一种新的基于局部搜索的扩展规则推理方法

杨 洋[1] 刘 磊[1] 李广力[1] 张桐搏[1] 吕 帅[1),2),3)]

[1)](吉林大学计算机科学与技术学院 长春 130012)

[2)](吉林大学数学学院 长春 130012)

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

摘 要 作为与归结推理方法互补的推理方法,扩展规则推理方法得到了国内外广泛认可.目前扩展规则推理方法的研究主要集中于完备推理方法,由于极大项变换方式过于机械,导致该方法处理公式的规模比较局限.该文在深入分析扩展规则推理和局部搜索关联性的基础上,利用子句集中的"极大项"概念设计了一种反向求解策略,进而设计并实现一种新的基于局部搜索的扩展规则推理框架.在此基础上,为了使得极大项搜索过程更加适配该框架,提出了一种基于精确格局检测实现和双向半扩展规则策略的两阶段局部搜索算法.实验结果表明,该文提出的基于精确格局检测的新型扩展规则推理算法ERACC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,使得扩展规则推理方法不再受公式规模制约,可以用于知识编译和可能性推理等多方面的应用.

关键词 自动推理;扩展规则;局部搜索;精确格局检测;双向半扩展规则;反向求解

中图法分类号 TP18 **DOI号** 10.11897/SP.J.1016.2018.00825

A Novel Local Search-Based Extension Rule Reasoning Method

YANG Yang[1] LIU Lei[1] LI Guang-Li[1] ZHANG Tong-Bo[1] LÜ Shuai[1),2),3)]

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

[2)](*College of Mathematics, Jilin University, Changchun* 130012)

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

Abstract Extension rule was proposed by Lin et al. in 2003, which is a new rule in automated reasoning. As a "complementary" reasoning method with resolution by Martin Davis, an expert in artificial intelligence, extension rule-based reasoning methods have been widely recognized around the world. Lots of excellent achievements which related to extension rule were proposed in succession, such as knowledge compilation, possibility reasoning and model counting methods. Especially in knowledge compilation, a theory called EPCCL (each pair contains complementary literal), which was the result of extension rule-based knowledge compilation method, was highly appreciated by Murray and Rosenthal. Throughout the achievements related to extension rule, the main researches on extension rule are among the aspects of complete reasoning methods, and researches among incomplete reasoning methods were rare. The original algorithm of extension rule is high memory consumption owing to the process of inclusion-exclusion principle, which

收稿日期:2016-04-25;在线出版日期:2017-01-09.本课题得到国家自然科学基金(61300049,61502197,61503044)、教育部高等学校博士学科点专项科研基金(20120061120059)、吉林省重点科技攻关项目(20130206052GX)、吉林省青年科研基金项目(20140520069JH,20150520058JH)、吉林省自然科学基金项目(20150101054JC,20180101053JC)资助.**杨 洋**,男,1992年生,硕士,中国计算机学会(CCF)学生会员,主要研究方向为智能规划与自动推理.E-mail:yang_freezing@sina.com.**刘 磊**,男,1960年生,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为软件理论与技术.**李广力**,男,1992年生,硕士研究生,中国计算机学会(CCF)学生会员,主要研究方向为自动推理与云计算.**张桐博**,男,1995年生,硕士研究生,中国计算机学会(CCF)学生会员,主要研究方向为智能规划与自动推理.**吕 帅**(通信作者),男,1981年生,博士,副教授,中国计算机学会(CCF)高级会员,中国计算机学会(CCF)理论计算机科学专业委员会委员,中国计算机学会(CCF)形式化方法专业组委员,中国计算机学会(CCF)教育专业委员会委员,主要研究方向为智能规划与自动推理.E-mail:lus@jlu.edu.cn.

results in being replaced by NER (novel extension rule). Transforming mode of the maximum term is the key of NER, and transforming in order guarantees the completeness of NER. Owing to the unintelligent transforming mode of the maximum term in NER, the scale of formulae it can handle are still very small. In this paper, after analyzing the relevance of extension rule and local search, the concept of "maximum term" in logic is utilized, which aimed at designing and implementing the novel reasoning framework of extension rule based on local search. In order to accelerate the computing in greedy mode, a method for computing scores based on different sets is proposed. On this basis, to make the searching process of maximum terms more suitable for the framework, two implementations of accurately configuration checking for different scale of formulae are proposed, which are based on the proposal of algorithm SWCC (smoothed weighting with configuration checking). After that, for further limitation of selecting variables, a new strategy called double-sided semi-extension rule is proposed, whose inspiration comes from SER(semi-extension rule) in 2009. Based on the framework and heuristics mentioned above, the first incomplete reasoning algorithm in extension rule called ERACC (extension rule based on accurate configuration checking) is designed and implemented. The experimental results show that the algorithm ERACC greatly breakthrough the bottleneck of performance in the reasoning methods related to extension rule, and get much improved in the ability of processing formulae with large scale. In the problems of random instances and structured instances, algorithm ERACC outperforms several existed reasoning methods based on extension rule. In addition, the robustness of ERACC has shown among the benchmarks used above. The comparison with state-of-the-art solvers in performance shows that ERACC is still not better than state-of-the-art solvers, but it also has comparative performance with some of them, which shows the great potential of algorithm ERACC. The proposal of ERACC is the first step of the researches of incomplete reasoning framework, and it shows the meaning for exploring more powerful heuristics to make the framework much stronger. This framework makes the extension rule no longer restricted by the scale of formulae, and it can also be used in the field of knowledge compilation and possibility reasoning.

Keywords automated reasoning; extension rule; local search; accurate configuration checking; double-sided semi-extension rule; reverse solving

1 引 言

可满足性问题(Satisfiability Problem,SAT)是一类著名的 NP 完全问题,局部搜索方法是 SAT 问题高效的求解方法,通过随机产生一个命题变量集合空间上的真值指派,利用公式特定信息进行变量翻转,达到不断向可满足解靠近的目的.1992 年,Selman 等人[1]提出的 GSAT 算法展示了其在 SAT 问题上强大的求解能力,有力地证明了局部搜索算法可以用于求解 SAT 问题,奠定了局部搜索用于可满足性判定的基础.

然而,局部搜索算法通常会导致求解过程陷入局部最优的死循环中,即在一部分候选解空间上反复迭代,始终无法找到问题的最优解.为了解决该问题,众多学者在该领域相继提出了许多杰出的研究成果.禁忌搜索是一种著名的控制策略,该策略被 Mazure 等人[2-5]广泛用于局部搜索算法中,通过设定一个阈值来禁止最近一段搜索过程发生循环.由于 SAT 比赛的蓬勃发展,陆续出现了一批强大而实用的局部搜索求解器,在随机测试用例上展现了强劲的求解能力.

2002 年,Hutter 等人[6]通过降低权重更新复杂度优化 ESG(Exponentiated Sub-Gradient)算法,设计并实现了能够自适应调参的动态局部搜索算法 RSAPS(Reactive version of Scaling And Probabilistic Smoothing),战胜了 ESG 和 Novelty+算法.同年,Hoos[7]为 WalkSAT 类局部搜索算法设计了一种自适应调节噪声参数的策略,使得多种 WalkSAT 算法的变体在处理不同问题时能够接近手动

调参时的峰值性能. 2004 年，Thornton 等人[8]利用加法子句权重变更模式设计了 PAWS 算法，在多个测试问题上战胜了乘法子句权重策略. 2005 年，Li 等人[9]设计了利用贪心结合随机选择的双模式进行搜索的 G2WSAT 算法，展现出了良好的性能. 同年，Prestwich[10]设计了一种高效的变量加权策略 VM，有力地保障了 WalkSAT 类算法的性能，获得了 SAT'2005 的铜奖；Pham 等人设计出的 Ranov 求解器①通过预处理阶段的归结方法优化了子句结构，获得了 SAT'2005 的金奖. Li 等人将自适应噪声机制和 g2wsat 算法良好结合，提出了 adaptg2wsat 类算法②，展现了强劲的性能. 随后，其设计了一种向前一步看策略，集成到 adaptg2wsat 类算法中[11]，战胜了 g2wsat 算法，与当前主流求解器的求解能力相当. 其中，adaptg2wsat0 和 adaptg2wsat＋分别获得了SAT'2007的银奖和铜奖. 2007 年，Pham 等人[12]在借鉴 AdaptNovelty＋和 g2wsat 算法的基础上，混合两种子句加权启发式策略，提出了 gNovelty＋求解器，在随机测试用例集上获得了 SAT'2007 的金奖. Balint 等人[13]设计的 Sparrow 算法利用弄假子句中变量的概率分布属性用于指导随机模式下翻转变量的选择，在 3-SAT 测试用例上战胜了 SAT'2009 中的众多求解器. 2011 年，改进的 Sparrow2011 获得了 SAT'2011 的金奖；Li 等人[14]提出了一种利用变量弄真历史记录的局部搜索求解器 SatTime，获得了 SAT'2011 的银奖. 2016 年，KhudaBukhsh 等人[15]提出了一个能够结合多个高性能局部搜索求解器的 SATenstein 求解框架和自动配置程序，战胜了众多主流求解器.

