软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(2):395−405 [doi: 10.13328/j.cnki.jos.005642]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于格局检测的模型计数方法*

贺甫霖[1,2], 刘 磊[1], 吕 帅[1,2], 牛当当[1,2,3], 王 强[1,2]

[1](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[2](符号计算与知识工程教育部重点实验室(吉林大学),吉林 长春 130012)

[3](西北农林科技大学 信息工程学院,陕西 杨凌 712100)

通讯作者: 吕帅, E-mail:lus@jlu.edu.cn

摘 要: 模型计数是指求出给定命题公式的模型数,是 SAT 问题的泛化.模型计数在人工智能领域取得了广泛应用,很多现实问题都可以规约为模型计数进行求解.目前,常用的模型计数求解器主要有 Cachet 与 sharpSAT,它们均采用完备方法且具有高效的求解能力,但其求解效率对模型数不敏感.有理由猜测:当给定问题的模型较少时,不完备算法可能发挥其效率优势而更适合模型计数.局部搜索是求解 SAT 问题的高效不完备方法,Cai 等人提出了格局检测策略,并将其应用到局部搜索方法中,提出了 SWcc 算法,具有很高的求解效率.对 SWcc 算法进行扩充,分别得到了迭代法与优化后的增量法两种效率较高的不完备模型计数方法,给出了两种方法的思路和具体实现.最后给出了大量测试样例的实验结果,以验证当给定合取范式的模型较少时,该迭代法与优化后的增量法的求解效率有所提升.

关键词: 模型计数;局部搜索;格局检测

中图法分类号: TP18

中文引用格式: 贺甫霖,刘磊,吕帅,牛当当,王强.基于格局检测的模型计数方法.软件学报,2020,31(2):395−405. http://www.jos.org.cn/1000-9825/5642.htm

英文引用格式: He FL, Liu L, Lü S, Niu DD, Wang Q. Model counting methods based on configuration checking. Ruan Jian Xue Bao/Journal of Software, 2020,31(2):395−405 (in Chinese). http://www.jos.org.cn/1000-9825/5642.htm

Model Counting Methods Based on Configuration Checking

HE Fu-Lin[1,2], LIU Lei[1], LÜ Shuai[1,2], NIU Dang-Dang[1,2,3], WANG Qiang[1,2]

[1](College of Computer Science and Technology, Jilin University, Changchun 130012, China)

[2](Key Laboratory of Symbolic Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun 130012, China)

[3](College of Information Engineering, Northwest A&F University, Yangling 712100, China)

Abstract: Model counting is the problem of calculating the number of the models of a given propositional formula, and it is a generalization of the SAT problem. Model counting is widely used in the field of artificial intelligence, and many practical problems can be reduced to model counting. At present, there are two complete solvers which are commonly used for model counting, i.e. Cachet and sharpSAT, both of which have high solving efficiency. But their solving efficiency does not relate to the numbers of the models. It is reasonable to suppose that incomplete methods are more likely to take their advantage in efficiency and maybe they could be more suitable to solve model counting problems when the number of the models of given propositional formula is little. Local search is an efficient incomplete method for solving SAT problem. A new strategy called configuration checking (CC) has been proposed by Cai *et al*. and it is adopted to the local search. In this way, the SWcc algorithm has been proposed with high solving efficiency. This study puts forward two incomplete methods on basis of the SWcc algorithm, i.e. the iteration method and the improved incremental method, both of

* 基金项目: 国家自然科学基金(61300049, 61763003); 吉林省自然科学基金(20180101053JC)

Foundation item: National Natural Science Foundation of China (61300049, 61763003); Natural Science Research Foundation of Jilin Province (20180101053JC)

收稿时间: 2017-11-06; 修改时间: 2018-03-20, 2018-05-29; 采用时间: 2018-08-29

© 中国科学院软件研究所 http://www.jos.org.cn

which have high efficiency. Then, the specific implementation process of the two methods is presented. At last, the experimental results of a large amount of benchmarks, according to which is found, show the advantages of the iteration method and the improved incremental method in terms of time, when the amount of the models of given conjunctive normal formula is small.

Key words: model counting; local search; configuration checking

命题可满足性问题(SAT)是人工智能领域中的重要问题.近年来,国内外学者不断致力于提出新的 SAT 求解方法,追求更高的求解能力[1−4].局部搜索方法是 SAT 问题高效的求解方法.该方法根据特定的启发式规则,将一个随机指派通过不断的变量翻转操作,使其成为一个可满足的真值指派.近年来,由 Cai(蔡少伟)等人提出的格局检测策略开始在局部搜索领域显露头角,该策略是目前华人在国际 SAT 比赛中提出的最具影响力的策略之一.2011 年,Cai 等人首次将格局检测策略用于 SAT 求解中的局部搜索方法,设计的 SWcc 算法[5]取得了骄人的成绩.不仅如此,格局检测策略在求解最小顶点覆盖问题时可以很好地处理循环问题,以此提高求解效率[6].2012 年,Cai 等人基于格局检测策略提出了一种新的启发式策略,设计了 SWcca 算法[7].该算法在求解随机实例与工业化实例时均能得到较高的求解效率.上述两种算法在求解 3-SAT 实例上效果较好,为了在求解 *k*-SAT 实例时同样能够得到较高的求解效率,Cai 等人在文献[3]中提出了 DCCA 启发式策略.该策略将两种对格局有不同定义的格局检测策略结合在一起.2015 年,Cai 等人基于格局检测策略设计了算法 CCAnr[8].该算法在求解非随机实例时具有很高的求解效率.

