基于 IAPS 的扩展规则局部搜索算法

王金艳[1,2],胡 春[2],牛当当[1,3],李先贤[1,2]
(1. 广西多源信息挖掘与安全重点实验室(广西师范大学),广西桂林 541004;
2. 广西师范大学计算机科学与信息工程学院,广西桂林 541004; 3. 西北农林科技大学信息工程学院,陕西杨凌 712100)

摘 要: ERACC (Extension Rule Based on Accurate Configuration Checking)算法由杨洋等人基于扩展规则和格局检测提出,具有较高的推理效率. 为进一步提高 ERACC 算法在大规模 SAT(Satisfiability)问题求解上的性能,本文在搜索由极大项组成的空间时,首先利用 IMOM(Improved Maximum Occurrences on Clauses of Maximum Size)思想生成初始极大项,接着设计了适用于扩展规则推理的 CCA_ER(Configuration Checking with Aspiration for Extension Rule-Based Reasoning)启发式策略,为极大项中格局信息未发生变化的变量对应文字提供一定的翻转机会. 同时,为进一步提高扩展规则推理算法在 *k*-SAT 问题求解上的性能,设计了适用于扩展规则推理的 PAWS_ER(Pure Additive Weighting Scheme for Extension Rule-Based Reasoning)策略,并且给出变量的 *Subscore_ER*(Subscore for Extension Rule-Based Reasoning),*CScore_ER*(Comprehensive Score for Extension Rule-Based Reasoning)和 *HScore_ER*(Hybrid Score for Extension Rule-Based Reasoning)属性. 在此基础上,提出了 ERACC_IAPS(ERACC with IMOM,CCA_ER,PAWS_ER and Subscore_ER)和 CERACC_IAPS(ERACC with IMOM,CCA_ER,PAWS_ER,CScore_ER and HScore_ER)算法. 实验结果表明:ERACC_IAPS 和 CERACC_IAPS 算法的效率明显优于 ERACC 算法,最高可将其求解效率提高 1000 多倍.

关键词: 扩展规则; 自动推理; 局部搜索; 格局检测
中图分类号: TP181, TP301 **文献标识码:** A **文章编号:** 0372-2112 (2020)05-0899-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2020.05.009

An Extension Rule Algorithm Based on Local Search with IAPS Strategies

WANG Jin-yan[1,2], HU Chun[2], NIU Dang-dang[1,3], LI Xian-xian[1,2]
(1. *Guangxi Key Lab of Multisource Information Mining & Security (Guangxi Normal University), Guilin, Guangxi* 541004, *China*;
2. *School of Computer Science and Information Engineering, Guangxi Normal University, Guilin, Guangxi* 541004, *China*;
3. *College of Information Engineering, Northwest A&F University, Yangling, Shaanxi* 712100, *China*)

Abstract: The algorithm ERACC proposed by Yang et al. has high reasoning efficiency. To further improve performance of ERACC algorithm in solving large-scale SAT problem, firstly, IMOM is used to generate the initial maximum term. Then a CCA_ER is designed to provide certain opportunities for the literals of variables whose configurations have not been changed since their last flips in maximal terms. At the same time, in order to further improve the performance of extension rule-based reasoning algorithm on the *k*-SAT problem, PAWS_ER is designed. Then the *Subscore_ER*, *CScore_ER* and *HScore_ER* attributes for variables are further designed. Finally, ERACC_IAPS and CERACC_ER algorithms are proposed. The experimental results show that the efficiency of ERACC_IAPS and CERACC_IAPS algorithms is obviously better than that of ERACC algorithm, and the maximum solution efficiency can be improved by more than 1000 times.

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

1 引言

布尔可满足性问题(SAT)是第一个被证明的 NP-完全问题. 目前,SAT 问题的求解方法主要分为完备算法和不完备算法. 局部搜索算法属于不完备算法,是求解大规模 SAT 问题的常用方法,通常工作在两种模式:

收稿日期:2019-11-15;修回日期:2020-02-10;责任编辑:马兰英
基金项目:国家自然科学基金(No. 61763003, No. 61672176, No. 61941201);广西多源信息挖掘与安全重点实验室系统性研究课题基金(No. 19-A-02-01);广西多源信息挖掘与安全重点实验室开放基金(No. MIMS19-05);广西高等学校千名中青年骨干教师培育计划;"八桂学者"工程专项经费资助项目;广西大数据智能与应用人才小高地;广西区域多源信息集成与智能处理协同创新中心

贪心模式和随机模式[1]. 在贪心模式下,选择能降低不满足子句数目(或者在有权重的随机局部搜索算法中,选择能降低不满足子句的总权重)的变量进行翻转. 在随机模式下,算法倾向于探索搜索空间并且避免陷入局部最优,通常的做法是使用随机策略,然后利用变量的多元属性选择变量翻转,例如变量的 age 或者翻转次数. 但局部搜索方法容易陷入局部循环,对此蔡少伟等提出了格局检测(Configuration Checking,CC)策略,基于该策略设计开发了一系列先进的 SAT 求解器,包括 Swcc (Smoothed Weighting with Configuration Checking)[2], Swcca (Smoothed Weighting and Configuration Checking with Aspiration)[3],CCASat(Configuration Checking with Aspiration and Subscore for SAT)[4] 和 CScoreSAT(Comprehensive Score Based SAT Algorithm)[5]等,这些求解器在 SAT 竞赛取得了很好的成绩.