近年来，由蔡少伟等人提出的格局检测策略在局部搜索领域崭露头角，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，蔡少伟等人[16]首次将格局检测策略用于 SAT 求解中的局部搜索，设计的 SWCC 取得了惊人的成绩. 随后，蔡少伟等人在基于"鼓励刺激"机制、子句状态检测、双状态格局检测以及双得分函数等策略的基础上，陆续设计了 SWCCA[17]、CCASat[18]、CScoreSAT[19]、CCAnr[20]、CSCCSat[21]和 DCCASat[22]等强劲的求解器. 其中，CCASat 获得了 SAT'2012 的冠军，CSCCSat 获得了 2014 年 SAT 比赛的铜奖. 同时，其将格局检测用于求解 Max-SAT 等问题，取得了巨大的成功[23-24]，CCLS 在 MAX-SAT'2013 中在非完备性求解器组中取得了 4 项第 1. 格局检测策略使得 SAT 局部搜索算法的性能得到了巨大的提升.

目前国际上主流的局部搜索算法大多基于 DPLL(Davis-Putnam-Logemann-Loveland)和归结方法，真值指派是其求解问题的核心. 2003 年，Lin 等人[25]提出了一种全新的推理方法——扩展规则推理方法，其通过不断搜索给定子句集的极大项空间来判定当前公式的可满足性. 该方法被人工智能专家 Martin Davis 誉为与归结方法互补的推理方法. 基于扩展规则推理方法，Lin 等人[26]在 2004 年提出了一种新型的知识编译方法 KCER. 任意给定的命题子句集在经过 KCER 处理后所得到的公式集称为 EPCCL(Each Pair Contains Complementary Literal)理论，得到了非子句归结的提出者 Murray 教授[27]的高度赞扬，他认为该理论的应用能够得到令人吃惊的成果并且认为该方法是一种完全不同于现有知识编译方法的全新方法. 2009 年，李莹等人[28]设计了一种新型基于扩展规则的推理方法 NER，使用时间复杂度换取空间复杂度，巧妙地解决了内存限制的问题. 之后，李莹等人[29]针对 IER 算法设计了 IMOM 和 IBOHM 两种启发式策略，更加科学地对搜索子空间进行选取. 同时，张立明等人[30-32]设计并实现了改进的扩展规则推理方法——半扩展规则推理方法，并针对其执行特性设计了相应的并行算法和分解定理证明方法. 许有军[33]在其博士论文中提出了一种新的结合 DPLL 分裂规则的扩展规则推理方法，并且设计了 MOAMD 启发式策略用于指导分裂变量的选择顺序，在 ER 的基础上将效率提升了 1～2 个数量级. 在扩展规则推理方法的应用方面，殷明浩、刘磊和赖永等人分别将扩展规则推理用于可能性推理[34]、#SAT 求解[35]、知识编译[36]和模型计数[37]等领域，对于特定的公式集取得了较好的推理效果.

SAT 问题的通用求解方法包括完备方法和不完备方法. 扩展规则推理的理论基础表面上只给出了一种新的完备推理算法的求解思路，但暗含着不完备的求解思路. 事实上，现有与扩展规则相关的研究成果几乎全部集中或基于完备推理算法，不完备推理的相关成果几乎空白，而不完备推理却恰恰是处理特定公式集的强大手段. 局部搜索思想完全可

① Pham D N, Anbulagan A. Resolution enhanced SLS solver: R+AdaptNovelty+. solver description. SAT Competition, 2007. http://www.satcompetition.org/2007/ranov.pdf

② Li C M, Wei W, Zhang H. Combining adaptive noise and promising decreasing variables in local search for SAT. SAT Competition, 2009. http://www.satcompetition.org/2007/adaptG2WSAT.pdf

以应用到极大项概念上,从而得到更加高效的极大项变换方式,达到加快问题求解的目的.

本文将结合格局检测设计基于局部搜索的新型扩展规则推理方法.扩展规则推理方法有着自己独特的适用性和求解优势,本文拟在深入研究扩展规则推理方法求解问题特性的基础上,设计并实现基于局部搜索的扩展规则推理框架.随后,结合现有的局部搜索策略以及本文提出的策略,给出朴素的基于局部搜索的新型扩展规则推理算法,希望能够进一步提升扩展规则的执行效率.

2 基础知识

2.1 局部搜索

定义 1. 给定一组布尔变量集 $V=\{x_1,\cdots,x_n\}$,一个文字为变量 x 或者其否定 $\neg x$.一个 CNF(Conjunctive Normal Form)公式为一组子句集的合取(命题公式的一种标准型),找到一个或多个能够使得该 CNF 公式为真的完全真值指派的过程称为 SAT 求解.

SAT 问题的求解可以概括为对于给定命题变量集上的公式,找到一组部分或者全部变量的真值指派使得所有公式可满足的过程.完备算法的核心思想在于按照某种特定的指派规则对变量集所有真值指派的可能性进行选取或剪枝,判断公式集是否可满足.而局部搜索却采取了一种完全不同的思路来进行问题的求解,该策略通过不断改变所有变量集的真值指派使得可满足的子句数向不断增加的方向进行,最终达到所有子句可满足的目的.对于不可满足的公式而言,因为永远无法找到一组变量集的真值指派使得公式集可满足,算法会陷入无限循环.因此,局部搜索只能判定可满足的公式集,无法对不可满足的公式集进行判定.

2.2 扩展规则

扩展规则推理方法主要利用了逻辑学中的"极大项"概念进行公式求解.基于扩展规则的定理证明方法的理论基础为:给定命题变量上的所有极大项组成的子句集不可满足.下面首先给出扩展规则的基本定义.

定义 2[25]. 给定一个子句 C 和不在 C 中出现的原子 a,$D=\{C\vee a, C\vee\neg a\}$,称由 C 到 D 的推导过程为扩展规则,D 为子句 C 关于原子 a 应用扩展规则的结果.

显然,公式集 D 在真值指派上和公式集 C 逻辑等价.因此,可以对子句集中的任意子句应用类似上述的推导过程,逐步扩展成只包含形式为极大项的子句集.朴素的扩展规则推理[25]只是利用极大项的概念对子句集进行模型计数,通过容斥原理计算子句集能够扩展出的极大项个数,从而达到判断子句集是否可满足的目的.该方法受限于子句的个数,一旦子句数过大将造成内存溢出,导致程序崩溃.李莹等人提出的 NER[28]则采取了使用命题变量集上的极大项空间来判断能否全被子句集扩展出来,从而达到判断子句集是否可满足的目的.该方法事实上是机械地变换极大项来对公式集进行判定,单纯地追求完备性而采用了暴力求解的方式,损失了过多的启发式信息,从而导致时间复杂度过高.而且,该方法受限于变量个数,与变量个数呈指数级对应关系,一旦变量数过大,会导致求解时间过长.

扩展规则类推理方法的本质在于:在给定变量集的极大项空间上寻找不可扩展的极大项.而这一搜索过程从一定角度上可以借鉴局部搜索方法的搜索过程,即:利用贪心策略等限定极大项的搜索空间,减少扩展过程中无用的搜索过程,从而提升扩展规则类推理方法的性能.

3 基于局部搜索的扩展规则推理框架

基于局部搜索的扩展规则推理框架的核心思想是给出一种更加高效的极大项变换方式,在牺牲完备性的前提下得到一种高效处理方法.

令待判定的子句集为 F,C_i 为子句集 F 中任意一个子句.其中一个重要的操作是使用给定极大项对子句集 F 中的子句进行判定能否被扩展,在 NER 中定义了该判定函数为 CanBeExpand.

令当前极大项 $T=\{l_1,\cdots,l_n\}$,其中 $l_k(1\leq k\leq n)$ 表示该极大项中的文字,令子句 $C_i=\{l'_1,\cdots,l'_k\}$,函数 CanBeExpand 判定极大项 T 能否被子句 C_i 扩展.文献[28]中对该操作的时间复杂度的分析为 $O(k^2)$,即为判定文字集合的包含关系的时间复杂度,其中 k 为子句的平均长度.

而在本文设计的算法里,通过对原子句集和极大项重新进行信息编码和数据结构优化,优化了该基本操作.主要思想是将原 CNF 公式的命题变量集进行编号,将每一个极大项按照文字对应变量编号的大小进行有序存储,并且将原子句集进行编号存储.通过该处理,可以将基本判断操作由原先的 $O(k^2)$ 的时间复杂度降低为 $O(k)$.在多次迭代进行

极大项变换的搜索框架中，把某一操作降低一个数量级将对于求解效率有着较大的提高.