可满足性问题是 NP 完全问题.然而,约束问题仅判定可满足性是远远不够的,例如,将概率推理问题[9]、一致性概率规划问题[10]转换为命题公式集合后,还需要计算出该公式集合中所有模型(可满足指派)的个数.#SAT 问题即模型计数问题,需要计算出给定命题公式集合的模型个数,其计算复杂度高于 SAT 问题,是#P 完备的.

模型计数问题的求解方法依据求解准则的不同可以划分为不同的类别:依据求解的精确程度可以分为精确求解与近似求解,依据求解方式的不同可以分为直接求解与基于编译的求解方法.精确算法最终求解得到的模型数是精确的.Sang 等人在 SAT 求解器 ZChaff[11,12]的基础上,利用组件缓冲策略和子句学习策略,提出了 Cachet 求解器[13,14],通过记录已经出现的冲突和子公式的模型个数来避免重复计算.在文献[15]中,Sang 等人在 Cachet 的基础上设计了 Cachet-WMC 系统.该系统可以处理加权模型计算问题.Cachet 求解器是目前应用范围最广泛的#SAT 求解器之一.Thurley 提出了 sharpSAT 求解器[16].它采用了一种新的编码方式,从而极大地降低了子句的存储规模;同时,它还采用了 BCP 策略和组件缓冲策略,求解效率比 Cachet 求解器有进一步的提升.殷明浩等人设计了一种基于扩展规则的#SAT 求解算法 CER[17].该算法可以找出哪些指派不满足命题公式,并记录下总数,然后用极大项空间的大小减去记录下的总数,即为模型数.实验结果表明,当互补因子较高时,基于扩展规则的方法一般要优于基于归结的方法.赖永等人设计了#SAT 求解算法#ER[18],并结合#DPLL[19]和#ER 的求解能力提出了混合#SAT 求解算法#CDE.该算法结合前两种算法的优点,表现出了比#ER 更高的求解效率,在互补因子较低的子句也有很好的表现.2015 年,贾凤雨等人提出了增量求解算法 RCER[20].该算法优先计算扩展极大项相交集对应的极大项个数,并重用此计算结果.2016 年,他们又提出结合互补度求解算法 CDCER[21].这两种算法都减少了 CER 算法的冗余计算,在互补因子较低的测试用例中,求解效率比 CER 提升显著.2017 年,王强等人发现选择规约子句的顺序对极大项空间的大小有着较大影响,提出了两种加速#SAT 求解的启发式策略——MW 和 LC&MW,分别设计了算法 CER_MW 和 CER_LC&MW,在求解效率和求解能力上都有显著提高[22].

近似算法先精确求解某个子空间的模型数,通过子空间占整体解空间的比例大小和子空间的精确模型数来估算整体解空间的近似模型数,是一类不完备的算法,通常用来求解规模较大的问题.Gomes 等人提出的 SampleCount[23]算法是一种近似算法.该算法可以给出给定合取范式的模型数的下界.实验结果表明,其计算得出的下界的精确度很高.该算法首先会用采样的方法选中一个变量集合,后续的计算不考虑选中的变量,用精确的求解器 Cachet 求解剩余合取范式模型个数,依据此模型个数和选中的变量个数可近似计算出原合取范式的模型总数.王金艳等人利用扩展规则提出了两种近似求解方法 ULBApprox 和 SampleApprox[24].ULBApprox 方法首先计算出不满足指派的近似个数,依据此数目推算出近似模型总数.SampleApprox 算法基于 SampleSat[25]算法,利用扩展规则可以得到精确度较高的模型数.

© 中国科学院软件研究所 http://www.jos.org.cn

上述介绍的方法均为直接求解方法,在求解过程中,原合取范式的形式并未发生改变.而编译求解的思想主要是利用知识编译的方法将原子句集进行编译,在目标语言上再使用模型计数算法进行求解,经编译得到的目标语言往往更易求解,且可以确保计算得到的模型总数不变.Darwiche 等人提出了一种自顶向下的#SAT 求解器 C2D[26,27].C2D 将给定的合取范式编译成 d-DNNF(deterministic decomposable negation normal form)形式.处理给定的合取范式生成一棵分解树.该分解树引导 DNNF 形式中决策节点的生成.对于部分实例,d-DNNF 是比 OBDD(ordered binary decision diagram)更简洁的目标语言,降低了求解的时间复杂度.Muise 等人提出了求解器 Dsharp[28].它同样是一个自顶向下的生成 d-DNNF 目标语言的求解器.但不同于 C2D,Dsharp 求解器通过一种动态的分解方法控制决策节点的生成.该方法可对不相交的组件进行分析处理.Dsharp 求解器采用了 BCP(binary constraint propagation)策略和预处理技术,其求解效率优于 C2D 求解器.上述基于编译的求解方法所求解出的模型数都是精确的.

精确的直接求解器 Cachet 与 sharpSAT 的求解能力很强,即便当给定的合取范式的模型较多时,也可以很快地求解出其精确的模型个数.但当模型较少时,采用局部搜索方法会得到更高的求解效率.本文在 Cai 等人的 SWcc 算法[5]上进行扩充,充分利用格局检测的优越性,得到两种不完备的直接求解方法:迭代法与增量法,将满足给定合取范式的完全指派依次求解出,实现模型计数.

本文第 1 节给出基础知识与 Cai 等人的 SWcc 算法[5]的相关定义.第 2 节给出迭代法的实现方法.第 3 节给出增量法的实现方法.第 4 节给出增量法的优化方法.第 5 节给出实验结果及分析,验证当给定的合取范式的模型较少时,采用本文的两种方法进行求解效率更高.

