软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(9):2744−2754 [doi: 10.13328/j.cnki.jos.005974]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于局部搜索的并行扩展规则推理方法*

李 壮[1,2], 刘 磊[1], 张桐搏[1,2], 周文博[1,2], 吕 帅[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)

通讯作者: 吕帅, E-mail: lus@jlu.edu.cn

摘 要: 扩展规则推理方法在经典的可满足性问题求解中已得到广泛应用,若干个基于扩展规则的推理方法已被提出,皆得到国内外的认可,例如完备的NER,IMOMH_IER,PPSER算法以及基于局部搜索的不完备算法ERACC等,都具有良好的求解效果.其中,ERACC 算法是当前扩展规则求解器中求解效率最高、能力最强的算法.但是,串行的ERACC 算法在启发式和预处理上仍然具有可提升的空间.基于此,设计了相应的并行框架,提出了 PERACC 算法.该算法基于格局检测的局部搜索方法,从变量赋初始值、化简解空间和启发式这 3 个阶段出发,将原极大项空间分解成为若干极大项子空间,并对原子句集进行化简后,并行处理各个子空间.通过实验显示:该算法与原算法相比,不仅在求解效率方面有较大提高,而且可以求解规模更大的测试用例,使扩展规则方法再次突破公式规模的限制.

关键词: 自动推理;局部搜索;扩展规则;格局检测;并行框架

中图法分类号: TP181

中文引用格式: 李壮,刘磊,张桐搏,周文博,吕帅.基于局部搜索的并行扩展规则推理方法.软件学报,2021,32(9):2744−2754. http://www.jos.org.cn/1000-9825/5974.htm

英文引用格式: Li Z, Liu L, Zhang TB, Zhou WB, Lü S. Local search-based parallel extension rule reasoning method. Ruan Jian Xue Bao/Journal of Software, 2021,32(9):2744−2754 (in Chinese). http://www.jos.org.cn/1000-9825/5974.htm

Local Search-based Parallel Extension Rule Reasoning Method

LI Zhuang[1,2], LIU Lei[1], ZHANG Tong-Bo[1,2], ZHOU Wen-Bo[1,2], LÜ Shuai[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), Changchun 130012, China)

Abstract: Extension rule reasoning method has been widely used in solving the classical satisfiability problem. Several reasoning methods based on extension rule have been proposed, which have been recognized around the world. These methods include the complete algorithms like NER, IMOMH_IER, PPSER, and local search-based incomplete algorithm like ERACC. All of them have sound performance. ERACC algorithm is the most efficient and powerful algorithm in the current extension rule solver. However, the serial algorithm ERACC still needs improvement on heuristics and preprocessing. Therefore, a parallel framework is designed and it is called PERACC algorithm. Based on the configuration checking of local search method, the algorithm decomposes the original maximum term space into several maximum term subspaces, from three stages of assigning initial values to variables, simplifying solution space, and heuristic, simplifying the original clause set, then processing each subspace in parallel. Experiments show that, compared with the original

* 基金项目: 国家重点研发计划(2017YFB1003103); 国家自然科学基金(61300049, 61763003); 吉林省自然科学基金(20180101053JC, 20190201193JC, 20190103005JH); 吉林大学研究生创新基金(101832018C025)

Foundation item: National Key Research and Development Program of China (2017YFB1003103); National Natural Science Foundation of China (61300049, 61763003); Natural Science Research Foundation of Jilin Province of China (20180101053JC, 20190201193JC, 20190103005JH); Graduate Innovation Fund of Jilin University of China (101832018C025)

收稿时间: 2019-02-28; 修改时间: 2019-08-25; 采用时间: 2019-10-30; jos 在线出版时间: 2019-12-05

CNKI 网络优先出版: 2019-12-05 14:55:15, http://kns.cnki.net/kcms/detail/11.2560.TP.20191205.1454.007.html

algorithm, the proposed algorithm not only can improve the efficiency of solution, but also can solve larger benchmark, which makes the extension rule method break through the limitation of formula size again.

Key words: automated reasoning; local search; extension rule; configuration checking; parallel framework

可满足性问题(SAT)是公认的第一个 NP-完全问题,是人工智能领域最重要的研究方向之一.自人工智能发展初期就引起了研究者的广泛关注,当今 SAT 求解器已能够解决很多现实问题.近年来,国内外的学者致力于求解 SAT 问题的效率和求解能力,提出了多种方法.

1992 年,Selman 等人提出的 GSAT 算法证明了局部搜索算法可以应用于求解 SAT 问题,其强大的求解能力为局部搜索求解 SAT 问题奠定了基础[1].随着局部搜索方法在该领域的广泛应用,自 2002 年 SAT 比赛的开展以来,众多基于局部搜索算法的求解器也相继问世并展现出了各自强大的求解能力.2005 年,Li 等人提出了 g2wsat 类算法,该算法结合随机选择和贪心双模式,在求解的过程中展现出良好的性能[2].2011 年,Li 等人提出了利用变量可满足历史记录的局部搜索求解器 SatTime,获得 SAT 2011 随机组银奖[3].2016 年,KhudaBukhsh 等人提出了 SATenstein 求解框架,该框架结合了多个高性能局部搜索求解器,其求解效果胜过多个主流求解器[4].近年来,由 Cai 等人提出的基于局部搜索的格局检测策略成为 SAT 比赛中最具影响力的策略之一.2011 年,Cai 等人设计的 SWCC 算法及相关求解器皆使 SAT 求解器的计算能力得到巨大地提升,并取得惊人的成绩[5].随后,Cai 等人于 2012 年提出的 CCASat 获得 SAT 2012 年随机组金奖[6].2014 年,Cai 等人提出的 CSCCSat 获得 SAT 2014 年比赛 Hard-combinatorial 组铜奖和 SAT 2016 比赛 Random track 组银奖[7].同时,基于局部搜索的格局检测策略用于求解 Max-SAT 问题也取得了巨大的成功[8,9],其中,CCLS 在 MAX-SAT 2013 非完备性求解器组中获得 4 项金奖.2018 年,Cai 等人基于局部搜索算法,在完备算法上设计了 ReasonLS 求解器,并在同年 SAT 国际比赛中获得 No-Limits track 组金奖[10].以上不难看出,局部搜索算法的广泛应用和强大的效率已得到不断突破.