对 CanBeExpand 函数进行优化后，下面给出朴素的基于局部搜索的扩展规则推理框架. 下述仅仅是局部搜索思想在扩展规则推理中的执行框架，通过更换算法中的 Transform 函数模块可以得到一系列衍生算法.

算法 1.　基于局部搜索的扩展规则推理框架 LSER(Local Search in Extension Rule).

输入：CNF 公式 $F=\{C_1,\cdots,C_m\}$，
　　当前极大项 $T=\{l_1,\cdots,l_n\}$
输出：不可扩展的极大项 $T'=\{l'_1,\cdots,l'_n\}$ 或
　　最大尝试次数 $MaxTries$

```
1.  初始化：当前变换次数 current←0；MaxTries←C
2.  T'←T
3.  WHILE current<MaxTries DO
4.    IF CanBeExpand(T',F)
5.    THEN T'←Transform(T')
6.         current++
7.    ELSE
8.         RETURN T'
9.  RETURN MaxTries
```

算法 1 中 $Transform$ 函数的功能为对当前极大项 T' 进行变换，形成新的极大项. 算法 1 给出了基于局部搜索的扩展规则推理框架，核心本质在于极大项的变换. 而实际上该推理框架将完备算法和不完备算法进行了统一，唯一的区别在于：完备算法在进行极大项变换时，不利用当前极大项与子句集的包含扩展信息，只是保留一个判定函数返回的布尔值从而机械地进行极大项变换，极大地降低了推理的效率；而不完备算法则充分地利用了这些信息，把最大限度地向解空间靠近作为唯一的目标.

定理 1.　算法 1 在最好情况下的时间复杂度为 $O(m\times k)$，在最坏情况下的时间复杂度为 $O((m\times k+C)\times M)$，所需要的空间复杂度为 $O(n)$. 其中：m 为输入公式的子句数，k 为子句的平均长度，M 为极大项的最大变换次数，C 为进行一次变换所需要的时间开销，n 为变量数.

证明.　算法 1 执行过程中，最好情况为：初始极大项就不能被所有子句集扩展出来，并其在对每个子句进行包含扩展判定时每个子句的第一个文字和当前极大项中的文字不同. 因此，对于每个子句可以在 $O(1)$ 的时间复杂度内完成判定，最优情况下的时间复杂度为 $O(m)$. 而最坏情况则对应着经过最大变换次数后的尝试，极大项仍能被子句集中某一子句扩展出来. 实际上，只要当前极大项被子句集中某一子句扩展出来，就不用再做后续判定，而直接进行变换操作. 纯暴力的搜索方法并没有最大限度地利用当前极大项的信息. 本文利用了上述信息，需要计算当前极大项和所有子句的差集，因此，不管当前极大项能否被某一子句扩展出来，都会继续判定后续子句. 最坏情况下每一次产生的极大项都需要和每个子句的每个文字进行比较，并且经历了 M 次变换. 因此，最坏情况下的时间复杂度为 $O((m\times k+C)\times M)$. 同理，为了计算差集，最优时间复杂度为 $O(m\times k)$. 在任何情况下，算法 1 所需要的空间复杂度仅为保存一个极大项的空间，即 $O(n)$.

上述仅给出了基本的搜索框架，将局部搜索的思想运用到极大项的变换过程中，完全不同于经典扩展规则推理的搜索框架. 该框架的核心本质在于 Transform 函数，即极大项的变换方式. 如何高效地进行极大项的变换，是算法取得性能优胜的关键，下文将对变换的策略进行进一步描述.

4　基于极大项的启发式策略

4.1　基于极大项的贪心策略

贪心策略是局部搜索中一种主要的策略，基本的贪心模式为：在翻转变量真值时，寻找某个能够使得“翻转后弄假子句的权值之和最大程度减小”的变量，但是该策略仅限于真值指派类算法. 在基于局部搜索的扩展规则推理框架中，由于完全不同的判定模式，以往的贪心策略将不再适用，因此需要设计一种基于极大项的贪心策略来适用于基于局部搜索的扩展规则推理框架.

令 F 为 CNF 公式，T 为当前极大项，W_i 为子句 C_i 相关联的权值，$|F|=k$，则公式 F 下的当前极大项 T 的增益函数为：$Cost(F,T)=\sum_{i=1}^{k}(W_i\cdot\delta(T,C_i))$，其中 δ 为二元谓词特征函数(双变量判定函数，求值返回 true 或 false)，具体定义为

$$\delta(T,C_i)=\begin{cases}1, & \text{当极大项 } T \text{ 能被子句 } C_i \text{ 扩展出来}\\ 0, & \text{否则}\end{cases}.$$

基于上述特征函数，可以计算对特定文字取否定来进行极大项变换的增益函数. 令当前即将取否定的文字对应变量为 v，则 v 的得分函数为

$$Score(v)=Cost(F,T)-Cost(F,T'),$$

其中 T' 表示对变量 v 在当前极大项 T 中对应文字取否定之后生成的新极大项.

基于上述公式,下面给出朴素的基于极大项的贪心搜索算法.在该策略下,每次极大项变换时,根据变量的得分函数选取一个得分最高的变量相应的文字进行变换,从而产生一个新极大项来进行下一次判定.

算法 2. 基于极大项的贪心搜索算法(Greedy Search Based on Maximum Term).

输入:CNF 公式 $F=\{C_1,\cdots,C_m\}$,
当前极大项 $T=\{l_1,\cdots,l_n\}$

输出:得分函数 Score 最高的变量 v 对应的得分

```
1.  初始化得分集合 s[]←{-1}, i←0
2.  WHILE i<n DO
3.    T'←T/ref(T,i)⋃¬ref(T,i)
4.    s[i]←Cost(F,T)-Cost(F,T'))
5.    T←T'/ref(T',i)⋃¬ref(T',i)
6.    i++
7.  v←getMax(s)
8.  RETURN s[v]
```

其中,$ref(T,i)$表示当前极大项 T 中变量 i 对应的文字.算法 2 实际上仅给出了极大项变换策略中的计算函数,其效率将直接影响整个框架的执行效率,接下来详细描述其实现细节.

在对选取的变量 v 对应的文字取否定操作之后,会生成新的极大项,关于每个变量的得分 $Score(v)$ 都会发生变化.

通常的做法是:每次生成一个新的极大项,都重新扫描整个子句集,然后根据取否定之后的特性结合是否能包含扩展,重新计算整个变量集的得分集合.在公式规模较小时,进行这样的计算不会造成太大的时间开销,但当变量数上升至几千、子句集个数上升至几万后,扫描一次的开销过大.因此,如何采取高效的得分集合计算方式将对算法执行的时间复杂度有着至关重要的影响.

本文采取了一种基于差集的计算方式来辅助计算各个变量的得分集合.

令 F 为 CNF 公式,当前极大项 $T=\{l_1,\cdots,l_n\}$,子句集中任意子句 $C_i=\{l'_1,\cdots,l'_k\}$,W_i 为子句 C_i 相关联的权值,$|F|=k$,$D_i=C_i-T$.则变量 v 基于差集的得分函数为:$Score(v)=\sum_{i=1}^{k}\delta(D_i,v)\times W_i$.

在该公式中,δ 为一种特殊的二元谓词特征函数,表现为多值逻辑的取值,即根据多种不同的情况判定得到几种相异的值.令$\{v\}$表示变量 v 的肯定或者否定文字,$ref(T,v)$表示当前极大项 T 中变量 v 对应的文字.δ 的具体定义如下:

$$\delta(D_i,v)=\begin{cases}0, & |D_i|>1 \text{ 或}(|D_i|=1 \text{ 且}\{v\}\notin D_i\\ 1, & |D_i|=0 \text{ 且 } ref(T,v)\in C_i\\ -1, & |D_i|=1 \text{ 且}\neg ref(T,v)\in D_i\\ 0, & |D_i|=0 \text{ 且}\{v\}\notin C_i\end{cases}.$$

该公式给出了基于差集的得分集合的计算方式,基本思想为:根据差集的元素个数来判定当前子句的权值在对当前极大项中某一文字取否定之后的得分值是否有贡献.因此,在对当前极大项某一文字取否定后,无需重新计算所有变量的得分集合,只需根据每一子句计算相应的差集元素的个数,即可得出所有变量的得分集合.