扩展规则推理方法[6]是由林海等于 2003 年提出的被人工智能专家 Martin Davis 誉为与归结方法互补的推理方法. 李莹等[7]设计了 NER(Novel Extension Rule Based Theorem Proving)算法,提高了扩展规则推理算法的效率. 张立明等[8]设计了基于半扩展规则的 SER(Semi-Extension-Rule-Based Theorem Proving)算法,使其效率得到进一步提高. 由于基于局部搜索的算法在求解 SAT 问题时获得了巨大的成功,杨洋等[9]将其与扩展规则相结合,设计并实现了一种基于局部搜索的扩展规则推理方法 ERACC,主要将格局检测融入扩展规则推理算法中. ERACC 算法在搜索由极大项组成的搜索空间时,首先初始化一个极大项,然后利用局部搜索技术对极大项空间进行搜索,并且采用新的格局检测避免陷入局部最优,算法每次选择极大项中一个变量对应的文字进行翻转,直至找到一个不能扩展的极大项. 此算法与现有基于扩展规则的算法相比,具有较高的求解效率. 另外,扩展规则推理方法也可应用知识编译[10]和模型计数[11]等领域.

本文在 ERACC 的基础上,提出基于 IAPS 的扩展规则局部搜索算法,其中 I、A、P 和 S 分别是 IMOM 启发式,CCA_ER 启发式,子句更新策略 PAWS_ER 和子句次贡献值 *Subscore_ER* 的简称. 首先利用 IMOM 思想[12]初始化极大项,然后设计了适用于扩展规则的 CCA_ER 启发式以及 PAWS_ER 策略,并且提出变量的 *Subscore_ER* 属性以及衍生的 *CScore_ER* 属性和 *HScore_ER* 属性. 其中 CCA_ER 启发式给超过一定阈值但自从上次翻转以来格局没有发生改变的变量对应文字一定的翻转机会,PAWS_ER 用于更新子句权重,*Subscore_ER* 用于打破平局(即当有多个变量具有最大贡献值时),*CScore_ER* 和 *HScore_ER* 分别用于贪心过程和随机过程翻转文字对应变量的选择.

2 基础知识

2.1 扩展规则

定义 1[6] 给定一个子句 C 和一个原子集合 M. $D=\{C\vee a, C\vee \neg a \mid a\in M$ 并且 a 和 $\neg a$ 都不在 C 中出现$\}$,将从 C 到 D 的推导过程称为扩展规则(Extension Rule,ER),并将 D 中的元素称为应用扩展规则的结果.

基于扩展规则的命题逻辑推理方法主要分为两种:基于容斥原理的推理方法和基于极大项搜索的推理方法. 前者使用容斥原理计算输入子句集 F 所能扩展出的极大项个数 MT,然后与 $2^{|V(F)|}$ 进行比较,若 $MT=2^{|V(F)|}$,则返回 F 为不可满足,否则返回 F 可满足,其中 $V(F)$ 表示出现在 F 中的变量集. 后者通过搜索极大项空间中是否存在不能由 F 扩展出的极大项,若存在,则可知 F 所能扩展出的极大项个数 $MT\neq 2^{|V(F)|}$,返回 F 可满足,否则返回 F 不可满足.

对于完备的极大项搜索算法而言,如 NER[7] 和 SER[8],在搜索完成之后可以判断是否存在 F 不能扩展出的极大项;而对于不完备的极大项搜索算法而言,如 ERACC[9],只有当搜索到 F 不能扩展出的极大项时才可判断 F 的可满足性,否则不能判断. 当 F 不可满足时,不完备的搜索算法不能判定其可满足性. 不完备的推理方法不能保证一定可以判断出输入公式的可满足性,然而它们执行效率高,能够处理超大规模的问题,且在使用合适的启发式策略之后,具备较高的准确率,因此得到了广泛应用.

