF公式，输出成功返回OK，否则返回ERROR。

（7）Clause\* copyList(Clause\* root)；

算法功能：复制CNF公式。

算法设计：使用遍历指针遍历链表，依次复制整个CNF公式，输出成功返回复制的链表地址，否则返回NULL。

（8）status destroyList(Clause\* root)；

算法功能：删除CNF公式。

算法设计：使用遍历指针遍历链表，反复调用删除子句函数（destroyClause）删除整个CNF公式，删除成功返回OK，否则返回ERROR。

（9）status addliter(Clause\* p,int lit)；

算法功能：将文字加入CNF子句。

算法设计：使用首插法将文字结点加入CNF子句，成功则返回OK，否则返回ERROR。

### 3.2.2 核心DPLL模块算法设计

核心DPLL模块的主要功能是对已经解析/生成的CNF公式进行求解，判断其是否有可满足的解，并按照要求的格式输出求解的结果。根据DPLL算法的思想和求解规则，核心DPLL模块的算法设计主要需要完成3个功能，第一，递归实现的DPLL处理功能；第二，选词功能；第三，合法的结果输出功能，具体功能实现及其函数算法设计如下。

（1）status preSolve(Clause\* root,int num\_of\_liter,int &num\_of\_clause)；

算法功能：对CNF公式进行预处理，以减少运算量。

算法设计：预处理分为两个部分，第一个部分是判断是否存在空子句，若出现空子句则表示不可满足，直接返回ERROR；第二个部分是使用单子句规则处理CNF公式，由于不存在选词故不需要回溯，若处理过后出现CNF为空的情况，表明该CNF公式已经满足，返回OK，若出现CNF公式中存在空子句的情况，表明一定不可满足，返回ERROR。

（2）law\_of\_UnitClause(Clause\* root,Clause\* p\_of\_unitCla,int &num\_of\_clause)；

算法功能：使用单子句规则处理CNF公式。

算法设计：由于本次实验采用的数据结构为链表形式，且不设置标记位，为方便回溯，防止修改结点影响回溯效果，采用了复制旧有链表进行处理，回溯时删除链表的方式进行操作。所以在使用单子句规则处理CNF公式时，只需要将含有文字的子句直接删除，含有反文字的结点删除即可。故需定义遍历指针遍历CNF公式，当结点文字为单子句文字时，将整句删去，结点文字为单子句文字的反文字时，删去结点即可（算法流程图见图3-2）。

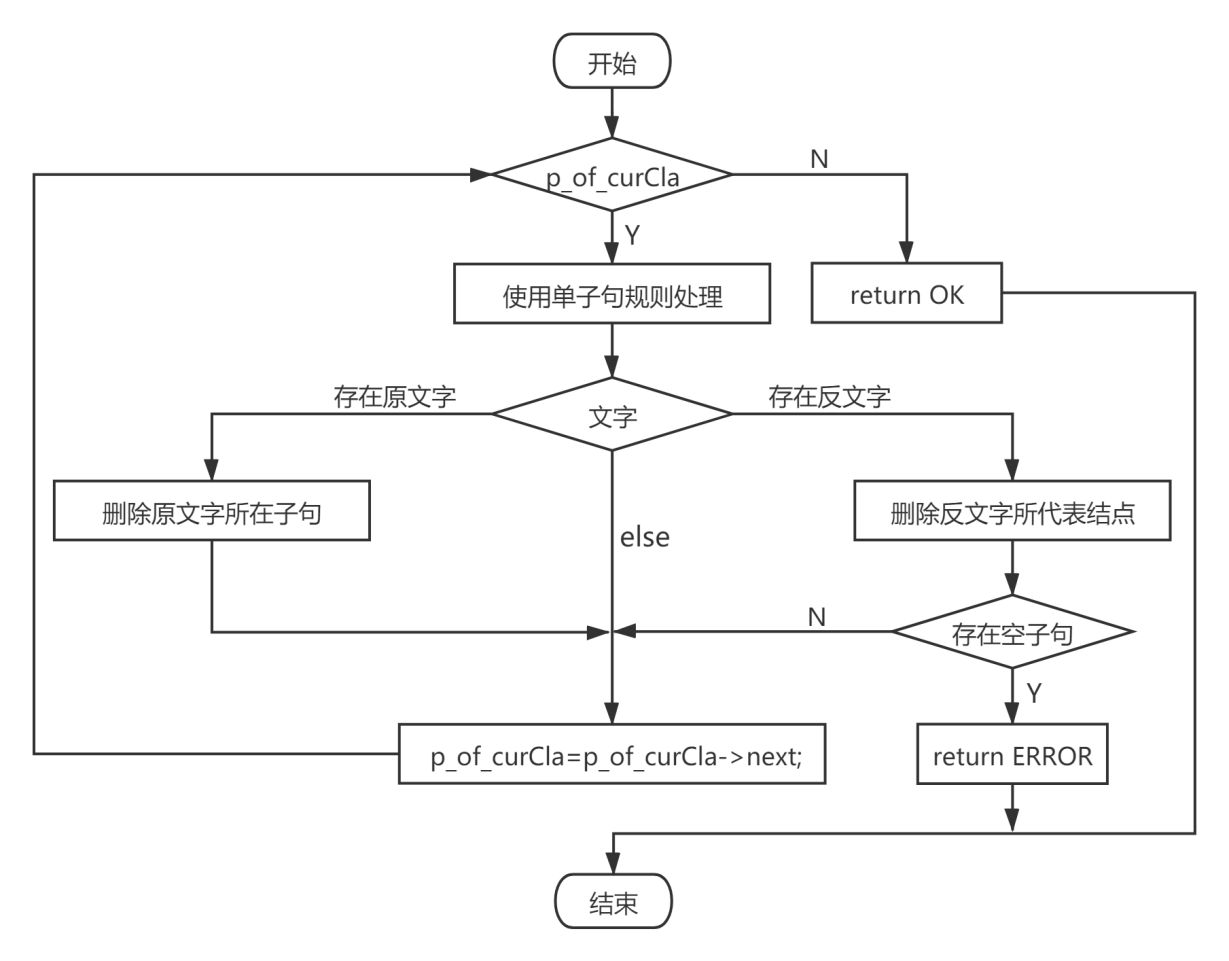


图3-2 单子句规则处理函数流程图

（3）status DPLL(Clause\* root,int lit,int num\_of\_liter,int num\_of\_clause)；

算法功能：核心DPLL函数，求解可满足性。

算法设计：为防止影响回溯效果，定义一个Clause\*类型指针rNew指向新复制的CNF公式，变量lit为选中的文字，将lit中的文字转化为单子句加入CNF公式中，然后使用单子句规则简化CNF公式。若law\_of\_UnitClause函数的返回值为ERROR则表示不满足，删除当前复制链表，然后返回FALSE开始回溯；若返回值为TRUE则继续简化，直到遍历完整个CNF公式。当遍历完成后，若CNF公式为空，代表可满足，得出结果返回TRUE；若不为空则调用选词模块，进行下一轮递归，直到递归完整棵搜索树或得出可满足解。

（4）status Recovery\_lit(Clause\* p)；

算法功能：恢复文字，用于递归失败回溯时，恢复之前的选词文字。

算法设计：采用遍历指针依次读取链表中的文字，对数组进行恢复。

（5）status Literalsum(Clause \*root,int num\_of\_liter)；

算法功能：文字选择函数，用于文字选择（JW策略）。

算法设计：本函数采用JW策略选择文字，通过对子句的2次遍历，先统计子句长度，再统计J值，采用遍历指针遍历CNF公式，完成统计返回OK（JW策略详细信息如图3-3所示）。