1 基础知识

1.1 相关定义

对于在变量集合$\{x_1,x_2,\ldots,x_n\}$上的合取范式 F,子句 C_i 是指一些文字的析取式,每个文字可以是一个变量的正文字形式 x_k 或者是其负文字形式$\neg x_k$.一个文字 l 出现在子句中,是指这个子句中包含 l;而一个变量 x 出现在子句中,是指这个子句中包含 x 或者$\neg x$.

用 $V(c)$表示在子句 c 中出现过的所有变量,$V(F)$表示在合取范式 F 中出现过的所有变量.两个变量互为邻居变量当且仅当它们至少在 1 个子句中同时出现过.$N(x)=\{y|y\in V(F)$并且 y 和 x 至少在同一个子句中出现过$\}$是变量 x 的所有邻居变量的集合.

1.2 格局检测策略及子句的加权方式

在 Cai 等人的局部搜索算法[5]中,提出了格局检测策略.Cai 等人定义了一个数组 *confChange*,数组的值表示对应变量的环境情况:*confChange*[x]=1 表示变量 x 的环境自从它最后一次翻转后发生了变化,*confChange*[x]=0 则表示相反的含义.用如下两条规则来修改和维持 *confChange* 数组.

规则 1. 在初始情况下,对于每一个变量 x,*confChange*[x]的值都被初始化为 1.

规则 2. 当选择变量 x 进行翻转操作时,*confChange*[x]的值被重置为 0,并且对于每一个变量 $y\in N(x)$,*confChange*[y]都被赋值为 1.

SWcc 算法采用平滑子句的加权方式.当陷入局部最优情况时,所有不满足的子句的权值都会加 1.此外,SWcc 算法运用一种平滑机制来定期平滑子句的权值:当所有子句的平均权值 $\overline{w}$ 超过了一个特定的阈值γ时,所有子句的权值都会按照公式(1)进行平滑.

$$w(C_i)=\rho\times w(C_i)+(1-\rho)\times \overline{w} \quad (0<\rho<1) \tag{1}$$

Cai 等人把一个合取范式 F 和一个加权函数 w 相关联[5].对于在范式 F 中出现的每个子句 c,都会有一个正整数 $w(c)$与其相关联并作为其权值.合取范式 F 在任意一个随机指派 s 下的花销的计算规则如公式(2)所示.

$$cost(F,s)=\sum_{c\in F,s使c不满足} w(c) \tag{2}$$

该公式表明了在随机指派 s 下,所有不满足当前完全指派的子句的权值总和.

© 中国科学院软件研究所 http://www.jos.org.cn

对于任意一个在范式 F 中出现的变量 x,

$$dscore(x)=cost(F,s)-cost(F,s') \tag{3}$$

该公式计算了翻转变量 x 之后的收益情况,其中,s'是由 s 翻转变量 x 的真值后所得到完全指派.

2 迭代法

SWcc 算法可以随机地生成一个指派,并通过翻转操作使其可满足.如何在得到一个可满足指派后,在接下来的求解过程中能够找到不同的指派,是模型计数的关键.迭代法利用子句学习策略来解决该问题.

子句学习的思路如下:在求得一个可满足解后,依据求出的真值指派添加一个子句,该子句所含文字为该真值指派的否定,能够使得刚刚求出的可满足解不满足.在下一次的求解过程中,这个指派可能会重复出现,但是不会再进行累加计数.为此,我们设计了 *add_a_clause*(·)函数以完成该功能.SWcc 迭代法描述了局部搜索思想在求解模型计数问题时的执行框架,如算法 1 所示.

算法 1. SWcc 迭代法.

```
1  Input:合取范式 F,maxSteps.
2  Output:满足合取范式 F 的全部真值指派的个数.
3  begin
4    while (true)
5      S←随机产生的一个真值指派;
6      将所有的子句权值初始化为 1,并计算每个变量的 descore(x);
7      对于每个变量 x,confChange[x]初始化为 1;
8      对于每个变量 x,timeStamp[x]初始化为 0;
9      for step←1 to maxSteps do
10       if s 满足 F then break;
11       if G={x|dscore(x)>0 and confChange[x]=1}≠∅ then v←x0;
12         (x0∈G,并且∀x∈G,dscore(x0)≥dscore(x))
13       else
14         for each clause Ci in unsat_stack
15           w(Ci)++;
16         if  w̄ >γ then
17           for each clause Ci
18             w(Ci)=ρ×w(Ci)+(1−ρ)× w̄ ;
19         c←随机选择一个不满足的子句 c;
20         v←子句 c 中具有最大时间戳的变量;
21         翻转变量 v,根据规则 2 更新 confChange 数组;
22         timestamp[v]←step;
23     if s 满足 F then
24       num_of_solutions++;
25       add_a_clause(·);
26     else
27       return “solution not found”;
28       break;
29 end
```

算法 1 第 4 行~第 28 行均置于 while 死循环中,算法在达到最大翻转步骤仍未找到可满足指派时退出.第 5

© 中国科学院软件研究所 http://www.jos.org.cn

行~第 8 行为每次迭代生成一个可满足指派后的变量初始化.第 9 行~第 22 行为 for 循环的作用域,为局部搜索的核心,在集合 G 中选取格局检测策略允许翻转的 *dscore* 值最大的变量进行翻转操作.若集合 G 为空,则更新子句权值.子句权值的变换规则是:首先,将所有不满足的子句权值递增(第 14 行、第 15 行);当子句的平均权值超过预先设定的阈值时,利用公式(1)对所有的子句权值进行平滑处理(第 16 行~第 18 行);当找到一个可满足指派或者达到最大翻转步数时,算法跳出 for 循环;如果成功找到一个可满足指派,则将模型个数累加,并将该指派的否定加入原合取范式中,以保证下次搜索到该指派不再满足,程序返回第 5 行进行下一次迭代过程;如果经过 for 循环仍未找到可满足指派,那么程序退出.