2.2 ERACC 算法

扩展规则推理方法通过判断当前极大项与所有输入子句之间的关系来决定输入子句集的可满足性. 用 $cost\_ER(F,T)$ 表示 F 中能扩展出当前极大项的子句权值之和,$score\_ER(x)$ 表示在当前极大项中翻转变量 x 对应的文字得到的贡献值,则 $score\_ER(x)=cost\_ER(F,T)\text{-}cost\_ER(F,T')$,其中 T' 为翻转 x 对应文字之后得到的极大项. 用数组 *confChange_ER* 表示每个变量的格局改变状态,对于任意变量 x,$confChange\_ER[x]=0$ 则表示 x 的格局自 x 对应的文字上次翻转之后未发生改变,$confChange\_ER[x]=1$ 则相反,其中变量 x 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 的所有邻居变量的相应文字,x 的邻居变量集[2]为 $N(x)=\{y\mid y\in V(F)$ 且 y 与 x 至少出现在一个相同的子句中$\}$. 称一个变量 x 是 CCD_ER(Configuration Changed Decreasing for ER Reasoning)变量当且仅当 $score\_ER(x)>0$ 并且 $confChange\_ER[x]=1$.

CC_ER(Configuration Checking for ER Reasoning)启发式[9]:贪心模式下从 CCD_ER 变量集中选择具有最大 *score_ER* 值变量对应文字进行翻转;在随机模式下,

随机选择一个能扩展出当前极大项的子句,从中选择最长时间没有翻转的变量,并对其文字进行翻转.

ERACC 算法[9]在使用 CC_ER 启发式的同时,使用了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模式. 该算法给出了基于局部搜索的扩展规则推理框架,为扩展规则推理算法在更多现实问题领域中的应用奠定了良好基础.

3　ERACC_IAPS 算法

3.1　初始化极大项

基于局部搜索的扩展规则算法需要找到不能由子句集扩展出来的极大项,因此极大项的初始化非常重要,如果初始极大项趋近目标,则需要较少的迭代次数就能找到问题的解. 本文利用限定极大项搜索空间的 IMOM 策略[12]对极大项进行初始化.

算法 1　IniMT_IMOM(CNF:F)

1　$IniMT$ = IMOM(F);
2　V_IMOM = GetVar($IniMT$);
3　$V\_ran = V(F)/V\_IMOM$;
4　For each $x_i \in V\_ran$;
5　　generate randomly a literal L_{xi};
6　　$IniMT = IniMT \cup \{L_{xi}\}$;
7　Return $IniMT$;

算法 1 的输入为给定的 CNF 公式 F,输出为得到的初始极大项. 首先利用 IMOM 策略得到用于限定极大项空间的子句,即确定初始极大项的部分文字,存储在集合 $IniMT$ 中. $GetVar(IniMT)$用于提取 $IniMT$ 中的每个文字的对应变量. 对于其它变量,随机产生对应的文字,将其并入 $IniMT$,得到初始极大项. 通过以上步骤初始化的极大项,不易被公式集中的子句扩展,使得初始解更加靠近问题的解,从而减少翻转次数.

3.2　CCA_ER 启发式

面向扩展规则推理的格局检测启发式 CC_ER 要求 ERACC 算法在贪心选择阶段翻转极大项中格局信息发生变化的变量对应文字. 受 CCA(Configuration Checking with Aspiration)启发式思想[3]的启发,我们为极大项中格局信息未发生变化的变量对应的文字提供一定的翻转机会,提出了 CCA_ER 启发式. 若存在一个变量 x 满足 $score\_ER(x) > g\_ER$,其中 g_ER 为所有子句权值的平均值,则称该变量是一个 SD_ER(Significant Decreasing for ER Reasoning)变量.

CCA_ER 启发式:与 CC_ER 启发式不同的是,若当前极大项中不存在 CCD_ER 变量,但存在 SD_ER 变量,则从 SD_ER 变量集中选择具有最大 $score\_ER$ 值变量对应文字进行翻转.

不同于 CCA 启发式,CCA_ER 启发式从极大项中选择变量,在贪心选择阶段,通过比较极大项中 CCD_ER 变量或 SD_ER 变量的 $score\_ER$ 值选择变量,在随机选择阶段,从能扩展出当前极大项的子句中随机选择子句,然后选取此子句中最长时间没有翻转的变量.

3.3　子句权重更新策略

在 ERACC 算法中,当不存在 CCD_ER 变量时,则使用 SWT_ER(Smooth Weighting Based on Threshold for ER Reasoning)策略更新子句集中的子句的权值. Cai 等[4]验证 SWT(Smooth Weighting Based on Threshold)并不适合 k-SAT 问题上的求解,因此在 CCApaws 算法中使用了 PAWS(Pure Additive Weighting Scheme)策略[13]对子句权重进行更新. 基于该思想,本文设计了适用于基于扩展规则的 PAWS_ER 策略:以一定的概率将所有权值大于 1 的不能扩展出当前极大项的子句权值减 1;否则将所有能扩展出当前极大项的子句权值加 1.