经典的 SAT 求解器大多基于 DPLL 算法和归结方法,2003 年,Lin 等人提出的扩展规则方法打破了传统思维的限制,开辟了 SAT 求解的新思路[11].作为归结方法的互补方法,通过寻找子句的极大项来判断 CNF 公式的可满足性.他提出的 ER 算法得到了国内外的广泛认可,并为日后的研究打下了坚实的基础.基于 ER 的算法,Lin、Wu、李莹和张立明等人分别提出了 IER[11],RER[12],NER[13],SER[14]等算法,将完备的 ER 算法的求解效率提升了几个数量级的高度.在扩展规则的应用方面,2010 年,殷明浩等人在可能性扩展规则的基础上提出了 EPPCCCL 理论,可作为可能性知识编译的目标语言[15].2016 年,刘磊等人提出了扩展反驳方法,并在该推理方法与知识编译之间建立了联系[16].2017 年,王强等人提出两种加速#SAT 求解的启发式策略,提出了 CER_MW 和 CER_LC&MW 两种算法,其求解速度是其他算法的 1.4 倍~100 倍,求解能力也在限定时间内可解更多的测试用例[17].2018 年,杨洋等人在 SWCC 算法的基础上提出了 ERACC 算法,该算法突破了传统扩展规则推理对公式规模的局限,求解效率有了极大提高[18,19].

不仅如此,各种经典的技术也相继应用于扩展规则求解的过程中,如启发式和并行处理等技术.2009 年,李莹等人针对 IER 算法提出了 IMOM 和 IBOHM 两种启发式,其求解效率较 IER 提高了 10 倍~200 倍[20].Zhang 等人基于其 PSER 的半扩展规则策略提出了分割解空间的 PPSER 算法,证明了该算法具有合理的执行效率,并优于 NER、DR、IER 等算法[21].

由以上不难看出,基于扩展规则的推理方法如今已得到了广泛的应用.但与局部搜索等推理方法相比,仍然具有很大的提升空间和完善度.本文基于 ERACC 算法,从李莹的 IMOM 启发式算法和张立明的 PPSER 并行算法中得到启示,提出了 PERACC 算法.该算法基于杨洋等人[18,19]的 ERACC 算法,调用启发式算法 CPVI,通过选择变量并调用 PERM 算法,用变量的组合将原解空间分割成不同的子空间,每个子空间通过 SIMT 算法选择初始极大项,随后进行并行处理.最后实验证明,该算法较 ERACC 算法的求解效率有较大的提高[22].

本文第 1 节介绍了局部搜索和扩展规则的基础知识并给出了相关定义.第 2 节提出启发式策略 CPVI 算法、并行处理策略 PERM 算法和变量赋初始值策略 SIMT 算法,并将它们有机结合,在 ERACC 算法基础上提出了 PERACC 算法.第 3 节进行了实验的对比,证明了我们的算法在特定的问题上较原算法快出 1.6 倍~117 倍不等.

1 基础知识

1.1 局部搜索

定义 1. 在一个布尔变量集合 $V=\{x_1,x_2,\ldots,x_n\}$中,每个变量呈正、负两种文字形式 x_i和$\neg x_i$以析取的形式存在于 CNF(conjunctive normal form)公式的每个子句 C_i中,公式中的每个子句之间是合取的.找到一个或多个解释能使 CNF 公式为可满足的真值指派过程,称为 SAT 求解.

SAT 问题的求解是按照某种特定规则对变量进行选取、赋值和剪枝等过程,找到可满足的解释.完备的算法如 DPLL,其功能在于不仅可以判断子句集的可满足性,而且可以找到可满足的解.局部搜索作为不完备的方法采取完全不同的方式对 SAT 问题进行求解,该方法通过不断变换部分子句空间所有变量赋值使得可满足的子句不断增加,最终达到子句集可满足的目的.当处理不可满足的问题时,由于子句集无解而使变量赋值无限地变换下去.因此,局部搜索只能求解可满足的子句集,且其求解效率远高于完备的算法,却无法处理不可满足的子句集.

1.2 扩展规则

传统的归结方法是通过归结出空子句来判断子句集不可满足;而作为归结的互补方法,扩展规则是通过对 CNF 公式中的子句能否扩展出所有极大项来判断其可满足性.其基本定义如下.

定义 2. CNF 公式中的一个子句 C,变量 l 的文字形式不在 C 中出现.有 $D=\{C\vee l, C\vee\neg l\}$,则称 C 到 D 的过程为扩展规则,D 是子句 C 关于变量 l 扩展后的结果.