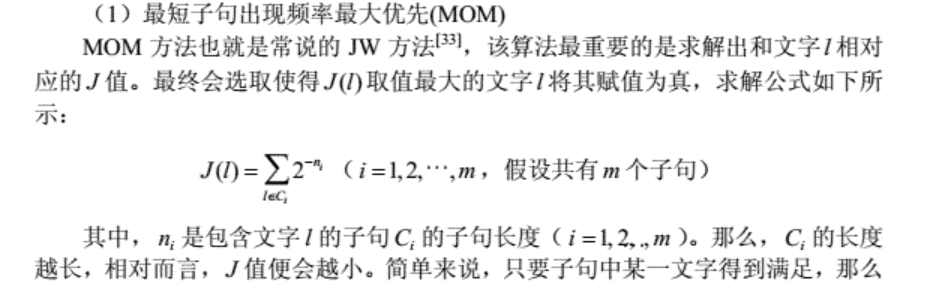


图3-3 最短子句出现频率最大优先规则详图

（6）Sum\_of\_Liter Literalmax(int num\_of\_liter)；

算法功能：与Literalsum函数一起构成选词模块。

算法设计：遍历所有文字，选出J值最大的文字作为备选文字。

（7）int find\_lit(int num\_of\_liter)；

算法功能：用于未优化DPLL函数的暴力选词方式。

算法设计：按照文字从小到大的顺序选择未被赋值的文字。

（8）status SaveFile(int time,char filename[],int flag,int num\_of\_liter)；

算法功能：按规定格式输出DPLL处理结果。

算法设计：使用字符串变量filename传递文件名，然后按照规定格式，使用fopen函数创建文件，fprintf函数输出结果。

### 3.2.3 二进制数独模块算法设计

根据本次课程设计的具体实验要求，二进制数独模块需要有以下几个功能，第一，能够生成具有唯一解的二进制棋盘格局，或者可以从外界读入棋盘格局；第二，能将二进制数独棋盘格局转化为可操作的CNF公式，并具有输出为文件的功能；第三，能够求解二进制数独棋局，并以合理的方式输出，为实现以上三个功能，本模块的函数较多，具体功能实现及其函数算法设计如下。

（1）int trans\_of\_comliter(int i,int j,int n,int flag)；

算法功能：将二进制数独单元格转化的布尔变元转化为连续自然数（n\*n）。

算法设计：对于由二进制数独单元格直接转化来的布尔变元编码，任意第i行j列单元格，采用(i-1)×n+j作为转化公式进行转化。为方便后续函数同时对行与列进行处理，设置了flag标志变量作为行序/列续优先的标志变量，返回值为转化后的文字。

（2）int trans\_of\_extraliter(int a,int b,int c,int d,int e,int n)；

算法功能：将引入的附加变元转化为连续自然数（3\*n\*n\*(n-1)个变元）。

算法设计：根据附加变元的生成规则可知，附加变元由5个整数组成，这里使用a、b、c、d、e五个整型变量代表附加变元的5个部分。变量a只能取1或2，其中1为行标志数，2为列标志数；变量b、c、d都可以从1取到n（n可以为二位整数），变量b和c分别表示不同的两行/两列，其中c需要大于b；变量e为正反与标志变量，其中0表示反文字，1表示正文字，2表示空缺，具体的转化公式为n\*n+(a-1)\*n\*(n-1)\*n\*3/2+((c-1)\*(c-2)/2+b-1)\*3\*n+3\*d+e-2。

（3）status firstlaw(Clause\* root,int &num\_of\_clause,int n,int flag)；

算法功能：使用约束1处理布尔变元，生成相对应的子句。

算法设计：根据约束1的要求，需要二进制数独在每一行、每一列中不允许有连续的3个1或3个0出现，为防止重复，规定子句中的三个文字从小到大排列，采用三重循环的方式，调用子句创建函数创造创造子句。

（4）status secondlaw(Clause\* root,int &num\_of\_clause,int n)；

算法功能：本函数与secondlaw\_of\_DEP函数共同完成约束2的处理。

算法设计：本函数是递归函数secondlaw\_of\_DEP的门函数，使用FOR循环计数器变量i由1增加到n，将每一行、每一列加上正负共四种情况分别调用函数secondlaw\_of\_DEP进行处理。

（5）status secondlaw\_of\_DEP(Clause\* root,

int &num\_of\_clause,int layer,int i,int j,int n,int flag1,int flag2)；

算法功能：本函数与secondlaw函数共同完成约束2的处理。

算法设计：本函数采用递归实现，递归的深度由变量layer控制，最大深度为n/2+1，变量i用于传递行列数，变量j用于传递上一个文字的位置，以便于控制递归深度较深的文字的数值大于较浅的的文字，以防止重复重复，然后再递归最后一层调用创建子句函数创建子句。

（6）status thirdlaw(Clause\* root,int &num\_of\_clause,int n)；

算法功能：根据约束3完成对附加变元的处理，生成子句。

算法设计：使用互补律拓展附加变元表达式，使用四重循环分别以变量a、b、c、d作为计数器，e分别为0、1、2三种情况进行处理，分别调用子句创建和加入函数创建子句并加入CNF公式中。

（7）status ReadPuzzle\_File(int \*puzzle,int n)；

算法功能：从文件中读取二进制数独棋盘格局。

算法设计：使用scanf函数从缓冲区读入文件名，使用fopen函数打开文件，fscanf函数读取棋盘格局并用puzzle数组进行转化和存储。

（8）status ReadPuzzle\_Buffer(int \*puzzle,int n)；

算法功能：从缓冲区读入二进制数独棋盘格局。

算法设计：使用scanf函数从缓冲区读取二进制数独棋盘格局，用puzzle数组进行转化和存储。

（9）SaveCNFFile(Clause\* root,int \*puzzle,int n,int num\_of\_liter,int num\_of\_clause)

算法功能：将CNF公式保存为文件的形式。

算法设计：使用scanf函数从缓冲区读入文件名，使用fopen函数创建文件，使用二重循环的方式按照一定格式输出CNF公式。

（10）status transPuzzle\_to\_CNF(Clause\* root,int &num\_of\_clause,int n,int \*a)；

算法功能：将二进制数独棋盘格局转化为CNF公式。

算法设计：棋盘的转化过程分为四步，第一步，使用约束1处理CNF公式；第二步，使用约束2处理CNF公式；第三步，使用约束3引入附加变元，生成CNF子句；第四步，将所有已经填入的空格转化为单子句。

（11）status puzzleslover(Clause\* root,int n,int num\_of\_clause)；

算法功能：二进制数独棋盘处理器，用于处理二进制棋盘。

算法设计：分别使用预处理函数和DPLL函数处理CNF公式。

（12）status puzzles(int n,int \*a)；

算法功能：二进制数独棋盘处理器，用于处理二进制棋盘。

算法设计：调用CNF公式转化函数生成CNF公式，并用二进制数独棋盘处理器处理二进制数独棋盘。

（12）status generate\_of\_puzzle(int n,int \*c)；

算法功能：生成具有唯一解的二进制数独棋局。

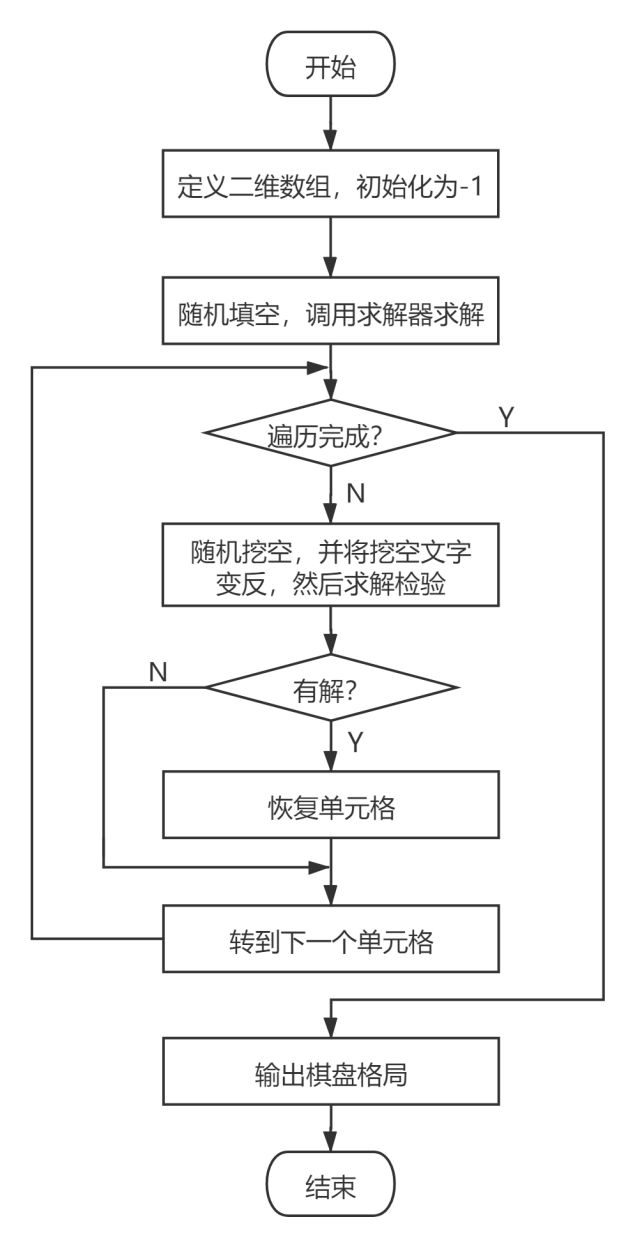
算法设计：要生成一个具有唯一解的二进制数独棋盘主要分为两个步骤，第一个步骤是先随机生成一个合法的完整的二进制数独，第二个步骤是采用挖洞法随机挖去部分单元格，生成有唯一解的二进制数独棋盘，具体生成方式如下。先创建一个n\*n的二维数组存储棋局并将每个单元初始化为-1，然后随机填入几个空（对于4阶和6阶的二进制数独应适当减少填入数字的个数以免出现无法求解的情况），调用二进制数独处理函数求解数独，生成合法的解。当合法的二进制数独产生后，使用挖洞法挖空，对于每个单元格设置一定概率可以将空挖去，然后将挖去的空改为相反的文字（0变1,1变0），再调用求解器求解，若有解则表示解不唯一，该空跳过遍历整个二维数组完成挖空（函数流程图见图3-4）。