不同于 PAWS 策略,PAWS_ER 更新权值的对象是能扩展出当前极大项的子句和不能扩展出当前极大项的子句,而 PAWS 更新权值的对象是当前指派下可满足的子句和当前指派下不可满足的子句.

3.4　子句的次贡献值

在局部搜索方法中,主要使用变量的 $score$ 或者 $break$ 属性指导翻转变量的选择. 其中 $break$ 属性表示翻转变量 x 后,公式集中可满足子句变为不可满足子句的数目. 对于 $k>3$ 的实例,由于可满足子句中可满足文字个数变化区间比较大,因此 Cai 等[4]设计了变量的 $Subscore$ 属性. 给定公式集 F 和一真值指派,一个可满足子句为一个关键子句当且仅当在该赋值下只有一个文字为真,否则为稳定子句. 变量 x 的 $Subscore(x) = Submake(x)$-$Subbreak(x)$,其中 $Submake(x)$ 和 $Subbreak(x)$ 分别表示翻转变量 x 后,关键子句变为稳定子句和稳定子句变为关键子句的个数. 受该思想的启发,我们设计基于极大项的 $Subscore\_ER$ 属性.

定义 2　给定公式集 F 和当前极大项,一个不可扩展当前极大项的子句为关键子句当且仅当因一个文字的差别而不能扩展当前极大项,否则为稳定子句.

给定 CNF 公式集 F,T 为当前极大项,T_x' 为在 T 中翻转变量 x 对应的文字得到的极大项,$w(C_i)$ 为子句 C_i 相关的权值,子句个数 k,则:

$$Submake\_ER(F,T,T_x') = \sum_{i=1}^{k} w(c_i) * \delta_1(T,T_x',C_i) \tag{1}$$

其中,$\delta_1(T,T_x',C_i)=1$,当 T 翻转为 T_x',C_i从关键子句变为稳定子句;否则 $\delta_1(T,T_x',C_i)=0$.

$$Subbreak\_ER(F,T,T_x') = \sum_{i=1}^{k} w(c_i) * \delta_2(T,T_x',C_i) \tag{2}$$

其中,$\delta_2(T,T_x^{'},C_i)=1$,当 T 翻转为 $T_x^{'}$,C_i从稳定子句变为关键子句;否则 $\delta_2(T,T_x^{'},C_i)=0$.

基于上述特征函数,可以计算出翻转变量 x 对应文字后的次贡献值.

$$Subscore\_ER(F,T,T_x^{'})=Submake\_ER(F,T,T_x^{'})-Subbreak\_ER(F,T,T_x^{'}) \quad (3)$$

利用上述初始化极大项策略 IniMT_IMOM、CCA_ER 启发式、子句权重更新策略 PAWS_ER 和变量的 Subscore_ER 属性,我们设计了 ERACC_IAPS 算法,具体流程如算法 2 所示:

算法 2 ERACC_IAPS(CNF:F,Maxsteps)

```
1  T = IniMT_IMOM(F);
2  For step: = 1 to MaxSteps do
3    If ~ CanBeExpand(T,F) Then
4      Return T;
5    (left,right) ←CRPV(F,T);
6    P = {v | score_ER(v) >0 & confChange_ER[v] = 1};
7    If P ≠ ∅ Then
8      If right ≥ left Then
9        S = {v | score_ER(v) >0 & v ∈ [left,right] & confChange_ER
         [v] = 1};
10       If S ≠ ∅ Then
11         x = v, where v ∈ S with largest score_ER, breaking ties pre-
           ferring the one with greatest Subscore_ER;
12         Else x = v, where v ∈ P with largest score_ER, breaking ties
           preferring the one with greatest Subscore_ER;
13       Else x = v, where v ∈ P with largest score_ER, breaking ties pre-
         ferring the one with greatest Subscore_ER;
14     Else if R = {v | score_ER(v) > g_ER} ≠ ∅ Then
15       x = v, where v ∈ R with largest score_ER, breaking ties prefer-
         ring the one with greatest Subscore_ER;
16     Else x ←PAWS_ER(F,T,W[ ],time[ ]);
17   T' = T/ref(T,x) ∪ ¬ ref(T,x);
18   Update configuration by ACCC or ACCBE;
19 Return Maxsteps;
```

算法 2 中,*Maxsteps* 为一个正整数,表示最大翻转次数;CanBeExpand(T,F)函数[17]用于判断极大项 T 是否可由公式 F 扩展出来. CRPV(Computing Range for Picking Variable)函数是 ERACC 中原有的函数,其功能是限定变量的选取范围;而 ACCC(Accurate Configuration Checking Based on Counters)或 ACCBE(Accurate Configuration Checking Based on Binary Encoding)也是 ERACC 中的函数,其功能就是更新变量的格局;PAWS_ER 函数是用 PAWS_ER 策略更新子句权重,然后随机选择一个可扩展当前极大项的子句,并从中选择一个最长时间没有翻转的变量;$ref(T,x)$表示当前极大项 T 中变量 x 对应的文字.