例 1. 令 CNF 公式 $F=\{A\vee B\vee C,\neg A\vee\neg D,\neg B\vee C\vee D,A\vee D,\neg C\}$,变量集 $V=\{A,B,C,D\}$,当前极大项为 $T=\{A,\neg B,C,D\}$,每个子句相关联的权值为 1.则每个子句与当前极大项的差集分别为:$D_1=\{B\}$,$D_2=\{\neg A,\neg D\}$,$D_3=\{\}$,$D_4=\{\}$,$D_5=\{\neg C\}$.根据这些差集可以计算出任意一个变量的得分:

$Score(A)=0\times1+0\times1+0\times1+1\times1+0\times1=1$;

$Score(B)=(-1)\times1+0\times1+1\times1+0\times1+0\times1=0$;

$Score(C)=0\times1+0\times1+1\times1+0\times1+(-1)\times1=0$;

$Score(D)=0\times1+0\times1+1\times1+1\times1+0\times1=2$.

由例 1 可以看出,仅仅扫描一遍子句集即可计算出所有变量的得分,而在选取下一阶段要取否定的文字时,即可选取得分最高的变量(例 1 中为 D).对在当前极大项中的对应文字取否定操作,从而得到一个新极大项来继续对原子句集进行包含扩展操作.

4.2 格局检测的精确实现

定义 3[16]. 给定一个 CNF 公式 F 和一个命题变量集 $V(F)$上的真值指派 s,一个变量 $x\in V(F)$ 的格局为一个状态向量,该状态向量包含了变量 x 的邻居变量(与变量 x 出现在同一子句中的变量)的真值指派.

格局概念的提出是防止在传统局部搜索算法中运用贪心策略的过程中陷入局部循环的无限搜索.在基于局部搜索的扩展规则推理框架中,由于不存在真值指派这一概念,执行的过程中是通过极大项包含扩展来判定,因此需要对格局检测的定义稍作调整.

定义 4. 给定一个 CNF 公式 F 和一个命题变量集 $V(F)$上的极大项 T,一个变量 $x\in V(F)$ 的格局为一个文字集合,该集合包含了当前极大项 T 中

变量 x 的邻居变量的相应文字.

本节详细讨论格局检测在本文中的实现.

蔡少伟等人[16]提出的基于格局检测的局部搜索算法中,并没有严格按照格局检测的定义进行算法实现,而是采取了一种近似的实现策略来降低格局检测这一操作的时间复杂度.其没有对格局检测进行精确实现的原因在于:其认为线性复杂度的检测过程代价过大.在算法的实际执行过程中,其采用的 $conf$ 数组和近似实现策略可以将检测一个特定变量的格局信息的时间复杂度降低到 $O(1)$,原因在于其进行变量格局检测时只需要判断该变量对应数组的标志位,而且存储的空间复杂度也得到了大大降低,牺牲的是格局检测的精确性.

本文针对格局检测这一策略在不增加时间复杂度的情况下进行了精确实现,下面给出精确格局检测的具体实现策略.

实际测试中,变量格局信息的存储空间并不会成为太大的负担(在现有的硬件配置水平和大部分 benchmark 规模下).进行格局检测这一策略的关键实现步骤在于检测某个特定变量的格局信息.在原算法实现中,只是进行了检测某一数组的标志位,而其精确实现需要进行线性的检测操作,从而导致时间复杂度增加.在深入研究公式特性的基础上,本文拟采用两种办法对格局检测进行精确实现.

4.2.1　适用于小规模公式的实现策略

在基于局部搜索的扩展规则推理框架中,一个变量的格局信息为其邻居变量在当前极大项中对应的文字情况,只有肯定和否定两种可能性.因此,可以将某个特定变量的格局信息看成一个二进制状态向量.首先将变量编码成十进制整数,然后分别将变量的肯定或者否定情况标记成该十进制整数的肯定与否定情况.

假定需要检测格局信息的变量为 v,其邻居变量集合为 $N(v)$,则其邻居变量集合在当前极大项中对应的文字集合为 $LN(v)$.将 $LN(v)$ 集合有序化后,升级为有序数组,则可将其中的文字情况进行编码转化为状态向量 $VLN(v)$.在得到状态向量 $VLN(v)$ 后,基于上述的编码方式将状态向量转化为一个二进制数,通过判断该二进制数的变化情况来进行格局信息的检测.

基于该编码方式,即可设计出初步的精确格局检测实现方法.首先给出实现精确格局检测前的预处理工作.

算法 3.　精确格局检测的预处理(Preprocessing for Accurate Configuration Checking).

输入:CNF 公式 $F=\{C_1,\cdots,C_m\}$,当前极大项 $T=\{l_1,\cdots,l_n\}$

输出:各变量的状态向量 $VLN[]$

　　各变量的初始二进制编码 $Initial\_info[]$

　　各变量的当前二进制编码 $Current\_info[]$

1.　初始化 $VLN[]$为空,$Initial\_info[]\leftarrow\{0\}$,$Current\_info[]\leftarrow\{0\}$,$i\leftarrow 0$
2.　**WHILE** $i<n$ **DO**
3.　　$VLN[i]\leftarrow Scan(F,i,T)$
4.　　$Initial\_info[i]\leftarrow Compute(VLN[i])$
5.　　$Current\_info[i]\leftarrow Initial\_info[i]$
6.　　$i{+}{+}$
7.　**RETURN** ($VLN[]$,$Initial\_info[]$,$Current\_info[]$)

算法 3 中的 $Scan$ 函数为扫描变量 i 在当前 CNF 公式 F 中的邻居变量在当前极大项 T 中的文字集合.算法初始化时已经将文字以及变量集合进行有序处理,因此可以直接得到状态向量 $VLN[]$.$Compute$ 函数为根据状态向量计算出当前变量的二进制数,初始情况下各变量的初始二进制编码与当前二进制编码值相同.

算法 3 给出了在小规模公式中进行精确格局检测实现前的预处理工作,完成了二进制信息编码的工作.下面给出基于二进制编码的精确格局检测算法.

算法 4.　基于二进制编码的精确格局检测(Accurate Configuration Checking based on Binary Encoding)

输入:选定的变量 v,$Initial\_info[]$,$Current\_info[]$

　　邻居集合 $N(v)$,极大项 $T=\{l_1,\cdots,l_n\}$

输出:更新后的极大项 T',$Initial\_info[]$,$Current\_info[]$

1.　$T'\leftarrow T/ref(T,v)\cup\neg ref(T,v)$
2.　$Initial\_info[v]\leftarrow Current\_info[v]$
3.　**FOR** i in $N(v)$
4.　　$l\leftarrow ref(T,v)$
5.　　**IF** $l=v$　**THEN** $Current\_info[i]-{=}2^i$
6.　　**ELSE IF** $l=\neg v$
7.　　**THEN** $Current\_info[i]+{=}2^i$
8.　**RETURN** (T',$Initial\_info[]$,$Current\_info[]$)

事实上,算法 4 给出了一种在该实现方法下更新变量格局信息的一种快速方法,其时间复杂度与原近似实现的方法一样,均为 $O(V(F))$.而在贪心模式下选取变量时,考虑变量的格局信息是否改变的判断标准为判定当前变量的 $Initial\_info[]$ 与 $Current\_info[]$ 是否一致,从而进一步判定该变量

是否被纳入候选.判定变量的格局信息是否改变,显然和原始的近似实现达到了相同的时间复杂度,即为 $O(1)$.因此,该方法作为小规模公式中的格局检测策略的精确实现方法,是理论可行的.

例 2. 类似例 1,$F=\{A\vee B,\neg A\vee\neg D,\neg B\vee C\vee D,A\vee D,\neg C\}$,$T=\{A,\neg B,C,D\}$,$V=\{A,B,C,D\}$.

预处理阶段,各变量的邻居集合为:$N(A)=\{B,D\}$,$N(B)=\{A,C,D\}$,$N(C)=\{B,D\}$,$N(D)=\{A,B,C\}$.

则各变量的邻居集合在当前极大项中的文字集合为:$LN(A)=\{\neg B,D\}$,$LN(B)=\{A,C,D\}$,$LN(C)=\{\neg B,D\}$,$LN(D)=\{A,\neg B,C\}$.

在对各变量对应的文字集合进行有序化二进制编码后,得到每个变量对应的格局信息为:$C(A)=(\{\neg B,D\},8,8)$,$C(B)=(\{A,C,D\},13,13)$,$C(C)=(\{\neg B,D\},8,8)$,$C(D)=(\{A,\neg B,C\},5,5)$.

经过上述编码后,对于每个变量可以得到一个三元组结构,用于格局检测.

假定当前要取否定的为变量 C 对应的文字,则当前极大项变为:$T'=\{A,\neg B,\neg C,D\}$.

下一步,将修改变量 C 所有邻居的格局信息,并且修改 C 变量所有邻居的当前二进制编码.C 变量的邻居集合为$\{B,D\}$,因此修改变量 B 和 D 对应的三元组结构,分别修改有序文字状态和当前二进制编码,修改为:$C(B)=(\{A,\neg C,D\},13,9)$,$C(D)=(\{A,\neg B,\neg C\},5,1)$.