3 增量法

由于迭代法在每次迭代的过程中都会返回到程序初始化部分,会不断重复计算每个变量 *dscore* 值等过程,每次迭代的计算成本较大.增量法是另一种可行的解决模型计数问题的方法.增量法同样需要采用增添子句的方法,但增量法与迭代法的不同之处在于:增量法没有返回到程序最初始的位置,该方法可以跳过初始化过程.

增量法能够跳过初始化阶段的原因是:每当新加入一个子句后,程序都需要计算新加入子句的邻居信息,并根据此信息更新集合 G,从而避免了重新生成一个随机指派,并根据这个指派重新计算每个变量 *dscore* 值等操作.为实现该功能我们设计了 *update*(·)函数,该函数在迭代的过程中可以代替原程序中的 *init*(·)函数.基于格局检测的 SWcc 增量法的推理框架描述如算法 2 所示.

算法 2. SWcc 增量法.

```
1  Input:合取范式 F,maxSteps.
2  Output:满足合取范式 F 的全部真值指派的个数.
3  begin
4    S←随机产生的一个真值指派;
5    将所有的子句权值初始化为 1,并计算每个变量的 dscore(x);
6    对于每个变量 x,confChange[x]初始化为 1;
7    对于每个变量 x,timeStamp[x]初始化为 0;
8    while (true)
9      for step←1 to maxSteps do
10       if s 满足 F then break;
11       if G={x|dscore(x)>0 and confChange[x]=1}≠∅ then v←x0;
12         (x0∈G,并且∀x∈G,dscore(x0)≥dscore(x))
13       else
14         for each clause Ci in unsat_stack
15           w(Ci)++;
16         if  w̄ >γ then
17           for each clause Ci
18             w(Ci)=ρ×w(Ci)+(1−ρ)× w̄ ;
19         c←随机选择一个不满足的子句 c;
20         v←子句 c 中具有最大时间戳的变量;
21         翻转变量 v,根据规则二更新 confChange 数组;
22         timestamp[v]←step;
23     if s 满足 F then
24       num_of_solutions++;
25       update(·);
```

© 中国科学院软件研究所 http://www.jos.org.cn

```
26          add_a_clause(·);
27      else
28          return "solution not found";
29          break;
30 end
```

增量法与迭代法的主要区别在于新加入了函数 *update*(·),加入该函数后,可将第 4 行~第 7 行的一系列变量初始化过程置于 while 循环之外,从而在每次迭代过程中减少计算量.*update*(·)函数的主要功能为将所有变量之间建立起邻居变量的关系,将新产生的不满足子句压入存放不满足子句的保留栈,将该子句对应的 *sat_count* 值赋值为 0,同时将所有变量的 *dscore* 值均增加 1.完成上述操作后,重新更新集合 *G*.增量法的其余求解过程与迭代法相同,不再赘述.

4 增量法的优化

上文给出了增量法的具体实现过程.增量法的优点是可以避免重复产生随机解等初始化过程.但是在第 1 次添加新子句后,所有变量之间就全部建立起邻居关系,破坏了格局检测的优越性.实验结果表明,增量法的求解效率较低.本节提出一种优化策略提升其求解性能.

该优化策略的思想是:依次遍历合取范式中的所有子句,如果所遍历的子句的变量集是准备加入子句的变量集的子集(即准备加入的子句包含了遍历到的子句中的所有变量),且所遍历的子句中有且仅有 1 个文字与准备加入子句的文字互补,则在新加入子句时删除该互补文字.

按照上述策略执行,遍历所有子句后,有极大的可能在加入的子句中减少变量个数,从而保证了格局检测的优越性.该优化策略如算法 3 所示.

算法 3. 解的优化策略.

```
1  Input:一个可满足的完全指派.
2  Output:一个被标记的完全指派(被标记的文字将不被添加到原合取范式中).
3  begin
4      for (w=1; w≤num_vars; w++)   //数组 flag[·]标记变量是否被删除,0 代表存在,1 代表已删除
5          flag[w]←0;
6      for (c=0; c<num_clauses; c++)
7          first←clauses[c];
8          while (first≠NULL) do   //依次遍历子句中的每个元素
9              if (flag[first→var_num]==1) then   //遇到已删除的元素
10                 fflag←1;   //fflag 元素为 1,表示当前所遍历子句的变量集不是准备加入子句变量集的子集
11                 break;
12             if (first→sense==best_soln[first→var_num]) then   //遇到一个文字与准备加入子句的文字互补
13                 notsame←first→var_num;   //记录下互补文字的标号,以待删除
14                 notnum++;   //互补文字个数加 1
15             if (notnum>1) then   //如果互补文字的个数大于 1
16                 break;
17             first←first→next_in_clause;
18     if ((fflag==0) && (notnum==1))   //如果子集条件与互补文字个数为 1 的条件同时满足
19         flag[notsame]←1;   //将待删除的文字 flag 标志为 1,正式删除
20 end
```

该优化算法运用数组 *flag*[·]来标记变量是否被删除:0 代表存在,1 代表已删除.初始情况下,将所有变量对应

© 中国科学院软件研究所 http://www.jos.org.cn

的 *flag* 值设为 0.依次遍历范式 F 中的所有子句,如果遇到了一个 *flag* 值为 1 的变量,则代表该变量在即将被添加的子句中已删除.此时已不满足子集条件(当前所遍历子句的变量集是准备加入子句变量集的子集),跳出当前循环遍历下一个子句.将 *fflag* 值设置为 1,代表子集条件不满足.接下来判断该变量在遍历到的子句中与要添加的子句是否含有互补文字:如果互补,则记录下变量编号,并将互补文字个数加 1.如果互补文字个数超过 1 个,则同样不满足互补文字个数为 1 的条件,跳出当前循环,遍历下一个子句.只有当两个条件同时满足时,才可将先前记录下的互补文字从新加的子句中删除.