ERACC_IAPS 与 ERACC 算法的主要区别在于:(1)使用 IniMT_IMOM 函数得到初始极大项;(2)增加了候选集合 R,给予极大项中格局状态未发生改变的变量对应文字一定的翻转机会;(3)更新子句权重方面使用了 PAWS_ER 策略;(4)利用 *Subscore_ER* 属性更有效地选取变量并对其文字进行翻转.

4 CERACC_IAPS 算法

由于长子句 SAT 问题结构更为复杂,需要设计更为巧妙的策略进行求解. 受文献[5]启发,我们将 *score_ER* 和 *Subscore_ER* 进行线性组合,给出了极大项中变量的综合贡献值 *CScore_ER* 和混合贡献值 *HScore_ER*,进而提出了 CERACC_IAPS 算法,具体流程如算法 3 所示.

算法 3 CERACC_IAPS(CNF:F,Maxsteps)

```
1  T = IniMT_IMOM(F);
2  For step: = 1 to MaxSteps do
3    If ~ CanBeExpand(T,F) Then
4      Return T;
5    (left,right) ←CRPV(F,T);
6    P = {v | score_ER(v) >0 & confChange_ER[v] = 1& CScore_ER
     (v) >0};
7    If P ≠ ∅ Then
8      If right ≥ left Then
9        S = {v | score_ER(v) >0 & v ∈ (left,right) & confChange_ER
         [v] = 1 & CScore_ER(v) >0};
10       If S ≠ ∅ Then
11         x = v, where v ∈ S with largest CScore_ER, breaking ties pre-
           ferring the oldest one ;
12         Else x = v, where v ∈ P with largest CScore_ER, breaking ties
           preferring the oldest one;
13       Else x = v, where v ∈ P with largest CScore_ER, breaking ties
         preferring the oldest one;
14     Else if R = {v | score_ER(v) > g_ER} ≠ ∅ Then
15       x = v, where v ∈ R with largest CScore_ER, breaking ties prefer-
         ring the oldest one;
16     Else update clause weights according to PAWS_ER and x = v with
       largest HScore in a selected randomly clause which can extend
       to T;
17   T' = T/ref(T,x) ∪ ¬ ref(T,x);
18   Update configuration by ACCC or ACCBE;
19 Return Maxsteps;
```

给定一个 *CNF* 公式集 F,对于变量为 x,$CScore\_ER(x)=score\_ER(x)+Subscore\_ER(x)/d$,其中 d 为正整数,当多个变量的 *score_ER* 值一样时,优先选择 *Subscore_ER* 值较大的变量;$HScore\_ER(x)=CScore\_ER(x)+age(x)/b$,其中 b 为正整数,当多个变量的 *age* 值一样时,优先选择 *CScore_ER* 值较大的变量.

CERACC_IAPS 算法进一步利用变量的 CScore_ER

属性以及 HScore_ER 属性,在贪心步骤中优先选择具有综合贡献值最大的变量,在随机步骤中从能够扩展出当前极大项的子句中选择具有最大混合贡献值的变量.

ERACC_IAPS 和 CERACC_IAPS 算法都包含贪心模式和随机模式. 在贪心步骤中前者优先选择具有贡献值最大的变量,后者优先选择具有综合贡献值最大的变量. 在随机步骤中,前者从能够扩展出当前极大项的子句中选择最长时间没有翻转的变量,后者从能够扩展出当前极大项的子句中选择具有最大混合贡献值的变量.

5　实验结果与分析

我们实现了 ERACC_IAPS 和 CERACC_IAPS 算法,并使用扩展规则推理算法 NER、SER 和 ERACC 以及基于真值赋值的局部搜索算法 CCASat、CScoreSAT、probSAT[14] 和 yalsat[15] 进行对比. 测试实例包括 2016 和 2017 年 SAT 竞赛实例,以及随机器产生的实例. 实验环境为:Linux(Ubuntu 18.04.1),CPU Intel Core i5 3.20GHz,内存 3.2GB. 测试过程中对每一个测试用例算法均执行 30 次,取平均值作为测试结果,*cutoff time* = 3000s.

5.1　ERACC_IAPS 算法的测试结果与分析

本节将 ERACC_IAPS 算法与完备扩展规则推理算法 NER、SER 以及基于局部搜索的扩展规则算法 ERACC 进行对比分析.