这样即为更新邻居变量对应的三元组信息,下一次进行格局检测时,只需判定三元组中的后两个数是否相等即可得到格局信息是否改变的结论.

上述格局检测策略的精确实现仍然具有一定的局限性.当特定变量的邻居集合的大小一旦超过某个限制,那么其转化成的二进制整数将溢出.因此,必须寻求一种更加通用的方法来实现精确的格局检测策略,以此来适应更大规模的公式.

4.2.2 适用于大规模公式的实现策略

上一小节,本文尝试将变量的格局信息与一个确定编码的二进制数一一对应,而这个二进制数是严格基于变量的状态向量的每一位生成的,在这种编码形式中存在较大的冗余.在进行格局检测时,需要考虑变量是否存在某一邻居的状态发生改变,不改变的邻居信息对于判定该变量是否进入候选毫无意义.因此,需要寻求一种重点考虑特定变量状态向量中被改变的邻居状态的编码方式.

在上一小节的基础上进行修改,不再采取按位编码的方式来保存变量的格局信息,而是拟采用一个计数器来记录特定变量的状态向量中被改变的邻居个数,对于特定变量状态向量中的每一位设置一个标志位来表示该位是否被改变过.下面给出该编码形式下的格局检测策略的精确实现.

算法 5. 基于计数器的精确格局检测(Accurate Configuration Checking based on Counters).

输入:选定的变量 v,*Neighbou_info*[][],*Flag_info*[][],*Counter*[],当前极大项 $T=\{l_1,\cdots,l_n\}$

输出:更新后的极大项 T',*Neighbour_info*[][],*Flag_info*[][],*Counter*[]

```
1.  T' ← T/ref(T,i) ∪ ¬ref(T,i)
2.  Counter[v] ← 0
3.  FOR flag in Flag_info[v]
4.      flag ← false
5.  FOR i in N(v)
6.      IF Flag_info[i][v] ← false
7.      THEN Flag_info[i][v] ← ~Flag_info[i][v])
8.           Neighbou_info[i][v] ←
                 ~Neighbour_info[i][v]
9.           Counter[i]++
10.     ELSE Flag_info[i][v] ← ~Flag_info[i][v]
11.          Neighbou_info[i][v] ←
                 ~Neighbour_info[i][v]
12.          Counter[i]--
13. RETURN (T', Neighbour_info[][], Flag_info[][],
           Counter[])
```

基于算法 5,可以得到大规模公式中精确实现格局检测的方法.在贪心模式下选取变量时,考虑变量的格局信息是否改变的判断标准为判定当前变量的变量计数器是否为 0.如果为 0,则格局信息未发生改变,不纳入候选范围;反之则格局信息发生改变,将其纳入候选范围.

大规模公式中的格局检测策略的精确实现,与小规模公式中的精确实现相比,增加的时间复杂度主要在于:在文字取否定时,需要修改信息的操作.针对该操作,前者需要将当前变量状态向量中的所有标志位复位,而后者只需要简单地将初始二进制编码的值更新为当前二进制编码的值即可.从时间复杂度来看,相比于后者,前者只不过是在对文字取否定后执行了一个线性操作,而实际上在对文字取否定之后还要对该文字的所有邻居变量进行更新操作,因此只是一个线性操作并不会使得算法的时间复杂度增加.

例 3. 类似例 1,$F=\{A\vee B,\neg A\vee\neg D,\neg B\vee C\vee D,A\vee D,\neg C\}$,$T=\{A,\neg B,C,D\}$,$V=\{A,B,C,D\}$.

各变量的邻居集合在当前极大项中的文字集合为：$LN(A)=\{\neg B, D\}$，$LN(B)=\{A, C, D\}$，$LN(C)=\{\neg B, D\}$，$LN(D)=\{A, \neg B, C\}$.

各变量的初始格局信息为

$C(A)=(\{(\neg B, \text{false}), (D, \text{false})\}, 0)$；

$C(B)=(\{(A, \text{false}), (C, \text{false}), (D, \text{false})\}, 0)$；

$C(C)=(\{(\neg B, \text{false}), (D, \text{false})\}, 0)$；

$C(D)=(\{(A, \text{false}), (\neg B, \text{false}), (C, \text{false})\}, 0)$.

经过这种信息编码后，对于每个变量可以得到一个二元组结构，用于格局检测.

假定当前要取否定的为变量 C 对应的文字，则当前极大项变为 $T'=\{A, \neg B, \neg C, D\}$.

下一步，修改变量 C 所有邻居的格局信息，并且修改变量 C 所有邻居的变量计数器. 变量 C 的邻居集合为 $\{B, D\}$，因此修改变量 B 和 D 对应的二元组结构，分别修改有序文字状态、标志位以及变量计数器，修改为：$C(B)=(\{(A, \text{false}), (\neg C, \text{true}), (D, \text{false})\}, 1)$，$C(D)=(\{(A, \text{false}), (\neg B, \text{false}), (\neg C, \text{true})\}, 1)$.

通过该操作，即可更新邻居变量对应的二元组信息，下一次进行格局信息检测时只需判定二元组中的变量计数器是否为 0 即可.

事实上，在实现上述精确格局检测策略时，为每个变量保存的信息不仅包括一个二元组，还为每个变量增加了一个栈. 利用栈结构，可以存储自该变量在极大项中对应的文字最近一次取否定至今，哪些邻居变量对应的文字曾经变化.

例如，对于任意变量 x，其格局信息中用一个栈来保存变化的邻居变量. 假定其邻居变量为 y 和 z，若 x 最近一次取否定时，变量 y 和 z 在当前极大项中对应的文字分别为 $\neg y$ 和 z，则该栈初始情况为空. 某一次选变量时若选中 y，则 y 在极大项中由 $\neg y$ 变成 y，发生了变化，则在 x 的格局信息中，y 变量就需要入栈，而 z 不用. 在每次判定每个变量的格局信息的时候，只需要判断该变量格局信息中的栈是否为空即可知道该变量自从上一次取否定后格局信息是否发生了变化. 而该变量对应的文字取否定的时候，则把栈中的变量对应的标志位全部置为 false，栈顶指针清零，恢复初始状态. 通过该种做法，可以最大化优化更新各变量的格局信息所消耗的时间复杂度.

从上面的分析可以得出：在使用当前极大项 T 对子句集 F 进行包含扩展判定时，经常遇到能够扩展出 T 的子句而立即判定当前极大项 T 已经失败，需要进行极大项变换；然而却没有充分利用当前失败的信息：能够扩展出当前极大项 T 的子句可以对下一次的极大项变换产生指导性的作用. 基于此，下面给出“向前一步看”策略——双向半扩展规则，通过能够扩展出当前极大项 T 的子句信息，限制取否定的变量的选取范围，达到剪枝优化的目的.

4.3　双向半扩展规则

两模式(two-mode)局部搜索算法，指包含贪心和随机双模式的局部搜索机制，即贪心模式下选择翻转得分值最高的变量进行翻转，随机模式下在当前指派下弄假的子句中随机选择一个变量翻转. 在基于局部搜索的扩展规则方法中，贪心模式下的目标是朝着寻找不可扩展出的极大项的方向进行，而格局检测策略的主要功能是防止在过渡贪心模式下陷入局部循环而长时间找不到解. 在贪心模式下预测下一步变量选取方面还可以加强，通常的做法是采取计算第二层得分(下一次翻转变量的权值信息)的方法来进行“向前一步看”的操作.

事实上在进行第二层得分计算的过程中，往往要采取和第一层得分计算复杂度较为相似或者高于其复杂度的方法来计算，而这种计算方法经常会提高变量选取阶段的时间复杂度.

本文拟采取类似半扩展规则[30-31]的思想来更近一步限制选取变量，而选取的变量的限定性完全来源于当前步骤的考虑，是一种动态更新的策略.

定义 5[30]. 令原子集 $M=\{V_1, V_2, \cdots, V_m\}$，$|M|=m$，子句 $C=l_i \vee \cdots \vee l_j \vee \cdots \vee l_d$. 其中 l_i 为 V_i 对应的文字，$1\leq i\leq j\leq d\leq m$，称 d 为子句 C 的度. $Y=\{C\vee V_k, C\vee \neg V_k \mid d<k\leq m\}$，把从 C 到 Y 中元素的推导过程叫做半扩展规则，Y 中的元素叫做应用半扩展规则的结果.

朴素半扩展规则算法的核心思想在于如果一个极大项能够被一部分子句扩展出来，则下一次不用完全按照编号顺序进行极大项变换，而是利用当前的包含扩展情况找出最小的度，直接进行高位变换，而不是从最低位进行极大项变换，一次可跳过多个极大项的包含扩展判定.