图3-4 二进制数独棋盘格局生成函数流程图

（12）status printpuzzle(int \*puzzle,int n)；

算法功能：打印棋盘格局，便于图形化输出。

算法设计：使用双重循环进行输出，当puzzle单元格的数值为-1时表示未填写，输出符号“\_”，数值为0或1时直接输出即可，每次输出一行后换行输出。

# 4 系统实现与测试

## 4.1 系统实现

### 4.1.1 系统功能

本系统的主要实现目标为，基于DPLL过程实现一个较高效SAT求解器，并对于给定的中小规模算例进行求解，按照输出规则将求解结果以及求解时间输出为合法文件。根据系统要求，达成如下6个功能：

1. 输入输出功能：实现程序执行参数的输入，SAT算例CNF文件的读取，执行结果的输出与文件保存等。

2. 公式解析与验证：读取CNF算例文件，解析文件，建立公式的内部表示；实现对解析正确性的验证功能，即遍历内部结构逐行输出与显示每个子句。

3. 基于DPLL的求解器：基于DPLL算法框架，实现SAT算例的求解。

4. 时间性能的测量：实现基于相应的时间处理函数，记录DPLL过程执行时间，并作为输出信息的一部分。

5. 程序优化：对基本DPLL的实现进行存储结构、分支变元选取策略等某一方面进行优化设计与实现，提供明确的性能优化率结果。

6. SAT应用：将数独游戏问题转化为SAT问题，并集成到上面的求解器进行问题求解，游戏可玩，具有简单的交互性。

### 4.1.2 编程环境与运行环境描述

编程环境：采用CodeBlocks 16.01编程软件编写。

运行环境：操作系统 Microsoft Windows 10 (64位)

CPU Intel i5-7200U

内存 8.00 GB

显卡 NVIDIA GeForce 940MX

### 4.1.3 头文件、预定义常量以及数据结构定义说明

1. 头文件

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

2. 预定义常量

#define N 100000

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

3. 类型表达式以及数据结构定义说明

（1）表示状态的类型定义表达式：typedef int status;

（2）CNF公式链表文字表结点数据结构，结点用于存储文字信息，具体数据结构结构体定义如下：

typedef struct Literal{

int liter; //用于存储文字信息

Literal\* next; //表结点指针，指向后继结点

}Literal;

（3）CNF公式链表的表头结点，其含义为一个CNF公式子句的开始，结点中仅需包含两个指针即可，具体数据结构结构体定义如下：

typedef struct Clause{

Literal\* first; //指向链表的头结点，以便进行删除或插入操作

Clause\* next; //指向下一个后继表头结点

}Clause;

（4）CNF公式文字信息结构体，其作用主要为保存各个文字的文字信息，文字信息主要包括，文字数值、J值（JW选词策略选词指标）、正负文字的个数共四条信息，其具体结构体定义如下：

typedef struct{

int liter;

double sum;

int num\_of\_positive;

int num\_of\_negative;

}Sum\_of\_Liter;

### 4.1.4 函数声明以及各模块函数调用说明

本次课程设计实验的整个系统分为四个模块，分别为主控、交互与显示模块（仅包含主函数）、CNF解析模块、核心DPLL模块以及二进制数独模块，其中二进制数独模块需要完成二进制数独游戏的格局的生成、归约以及求解和输出功能，各个模块的函数声明以及调用情况如下。

1. CNF解析模块

该模块的主要功能为从文件中读取完整的CNF公式，并对其进行解析，生成可操作的CNF公式数据结构，除此之外，CNF解析模块还需要为其他模块的操作提供基础性的辅助功能，各个功能的实现以及函数声明如下。

（1）读取CNF文件并解析生成可操作的CNF公式；

函数声明：

子句创建函数：Clause\* createClause(int n,int lit[]);

子句增加函数：status addClause(Clause\* root,Clause\* pNew);

读CNF文件函数：status ReadFile(Clause\* root,int &num\_of\_liter,

int &num\_of\_clause,char filename[]);

函数调用：主函数调用ReadFile函数读取CNF文件并生成CNF公式，ReadFile函数调用fopen函数以及fscanf函数读取CNF文件的内容，调用createClause函数生成CNF子句，addClause函数将子句加入CNF公式。

（2）对CNF公式及其子句文字的基础性辅助操作；

函数声明：

子句创建函数：Clause\* createClause(int n,int lit[]);

子句销毁函数：status destroyClause(Clause\* p);

子句增加函数：status addClause(Clause\* root,Clause\* pNew);

判断单子句函数：status isUnitClause(Clause\* p);

打印CNF公式函数：status printClause(Clause\* root);

复制CNF公式函数：Clause\* copyList(Clause\* root);

销毁CNF公式函数：status destroyList(Clause\* root);

加入文字函数：status addliter(Clause\* p,int lit);

2. 核心DPLL模块

该模块的主要功能是对已经解析/生成的CNF公式进行求解，判断其是否有可满足的解，并按照要求的格式输出求解的结果，功能实现以及函数声明如下。

（1）文字选词功能；

函数声明：

未优化暴力求解选词函数：int find\_lit(int num\_of\_liter);

策略J值计算函数：status Literalsum(Clause \*root,int num\_of\_liter);

优化求解选词函数：Sum\_of\_Liter Literalmax(int num\_of\_liter);

函数调用：未优化选词是从小到大选择第一个没有被赋值的文字作为操作词，优化后的选词采用了JW策略，当需进行选词操作时，先调用计算函数Literalsum计算J值，然后调用Literalmax函数选择J值最大的且未赋值的文字作为操作词。

（2）合法的结果输出功能；

函数声明：

结果保存函数：status SaveFile(int time,char filename[],int flag,int num\_of\_liter)

函数调用：调用fopen函数创建文件，fprintf函数按照格式输出结果。

（3）递归实现的DPLL处理功能；

函数声明：

文字赋值恢复函数：status Recovery\_lit(Clause\* p);

核心DPLL处理函数：status DPLL(Clause\* root,int lit,int num\_of\_liter,

int num\_of\_clause);

单子句规则处理函数：status law\_of\_UnitClause(Clause\* root,

Clause\* p\_of\_unitCla,int &num\_of\_clause);

函数调用：主函数调用DPLL函数进行CNF公式的求解，进入DPLL函数后，先调用copyList函数复制CNF链表进行操作，然后调用createClause和addClause函数创建单子句并加入新复制的CNF公式中，再调用law\_of\_UnitClause函数进行单子句处理，处理完毕后，调用选词函数Literalsum和Literalmax进行选词，最后递归调用DPLL函数处理，当递归失败回溯时，需要使用Recovery\_lit函数恢复赋值，destroyList函数删除CNF链表，以防止内存溢出（函数调用关系如图4-1）。