(1)四种算法分别在 2017 年 SAT 竞赛以及随机器产生的 5-SAT 实例上进行测试,实验结果如表 1 和表 2 所示. 在表 1 中,当 $v = 360$ 时,ERACC_IAPS 的效率是 ERACC 的 300 倍. 在表 2 中 ERACC_IAPS 可以解决所有的实例,ERACC 不能求解任何一个.

表 1　2017 年 SAT competition 5-SAT 实例上的测试结果

Benchmarks	NER	SER	ERACC	ERACC_IAPS
5SAT-*v*280	Time out	Time out	47.58	**1.51**
5SAT-*v*290	Time out	Time out	12.44	**10.84**
5SAT-*v*300	Time out	Time out	115.67	**9.5**
5SAT-*v*310	Time out	Time out	2161.21	**235.51**
5SAT-*v*320	Time out	Time out	1177.41	**18.62**
5SAT-*v*340	Time out	Time out	2782.13	**146.91**
5SAT-*v*360	Time out	Time out	21.05	**0.07**
5SAT-*v*370	Time out	Time out	235.73	**6.37**
5SAT-*v*420	Time out	Time out	Time out	**41.89**

(2)四种算法分别在 2016 年 SAT 竞赛以及随机器产生的 7-SAT 实例上进行测试,实验结果如表 3 和表 4 所示. 在表 3 中,当 $v = 94$ 时,ERACC_IAPS 比 ERACC 快 100 多倍;在表 4 中,对于 06 编号的实例,ERACC 出现了 Time out 即运行时间超过了 3000s,而 ERACC_IAPS 只花了 1.96 秒,是 ERACC 算法效率的 1000 多倍.

表 2　随机器产生的 5-SAT 实例上的测试结果

Benchmarks	NER	SER	ERACC	ERACC_IAPS
5SAT-*v*1200-01	Time out	Time out	Time out	**120.02**
5SAT-*v*1200-02	Time out	Time out	Time out	**35.42**
5SAT-*v*1200-03	Time out	Time out	Time out	**27.69**
5SAT-*v*1200-04	Time out	Time out	Time out	**87.17**
5SAT-*v*1200-05	Time out	Time out	Time out	**56.95**
5SAT-*v*1200-06	Time out	Time out	Time out	**132.57**
5SAT-*v*1200-07	Time out	Time out	Time out	**69.23**
5SAT-*v*1200-08	Time out	Time out	Time out	**23.06**
5SAT-*v*1200-09	Time out	Time out	Time out	**292.78**
5SAT-*v*1200-10	Time out	Time out	Time out	**125.31**

表 3　2016 年 SAT competition 7-SAT 实例上的测试结果

Benchmarks	NER	SER	ERACC	ERACC_IAPS
7SAT-*v*90	Time out	Time out	12.09	**3.59**
7SAT-*v*92	Time out	Time out	103.96	**4.81**
7SAT-*v*94	Time out	Time out	405.14	**3.94**
7SAT-*v*96	Time out	Time out	264.34	**12.58**
7SAT-*v*98	Time out	Time out	32.54	**12.22**
7SAT-*v*102	Time out	Time out	2352.32	**287.18**
7SAT-*v*104	Time out	Time out	313.86	**102.63**
7SAT-*v*106	Time out	Time out	85.86	**9.33**

表 4　随机器产生的 7-SAT 实例上的测试结果

Benchmarks	NER	SER	ERACC	ERACC_IAPS
7SAT-*v*200-01	Time out	Time out	Time out	**168.65**
7SAT-*v*200-02	Time out	Time out	Time out	**116.59**
7SAT-*v*200-03	Time out	Time out	Time out	**99.08**
7SAT-*v*200-04	Time out	Time out	Time out	**23.02**
7SAT-*v*200-05	Time out	Time out	Time out	**24.41**
7SAT-*v*200-06	Time out	Time out	Time out	**1.96**
7SAT-*v*200-07	Time out	Time out	Time out	**51.35**
7SAT-*v*200-08	Time out	Time out	Time out	**22.76**
7SAT-*v*200-09	Time out	Time out	Time out	**25.63**
7SAT-*v*200-10	Time out	Time out	Time out	**107.27**

5.2　CERACC_IAPS 算法的测试结果与分析

本节将 CERACC_IAPS 算法与基于局部搜索的扩展规则算法 ERACC 以及我们的 ERACC_IAPS 算法进行对比分析.

(1)我们的两种算法 ERACC_IAPS、CERACC_IAPS 与 ERACC 分别在 5-SAT 和 7-SAT 实例上进行测试,实验结果如表 5 和表 6 所示. 从表 5 可以看出,ERACC 不能解决变量高于 750 的实例,而 ERACC_IAPS 和 CERACC_IAPS 能全部解决,当 $v=250$、350、550、650 和 1000 时,CERACC_IAPS 优于 ERACC_IAPS,其余实例为 ERACC_IAPS 胜出. 从表 6 可以可知,ERACC_IAPS 和 CERACC_IAPS 均明显优于 ERACC. 在编号为 01 和 06 的实例中,CERACC_IAPS 算法的运行效率是 ERACC 算法的 1000 多倍. 在编号为 02 的实例中,CERACC_IAPS 输于 ERACC_IAPS,其余均 CERACC_IAPS 胜出.