按照上述过程,添加子句时,*flag*[·]值为 1 的元素跳过将不会添加到新子句当中,从而达到简化目的,充分保留了格局检测的优越性.

5 实验结果与分析

本节对上述提出的迭代法与优化后的增量法进行了实现,并将其与#SAT 推理领域主流的求解器(Cachet, sharpSAT)进行了对比,实验结果凸显出了基于局部搜索的模型计数方法推理框架的优势.

实验环境是:操作系统为 Linux(Ubuntu 12.04.5),CPU 为 Intel Core i7 3.60GHz,内存为 8GB.

对比实验主要记录下迭代法、优化后的增量法与两种完备求解器 Cachet 和 sharpSAT 求解同一实例所需的时间以进行性能对比.本文采取了多种测试用例对算法的性能进行了测试,对于每一问题测试用例,均将算法执行 50 次,最终取平均值作为结果.值得注意的是,本文提出的两种方法是不完备的,但对于在表 1~表 7 中的全部实例,迭代法与增量法求解出的模型数均是精确的.也就是说,实验部分是在求出给定问题的全部模型的前提下,与完备方法进行求解时间的比对.

1) 采用变量数为 20、子句数为 91 的小规模随机测试用例对迭代法、优化后的增量法、Cachet、sharpSAT 进行了对比实验,实验结果见表 1.

Table 1 Experimental results on random instances uf20 (s)

表 1 uf20 随机实例的实验结果 (s)

实例	模型数	优化后的增量法	迭代法	Cachet	sharpSAT
uf20-01	8	0.012 917	0.008 737	0.020 803	0.018 762
uf20-02	29	0.028 090	0.019 211	0.020 987	0.016 845
uf20-03	1	0.003 651	0.004 871	0.018 607	0.015 833
uf20-04	3	0.004 230	0.004 311	0.019 512	0.017 425
uf20-05	2	0.003 627	0.004 282	0.024 746	0.017 973
uf20-06	4	0.005 620	0.005 877	0.020 197	0.016 069
uf20-07	23	0.021 206	0.014 487	0.019 808	0.016 319
uf20-08	4	0.004 453	0.004 492	0.020 292	0.017 574
uf20-09	1	0.003 240	0.003 565	0.018 421	0.016 253
uf20-010	9	0.009 220	0.007 811	0.020 403	0.016 912

实验结果表明,

(1) 当模型数小于 5 时,优化后的增量法与迭代法较两种完备算法求解性能提升明显.对于优化后的增量法,Cachet 求解时间是其 3.5 倍~6.8 倍,sharpSAT 求解时间是其 2.8 倍~5 倍.对于迭代法,Cachet 求解时间是其 3.8 倍~5.7 倍,sharpSAT 求解时间是其 2.7 倍~4.5 倍.随着实例的模型数增加,优化后的增量法与迭代法优势减弱.

(2) 当模型数接近 10 时,对于优化后的增量法,Cachet 求解时间是其 1.6 倍,sharpSAT 求解时间是其 1.4 倍.对于迭代法,Cachet 求解时间均是其 2.3 倍,sharpSAT 求解时间均是其 2.1 倍.

(3) 当模型数超过 20 时,优化后的增量法较两种完备算法已不再具有优势,而迭代法的求解性能与两种完备算法十分接近.

2) 将实例的规模扩大,选取变量数为 50、子句数为 218,变量数为 75、子句数为 325,变量数为 100、子句数为 430 的随机实例进行对比实验,实验结果见表 2、表 3.

© 中国科学院软件研究所 http://www.jos.org.cn

Table 2 Experimental results on random instances uf50 (s)

表 2 uf50 随机实例的实验结果 (s)

实例	模型数	优化后的增量法	迭代法	Cachet	sharpSAT
uf50-01	24	0.073 095	0.045 042	0.022 632	0.021 866
uf50-02	6	0.013 061	0.021 857	0.024 639	0.021 575
uf50-03	1 362	15.721 511	18.745 151	0.024 036	0.018 666
uf50-04	8	0.014 703	0.012 608	0.024 277	0.018 219
uf50-06	4	0.005 137	0.006 692	0.020 334	0.009 589
uf50-07	140	0.213 027	0.203 158	0.019 309	0.017 807
uf50-08	2	0.003 811	0.007 985	0.021 190	0.018 392
uf50-09	156	0.277 022	0.303 012	0.021 655	0.019 773
uf50-010	156	0.268 106	0.290 537	0.023 974	0.019 558

Table 3 Experimental results on random instances uf75 and uf100 (s)

表 3 uf75 和 uf100 随机实例的实验结果 (s)

实例	模型数	优化后的增量法	迭代法	Cachet	sharpSAT
uf75-03	3	0.007 319	0.015 053	0.029 975	0.024 232
uf75-091	8	0.018 245	0.033 206	0.029 756	0.021 429
uf100-0276	8	0.029 843	0.063 782	0.081 073	0.048 389
uf100-0285	4	0.007 320	0.020 812	0.049 144	0.033 407
uf100-0326	8	0.019 339	0.075 778	0.065 657	0.040 628
uf100-0441	4	0.015 449	0.033 549	0.068 440	0.036 576
uf100-0602	6	0.016 279	0.035 804	0.062 002	0.038 896
uf100-0678	4	0.020 040	0.036 020	0.067 332	0.039 350
uf100-0808	6	0.016 951	0.037 470	0.063 907	0.039 531
uf100-0953	2	0.009 813	0.012 832	0.063 706	0.036 673