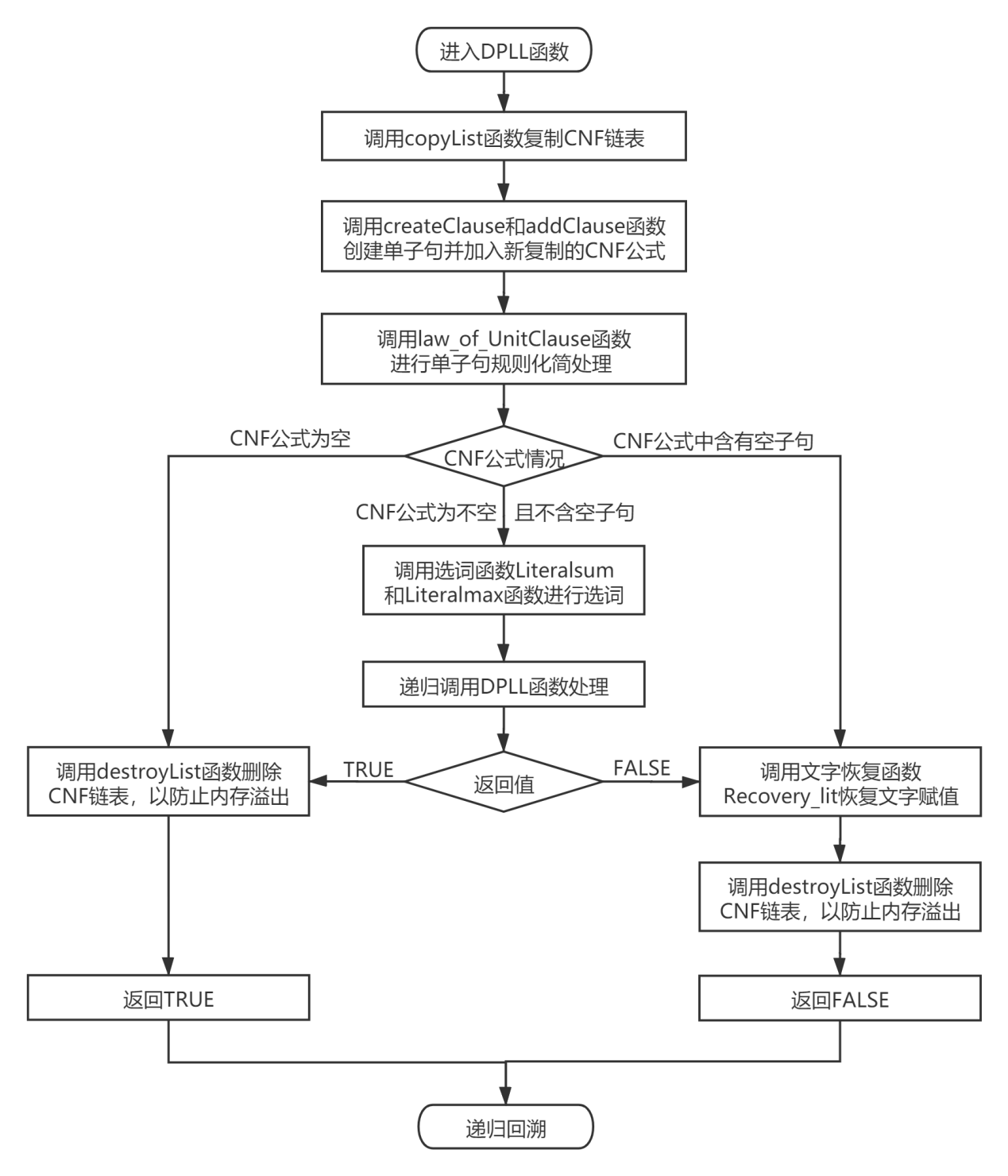


图4-1 DPLL模块函数调用关系图

3. 二进制数独模块

该模块的主要功能要完成二进制数独游戏的格局的生成（或者从外界读入）、归约以及求解和输出功能，其中需要确保生成的二进制棋盘格局具有唯一解，具体功能实现及其函数声明如下。

函数声明：

普通变元转化函数：int trans\_of\_comliter(int i,int j,int n,int flag);

附加变元转化函数：int trans\_of\_extraliter(int a,int b,int c,int d,int e,int n);

约束1处理函数：status firstlaw(Clause\* root,int &num\_of\_clause,

int n,int flag);

约束2处理递归函数：status secondlaw\_of\_DEP(Clause\* root,

int &num\_of\_clause,int layer,int i,int j,int n,int flag1,int flag2);

约束2处理端口函数：status secondlaw(Clause\* root,

int &num\_of\_clause,int n);

约束3处理函数：status thirdlaw(Clause\* root,int &num\_of\_clause,int n);

文件读取棋盘函数：status ReadPuzzle\_File(int \*puzzle,int n);

缓冲区读取棋盘函数：status ReadPuzzle\_Buffer(int \*puzzle,int n);

数独棋盘归约转化函数：status transPuzzle\_to\_CNF(Clause\* root,

int &num\_of\_clause,int n,int \*a);

CNF公式文件保存函数：status SaveCNFFile(Clause\* root,int \*puzzle,

int n,int num\_of\_liter,int num\_of\_clause);

数独求解器：status puzzles(int n,int \*a);

status puzzleslover(Clause\* root,int n,int num\_of\_clause);

数独生成函数：status generate\_of\_puzzle(int n,int \*c);

数独棋盘打印函数：status printpuzzle(int \*puzzle,int n);

（1）二进制数独棋盘格局的归约化功能；

函数调用：首先，扫描二进制数独棋盘，读取已经填入的数字，调用转化函数trans\_of\_comliter将单元格转化为布尔变元，然后调用createClause和addClause函数创建单子句并加入CNF公式中；然后，分别调用firstlaw、secondlaw及thirdlaw三个约束处理函数进行处理，将附加变元及其生产的子句加入CNF公式中，二进制棋盘归约化完成。

（2）二进制数独棋盘的求解功能；

函数调用：调用二进制数独求解器puzzles进行求解，首先调用棋盘格局归约函数transPuzzle\_to\_CNF将二进制数独棋盘转化为CNF公式，然后调用求解函数puzzleslover进行求解，其中puzzleslover函数中包含选词功能和DPLL求解功能。

（3）二进制数独游戏格局生成功能；

函数调用：主函数调用generate\_of\_puzzle函数进行二进制数独棋盘格局的生成；进入generate\_of\_puzzle函数后，先随机生成少数几个棋盘单元格，调用puzzles数独求解器进行求解，求解出合法的棋盘格局；然后从第一个单元格开始随机挖空，每次挖出一个空，都需要将挖出的空的值取反后调用puzzles求解器进行求解，已验证是否具有唯一解，当遍历完成后，输出棋盘（二进制棋盘格局生成功能模块调用关系如图4-2所示）。