该思想同样可以用于在局部搜索中的文字取否定判定上.

例 4.　类似例 1，$F=\{A\vee B, \neg A\vee \neg D, A\vee C, A\vee D, B\vee \neg C\vee D, \neg B\}$，$T=\{A, \neg B, C, D, E\}$，$V=\{A, B, C, D, E\}$.

当前 CNF 公式中一共 6 个子句，编号为①～⑥. 当使用当前极大项 T 扫描 CNF 公式，计算差集时，发现③、④和⑥号子句的差集为空集，也即是说这 3 个子句能够扩展出当前极大项 T. 利用③号子

句的包含扩展可以发现,下一次对当前极大项中的某个文字取否定时,取 D、E 都会直接导致下一次包含扩展判定失败;利用⑥号子句的包含扩展也同样可以发现,下一次对当前极大项中的某个文字取否定时,取 A、C、D 或者 E 都会导致下一次包含扩展判定直接失败.

在这种情况下需要限制这种直接失败的情况发生,因此需要限制取否定文字的变化范围.

下面给出计算双向半扩展规则的定量指标的算法.

算法 6. 计算选取变量的取值范围(Computing Range for Picking Variable).

输入:CNF 公式 $F=\{C_1,\cdots,C_m\}$,
当前极大项 $T=\{l_1,\cdots,l_n\}$
输出:选取变量的最小右度 Min_right_degree
选取变量的最大左度 Max_left_degree

1. 初始化 $Min\_right\_degree\leftarrow n$
$Max\_left\_degree\leftarrow -1$
2. **WHILE** $i<m$ **DO**
3. **IF** $Expand(T,C_i)$ **THEN**
4. $ld\leftarrow GetLeftDegree(C_i)$
5. $rd\leftarrow GetRightDegree(C_i)$
6. **IF** $Min\_right\_degree>rd$
7. **THEN** $Min\_right\_degree\leftarrow rd$
8. **IF** $Max\_left\_degree<ld$
9. **THEN** $Max\_left\_degree\leftarrow ld$
10. **RETURN** $(Max\_left\_degree, Min\_right\_degree)$

基于算法 6,可以在贪心模式下对候选变量集合进行限制,使得下一步的选择更加精准,加快算法的执行过程.

结合算法 1 至算法 6,下面给出基于局部搜索的扩展规则推理算法 ERACC.

算法 7. 基于精确格局检测的扩展规则推理算法 ERACC(Extension Rule based on Accurate Configuration Checking).

输入:CNF 公式 $F=\{C_1,\cdots,C_m\}$, $MaxTries$,
当前极大项 $T=\{l_1,\cdots,l_n\}$
输出:不可扩展的极大项 $T'=\{l'_1,\cdots,l'_n\}$或
最大尝试次数 $MaxTries$

1. **WHILE** $i<MaxTries$ **DO**
2. **IF** $\sim CanBeExpand(T,F)$ **THEN RETURN** T
3. $(left,right)\leftarrow CRPV(F,T)$
4. Compute the Candidate Set
5. $P=\{v|Score(v)>0$ and $conf[v]=1\}$
6. **IF** $P\neq\varnothing$ **THEN**
7. **IF** $right\geq left$ **THEN**
8. $S=\{v|Score(v)>0$ and $left\leq v\leq right$ and $conf[v]=1\}$
9. **IF** $S\neq\varnothing$ **THEN** $x\leftarrow v\in S$ with largest Score
10. **ELSE**
11. $x\leftarrow v\in P$ with largest Score
12. **ELSE**
13. $x\leftarrow v\in P$ with largest Score
14. **ELSE** $x\leftarrow SWT(F,T,W[],time[])$
15. $T'\leftarrow T/ref(T,x)\cup\neg ref(T,x)$
16. Update configuration according to $ACCC$ or $ACCBE$
17. $i++$
18. **RETURN** $MaxTries$

算法 7 中的 SWT(Smooth Weighting based on Threshold)函数[16]在贪心策略失败时在可扩展当前极大项的子句里随机选择一个,然后选取该子句中最长时间没有翻转的变量.

算法 ERACC 的执行过程大致如下:输入一个初始的极大项,该极大项一般由某种规则(随机或预处理函数)生成.在算法的循环阶段,首先判断当前极大项能否被子句集扩展,如果不能找到一个子句能够扩展出当前极大项,则算法结束,当前极大项即为判定所需要的极大项;否则,按照优先顺序计算两种集合.在计算两种集合时,使用当前极大项先找出能够扩展出该极大项的子句的最大左度和最小右度,再计算出格局信息改变并且得分值大于 0 的集合.如果集合为空,直接进入 SWT 模式;否则,再判断最大左度和最小右度的大小.如果最大左度大于最小右度,说明下一次极大项的变换无论如何都会失败,则退而求其次,选择集合中得分最高的变量在极大项中对应的文字取否定.如果最大左度不大于最小右度,说明下一次极大项变换有可能成功,使用该范围对候选集合进行清理.如果清理完集合不为空,则选取得分最高的变量;否则,同样说明下一次的极大项变换无论如何都会失败,继续选取原集合中得分最高的变量在极大项中对应的文字取否定.

5 实验部分

本节对算法 ERACC 进行了实现,并且将其与扩展规则推理领域主流的求解器 NER[28]、SER[30]以及当前国际主流的 SAT 求解器 SCC[16]、Adaptg2wsat2009++[11]、g2wsat[9]进行了对比,实验结果凸显出了基于局部搜索的扩展规则推理框架

的极大潜能.

实验环境为：操作系统 Linux(Ubuntu 14.04)、CPU Intel Core i3 3.40GHz、内存 2.2GB.

对比实验主要分为两部分：一部分是 ERACC 与 NER、SER 的性能对比；另一部分是 ERACC 与当前国际主流 SAT 求解器的性能对比. 本文采取了多种测试用例①对算法性能进行了测试，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. 其中，Adaptg2wsat2009＋＋和 g2wsat 的源代码均来自于 SAT Competition 的官网，而 SWCC 的源代码则源于蔡少伟的个人主页 http://lcs.ios.ac.cn/～caisw/SAT.html.

5.1　ERACC 与 NER、SER 的对比实验

(1)采用变量数为 20、子句数为 91 的处于相变区间的小规模随机测试用例对 ERACC、NER 和 SER 进行了对比实验，实验结果如表 1 所示.

表 1　随机测试用例 uf20 实验结果(CPU Time/s)

Problem	NER	SER	ERACC
uf20-01	0.0014	<0.0001	<0.0001
uf20-02	0.0010	<0.0001	<0.0001
uf20-03	0.0152	0.0004	<0.0001
uf20-04	0.0018	<0.0001	<0.0001
uf20-05	0.0128	<0.0001	<0.0001
uf20-06	0.0130	<0.0001	<0.0001
uf20-07	0.0132	<0.0001	<0.0001
uf20-08	<0.0001	<0.0001	<0.0001
uf20-09	0.0022	0.0004	<0.0001
uf20-010	0.0052	<0.0001	<0.0001

实验结果表明：针对小规模相变区间的测试用例，算法 ERACC、NER、SER 都表现出了较高的性能，算法 ERACC 占据绝对优势.

(2)采用变量数为 50、子句数为 218 的处于相变区间的小规模随机测试用例对 ERACC、NER 和 SER 进行了对比实验，实验结果如表 2 所示.

表 2　随机测试用例 uf50 实验结果(CPU Time/s)

Problem	NER	SER	ERACC
uf50-01	Time out	1.7350	<0.0001
uf50-02	Time out	0.2848	<0.0001
uf50-03	Time out	0.9020	<0.0001
uf50-04	Time out	2.7498	<0.0001
uf50-05	Time out	0.0368	<0.0001
uf50-06	Time out	2.4602	<0.0001
uf50-07	Time out	0.3966	<0.0001
uf50-08	Time out	1.9374	<0.0001
uf50-09	Time out	0.2306	<0.0001
uf50-010	Time out	0.9956	<0.0001

实验结果表明：当变量规模上升时，NER、SER 的劣势已经凸显. NER 面对这样规模的测试用例已然无能无力，SER 也需要一定的时间才能处理完毕，而 ERACC 却依然保持着优越的性能. 可以看出：ERACC 已经突破传统的基于扩展规则推理的算法的处理能力.

(3)为了更加全方位地测试推理框架以及算法的执行性能，选取了几类结构 SAT(AIM 类问题、JNH 类问题、CBS 类问题)对 ERACC、NER、SER 进行了测试，实验结果如表 3、表 4 和表 5 所示.