表 5 随机器产生的 5-SAT 实例上的测试结果

Benchmarks	ERACC	ERACC_IAPS	CERACC_IAPS
5SAT-v250	2. 55	0. 5	**0. 34**
5SAT-v350	15. 84	0. 7	**0. 2**
5SAT-v450	17. 01	**1. 98**	4. 32
5SAT-v550	30. 97	3. 91	**1. 65**
5SAT-v650	125. 79	4. 68	**1. 95**
5SAT-v750	Time out	**6. 45**	54. 73
5SAT-v850	Time out	**30. 66**	79. 85
5SAT-v950	Time out	**53. 53**	96. 78
5SAT-v1000	Time out	92. 18	**66. 37**

5.3 与其他局部搜索算法进行比较

本节将本文算法 ERACC_IAPS 和 CERACC_IAPS 与基于真值赋值的局部搜索算法 CCASat、CScoreSAT、probSAT 和 yalsat 进行对比分析.

(1)上述 6 种算法在 2016 年 SAT 竞赛的 7-SAT 实例上进行测试,实验结果如表 7 所示. 可以看出,ERACC_IAPS 与 CScoreSAT 相比,ERACC_IAPS 能解决全部的实例,而 CScoreSAT 在 $v=104$ 出现了 Time out 的情况,ERACC_IAPS 与 probSAT 相比,probSAT 在 $v=90$、92、98 和 104 胜出,其余为 ERACC_IAPS 胜出.

表 6 随机器产生的 7-SAT 实例上的测试结果

Benchmarks	ERACC	ERACC_IAPS	CERACC_IAPS
7SAT-v300-01	259. 65	0. 98	**0. 25**
7SAT-v300-02	138. 36	**0. 5**	2. 84
7SAT-v300-03	254. 88	9. 52	**0. 72**
7SAT-v300-04	157. 66	1. 47	**1. 19**
7SAT-v300-05	324. 69	4. 36	**0. 4**
7SAT-v300-06	489. 69	8. 4	**0. 39**
7SAT-v300-07	167. 62	5. 34	**1. 53**
7SAT-v300-08	151. 11	2. 02	**0. 93**
7SAT-v300-09	117. 89	4. 09	**0. 33**
7SAT-v300-10	236. 21	3. 58	**0. 24**

表 7 2016 SAT competition 7-SAT 实例上的测试结果

Benchmarks	CCASat	CScoreSAT	probSAT	yalsat	ERACC_IAPS	CERACC_IAPS
7SAT-v90	18. 66	14. 83	**0. 61**	18. 23	3. 59	14. 69
7SAT-v92	**0. 16**	0. 19	3. 81	9. 43	4. 81	1. 02
7SAT-v94	31. 12	5	6. 61	**1. 19**	3. 94	7. 66
7SAT-v96	81. 54	15. 61	38. 66	40. 97	12. 58	**3. 43**
7SAT-v98	2. 62	4. 56	**2. 55**	53. 98	12. 22	3
7SAT-v102	**37. 9**	1332. 26	1841. 09	201	287. 18	1837. 76
7SAT-v104	154. 12	Time out	**35. 06**	212. 42	102. 63	1435. 48
7SAT-v106	16. 64	7. 81	42. 44	**4. 23**	9. 33	51. 47

6 结论与展望

扩展规则方法是与归结互补的推理方法,本文在基于局部搜索的扩展规则算法 ERACC 的基础上,提出若干策略用于改进其在求解大规模 SAT 问题上的性能. 首先在初始化时利用 IMOM 思想得到初始极大项,使得初始解更加靠近问题的解,从而减少翻转次数. 然后设计了适用于扩展规则的 CCA_ER 启发式,给超过一定阈值但格局没有发生改变的变量对应文字一定的翻转机会,并提出 PAWS_ER 策略用于更新子句权重,最后给出变量的 *Subscore_ER*、*CScore_ER* 和 *HScore_ER* 属性用于指导不同情况下变量的选择. 实验结果表明,我们的两种算法 ERACC_IAPS 和 CERACC_IAPS 在 k-SAT 问题求解上明显优于已有的扩展规则推理方法,在一些实例求解上也优于目前部分流行的局部搜索算法. 在未来的工作中,我们将进一步深入研究扩展规则极大项的特点,设计出新的启发式方法,以便进一步提高扩展规则推理算法的求解效率.