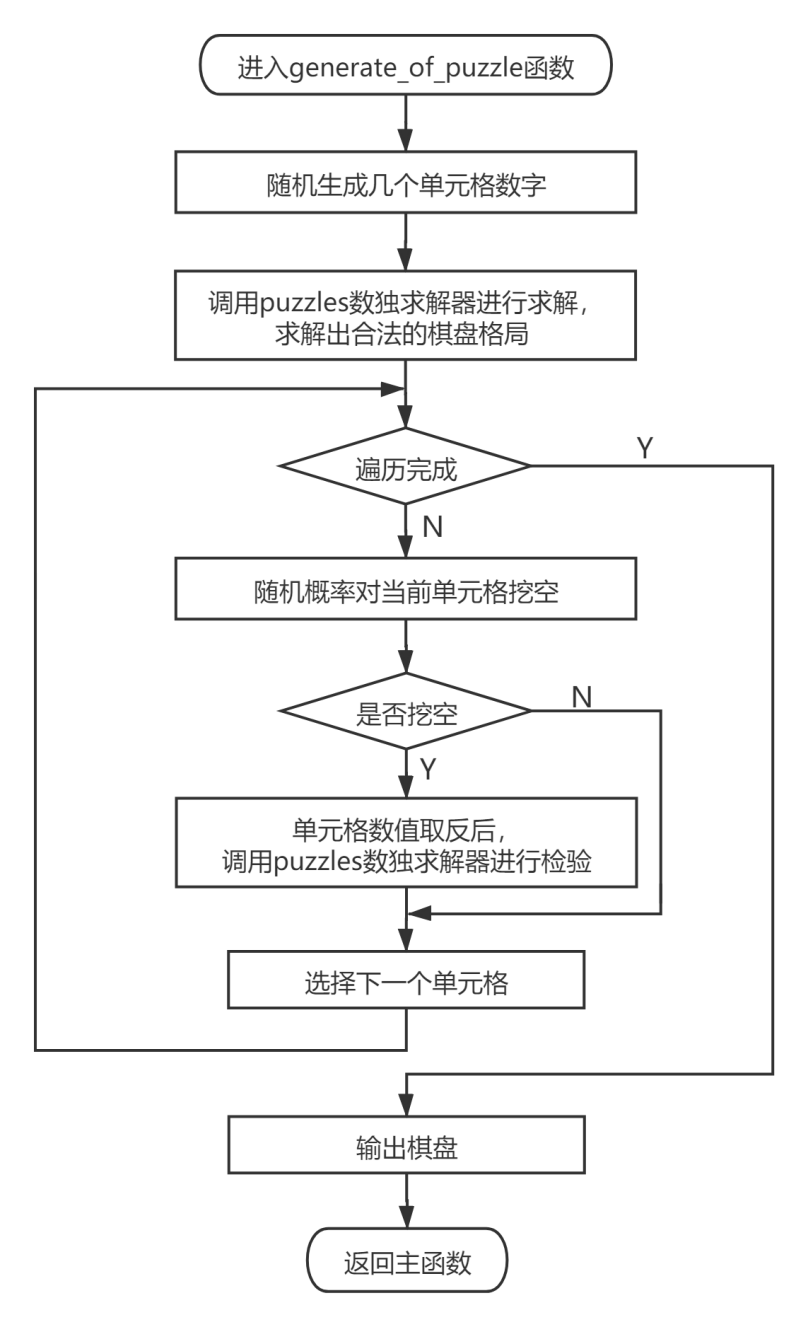


图4-2 二进制数独生成函数调用关系图

4.2系统测试

本次课程设计实验的系统测试主要对本系统的几个具有代表性和重要功能的功能模块进行测试，具体测试模块有主控、交互与显示模块，CNF文件读取模块，核心DPLL求解模块，二进制数独生成模块以及二进制数独求解模块。

1. 主控、交互与显示模块测试

本模块的主要功能就是通过合理的设计，实现清晰明确的显示和交互功能，从而有更加友好的人机交互环境。根据测试的需求，本次测试主要对合法和非法两种输入进行测试，具体测试的主要内容如下：

①根据菜单编号正确输入；

②没有根据菜单编号的输入错误即输入越界；

表4-1 主控、交互与显示模块测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试内容 | 测试输入 | 理论输出 | 实际输出 |
| ① | 1. 主界面输入1进入次级界面；  2. 次级界面输入0返回主界面；  3. 主界面输入0退出系统； | 1. 进入次级界面  2. 返回主界面  3. 退出系统成功 | （输出结果图，如图4-3-1所示） |
| ② | 1. 主界面输入200；  2. 主界面输入1进入次级界面，再输入200； | 1. 显示“输入非法”  2. 显示“输入非法” | （输出结果图，如图4-3-2及图4-3-3所示） |

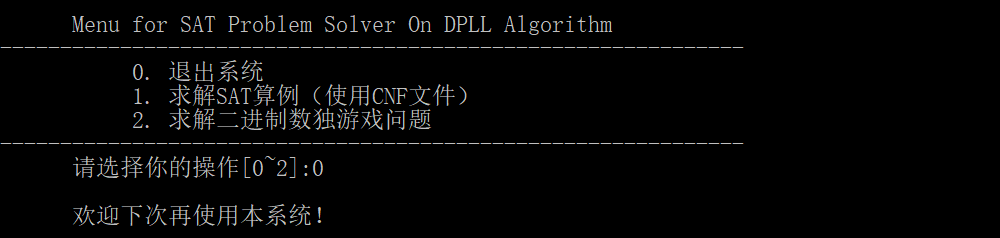


图4-3-1 主控、交互与显示模块测试内容1结果图

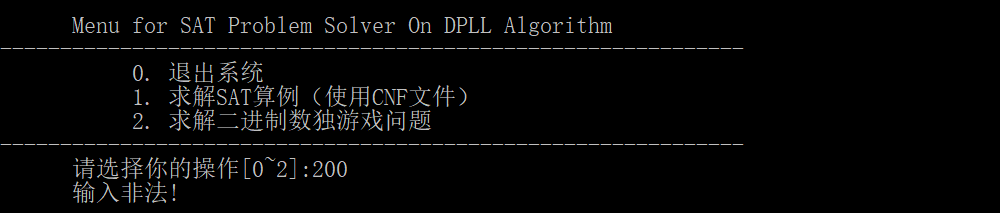


图4-3-2 主控、交互与显示模块测试内容2结果图1

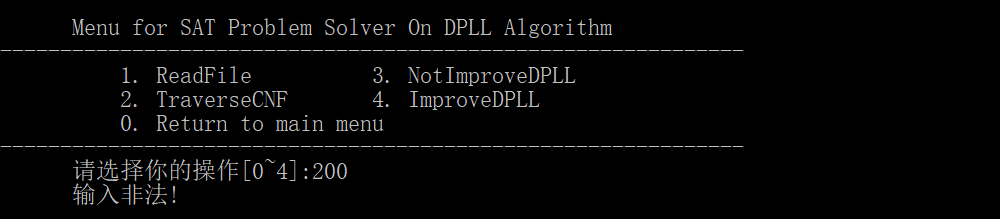


图4-3-3 主控、交互与显示模块测试内容2结果图2

2. CNF文件读取模块测试

本模块的主要功能是读取CNF文件，并输出验证是否读取正确。根据测试需求，具体测试的主要内容如下：

①读取CNF算例文件；

②调用输出模块，输出验证是否读取正确；

表4-2 CNF文件读取模块测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试内容 | 测试输入 | 理论输出 | 实际输出 |
| ① | 1. 主界面输入1进入次级界面；  2. 次级界面输入1，进入CNF文件读取选项；  3. 输入CNF文件路径； | 1. 进入次级界面  2. 显示提示语  3. 显示CNF文件中注释内容，并显示“CNF文件读取成功！” | （输出结果图，如图4-4-1所示） |
| ② | 1. 次级界面输入2，进入CNF公式输出模块； | 1. 显示文字数与子句数，并CNF公式内容 | （输出结果图，如图4-4-2所示） |

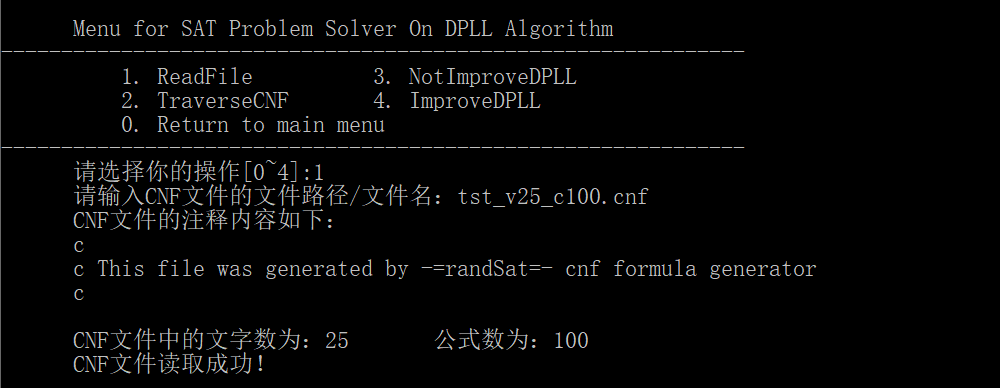


图4-4-1 CNF文件读取模块测试内容1结果图

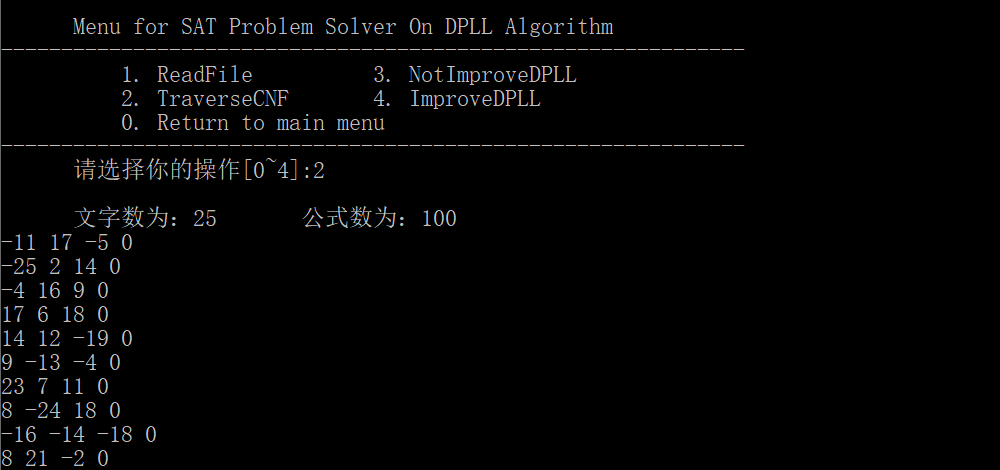


图4-4-2 CNF文件读取模块测试内容2结果图

3. 核心DPLL求解模块

本模块的主要功能是使用未优化以及优化后的DPLL算法求解CNF公式。根据测试需求，本次测试需要完成对满足算例以及未满足算例的求解，并给出优化前后的运行时间和优化率的对比，具体测试的主要内容如下：