定义 3. 若一个子句包含所有变量对应的正文字或负文字之一,则称该子句为变量集的一个极大项.

初始的扩展规则推理方法是通过计数每个子句所能扩展出的极大项个数,利用容斥原理计算所能扩展出极大项总数.如果极大项个数是 2^n(n 为变量个数),则 CNF 公式是不可满足的;否则是可满足的.这种完备算法的缺点在于:其处理的 CNF 规模十分有限,当子句数超过一定阈值,将会造成内存溢出.李莹等人提出的 NER 算法通过一定次序判断当前极大项 T 是否能被 CNF 公式中的子句扩展出来:当所有极大项都能被扩展出来时,子句集不可满足;当存在至少一个极大项不能被扩展,则子句集可满足.该算法的求解效率远优于 ER 和 IER 算法,但缺点在于机械地变换极大项,暴力的求解方式满足了算法的完备性,却丢失了很多启发式信息.杨洋等人提出的 ERACC 算法基于局部搜索的格局检测方法,通过得分机制选取变量,根据变量的邻居关系精确格局信息,并通过双向半扩展规则来限制变量的选取范围.该算法在时间复杂度和空间复杂度上都有了很大的改进,并且利用了局部搜索的高效性,使求解可满足问题的求解效率达到了质的飞跃.

1.3 基于精确格局检测的扩展规则算法

基于局部搜索的扩展规则推理方法的本质在于:以寻找极大项空间不可扩展的极大项为目标,利用贪心策略限定极大项空间,以变量的邻居变量设定格局信息,从而减少了冗余搜索步骤,牺牲了算法完备性,从而达到了高效求解的效果.

ERACC 算法的精髓在于对极大项中反转变量的选择,通过最大左度和最小右度的概念,限定两集合中的变量.如果集合为空,则退一步选择得分高的变量取反.精确了变量的选择直接控制了极大项的变换,使得推理过程更加精确(算法的具体流程见文献[18,19]).但是算法仍存在以下不足:初始极大项的选择并没有明确的启发式;串行的处理也使得 ERACC 算法无法发挥出极致的效果;忽略了预处理技术,使子句集没有化简的过程.下文将针对以上问题提出 PERACC 算法,使求解的性能发挥到极致.

2 PERACC 算法

传统的基于扩展规则的并行推理方法将子句集的极大项空间分解为若干个子空间,并对其进行逐一求解.这种并行方式并未加入启发式信息,使得分割空间无层次.即使分割的子空间彼此毫无关联,但整体推理的空间复杂度并无降低.

本文提出的并行处理是一种结合了对解空间化简的预处理方法.在选择特定的变量之后,对原子句集进行化简,化简后的子句集与原子句集是逻辑等价的,这也极大地降低了对整个子句集求解的空间需求;并在ERACC算法中加入了对初始值的赋值限制,改变了ERACC对初始值赋为1的单调做法,使求解器整体求解能力得到了较大地提升.

2.1 CPVI算法

我们通过选取变量的启发式算法,对出现在每个子句中的文字赋予权值.从文字出现在各个子句中的累计次数计算出得分最高的变量.首先,遍历原始子句集获得所有文字的数组,从数组中选取最大数对应的变量.如果同等数量的变量有若干个,那么从中随机选取需求个数作为初始变量,变量个数由所划分的解空间参数决定.变量选取算法如下,其中,vs=(var,$score$)表示变量及对应的得分,其中,var表示变量,$score$表示变量的得分.

算法 1. 计算选取的变量CPVI(computing the picking variables initially).

输入:CNF公式 $F=\{C_1,\ldots,C_m\}$,选取变量个数α;

输出:CNF公式 F 中包含最多文字的前α个变量的集合 $result$.

```
1  result←∅,vs←∅;
2  {v1,v2,…,vn}←getAL(F)                  //函数 getAL 返回 CNF 公式 F 中所有变量的集合
3  FOR (i←1; i<n+1; i++)                  //初始化数组 score
4     vsi.var←vi;
5     vsi.score←0;
6  FOR (i←1; i<n+1; i++)                  //对每个变量循环
7     FOR (j←1; j<m+1; j++)               //对每个子句循环
8        IF vsi.var∈Cj
9        THEN vsi.score=vsi.score+1;     //数组 score 的第 i 个元素表示第 i 个变量的得分
10 sv←sort(vs,n);                        //函数 sort 根据 score 域对数组 vs 进行从大到小的排序
11 ns←{svα.var};
12 i←α+1;
13 k←1;
14 IF svα.score=svi.score
15 THEN
16    WHILE (i<n+1 and svα.score=svi.score)
17       ns←ns∪{svi.var};
18       i←i+1;
19    i←α–1;
20    WHILE (i>0 and svα.score=svi.score)
21       ns←ns∪{svi.var};
22       k←k+1;
23       i←i–1;
24    svars←random(ns,k);                //函数 random 返回从集合 ns 中随机选取 k 个元素
25    result←{sv1.var,…,svα–k.var}∪svars;
26 ELSE result←{sv1.var,…,svα.var};
27 RETURN result.
```

算法1采取一种遍历求和排序的方式来计算各个变量的得分集合,从而选取权重最高的变量作为化简子句集的初始变量.算法第2行表示程序遍历了整个子句集F并统计了所有变量集合;第3行~第5行对所有的变量及其得分清零;第6行~第10行分别描述了对所有变量和每个变量在各个子句中的双重循环,得到变量集合

的得分数组 $vs_i.score$,并对其进行排序;第 12 行~第 24 行表示当出现得分均等的若干个变量时,从中随机选取所需个数变量;最后输出所选择的变量.