表 3　AIM 类问题测试实验结果(CPU Time/s)

Problem	NER	SER	ERACC
aim-50-1_6-yes1-1	Time out	35.28	<0.0001
aim-50-1_6-yes1-2	Time out	57.38	<0.0001
aim-50-2_0-yes1-1	Time out	345.51	<0.0001
aim-50-2_0-yes1-3	Time out	38.57	<0.0001
aim-50-3_4-yes1-1	Time out	1.63	<0.0001
aim-50-3_4-yes1-2	Time out	0.03	<0.0001
aim-100-1_6-yes1-3	Time out	Time out	Time out
aim-100-1_6-yes1-4	Time out	Time out	Time out

表 4　JNH 类问题测试实验结果(CPU Time/s)

Problem	NER	SER	ERACC
jnh1	Time out	Time out	0.0024
jnh7	Time out	Time out	<0.0001
jnh12	Time out	Time out	<0.0001
jnh17	Time out	Time out	0.0028
jnh201	Time out	Time out	<0.0001
jnh204	Time out	Time out	0.0004
jnh205	Time out	Time out	0.0002
jnh207	Time out	Time out	0.0058
jnh209	Time out	Time out	0.0008
jnh210	Time out	Time out	<0.0001

表 5　CBS 类问题测试实验结果(CPU Time/s)

Problem	NER	SER	ERACC
CBS_k3_n100_m403_b10_0	Time out	Time out	<0.0001
CBS_k3_n100_m403_b10_1	Time out	Time out	<0.0001
CBS_k3_n100_m403_b10_2	Time out	Time out	<0.0001
CBS_k3_n100_m403_b10_3	Time out	Time out	<0.0001
CBS_k3_n100_m403_b10_4	Time out	Time out	0.0002
CBS_k3_n100_m403_b10_5	Time out	Time out	<0.0001
CBS_k3_n100_m403_b10_6	Time out	Time out	0.0002
CBS_k3_n100_m403_b10_7	Time out	Time out	<0.0001
CBS_k3_n100_m403_b10_8	Time out	Time out	<0.0001
CBS_k3_n100_m403_b10_9	Time out	Time out	<0.0001

表 3 的实验结果表明：在求解结构 SAT 问题时，ERACC 与传统的基于扩展规则的推理算法执行效率的差距进一步拉大，继续保持稳定的性能. 但对于变量数为 100 的 AIM 类问题，却出现了“Time out”的情况. 经分析，该类测试用例结构性很强，变量个数以及正负文字的比例有着一定的规律，问题

① http://www.cs.ubc.ca/～hoos/SATLIB/index-ubc.html

的解个数较少,其求解的难度并不是和问题规模完全呈正相关.事实上,SWCC 也无法求解上述两个测试用例,主要根源在于局部搜索类算法在求解结构 SAT 问题时并不占优势.

表 4 和表 5 的实验结果表明:在这两类问题上,ERACC 和 NER、SER 已经拉开了较大的差距.NER 和 SER 在限定的求解时间内已经完全不能满足需要,而 ERACC 却已然保持强劲的求解能力.从这两类测试用例也可以看出,基于局部搜索的不完备扩展规则推理框架表现出了出色的性能.

5.2 ERACC 与 SWCC 等求解器的对比实验

(1)将 ERACC 与国际著名局部搜索 SAT 求解器进行了对比实验,测试指标主要为大规模随机测试用例中的执行时间的对比,实验结果如表 6 所示.

表 6 随机测试用例 uf250 实验结果(CPU Time/s)

Problem	ERACC	SWCC	Adaptg2wsat2009++	g2wsat
uf250-01	0.0018	0.0001	0.01234	0.00842
uf250-02	0.0688	0.0084	0.01254	0.01700
uf250-03	0.0302	0.0080	0.03568	0.03074
uf250-04	0.0008	0.0001	0.01192	0.01156
uf250-05	0.0242	0.0016	0.01466	0.01276
uf250-06	0.0058	0.0001	0.01248	0.01034
uf250-07	0.0504	0.0048	0.01340	0.01554
uf250-08	0.0106	<0.0001	0.01256	0.01484
uf250-09	0.0220	0.0030	0.01264	0.02608
uf250-10	0.0256	0.0022	0.01300	0.01692
f600	0.4936	0.0692	0.02744	0.02140

实验结果表明:ERACC 与 Adaptg2wsat2009++、g2wsat 在执行时间上保持了相当的水平,而和 SWCC 相比性能上要略差,差距主要来源于推理框架本身求解问题的特性以及算法的启发式信息指导上.

(2)类似地,使用一些结构 SAT 问题(inductive inference 类问题)对 ERACC 与主流求解器进行对比实验.测试指标主要为结构 SAT 问题中各算法的执行时间的对比,实验结果如表 7 所示.

表 7 Inductive inference 类问题测试实验结果(CPU Time/s)

Problem	ERACC	SWCC	Adaptg2wsat2009++	g2wsat
ii32b1	0.0014	<0.0001	0.00722	0.01336
ii32b2	0.0024	<0.0001	0.02636	0.03358
ii32b3	0.0140	0.0096	0.19436	0.18866
ii32b4	0.0202	0.0104	0.28088	0.28764
ii32c1	<0.0001	<0.0001	0.00724	0.01340
ii32c2	<0.0001	<0.0001	0.02824	0.03394
ii32c3	0.0010	0.0001	0.06798	0.07256
ii32d1	0.0010	0.0001	0.00784	0.01474
ii32d2	0.0104	0.0008	0.02786	0.03260
ii32e1	<0.0001	<0.0001	0.00754	0.01502
ii32e2	<0.0001	<0.0001	0.02540	0.03240
ii32e5	0.0228	0.0013	0.43914	0.41616

实验结果表明:在解决由真实世界的问题进行编码转化的 SAT 问题时,ERACC 表现出了优秀的性能.从实验结果上看,ERACC 和 SWCC 并没有表现出较大的差距;而与 Adaptg2wsat2009++和 g2wsat 相比则表现出了较强的性能优势,表现出较优的启发式指导作用.

从全文的算法设计与分析可知,基于极大项的推理求解方式本身是一种与基于真值指派的推理求解方式相反的求解过程,它是从问题解的反面来寻找答案,因此在不同的测试用例中有着一定的适用性与缺憾.而相比于 SWCC 这一较为成熟的局部搜索求解器,算法 ERACC 目前还处于起步阶段,有必要通过研究启发式策略进一步提升算法性能.

6 结论与展望

本文提出了一种基于局部搜索的扩展规则推理框架,并结合现有研究成果和本文提出的策略设计并实现了一个基于局部搜索的扩展规则推理算法 ERACC.实验结果表明:ERACC 突破了现有扩展规则推理的性能瓶颈,将扩展规则推理的效率提升了一个高度.经过与国际著名 SAT 求解器的对比,标注了 ERACC 的上升空间,也为扩展规则推理的后续研究和应用奠定了基础.

ERACC 算法是扩展规则方法领域不完备推理框架的一个大胆尝试,更加智能的启发式策略和多元的集成策略将会是该框架后续的重要研究方向.就目前考虑,有两个重要的研究方向可供选择.一方面,扩展规则完备推理方法中的启发式策略具有重要的借鉴意义,如 IMOM 和 IBOHM[29]等策略.另一方面,现有局部搜索的部分策略可以作为重要的参考蓝本,可以考虑向该框架中加入概率分布[13]等策略来加强随机模式的优化功能.当然,具体的变体策略需要经过大量的理论分析和实验认证来展现其与该框架的契合性,从而使得该框架更加丰满,求解性能和能力得到更大的提升.

参 考 文 献

[1] Selman B, Levesque H J, Mitchell D G. A new method for solving hard satisfiability problems//Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, USA, 1992: 440-446

[2] Mazure B, Saïs L, Grégoire É. Tabu search for SAT//Proceedings of the 1997 14th National Conference on Artificial Intelligence. Providence, USA, 1997: 281-285

[3] Battiti R, Protasi M. Reactive local search for the maximum clique problem. Algorithmica, 2001, 29(4): 610-637

[4] Smyth K, Hoos H H, Stützle T, Thomas S. Iterated robust tabu search for MAX-SAT//Proceedings of the 16th Conference of the Canadian Society for Computational Studies of Intelligence. Halifax, Canada, 2003: 129-144

[5] Di Gaspero L, Schaerf A. A composite-neighborhood tabu search approach to the traveling tournament problem. Journal of Heuristics, 2007, 13(2): 189-207

[6] Hutter F, Tompkins D A D, Hoos H H. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT//Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming. Ithaca, USA, 2002: 233-248

[7] Hoos H H. An adaptive noise mechanism for WalkSAT//Proceedings of the 18th National Conference on Artificial Intelligence, and the 14th Innovative Applications of Artificial Intelligence Conference. Edmonton, Canada, 2002: 655-660

[8] Thornton J, Pham D N, Bain S, Ferreira V. Additive versus multiplicative clause weighting for SAT//Proceedings of the 19th National Conference on Artificial Intelligence, and the 16th Innovative Applications of Artificial Intelligence Conference. San Jose, USA, 2004: 191-196