①基准满足CNF算例测试；

②基准不满足CNF算例测试；

③SAT算例18组测试（满足算例15组，不满足算例3组）；

表4-3 核心DPLL求解模块测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试内容 | 测试用例 | 理论输出 | 实际输出 |
| ① | sat-20.cnf | 求解成功！ | （输出结果图，如图4-5-1所示） |
| ② | unsat-5cnf-30.cnf | 求解失败！ | （输出结果图，如图4-5-2所示） |
| ③ | 具体测试用例见表4-4 | 输出运行时间与优化率 | 测试结果见表4-4 |

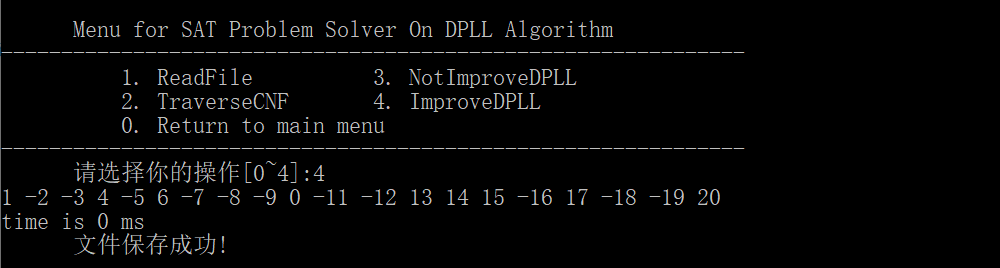


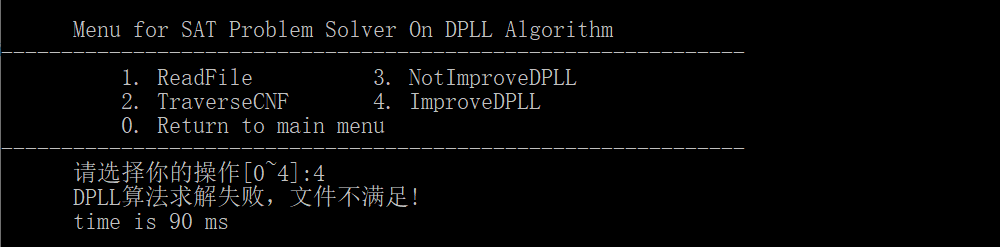
图4-5-1 核心DPLL求解模块测试内容1结果图

图4-5-2 核心DPLL求解模块测试内容2结果图

测试③的主要目的是**对比DPLL算法在优化前和优化后的运行时间**，并用优化率作为指标来数字化显示。本次算法在未优化时没有特殊文字选取策略，**经过测试大量不同的选词策略后，本次优化后的算法采取了JW策略进行选词**，测试证明，**对于大多数的测试算例，经过优化后的算法求解时间大幅减少**，对于少量由数独转化的CNF文件，优化算法的效率反而有所下降，具体测试样例以及结果如表4-4所示（注：若求解时间超过300000ms则填写**∞**）。

表4-4 CNF算例测试样例及结果表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 测试用例 | 变元数 | 公式数 | 未优化时间（ms） | 优化后时间（ms） | 优化率（%） |
| 1 | u-5cnf\_3500\_3500\_30f1.shuffled-30.cnf | 30 | 420 | 263 | 82 | 68.82 |
| 2 | u-problem7-50.cnf | 50 | 100 | 1268 | 120 | 90.54 |
| 3 | u-as.sat05-1171.cnf | 80 | 370 | 23234 | 3553 | 84.71 |
| 4 | problem2-50.cnf | 50 | 80 | 935 | 1 | 99.89 |
| 5 | problem6-50.cnf | 50 | 100 | 267 | 1 | 99.63 |
| 6 | problem9-100.cnf | 100 | 200 | 294809 | 21 | 99.99 |
| 7 | problem11-100.cnf | 100 | 600 | 483 | 15 | 96.89 |
| 8 | CBS\_k3\_n100\_m403\_b10\_0.cnf | 100 | 403 | 9238 | 5 | 99.95 |
| 9 | tst\_v100\_c425.cnf | 100 | 425 | 452 | 5 | 98.89 |
| 10 | problem3-100.cnf | 100 | 340 | 12354 | 34 | 99.72 |
| 11 | tst\_v100\_c400.cnf | 100 | 400 | 136 | 5 | 96.32 |
| 12 | problem12-200.cnf | 200 | 1200 | 32904 | 555 | 98.31 |
| 13 | bart17.shuffled-231.cnf | 231 | 1166 | **∞** | 515 | 100 |
| 14 | bw\_large.a.cnf | 459 | 4675 | 2406 | 229 | 90.48 |
| 15 | sw100-70.cnf | 500 | 3100 | 23971 | 171 | 99.29 |
| 16 | eh-dp04s04.  shuffled-1075.cnf | 1075 | 3152 | 43937 | 4137 | 90.58 |
| 17 | puzzle-8-2.cnf | 1408 | 6089 | 608 | 95 | 84.38 |
| 18 | ec-iso-ukn009.shuffled-as.sat05-3632-1584.cnf | 1584 | 16587 | **∞** | 817 | 100 |

4. 二进制数独生成模块

本模块的主要功能是生成一个具有唯一解的二进制数独棋盘格局。根据测试需求，具体测试的主要内容如下：

①生成4阶二进制数独测试；

②生成8阶二进制数独测试；

③生成10阶二进制数独测试；

表4-5 二进制数独生成模块测试用例表

|  |  |  |
| --- | --- | --- |
| 测试内容 | 测试阶数 | 实际输出 |
| ① | 4阶二进制数独 | （输出结果图，如图4-6-1所示） |
| ② | 8阶二进制数独 | （输出结果图，如图4-6-2所示） |
| ③ | 10阶二进制数独 | （输出结果图，如图4-6-3所示） |