变量得分的计算,令 F 为 CNF 公式,$C_i=\{l_1,\ldots,l_k\}$为子句集中任意子句($i\in(0,m]$),m 为子句数,W_i 为子句相关联文字的权值.则变量 v 基于 F 的得分函数为

$$Score(v)=\sum_{i=1}^{m}|\delta(C_i,l)|\times W_i.$$

在以上公式中,δ为二元谓词特征函数,代表多值逻辑的取值.令 l 表示在子句 C_i 中变量 v 所对应文字的取值.δ的具体定义如下:

$$\delta(C_i,l)=\begin{cases}1, & C_i\text{中的}l\text{为正文字}\\ -1, & C_i\text{中的}l\text{为负文字}\end{cases}.$$

以上公式给出了变量得分的计算方式,基本思想为:根据文字出现在子句中的形式取相应的值.正文字取 1(变量所对应文字为 l 时取 1),负文字取–1(变量所对应文字为$\neg l$ 时取–1).通过求和计算累计变量的得分,根据得分将变量排序,进而选取所需变量.该方法的时间复杂度是线性的,因此,该预处理方法所需时间在求解的过程中所占比重是极小的.

例 1:令 CNF 公式 $F=\{A\vee C\vee\neg D,\neg B\vee\neg G\vee H,\neg A\vee E\vee G\vee\neg H,C\vee E\vee H,\neg D\vee G\vee\neg H\}$,在该测试用例中,每个变量相对应的文字权值 W_i 为 1,则变量的得分如下:

- $Score(A)=|1\times1|+|0\times1|+|(-1)\times1|+|0\times1|+|0\times1|=2$;
- $Score(B)=|0\times1|+|(-1)\times1|+|0\times1|+|0\times1|+|0\times1|=1$;
- $Score(C)=|1\times1|+|0\times1|+|0\times1|+|1\times1|+|1\times1|=2$;
- $Score(D)=|(-1)\times1|+|0\times1|+|0\times1|+|0\times1|+|(-1)\times1|=2$;
- $Score(E)=|0\times1|+|0\times1|+|1\times1|+|1\times1|+|0\times1|=2$;
- $Score(G)=|0\times1|+|(-1)\times1|+|1\times1|+|0\times1|+|1\times1|=3$;
- $Score(H)=|0\times1|+|1\times1|+|(-1)\times1|+|1\times1|+|(-1)\times1|=4$.

由例 1 可以看出:给定 CNF 公式 F 中,所有文字的权值为 1,在对变量选择的过程中扫描一次即可得知变量的得分.在选取变量阶段,我们选取得分最高的变量(例 1 中的变量 F).在硬件条件允许的条件下,我们可以选择多个变量.如果我们选择 3 个变量,那么会根据得分依次选择变量 G 和变量 H,而第 3 个变量的选取则在得分相等的变量$\{A,C,D,E\}$中随机选取一个.

我们的目的在于对原解空间进行最大程度化简,因此选择了得分最高的变量作为初始变量.最理想的情况是存在某个变量出现在每个子句中,这样可以最大程度地降低子句集规模,得到的问题规模最小.即:在预处理阶段约简了原子句集,其约简后的结果与原子句集是逻辑等价的.

定理 1. 预处理阶段,化简后的子句集与原子句集是逻辑等价的.

证明:固定了极大项中的文字之后,原子句集中带有互补文字的子句可直接忽略,因为这样的子句是无法扩展出当前极大项的;而带有相同的文字可以忽略不计,只看除了该文字剩余部分的子句.这样,剩余变量的模型组成了新的极大项,与化简后的子句集相对应,即,同原极大项和原子句集是逻辑等价的.

例 2:令 CNF 公式 $F=\{A\vee B\vee C\vee D,\neg A\vee B\vee\neg C\vee E,\neg A\vee\neg B\vee\neg G,\neg A\vee B\vee\neg D\vee\neg E,B\vee C\vee\neg D\vee G\}$,按照算法 1 选取变量,我们应选择的变量为 A 和 B.如果我们单看其中的一个子空间,变量取值为$\neg A$ 和 B,那么化简的原子句集为 $F'=\{\neg C\vee E,\neg D\vee\neg E,C\vee\neg D\vee G\}$,新的解空间是由变量$\{C,D,E,G\}$所组成的极大项空间.

由例 2 可以看出:在保证了逻辑等价的前提下,化简后的子句集规模已经远远地小于原子句集的规模.实验证明:不论在时间还是空间上,都有了很大的提高.

2.2 PERM算法

选择变量的目的在于化简解空间,而并行的目的是基于化简基础上能够全面地对子句集求解.由于选择的变量组合的不同,对子句集化简的程度也不同.基于该问题,我们会选择几种化简后,求解效率最高的那个子空

间作为判断的标准.我们提出了 PERM 算法来解决这一问题.

PERM 算法过程如下:

算法 2. 并行扩展的推理方法 PERM(parallel extensionrulereasoning method).