[9] Li C M, Huang W Q. Diversification and determinism in local search for satisfiability//Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing. St Andrews, UK, 2005: 158-172

[10] Prestwich S. Random walk with continuously smoothed variable weights//Proceedings of 8th International Conference on Theory and Applications of Satisfiability Testing. St Andrews, UK, 2005: 203-215

[11] Li C M, Wei W, Zhang H. Combining adaptive noise and look-ahead in local search for SAT//Proceedings of the 10th International Conference on Theory and Applications of Satisfiability Testing. Lisbon, Portugal, 2007: 121-133

[12] Pham D N, Thornton J, Gretton C, Sattar A. Combining adaptive and dynamic local search for satisfiability. Journal on Satisfiability, Boolean Modeling and Computation, 2008, 4(2): 149-172

[13] Balint A, Fröhlich A. Improving stochastic local search for SAT with a new probability distribution//Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing. Edinburgh, UK, 2010: 10-15

[14] Li C M, Li Y. Satisfying versus falsifying in local search for satisfiability//Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing. Trento, Italy, 2012: 477-478

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

[16] Cai S W, Su K L. Local search with configuration checking for SAT//Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence. Boca Raton, USA, 2011: 59-66

[17] Cai S W, Su K L. Configuration checking with aspiration in local search for SAT//Proceedings of the 26th AAAI Conference on Artificial Intelligence and the 24th Innovative Applications of Artificial Intelligence Conference. Toronto, Canada, 2012: 434-440

[18] Cai S W, Su K L. Local search for Boolean Satisfiability with configuration checking and subscore. Artificial Intelligence, 2013, 204(9): 75-98

[19] Cai S W, Luo C, Su K L. Scoring functions based on second level score for k-SAT with long clauses. Journal of Artificial Intelligence Research, 2014, 51(1): 413-441

[20] Cai S W, Luo C, Su K L. CCAnr: A configuration checking based local search solver for non-random satisfiability//Proceedings of the 18th International Conference on Theory and Applications of Satisfiability Testing. Austin, USA, 2015: 1-8

[21] Luo C, Cai S W, Su K L, Wu W. Clause states based configuration checking in local search for satisfiability. IEEE Transactions on Cybernetics, 2015, 45(5): 1014-1027

[22] Luo C, Cai S W, Wu W, Su K L. Double configuration checking in stochastic local search for satisfiability//Proceedings of the 28th AAAI Conference on Artificial Intelligence, the 26th Innovative Applications of Artificial Intelligence Conference and the 5th Symposium on Educational Advances in Artificial Intelligence. Quebec City, Canada, 2014: 2703-2709

[23] Luo C, Cai S W, Wu W, et al. CCLS: An efficient local search algorithm for weighted maximum satisfiability. IEEE Transactions on Computers, 2015, 64(7): 1830-1843

[24] Cai S W, Jie Z, Su K L. An effective variable selection heuristic in SLS for weighted Max-2-SAT. Journal of Heuristics, 2015, 21(3):433-456

[25] Lin H, Sun J G, Zhang Y M. Theorem proving based on the extension rule. Journal of Automated Reasoning, 2003, 31(1): 11-21

[26] Lin H, Sun J G. Knowledge compilation using the extension rule. Journal of Automated Reasoning, 2004, 32(2): 93-102

[27] Murray N V, Rosenthal E. Duality in knowledge compilation techniques//Proceedings of the 15th International Symposium on Foundations of Intelligent Systems. Saratoga Springs, USA, 2005: 182-190

[28] Sun Ji-Gui, Li Ying, Zhu Xing-Jun, Lü Shuai. A novel theorem proving algorithm based on extension rule. Journal of Computer Research and Development, 2009, 46(1): 9-14 (in Chinese)
(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)

[29] Li Ying, Sun Ji-Gui, Wu Xia, Zhu Xing-Jun. Extension rule algorithms based on IMOM and IBOHM heuristics strategies. Journal of Software, 2009, 20(6): 1521-1527(in Chinese)

(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)

[30] Zhang Li-Ming，Ouyang Dang-Tong，Bai Hong-Tao. Theorem proving algorithm based on semi-extension rule. Journal of Computer Research and Development，2010，47(9)：1522-1529(in Chinese)
(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)

[31] Zhang L M，Ouyang D T，Zhao J，Bai H T. The parallel theorem proving algorithm based on semi-extension rule. Applied Mathematics & Information Sciences，2012，6(1)：119-122

[32] Zhang Li-Ming，Ouyang Dan-Tong，Zhao Yi. Theorem proving decomposition algorithm based on semi-extension rule. Journal of Software，2015，26(9)：2250-2261(in Chinese)
(张立明，欧阳丹彤，赵毅. 半扩展规则下分解的定理证明方法. 软件学报，2015，26(9)：2250-2261)

[33] Xu You-Jun. Research on Several SAT Issues Based on Extension Rule [Ph. D. dissertation]. Jilin University，Changchun，2011(in Chinese)
(许有军. 基于扩展规则的若干SAT问题研究[博士学位论文]. 吉林大学，长春，2011)

[34] Yin Ming-Hao，Sun Ji-Gui，Lin Hai，Wu Xia. Possibilistic extension rules for reasoning and knowledge compilation. Journal of Software，2010，20(11)：2826-2837(in Chinese)
(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)

[35] Yin Ming-Hao，Lin Hai，Sun Ji-Gui. Solving ♯SAT using extension rules. Journal of Software，2009，20(7)：1714-1725(in Chinese)
(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)

[36] Liu Lei，Niu Dang-Dang，Lü Shuai. Knowledge compilation methods based on the hyper extension rule. Chinese Journal of Computers，2016，39(8)：1681-1696(in Chinese)
(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)

[37] Lai Yong，Ouyang Dan-Tong，Cai Dun-Bo，Lü Shuai. Model counting and planning using extension rule. Journal of Computer Research and Development，2009，46(3)：459-469(in Chinese)
(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)

YANG Yang，born in 1992，M. S. His research interests include intelligent planning and automated reasoning.

LIU Lei，born in 1960，professor，Ph. D. supervisor. His main research interests include software theory and technology.

LI Guang-Li，born in 1992，M. S. candidate. His research interests include automated reasoning and cloud computing.

ZHANG Tong-Bo，born in 1995，M. S. candidate. His research interests include intelligent planning and automated reasoning.

LÜ Shuai，born in 1981，Ph. D.，associate professor. His main research interests include intelligent planning and automated reasoning.

Background

The research area of this paper is reasoning method in Extension Rule(ER). ER is a new reasoning method，first presented in 2003 and well received by experts at home and abroad. And the authors also put forth a new method KCER (Knowledge Compilation using the Extension Rule). The formulae compiled by KCER were called EPCCL theory，in which each pair of clauses contain complementary literals. The EPCCL theory was appreciated by Murray，who proposed non-clausal resolution. After that，Li et al. proposed NER (novel extension rule) based on the classical extension rule. In the same year，Li et al. put forward two heuristic strategies，IMOM and IBOHM. In addition，Zhang et al. proposed several reasoning methods based on semi-extension rule in 2010 and 2015. Much of the achievements in extension rule are about complete reasoning method，and there was no achievements about incomplete reasoning method based on extension rule so far. More strategies and methods should be a great reference for extension rule to make it more mature.

This paper proposes a framework of incomplete reasoning method based on extension rule，and then use several strategies to make the framework more filling. The algorithm ERACC beats different version of reasoning methods based on extension rule and make the ability of dealing large scale formulae in extension rule stronger. Compared with SWCC，

there is still some way to go for ERACC. But we can also see the great potential of it.

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61300049, 61502197, 61503044, the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20120061120059, and the Natural Science Research Foundation of Jilin Province of China under Grant Nos. 20130206052GX, 20140520069JH, 20150520058JH, 20150101054JC, 20180101053JC.

The performance of extension rule is an emergent problem to solve. It will be much influenced in many field based on extension rule, such as knowledge compilation, model counting and intelligent planning etc. After deep analysis of all the achievements of extension rule since it was proposed in 2003, we can see that the different version of extension rule mainly consist of ER, IER, NER, IMOM_IER, IBOHM_IER, SER and PSER etc. The proposal of extension rule is a remarkable event in the field of automated reasoning, but the performance of these algorithms almost cannot be regard as the state-of-art solvers in the world owing to the time. More great skills and method should be a great reference for extension rule to make the framework grown-up. The framework of LSER (local search in extension rule) is an expansion of extension rule and algorithm ERACC is a specific version of LSER. The experimental results show that algorithm ERACC totally break through the gap of extension rule in the scale of formulae, and make the performance of reasoning method based on extension rule more efficient. It should be a good start for more research for extension rule in the future.