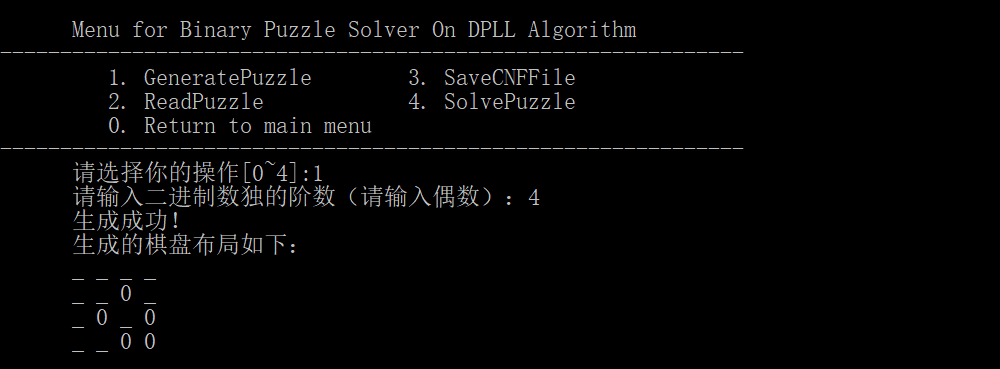


图4-6-1 二进制数独4阶生成结果图

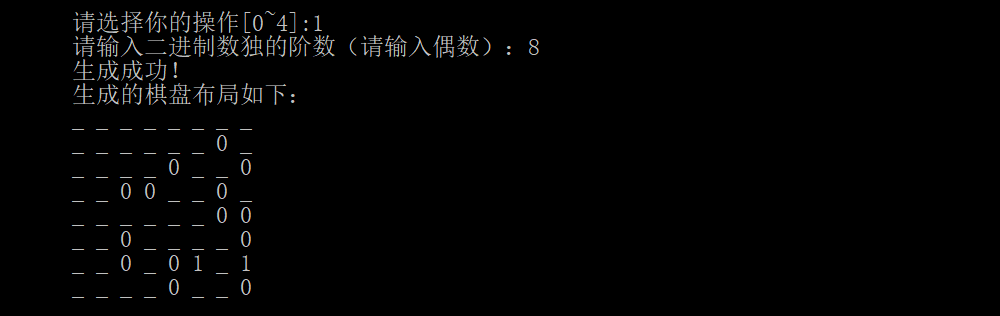


图4-6-2 二进制数独8阶生成结果图

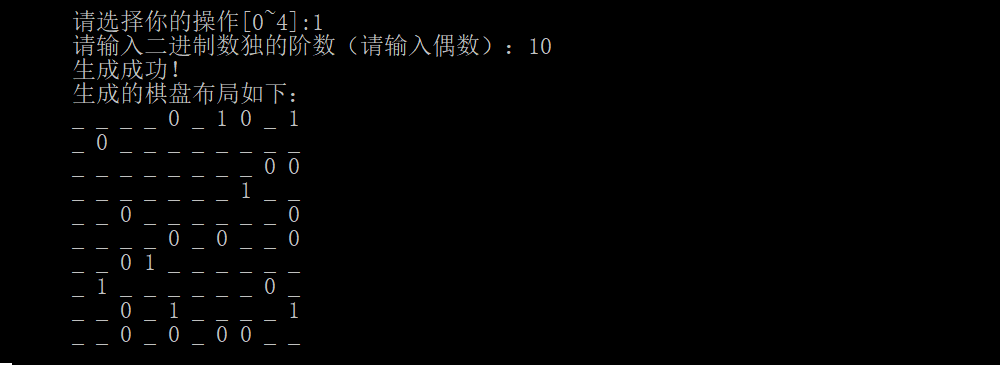


图4-6-3 二进制数独10阶生成结果图

5. 二进制数独求解模块

本模块的主要功能是求解二进制数独棋盘格局。根据测试需求，本次测试将对三种方式读入/生成的二进制数独格局分别进行测试，其中各个方式的具体测试的主要内容如下：

①生成一个8阶二进制数独并求解；

②从键盘输入二进制数独棋盘格局并求解；

③从文件读入二进制数独棋盘格局并求解；

表4-6 二进制数独求解模块测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试内容 | 测试用例 | 理论输出 | 实际输出 |
| ① | 程序自主生成一个8阶的  二进制数独格局并求解 | 求解出的格局 | （数独生成及求解结果图，如图4-7-1所示） |
| ② | 从键盘输入一个14阶的  二进制数独格局并求解 | 求解出的格局 | （结果图，如图4-7-2及图4-7-4所示） |
| ③ | 从文件读入一个14阶的  二进制数独格局并求解 | 求解出的格局 | （结果图，如图4-7-3及图4-7-4所示） |

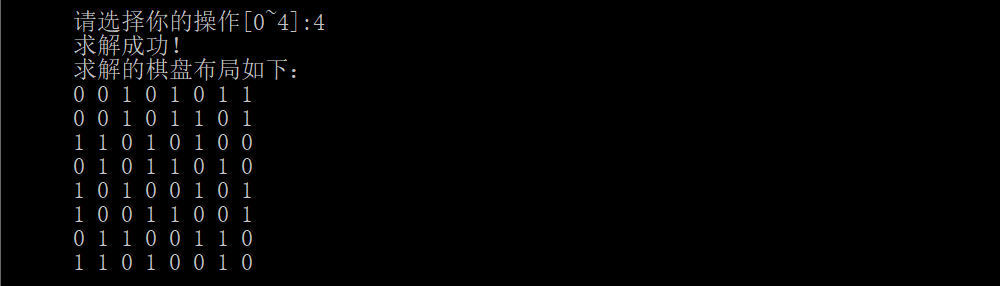
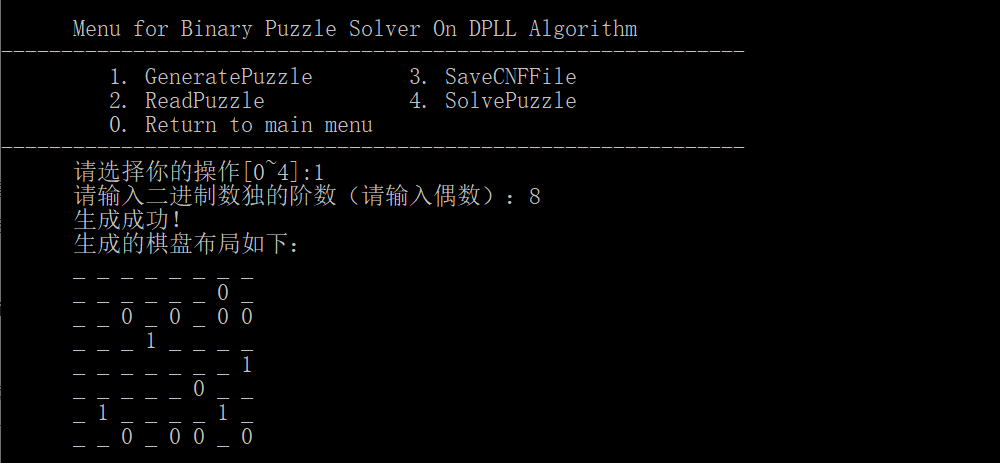