输入:CNF 公式 $F=\{C_1,\ldots,C_m\}$,F 的极大项 T,选择变量个数α;

输出:不可扩展的极大项 T 或最大尝试次数 *maxTries*.

```
1  num←2^α;                                  //解空间总数
2  FOR (i=0; i<num; i++)
3    startThread(i,F,T,ERACC);               //函数 startThread 启动解空间 i,调用 ERACC 算法
4  WHILE (1) DO
5    FOR (i=0; i<num; i++)
6      IF EndThread(i)                       //函数 EndThread 判断解空间 i 是否计算完
7      THEN RETURN resultThread(i).          //函数 resultThread 返回解空间 i 的结果
```

算法 2 给出在极大项子空间中分割原始极大项解空间的并行算法 PERM.首先输入原 CNF 公式和已选取的变量,预处理程序通过二进制方法将变量的 2^{α}种析取将子句集的极大项空间分解为 $thread_{num}$ 个独立的子空间,并记录每个子空间的返回结果,然后对每个极大项子空间并行地调用 ERACC 算法进行极大项的判定.当某一个极大项子空间 *EndThread*(*i*)中返回的结果为 SAT,即该子空间中存在某个极大项在规定时间内不能被扩展出来的子句,则子句集是可满足的.将该结果返回至主程序,推理过程结束.

2.3 SIMT算法

由第 1 节我们可知,NER 和 ERACC 两种算法都是以寻找子句集扩展不出的极大项为目标来进行推理的,所以我们采取的途径是变换和调整极大项的模型.在 ERACC 中已详细地说明了如何去变换调整极大项,忽略了对初始极大项的重视,采用了单纯的对初始极大项中没个变量赋值为 1.初始的极大项中,变量的赋值在整个推理过程中占据重要的位置.如果没有针对性地给初始极大项赋值,则会在求解过程中走很大的弯路,间接地增加了冗余的推理过程.

为了使推理过程更直接、高效,我们提出了 SIMT 算法,有针对性地解决对初始极大项中变量赋初始值的选择.由于极大项中的每个变量可以赋值为正和负两种文字形式,而每种文字形式的赋值概率为 50%,我们提出的算法是每个变量根据原子句集中文字出现的比率来确定初始极大项中变量的取值概率.实验结果表明,我们的策略是有效的.

SIMT 算法过程如下.

算法 3. 选择初始极大项方法 SIMT(selection of initial maximum term).

输入:CNF 公式 F;

输出:初始极大项 T.

```
1  i←1;
2  n←getNumLiteral(F);                       //返回 CNF 公式 F 的变量数目
3  {l1+,l2+,...,ln+}←getPosLiteral(F);       //返回 CNF 公式 F 中每个正文字的个数
4  {l1-,l2-,...,ln-}←getNegLiteral(F);       //返回 CNF 公式 F 中每个负文字的个数
5  WHILE i≤n DO
6    lni← li+ + li- ;
7    pi← li+ /n(li);
8    qi← li- /n(li);
9    li←random(pi,qi);                       //随机返回 pi 或 qi
10   i←i+1;
```

11 **RETURN** $\{l_1 \vee l_2 \vee \ldots \vee l_n\}$. //返回的是一个初始的极大项

算法 3 中,i 代表初始的极大项,通过遍历子句集得到 F 中的变量数 n,再根据每个变量所对应的文字统计其正、负文字出现的个数,$n(l_i)$表示变量对应的文字总个数.第 5 行~第 10 行是一个循环.根据正负文字出现的个数与变量对应的文字总个数,计算对应极大项中该变量应该赋值的概率,即有指向性地调整了变量赋值的概率.逐一对变量赋值之后,当赋值变量的个数完成了第 n 个后,说明初始极大项已经产生,算法结束.有针对性地选择初始极大项,会使下一步的推理更加精准,提高了推理效率.

2.4 PERACC算法

结合了以上算法,基于 ERACC 算法,我们提出了基于精确格局检测的并行扩展规则算法,算法过程如下.

算法 4. 基于精确格局检测的并行扩展规则算法 PERACC(parallel extension rule based on accurate configuration checking).

输入:CNF 公式 F,$MaxTries$,选取变量个数α;

输出:不可扩展极大项 T'或最长尝试次数 $MaxTries$.

```
1  result←CPVI(α)                          //选择变量数组
2  num←2^α;                                //解空间总数
3  FOR (i=0; i<num; i++)
4    startThread(i,F',T);        //函数 startThread 启动解空间 i,F'为化简后的子句集,T 为当前极大项
5  T←SIMT(F')                              //调用 SIMT 计算各个解空间的初始极大项
6  WHILE j<MaxTries DO
7    IF CanBeExpend(·) THEN RETURN T       //判定当前极大项是否能被扩展
8    Compute the Candidate Set(left,right)
9    P={v|Score(v)>0 and conf[v]=1}
10   IF P≠∅ THEN
11     IF left≤rignt THEN
12       S={v|Score(v)>0 and left≤0≤rignt and conf[v]=1}
13       IF S≠0 THEN x←v∈S with largest Score
14       ELSE x←v∈P with largest Score
15     ELSE x←v∈P with largest Score
16   ELSE x←SWT(F,T,W[·],time[·])
17   T'←T/ref(T,x)∪¬ref(T,x)
18   j++
19 WHILE (1) DO
20   FOR (i=0; i<num; i++)
21     IF EndThread(i)                     //函数 EndThread 判断解空间 i 是否计算完
22     THEN RETURN resultThread(i).        //函数 resultThread 返回解空间 i 的结果
```

算法 PERACC 的执行过程大致如下:第 1 行~第 4 行通过调用 CPVI 算法选择初始变量,通过变量选择的数量,得到分解解空间的数量,并对各个解空间进行化简,然后对各个解空间进行求解;第 5 行调用算法 SIMT 来计算各个解空间的初始极大项;第 6 行~第 18 行是调用 ERACC 算法对每个解空间的求解,其详细过程可参考文献[18,19];第 19 行~第 22 行是一个死循环,当各个解空间中,某一个解空间找到了扩展不出的极大项,那么跳出该循环.