参考文献

[1] Tompkins D A D, Balint A, Hoos H H. Captain Jack: new variable selection heuristics in local search for SAT[A]. Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing[C]. Berlin, Germany: Springer, 2011. 302 – 316.

[2] Cai S W, Su K L. Local search with configuration checking for SAT[A]. Proceedings of the 23rd International Conference on Tools with Artificial Intelligence[C]. New York, USA: IEEE Computer Society, 2011. 59 – 66.

[3] Cai S W, Su K L. Configuration checking with aspiration in local search for SAT[A]. Proceedings of the 26th AAAI

Conference on Artificial Intelligence[C]. Menlo Park,CA: AAAI Press,2012. 434 – 440.

[4] Cai S W, Su K L. Local search for Boolean Satisfiability with configuration checking and subscore[J]. Artificial Intelligence,2013,204:75 – 98.

[5] Cai S W, Su K L. Comprehensive score: towards efficient local search for SAT with long clauses[A]. Proceedings of the 23rd International Joint Conference on Artificial Intelligence[C]. Menlo Park,CA: AAAI Press,2013. 489 – 495.

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

[7] 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[J]. 计算机研究与发展,2009,46(1):9 – 14.
Sun Ji-Gui, Li Ying, Zhu Xing-Jun, LÜ Shuai. A novel theorem proving algorithm based on extension rule[J]. Journal of Computer Research and Development,2009,46(1): 9 – 14. (in Chinese)

[8] 张立明,欧阳丹彤,白洪涛. 基于半扩展规则的定理证明方法[J]. 计算机研究与发展,2010,47(9):1522 – 1529.
Zhang Li-Ming, Ouyang Dang-Tong, Bai Hong-Tao. Theorem proving algorithm based on semi-extension rule[J]. Journal of Computer Research and Development,2010,47 (9):1522 – 1529. (in Chinese)

[9] 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[J]. 计算机学报,2018,41 (4): 825 – 839.
Yang, Liu Lei, Li Guang-Li, Zhang Tong-Bo, LÜ Shuai. A novel local search-based extension rule reasoning method [J]. Chinese Journal of Computers, 2018, 41 (4): 825 – 839. (in Chinese)

[10] 谷文祥,王金艳,殷明浩. 基于 MCN 和 MO 启发式策略的扩展规则知识编译方法[J]. 计算机研究与发展, 2011,48(11):2064 – 2073.
Gu Wen-Xiang, Wang Jing-Yan, Yin Ming-Hao. Knowledge compilation using extension rule based on MCN and MO heuristic strategies[J]. Journal of Computer Research and Development,2011,48(11):2064 – 2073. (in Chinese)

[11] 王强,刘磊,吕帅. 基于扩展规则的启发式#SAT 求解算法[J]. 软件学报,2018,29(11):3517 – 3527.
Wang Qiang, Liu Lei, LÜ Shuai. #SAT solving algorithms based on extension rule using heuristic strategies[J]. Journal of Software,2018,29(11):3517 – 3527. (in Chinese)

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

[13] Thornton J, Pham D N, Bain S. Additive versus multiplicative clause weighting for SAT[A]. Proceedings of the 19th National Conference on artificial Intelligence and 16th Innovative Applications of Artificial Intelligence Conference [C]. Menlo Park,CA: AAAI Press,2004. 191 – 196.

[14] Balint A, Schöning U. Choosing probability distributions for stochastic local search and the role of Make versus break[A]. Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing [C]. Berlin, Germany: Springer, 2012. 16 – 29.

[15] Biere A. CaDiCaL, Lingeling, Plingeling treengeling and YalSAT entering the SAT competition 2018[A]. Proceedings of SAT Competition 2018: Solver and Benchmark Descriptions, volume B-2018-1 of Department of Computer Science Series of Publications B[C]. University of Helsinki,2018. 13 – 14.

作者简介

王金艳　女,1982 年 1 月 2 日生,广西资源人. 现为广西师范大学计算机科学与信息工程学院副教授,主要研究方向为自动推理和数据安全.
E-mail: wangjy612@ gxnu. edu. cn

胡　春　男,1995 年 3 月 1 日生,江西南昌人. 现为广西师范大学计算机科学与信息工程学院研究生,主要研究方向为自动推理和人工智能.
E-mail: huchungxnu@ 163. com

牛当当　男,1990 年 2 月出生,陕西周至人. 现为西北农林科技大学信息工程学院讲师,主要研究方向为自动推理和抽象论辩.
E-mail: niudd@ nwafu. edu. cn

李先贤(通信作者)　男,1969 年 9 月出生,广西阳朔人. 现为广西师范大学计算机科学与信息工程学院教授,主要研究方向为形式化理论和数据安全.
E-mail: lixx@ gxnu. edu. cn