从表 2 和表 3 的实验数据可以看出,对于优化后的增量法,可以得到与表 1 结果相似的结论,但是迭代法的求解效率有所下降.迭代法在求解模型数为 2 的实例 uf100-0953 时效率较好,Cachet 求解时间是其 4.9 倍,sharpSAT 求解时间是其 2.8 倍.当求解其他模型数不大于 6 的实例时,Cachet 求解时间是其 2 倍左右,sharpSAT 求解时间与其十分接近.当模型数达到 100 以上时,本文提出的两种方法的求解时间是两种完备方法的 20 倍以上.当模型数达到 1 000 以上时,本文提出的两种方法的求解效率与完备方法差距较大.

对表 3 实验结果进行分析,当实例规模扩大时,优化后的增量法的求解效率要优于迭代法的求解效率,原因是合取范式所包含的文字数会随着实例规模的增大而增多.应用局部搜索方法每求出一个真值指派后,利用解的优化策略,需要添加的子句中文字可约减的概率增大,个数也可能会增加.这样就继续保持了格局检测的优越性,并且在下一次迭代过程中减少了搜索计算量.而优化后的增量法的优越性在实例规模较小时体现得并不明显,原因在于,解的优化策略每一次执行都需要遍历一次已存在的全部子句.当遍历操作所耗费的时间大于解的优化策略所能节省的时间时,优化后的增量法的求解效率将不如迭代法.当无法约减任何文字时,优化后的增量法的求解效率同样不如迭代法.这种情况在实例规模较小时体现得较为明显.

为了更加全方位地测试推理框架以及算法的执行性能,本文选取结构化实例(AIM 类问题、blocksworld 类问题、PARITY 类问题以及 CBS 类问题)进行了对比实验,实验结果见表 4~表 7.

上述结构化实例中,合取范式中包含的子句不全是 3-SAT,有些子句仅包含两个或多个文字.而 SWcc 算法的优势在于求解 3-SAT 子句,故求解结构化实例的效率比求解随机化实例效率有所下降.在求解模型数为 1 的子句时,不完备方法仍具有优势.值得注意的是,表 4 中的 AIM 类实例的模型数均为 1 是测试样例的特点,而不是人为选取.对于该类实例,Cachet 求解时间是优化后的增量法与迭代法求解时间的 2 倍以上.sharpSAT 求解时间是优化后的增量法与迭代法求解时间的 1.5 倍以上.对于表 5 中的模型数为 1 的 blocksworld 类实例,Cachet 求解时间为本文所提两种算法求解时间的 3 倍~4 倍.sharpSAT 求解时间为本文提出的两种算法求解时间的 2 倍~3 倍.对于模型数为 2 的实例 medium,本文提出的两种算法同样具有优势.对于 PARITY 类实例,本文提出的两种算法的求解效率为完备方法的 5 倍左右.对于 CBS 类实例,由于采样的实例的模型数均不低于 80,本文提出的

© 中国科学院软件研究所 http://www.jos.org.cn

两种算法并不具备求解效率上的优势.

Table 4 Experimental results on AIM instances (s)

表 4 AIM 类问题实例的实验结果 (s)

实例	模型数	优化后的增量法	迭代法	Cachet	sharpSAT
aim-50-1_6-yes1-1	1	0.008 793	0.009 897	0.020 599	0.016 703
aim-50-1_6-yes1-2	1	0.010 955	0.012 724	0.021 670	0.017 356
aim-50-1_6-yes1-3	1	0.004 614	0.004 836	0.022 029	0.018 678
aim-50-1_6-yes1-4	1	0.004 658	0.004 528	0.021 020	0.017 558
aim-50-2_0-yes1-1	1	0.004 041	0.004 003	0.022 943	0.018 227
aim-50-2_0-yes1-2	1	0.003 194	0.003 506	0.022 088	0.017 371
aim-50-2_0-yes1-3	1	0.003 306	0.003 696	0.019 847	0.014 150
aim-50-2_0-yes1-4	1	0.004 033	0.004 894	0.025 345	0.014 684

Table 5 Experimental results on blocksworld instances (s)

表 5 Blocksworld 类问题实例的实验结果 (s)

实例	模型数	优化后的增量法	迭代法	Cachet	sharpSAT
anomaly	1	0.006 153	0.004 781	0.021 062	0.017 188
bw_large.a	1	0.012 060	0.011 486	0.032 475	0.031 691
bw_large.b	2	0.205 732	0.161 986	0.162 289	0.069 550
huge	1	0.016 661	0.015 780	0.029 559	0.036 054
medium	2	0.006 154	0.004 994	0.018 130	0.020 217

Table 6 Experimental results on PARITY instances (s)

表 6 PARITY 类问题实例的实验结果 (s)

实例	模型数	优化后的增量法	迭代法	Cachet	sharpSAT
par8-1-c	1	0.003 550	0.003 783	0.019 536	0.017 773
par8-2-c	1	0.002 921	0.003 592	0.019 990	0.016 749
par8-3-c	1	0.005 137	0.005 156	0.020 403	0.020 184
par8-4-c	1	0.003 641	0.004 203	0.019 482	0.019 699
par8-5-c	1	0.004 254	0.004 358	0.020 483	0.018 477