3 实验结果与分析

本节对算法 PERACC 算法进行实现,并且将其与扩展规则领域主流的求解器 SER,ERACC 以及国际主流

的 SWCC 求解器进行了对比.实验结果证明了 PERACC 算法不仅胜过了传统的扩展规则方法,而且大大地缩短了与国际主流求解器的距离.

实验环境:Arch Linux 16.04 操作系统,12 核 CPU,内存 8GB.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.由于本文提出的算法属于不完备算法,所以解决的问题都是可满足性的问题.测试用例来源于 https://www.cs.ubc.ca/~hoos/SATLIB/index-ubc.html,SWCC 的源代码来源于他的个人主页 https://lcs.ios.ac.cn/~caisw/SAT.hrml,g2wsat 的源码来源于国际 SAT 比赛官网 https://www.satcompetition.org/.

首先,我们采用处于相变区域的 Uniform Random-3-SAT 问题,不论是对于系统的 SAT 求解器或者是随机的局部搜索算法都是很难的问题.在以往的扩展规则算法测试中,由于受到求解规模的限制,只能求解一些小规模的测试用例.而在文献[18]中,ERACC 求解器已经展现出了其强大的求解能力,可以求解先前求解器无法求解的测试用例.所以在我们的测试中,首先选择了 ERACC 所能求解的规模最大的测试用例.我们选择变量 250、子句数为 1 065 的规模最大的问题.由于测试用例规模较小,所以该组实验我们将运行 30 秒未结束的测试用例定为 Time out.由表 1 不难看出:传统的 SER 算法已经无法解决规模庞大的测试用例,而 ERACC 算法和 PERACC 算法显然已经突破了传统算法的限制,发挥了极大的潜力.在这组测试用例中,PERACC 算法的求解效率并没有 ERACC 好.原因在于预处理的过程需要少量的时间,而 ERACC 算法没有预处理的过程.因此,在处理规模相对较小的问题时,并不能凸显出 PERACC 算法的优势,即先前所能求解的测试用例已经无法凸显我们算法的优势.

在表 1 实验中,我们默认 PERACC 算法在预处理过程中选择 4 个变量,并不确定选择 4 个变量就是最优的选择.

Table 1 Experiment results on uniform random-3-SAT instances uf250 (CPU time/s)

表 1 随机 3-SAT 测试用例 uf250 实验结果(CPU time/s)

Problem	SER	ERACC	PERACC
uf250-01	Time out	0.021	0.130
uf250-02	Time out	0.089	0.158
uf250-03	Time out	0.056	0.134
uf250-04	Time out	0.006	0.124
uf250-05	Time out	0.047	0.135
uf250-06	Time out	0.016	0.129
uf250-07	Time out	0.043	0.176
uf250-08	Time out	0.061	0.134
uf250-09	Time out	0.088	0.135
uf250-10	Time out	0.002	0.135

带着以上两个问题,我们做出了以下实验.

表 2 选择了图着色问题中一些较难的大型 Random-3-SAT 测试用例.图着色类测试用例包含 200 个顶点和 479 条边,将 GCP 测试用例编译为 SAT 问题后,成为 600 个变量和 2 237 条子句.我们选择以下测试用例的原因在于它们的复杂性更高,求解时间较长,所以该组实验我们将运行 500 秒未结束的测试用例定为 Time out.这些测试用例与表 1 相比规模更大,更能突出 PERACC 的性能,符合我们对上述问题的实验论证.

为了测试 SIMT 算法的有效性,我们在表 2 中加入了基于 ERACC 算法,结合了 SIMT 的算法 ERACC+.通过 ERACC 和 ERACC+两组实验我们不难看出:在 21 个测试用例中,ERACC+组有 18 个测试用例优于 ERACC 组,其中效果最好的一组对比数据提高了 82 倍.而平均时间也提高了近 3.5 倍.由此我们得知,加入 SIMT 算法比单纯的将初值赋为 1 效果要好很多.

PERACC(i)代表启发式过程中选择 i 个变量.从 ERACC(2)和 PERACC(4)两组数据可以看出:针对比较复杂的问题,PERACC(4)算法已经比较明显地凸显出了其性能的优势.从这两组数据也不难看出:选择变量的数量越多,其求解效率越高.结合表 1 分析,虽然 PERACC 算法在启发式阶段花掉了少许时间,但是整体来看,PERACC(4)比 ERACC 效率提高了 4 倍~25 倍不等,这也符合了磨刀不误砍柴工的道理.

我们将继续延伸我们的思路,加入了当前国际著名的局部搜索求解器 SWCC 进行对比.从数据上不难看出,PERACC 比 ERACC 的效率提升了 1.6 倍~117 倍不等.虽然 PERACC 的求解效率并没有超过 SWCC 求解器,但

是相对于当前扩展规则领域最高效的 ERACC 而言,PERACC 已大大地缩短了与 SWCC 的距离.