图4-7-1 二进制数独8阶生成及求解结果图

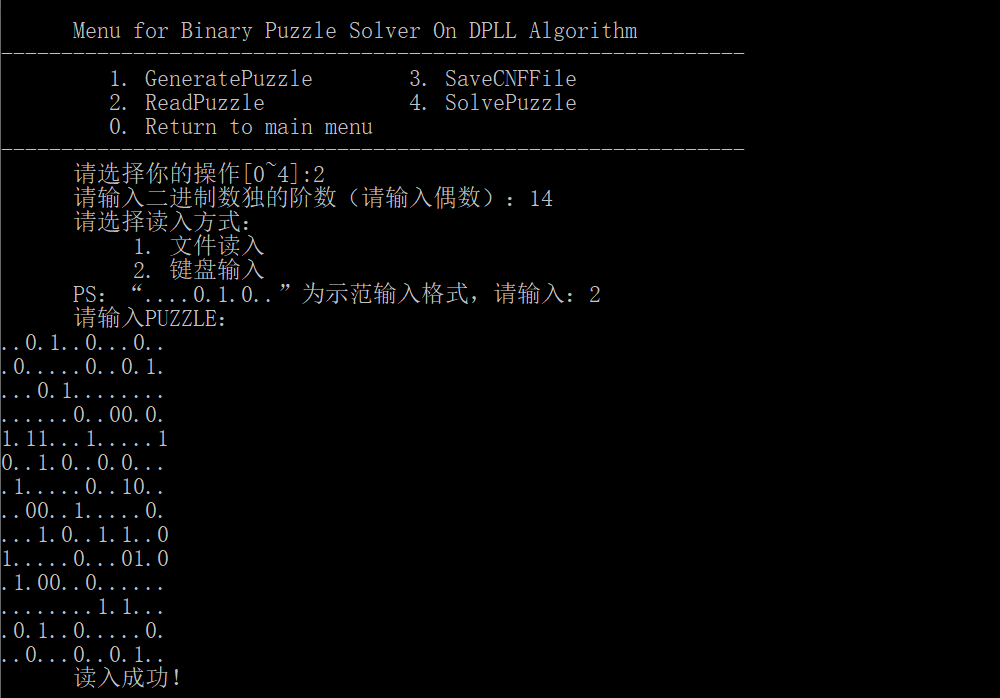


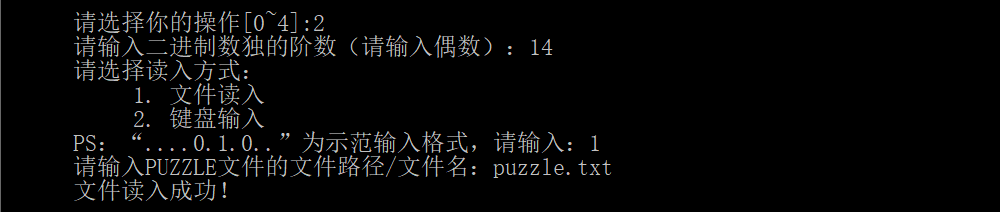
图4-7-2 二进制数独14阶键盘输入结果图

图4-7-3 二进制数独14阶文件读入结果图

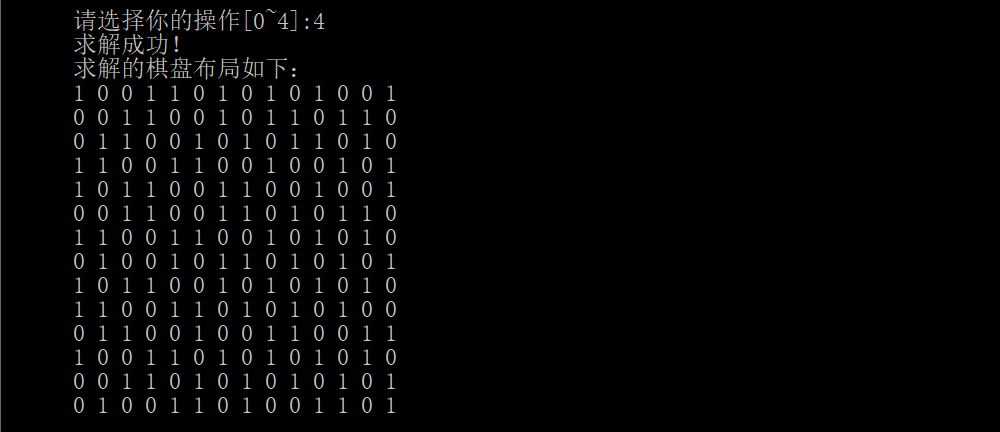


图4-7-3 二进制数独14阶求解结果图

# 5 总结与展望

## 5.1 全文总结

在本次课程设计实验中，圆满地完成了基于DPLL算法的SAT求解器设计，并实现了二进制数独游戏的求解系统，实现了较好的交互功能，主要工作如下：

（1）阅读课程设计任务书，并查阅了相关的资料，了解和学习SAT问题的相关知识，掌握了DPLL算法的算法思想与具体选词策略。同时，学习了数独问题转换为SAT问题的方法，为接下来的课程设计实验部分的开展奠定了坚实的理论基础。

（2）设计并实现主控、交互与显示功能，使得系统有一个较为清晰和美观的交互界面，完成较好的交互功能。

（3）设计并实现了对CNF文件和CNF公式的处理功能，实现对CNF文件的读取、对CNF公式的遍历输出与文件保存功能，并定义了一系列基础操作，以供调用。

（4）基于DPLL算法框架，设计并实现了较为高效的SAT求解器，实现正确求解CNF算例并按照规定的格式输出结果的功能。同时，通过选择尝试不同的选词策略，选择一个较为高效的选词策略对算法进行优化，大幅提高了对CNF算例求解速度（尤其是满足CNF算例）。

（5）基于设计并实现完成的SAT求解器完成对二进制数独问题的求解功能，并通过引入附加变元的方式实现对二进制数独问题的归化，通过挖洞法创建一个具有唯一解的二进制数独棋盘初始格局，并实现了简单的交互性。

## 5.1 工作展望

在本次实验中实现了一个较高效的SAT求解器，但是相较于实际需求，本次设计和实现的SAT求解器仍然效率过低，有许多地方有待改进，下面列出了几个将来的工作重点和研究方向：

（1）学习并了解对CNF公式预处理机制，对CNF公式进行预处理或化简，并通过引入部分非完备算法的方式来加快求解速度。

（2）学习并了解其他特殊的数据结构和配套的求解算法，通过引入更精妙的数据结构来提高文字和子句信息的处理速度，以提高算法效率。

（3）学习并了解基于冲突学习的子句学习和非时序回溯机制，通过子句学习减少无效搜索分支的搜索次数，并利用非时序回溯的方式来大幅加快算法的回溯速度。

# 6 体会

通过不懈的努力，本次课程设计实验任务顺利完成，也较为圆满地实现并测试了相关功能。在本次实验中，我遇到了许多前所未见的问题，在解决这些问题的过程中，我学到了很多东西，也收获了许多经验和教训，并从中看出了我现有的不足之处，以下我将对我在课程设计中遇到的问题和收获做出简述和总结。

1. 遇到的问题

（1）资料查询问题

由于本次课程设计实验是要设计基于DPLL算法这种我完全没有了解的算法实现的SAT求解器，对于SAT问题包括解决方法的不了解，造成了我在看到课程设计任务书时有些不知所措；再加上查阅资料的能力不足，导致我前期的进展较慢，不过随着我查询资料的熟练度水平逐渐提高，这个问题得到了很大的缓解。**在查阅资料的过程中，我明白了我之前按照课本学习的局限性，我明白了查阅资料能力的重要性，这也将是我之后一段时间的努力方向。**

（2）DPLL算法学习问题

本次实验采用的DPLL算法对于我来说是全新的知识，通过查阅资料和阅读不同的代码，我才逐渐了解了DPLL算法的可行的实现流程，在期间我走过非常多的弯路。**同时，对于DPLL算法的优化，我也花费了大量精力尝试不同的方式**，从UP策略到MOM策略等等，最终，通过比对运行时间，我选择了JW策略来进行选词，该策略既考虑了子句长度对单子句规则处理的影响也考虑了文字频度的影响，是我测试的策略中效果最好的一种。

（3）二进制数独的归化问题

本次实验不仅要设计一个有效地SAT求解器，同时要实现二进制数独的求解和生成问题，在我考虑二进制数独的求解问题时，最大的问题就是二进制数独的归化问题，由于要实现对不同阶数的二进制数独的附加变元的转化，附加变元的转化函数着实让我头疼，三种约束规则的子句转化问题也让我有些郁闷，不过通过不懈努力，最终还是较完满地实现了功能。

（4）二进制数独的生成问题

这个模块是我第二头疼的模块，由于要随机生成且要有唯一解，我采取了随机函数填空生成完整棋盘+挖洞法+求反求解判断的方式判断唯一性的方式生成有唯一解的棋盘。由于在判断唯一解的实现方向上存在一段时间的错误，导致我在这个模块花费了非常多的时间实现功能，在同学的提点下我才找到了正确的方向，在此我要感谢我优秀的室友们，感谢他们对我的帮助。

2. 取得的收获

（1）知识体系的完善

本次课程设计实验让我接触了全新的算法领域，填补了我知识体系的一个漏洞，这也是我在这次课程设计实验中最直接和最明显的收获。

（2）代码能力的巩固

由于上个学期较为繁忙的学业，导致我编写代码的时间略有不足。通过这次课程设计，我巩固了之前所学，不仅对C语言的理解更加深刻，对于数据结构的理解也上了一个台阶，这是我在本次实验中比较隐性但重要的收获。

（3）查阅资料能力的提高

由于长期按部就班地跟着老师的节奏学习，习惯于啃书本的学习方式，导致我对本次课程设计实验的不适应，这是我所遇见的困难也是我之后应该注意的重点，本次实验凸显了查阅资料以及整理信息的能力的重要性。通过本次实验的学习，我查阅资料的能力水平已经有了较大的提高，这是一个巨大的改变，也是自主学习能力的巨大进步，在之后的学习中，我也将继续加强这方面能力的培养。

# 参考文献

[1]□王静康,张凤宝,夏淑倩等.论化工本科专业国际认证与国内认证的“实质性”.高等工程教育研究,2014,5:1-4

[2]□Stone J A, Howard L P. A simple technique for observing periodic nonlinearities in Michelson interferometers. Precision Engineering,1998,22(4):220-232

[3]□朱印红,袁衍明.Dreamweaver完美网页设计——技术入门篇.(第一版).北京:中国电力出版社,2006:19～20

[4]□Lewis S L. Physics and chemistry of the solar system.北京:北京大学出版社,2014.1～2

[5]□陈剑.上博简《民之父母》“而得既塞於四海矣”句解释[EB/OL］.简帛研究网站，http://www.bamboosilk.org/Wssf/2003/chenjian03.htm．2003-01-18

……

# 附录：程序使用说明

1. 求解SAT算例功能步骤说明

（1）进入主菜单界面后，按照提示输入“1”进入求解SAT算例功能选项。

（2）进入次级菜单界面，首先使用输入“1”，使用ReadFile功能读入CNF算例文件，输入正确的文件路径，等待返回打开成功提示信息。

（3）根据需要选择输入“2”（TraverseCNF，即遍历输出CNF文件内容）或“3”（NotImproveDPLL，即未改进DPLL求解器）或“4”（ImproveDPLL，即改进后的DPLL求解器），使用三个功能进行操作。

（4）输入“0”可以返回主菜单。

2. 生成并求解二进制数独棋盘功能步骤说明

（1）进入主菜单界面后，按照提示输入“2”进入求解二进制数独功能选项。

（2）进入次级菜单界面，首先使用输入“1”或“2”，生成或读入合法的二进制数独棋盘格局。

（3）根据需要选择输入“3”（SaveCNFFile，即保存CNF公式）或“4”（SolvePuzzle，即求解二进制数独棋盘），进行操作。

（4）输入“0”可以返回主菜单。