Table 7 Experimental results on CBS instances (s)

表 7 CBS 类问题实例的实验结果 (s)

实例	模型数	优化后的增量法	迭代法	Cachet	sharpSAT
CBS_k3_n100_m441_b90_0	80	0.226 907	0.268 882	0.045 408	0.027 475
CBS_k3_n100_m441_b90_1	448	3.796 172	7.013 469	0.041 678	0.025 963
CBS_k3_n100_m441_b90_2	80	0.178 873	0.266 189	0.041 452	0.030 354
CBS_k3_n100_m441_b90_3	176	0.630 085	1.405 833	0.052 569	0.036 754
CBS_k3_n100_m441_b90_4	144	0.454 496	0.821 496	0.049 634	0.029 669
CBS_k3_n100_m441_b90_5	132	0.427 013	1.172 159	0.066 094	0.034 003
CBS_k3_n100_m441_b90_6	576	6.227 574	10.599 465	0.040 438	0.024 254
CBS_k3_n100_m441_b90_7	364	2.626 116	4.977 643	0.043 141	0.028 257
CBS_k3_n100_m441_b90_8	260	1.386 326	2.281 566	0.055 459	0.032 671
CBS_k3_n100_m441_b90_9	84	0.235 902	0.323 409	0.048 353	0.026 285

综上,优化后的增量法与迭代法在求解 uf 类随机实例时,当所求解的实例的模型数小于 10 时,求解能力比完备方法具有优势.当模型数小于 5 时,优势格外明显.当实例规模扩大时,优化后的增量法继续拥有求解优势,而迭代法的优势有所减弱.在求解结构化实例时,两种不完备的方法对于模型数为 1 以及部分模型数为 2 的实例具有求解优势,而对于其余模型数较大的实例则不具备求解优势.

本文方法求解效率提升的原因可以概括为:传统完备方法的求解效率对给定问题的模型数并不敏感,即对于同类问题,模型数的大小对于求解效率的影响并不大;而本文提出的利用局部搜索的模型计数求解方法,在每次的迭代过程中求出一个不重复的可满足指派.给定问题的模型越少,所需的迭代计算过程就越少,求解时间自然也会更短.因此在这种情况下,不完备方法具备超越完备方法的可能.

© 中国科学院软件研究所 http://www.jos.org.cn

6 结 语

本文提出了迭代法与优化后的增量法两种模型计数方法.由于完备的模型计数方法在求解模型较少的实例时性能一般,为了进一步提高对该类实例的求解性能,本文在 Cai 等人的 SWcc 算法基础上进行扩充,得到两种不完备的模型计数方法.实验结果表明,当给定合取范式的模型较少(模型数小于 10)时,本文提出的两种方法与完备方法相比,求解性能提升明显.增量法起初求解能力一般,但经优化后,保持了格局检测的优越性,求解能力最强.模型越少,两种不完备算法的优越性越能得到体现.但当给定的合取范式的模型较多时,不完备方法要将可满足解依次求出,其求解时间势必增加.

本文提出的两种不完备的模型计数方法很好地利用局部搜索,在求解模型较少的实例上具有优势,为高效推理提供了可能.对于给定问题的模型数较大的情况,仍有待进一步研究.

References:

[1] Monasson R, Zecchina R, Kirkpatrick S, Selman B, Troyansky L. Determining computational complexity from characteristic “phase transitions”. Nature, 1999,400(6740):133−137.

[2] Mezard M, Parisi G, Zecchina R. Analytic and algorithmic solution of random satisfiability problems. Science, 2002,297(5582): 812−815.

[3] Luo C, Cai SW, Wu W, Su KL. Double configuration checking in stochastic local search for satisfiability. In: Proc. of the 28th AAAI Conf. on Artificial Intelligence (AAAI 2014). 2014. 2703−2709.

[4] Cai SW, Su KL. Comprehensive score: Towards efficient local search for SAT with long clauses. In: Proc. of the 23rd Int’l Joint Conf. on Artificial Intelligence (IJCAI 2013). 2013. 489−495.

[5] Cai SW, Su KL. Local search with configuration checking for SAT. In: Proc. of the 23rd Int’l Conf. on Tools with Artificial Intelligence (ICTAI 2011). 2011. 59−66.

[6] Cai SW, Su KL, Satar A. Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artificial Intelligence, 2011,175(9):1672−1696.

[7] Cai SW, Su KL. Configuration checking with aspiration in local search for SAT. In: Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI 2012). 2012. 434−440.

[8] Cai SW, Luo C, Su KL. CCAnr: A configuration checking based local search solver for non-random satisfiability. In: Proc. of the 18th Int’l Conf. on Theory and Applications of Satisfiability Testing (SAT 2015). 2015. 1−8.

[9] Majercik SM, Littman ML. Contingent planning under uncertainty via stochastic satisfiability. Artificial Intelligence, 2003, 147(1-2):119−162.

[10] Palacios H, Bonet B, Darwiche A, Geffner H. Pruning conformant plans by counting models on compiled d-DNNF representations. In: Proc. of the 15th Int’l Conf. on Automated Planning and Scheduling (ICAPS 2005). 2005. 141−150.

[11] Moskewicz MW, Madigan CF, Zhao Y, Zhang L, Malik S. Chaff: Engineering an efficient SAT solver. In: Proc. of the 38th Design Automation Conf. (DAC 2001). 2001. 530−535.

[12] Zhang L, Madigan CF, Moskewicz MH, Malik S. Efficient conflict driven learning in a Boolean satisfiability solver. In: Proc. of the 2001 Int’l Conf. on Computer-aided Design (ICCAD 2001). 2001. 279−285.