Table 2 Experiment results on graph Colouring instances flat200-479 (CPU time/s)

表 2 图着色测试用例 flat200-479 实验结果(CPU time/s)

Problem	ERACC	ERACC+	PERACC(2)	PERACC(4)	SWCC
flat200-4	40.544	4.714	1.191	7.054	0.058
flat200-10	7.481	19.696	7.671	3.426	0.071
flat200-12	179.901	2.206	4.791	1.533	0.023
flat200-26	24.011	3.776	1.603	2.052	0.156
flat200-32	6.941	2.521	0.637	1.011	0.105
flat200-36	12.237	7.709	0.577	0.884	0.169
flat200-39	230.423	44.138	4.218	4.411	0.236
flat200-44	5.144	0.764	0.251	0.326	0.161
flat200-45	9.331	0.733	1.869	0.491	0.037
flat200-46	8.229	2.762	0.462	0.259	0.045
flat200-48	14.327	4.465	0.924	2.417	0.566
flat200-49	5.532	0.719	1.684	0.536	0.052
flat200-55	26.241	1.425	1.232	1.904	Time out
flat200-60	5.364	9.375	1.808	1.519	0.066
flat200-72	45.634	19.326	10.054	7.889	0.211
flat200-82	5.088	4.522	0.972	0.766	0.113
flat200-83	13.901	8.599	6.783	2.266	0.384
flat200-87	17.189	31.681	1.413	0.242	0.272
flat200-88	26.887	0.946	0.374	0.273	0.079
flat200-94	8.945	1.038	3.754	0.581	0.162
flat200-99	42.392	28.262	2.514	0.775	0.074
统计结果	735.742	199.377	54.782	40.615	3.04

我们将实验的证明继续扩大化,选择了 AIM 类问题.该类问题结构性很强,变量个数以及正负文字比例存在一定规律,解的个数只有一个,求解难度较高.1_6 和 2_0 代表子句与变量的比率,测试用例的变量为 100 个.所以,我们将运行 1 200s 未结束的测试用例定为 Time out.

从表 3 可以看出:在前 4 个测试用例中,并没有拉开 PERACC 和 ERACC 的差距,g2wsat 在规定时间内未找到解,SWCC 的数据两极分化比较严重;后 4 个测试用例中,仍然体现出了 PERACC 的高效性,并且大大地拉开了与 g2wsat 的求解差距.虽然在多半用例,我们仍然无法超越 SWCC.我们分析:对于结构性较强的 AIM 类问题,局部搜索求解结构 SAT 问题并不占优势.对于少数测试结果,PERACC 算法超过了 SWCC 求解器,这在扩展规则领域是史无前例的,从中也看出扩展规则方法在求解 SAT 问题上的巨大潜力.

Table 3 Experiment results on AIM instances (CPU time/s)

表 3 AIM 类问题测试实验结果(CPU time/s)

Problem	ERACC	PERACC	SWCC	g2wsat
aim-100-1_6-yes1-1	1 078.164	957.043	345.605	Time out
aim-100-1_6-yes1-2	933.439	895.485	338.992	Time out
aim-100-1_6-yes1-3	801.674	821.801	0.018	Time out
aim-100-1_6-yes1-4	753.224	740.931	0.017	Time out
aim-100-2_0-yes1-1	45.142	1.876 76	0.015	45.463
aim-100-2_0-yes1-2	21.203	1.033 97	160.536	17.531
aim-100-2_0-yes1-3	6.438	1.375 56	2.151	10.831
aim-100-2_0-yes1-4	16.321	0.151 98	3.362	235.010 6

以上实验结果表明:PERACC 算法在求解复杂性较高、规模较大的测试用例时,更能凸显 PERACC 的高效性能,同时缩短了与 SWCC 求解器的差距,得到了比较良好的效果.

基于扩展规则的推理方式本身是与归结方法截然相反的求解过程,它是从问题的解空间入手来寻求答案.基于扩展规则的 ERACC 算法相比于成熟的局部搜索求解器确有不足,因此,我们在原有基础上做出了一系列改进,进一步研究扩展规则的适用性.我们相信,虽然扩展规则推理方法还处于初级阶段,但与不完备的局部搜索方法相结合,必将发挥出它更大的潜力.

4 总结与展望

本文基于 ERACC 算法提出了并行框架,提出了 PERACC 算法,我们先后提出了预处理、化简解空间等技术,并行处理各个子空间.通过实验显示:该算法较原算法的求解效率有较大的提高,并不断缩短了与国际著名算法 SWCC 的差距.

PERACC 算法在原有算法的基础上进一步大胆地尝试了扩展规则方法领域中不完备推理的发展.下一步,我们决定将 DPLL 算法中的单文子传播机制加入其中,从而进一步完善预处理的效果.另一方面,我们将知识约简方法应用于并行前和并行后,通过对复杂问题约简手段的不断提高,才能让我们去不断地求解复杂性更高、规模更大的测试用例.具体的方法还需要进一步进行理论分析和大量的实验来证明想法的可行性,合理与 PERACC 结合,使算法的求解能力得到最大化的提升.

References:

[1] Selman B, Levesque HJ, Mitchell DG. A new method for solving hard satisfiability problems. In: Proc. of the 10th National Conf. on Artificial Intelligence. 1992. 400−446.

[2] Li CM, Huang WQ. Diversification and determinism in local search for satisfiability. In: Proc. of the 8th Int'l Conf. on Theory and Applications of Satisfiability Testing. 2005. 158−172.

[3] Li CM, Li Y. Satisfying versus falsifying in local search for satisfiability. In: Proc. of the 15th Int'l Conf. on Theory and Applications of Satisfiability Testing. 2012. 477−478.

[4] KhudaBukhsh AR, Xu L, Hoos HH, Leyton-Brown K. SATenstein: Automatically building local search SAT solvers from components. Artificial Intelligence, 2016,232(4):20−42.

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

[6] Cai SW, Su KL. Local search for Boolean satisfiability with configuration checking and subscore. Artificial Intelligence, 2013, 204(9):75−98.

[7] Luo C, Cai SW, Su KL, Wu W. Clause states based configuration checking in local search for satisfiability. IEEE Trans. on Cybernetics, 2015,45(5):1014−1027.

[8] Luo C, Cai SW, Wu W, Jie Z, Su KL. CCLS: An efficient local search algorithm for weighted maximum satisfiability. IEEE Trans. on Computers, 2015,64(7):1830−1843.

[9] Cai SW, Jie Z, Su KL. An effective variable selection heuristic in SLS for weighted Max-2-SAT. Journal of Heuristics, 2015,21(3): 433−456.

[10] Cai SW, Zhang XD. ReasonLS. In: Proc. of the SAT COMPETITION 2018 Solver and Benchmark Descriptions. 2018. 52−53.

[11] Lin H, Sun JG, Zhang YM. Theorem proving based on the extension rule. Journal of Automated Reasoning, 2003,31(1):11−21.

[12] Wu X, Sun JG, Lü S, Li Y, Meng W, Yin MH. Improved propositional extension rule. In: Proc. of the 1st Int'l Conf. on Rough Sets and Knowledge Technology. 2006. 592−597.

[13] Sun JG, Li Y, Zhu XJ, Lü S. A novel theorem proving algorithm based on extension rule. Journal of Computer Research and Development, 2009,46(1):9−14 (in Chinese with English abstract).

[14] Zhang LM, Ouyang DT, Bai HT. Theorem proving algorithm based on semi-extension rule. Journal of Computer Research and Development, 2010,47(9):1522−1529 (in Chinese with English abstract).

[15] Yin MH, Sun JG, Lin H, Wu X. Possibilistic extension rules for reasoning and knowledge compilation. Ruan Jian Xue Bao/Journal of Software, 2010,20(11):2826−2837 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3690.htm [doi: 10.3724/SP.J.1001.2010.03690]

[16] Liu L, Niu DD, Lü S. Knowledge compilation methods based on the hyper extension rule. Chinese Journal of Computers, 2016, 39(8):1681−1696 (in Chinese with English abstract).

[17] 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]

[18] Yang Y, Liu L, Li GL, Zhang TB, Lü S. A novel local search-based extension rule reasoning method. Chinese Journal of Computers, 2018,14(4):825−839 (in Chinese with English abstract).

[19] Yang Y. Research on the reasoning methods using the extension rule [MS. Thesis]. Changchun: Jilin University, 2017 (in Chinese with English abstract).

[20] Li Y, Sun JG, Wu X, Zhu XJ. Extension rule algorithms based on IMOM and IBOHM heuristics strategies. Ruan Jian Xue Bao/ Journal of Software, 2009,20(6):1521−1527 (in Chinese with English abstract). http://www.jos. org.cn/1000-9825/3420.htm [doi: 10.3724/SP.J.1001.2009.03420]

[21] Zhang LM, Ouyang DT, Zhao J, Bai HT. The parallel theorem proving algorithm based on semi-extension rule. Applied Mathematics & Information Sciences, 2012,6(1):119−122.

[22] Li Z. Studies on some issues of satisfiability problems [Ph.D. Thesis]. Changchun: Jilin University, 2020 (in Chinese with English abstract).

附中文参考文献:

[13] 孙吉贵,李莹,朱兴军,吕帅.一种新的基于扩展规则的定理证明算法.计算机研究与发展,2009,46(1):9−14.

[14] 张立明,欧阳丹彤,白洪涛.基于半扩展规则的定理证明方法.计算机研究与发展,2010,47(9):1522−1529.

[15] 殷明浩,孙吉贵,林海,吴瑕.可能性扩展规则的推理和知识编译.软件学报,2010,20(11):2826−2837. http://www.jos.org.cn/1000-9825/3690.htm [doi: 10.3724/SP.J.1001.2010.03690]

[16] 刘磊,牛当当,吕帅.基于超扩展规则的知识编译方法.计算机学报,2016,39(8):1681−1696.

[17] 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.tm [doi: 10.13328/j.cnki.jos.005298]

[18] 杨洋,刘磊,李广力,张桐搏,吕帅.一种新的基于局部搜索的扩展规则推理方法.计算机学报,2018,14(4):825−839.

[19] 杨洋.基于扩展规则的推理方法研究[硕士学位论文].长春:吉林大学,2017.

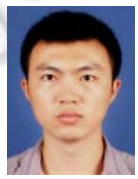
[20] 李莹,孙吉贵,吴瑕,朱兴军.基于 IMOM 和 IBOHM 启发式策略的扩展规则算法.软件学报,2009,20(6):1521−1527. http://www.jos.org.cn/1000-9825/3420.htm [doi: 10.3724/SP.J.1001.2009.03420]

[22] 李壮.可满足性问题的相关问题研究[博士学位论文].长春:吉林大学,2020.

李壮(1988－),男,博士,主要研究领域为人工智能,自动推理,机器学习.

周文博(1991－),男,博士,CCF 学生会员,主要研究领域为形式化方法,云计算.

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,机器学习,自动推理.

张桐搏(1995－),男,硕士,主要研究领域为人工智能,自动推理,云计算.