[13] Sang T, Bacchus F, Beame P, Beamel P, Kautzl HA, Pitassi T. Combining component caching and clause learning for effective model counting. In: Proc. of the 17th Int’l Conf. on Theory and Applications of Satisfiability Testing (SAT 2004). 2004. 20−28.

[14] Sang T, Beame P, Kautz HA. Heuristics for fast exact model counting. In: Proc. of the 8th Int’l Conf. on Theory and Applications of Satisfiability Testing (SAT 2005). LNCS 3569, 2005. 226−240.

[15] Sang T, Beame P, Kautz HA. Performing Bayesian inference by weighted model counting. In: Proc. of the 20th National Conf. on Artificial Intelligence (AAAI 2005). 2005. 475−482.

[16] Thurley M. sharpSAT—Counting models with advanced component caching and implicit BCP. In: Proc. of the 9th Int’l Conf. on Theory and Applications of Satisfiability Testing (SAT 2006). LNCS 4121, 2006. 424−429.

[17] Yin MH, Lin H, Sun JG. Solving #SAT using extension rules. Ruan Jian Xue Bao/Journal of Software, 2009,20(7):1714−1725 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3320.htm [doi: 10.3724/SP.J.1001.2009.03320]

[18] Lai Y, Ouyang DT, Cai DB, Lü S. Model counting and planning using extension rule. Journal of Computer Research and Development, 2009,46(3):459−469 (in Chinese with English abstract).

© 中国科学院软件研究所 http://www.jos.org.cn

[19] Birnbaum E, Lozinskii E. The good old Davis-Putnam procedure helps counting models. Journal of Artificial Intelligence Research, 1999,10:457−477.

[20] Jia FY, Ouyang DT, Zhang LM, Liu SG. Reconstructive algorithm based on extension rule for solving #SAT incrementally. Ruan Jian Xue Bao/Journal of Software, 2015,26(12):3117−3129 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4827.htm [doi: 10.13328/j.cnki.jos.004827]

[21] Ouyang DT, Jia FY, Liu SG, Zhang LM. An algorithm based on extension rule for solving #SAT using complementary degree. Journal of Computer Research and Development, 2016,53(7):1596−1604 (in Chinese with English abstract).

[22] Wang Q, Liu L, Lü S. #SAT solving algorithms based on extension rule using heuristic strategies. Ruan Jian Xue Bao/Journal of Software, 2018,29(11):3517−3527 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5298.htm [doi: 10.13328/j.cnki.jos.005298]

[23] Gomes CP, Hoffmann J, Sabharwal A, Selman B. From sampling to model counting. In: Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2007). 2007. 2293−2299.

[24] Wang JY, Yin MH, Wu JL. Two approximate algorithms for model counting. Theoretical Computer Science, 2017,657:28−37.

[25] Wei W, Erenrich J, Selman B. Towards efficient sampling: Exploiting random walk strategies. In: Proc. of the 19th National Conf. on Artificial Intelligence (AAAI 2004). 2004. 670−676.

[26] Darwiche A. Decomposable negation normal form. Journal of the ACM, 2001,48(4):608−647.

[27] Darwiche A. New advances in compiling CNF into decomposable negation normal form. In: Proc. of the 16th Eureopean Conf. on Artificial Intelligence (ECAI 2004). 2004. 328−332.

[28] Muise C, Mcilraith SA, Beck JC, Hsu EI. Dsharp: Fast d-DNNF compilation with sharpSAT. In: Proc. of the 25th Canadian Conf. on Artificial Intelligence (Canadian AI 2012). LNCS 7310, 2012. 356−361.

附中文参考文献:

[17] 殷明浩,林海,孙吉贵.一种基于扩展规则的#SAT 求解系统.软件学报,2009,20(7):1714−1725. http://www.jos.org.cn/1000-9825/3320.htm [doi: 10.3724/SP.J.1001.2009.03320]

[18] 赖永,欧阳丹彤,蔡敦波,吕帅.基于扩展规则的模型计数与智能规划方法.计算机研究与发展,2009,46(3):459−469.

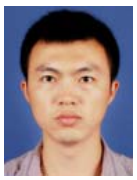
[20] 贾凤雨,欧阳丹彤,张立明,刘思光.结合扩展规则重构的#SAT 问题增量求解方法.软件学报,2015,26(12):3117−3129. http://www.jos.org.cn/1000-9825/4827.htm [doi: 10.13328/j.cnki.jos.004827]

[21] 欧阳丹彤,贾凤雨,刘思光,张立明.结合互补度的基于扩展规则#SAT 问题求解方法.计算机研究与发展,2016,53(7):1596−1604.

[22] 王强,刘磊,吕帅.基于扩展规则的启发式#SAT 求解算法.软件学报,2018,29(11):3517−3527. http://www.jos.org.cn/1000-9825/5298.htm [doi: 10.13328/j.cnki.jos.005298]

贺甫霖(1993－),男,吉林省吉林市人,硕士,主要研究领域为人工智能,自动推理.

刘磊(1960－),男,教授,博士生导师,CCF 专业会员,主要研究领域为软件理论与技术.



吕帅(1981－),男,博士,副教授,博士生导师,CCF 高级会员,主要研究领域为人工智能,智能规划,机器学习,自动推理.

牛当当(1990－),男,博士,讲师,CCF 专业会员,主要研究领域为人工智能,自动推理,抽象论辩.

王强(1994－),男,硕士,主要研究领域为人工智能,自动推理,机器学习.

© 中国科学院软件研究所 http://www.jos.org.cn