基于搜索信息反馈策略的MaxSAT非完备求解算法

徐振兴[1)] 何 琨[1)] 李初民[1),2)] 刘燕丽[3)] 郑迥之[1)]

[1)](华中科技大学计算机科学与技术学院 武汉 430074)

[2)](亚眠大学计算机科学系 亚眠 法国 80039)

[3)](武汉科技大学理学院 武汉 430081)

摘 要 MaxSAT问题是SAT可满足性问题的优化形式,具有NP难度.本文分析了传统的MaxSAT局部搜索求解器对工业算例求解存在的局限性,并基于此分析提出了新的初始解构造算法ASIF.ASIF是一个基于树形赋值的初始解构造算法,其中包含了一个全局信息反馈策略.该算法选取并定义了构造过程中有意义的统计量,使用这些量设计了一个全局搜索信息更新反馈机制,对初始解构造过程中的经验进行积累并为后续解的构造提供指导信息,再根据后续解的构造情况对全局经验进行反馈和更新,从而有效利用了解构造过程中的经验和信息.进一步地,将ASIF作为初始解构造算法,结合IPBMR算法中的路径截断(PB)策略,提出了新的算法PB-ASIF.实验设计与比较共分为三个阶段.第一阶段,将ASIF在300秒内首次找到的可行解与IPBMR求解300秒的结果进行对比.ASIF初始可行解更优的数量是IPBMR在300秒内求解的可行解更优数量的两倍多,其中非加权偏类算例更优解数量上前者更是后者的3.68倍.该阶段的实验结果表明,ASIF算法能快速构造优质的初始可行解.第二阶段,将PB-ASIF与IPBMR进行对比实验,在300秒求解时间内,PB-ASIF求得更优解的数量总体上是IPBMR的2.38倍,在非加权偏类算例更优解数量上前者更是后者的3.85倍.该阶段的实验结果表明,PB-ASIF算法求解工业算例的能力明显超过了IPBMR算法,有效改进了使用PB策略求解工业算例的效果.第三阶段,将PB-ASIF与其它优秀求解器进行联合求解,包括CCEHC求解器和SATLike3.0求解器.该阶段的实验结果表明,PB-ASIF算法与其它局部搜索类算法有很强的互补性,有提升其它求解器求解效果的能力.

关键词 组合优化;最大可满足性问题;非完备算法;搜索信息反馈;赋值算法

中图法分类号 TP301 **DOI号** 10.11897/SP.J.1016.2023.00711

Incomplete Solver based on Search Information Feedback for MaxSAT

XU Zhen-Xing[1)] HE Kun[1)] LI Chu-Min[1),2)] LIU Yan-Li[3)] ZHENG Jiong-Zhi[1)]

[1)](*Department of Computer Science, Huazhong University of Science and Technology, Wuhan* 430074)

[2)](*Department of Computer Science, University of Picardie Jules Verne, Amiens, France* 80039)

[3)](*Department of Science, Wuhan University of Science and Technology, Wuhan* 430074)

Abstract MaxSAT problem is the optimization version of the SAT problem, and it is NP hard. There are two classes of algorithms for solving the MaxSAT problem, the complete algorithms and incomplete algorithms. A complete algorithm requires to return a feasible solution that is optimal. An incomplete algorithm requires to return the best-quality feasible solution within the specified time or steps. The two classes of algorithms have made great progress in recent

收稿日期:2022-09-06;在线发布日期:2022-12-06.本课题得到科技部高端外国专家引进计划项目智能学习与优化核心算法研究(No. G2022154012L)、微软亚洲研究院联合研究基金基于学习的组合优化问题求解(No. 100338928)资助.**徐振兴**,博士,主要研究领域为NP难度问题求解、组合优化.E-mail:lxadd515@hust.edu.cn.**何 琨**(通信作者),博士,教授,中国计算机学会(CCF)杰出会员,主要研究领域为机器学习、对抗学习、组合优化、图数据挖掘.E-mail:brooklet60@hust.edu.cn.**李初民**,博士,教授,主要研究领域为NP难度问题求解、组合优化.**刘燕丽**,博士,副教授,中国计算机学会(CCF)会员,主要研究领域为NP难度问题求解、组合优化.**郑迥之**,博士研究生,主要研究领域为NP难度问题求解、组合优化.

decades. The local search algorithm is a typical and effective kind of incomplete algorithm. The structure of different types of MaxSAT instances varies greatly, making it hard for a method to have good performance on all types of instances. This paper analyzes the limitations of traditional local search MaxSAT solvers in solving industrial instances, and proposes a new algorithm for the initial solution construction based on the analysis, termed ASIF (Assignment approach by Search Information Feedback). ASIF is an initial solution construction algorithm based on tree assignment, which includes a global information feedback strategy. This algorithm selects and defines meaningful quantities in the construction process, uses these quantities to design a global search information update feedback mechanism, accumulates the experience in the construction process of the initial solution and provides guidance information for the construction of subsequent solutions, and then feeds back and updates the global experience according to the construction of subsequent solutions. The goal is to effectively use the experience and information in the construction process. Taking ASIF for the initial solution construction, we combine it with the Path-Breaking (PB) strategy in IPBMR algorithm, and propose a new algorithm called PB-ASIF (Path-Breaking approach with ASIF strategies). There are three stages in our experiments. In the first stage, the first feasible solution found by ASIF in 300 seconds was compared with the best solution found by IPBMR in the same time limit. The number of better feasible solutions found by ASIF is more than twice of the number found by IPBMR in 300 seconds. More significantly, the number of better solutions of ASIF is 3.68 times of IPBMR on unweighted partial benchmarks. The experimental results in this stage show that the ASIF algorithm can quickly construct high-quality initial feasible solutions. The second stage is designed to compare PB-ASIF with IPBMR. In the 300 seconds of time limit, the number of better solutions found by PB-ASIF is 2.38 times of IPBMR in general, and the number of better solutions found by PB-ASIF is 3.85 times of IPBMR in unweighted partial benchmarks. The results show that the ability of PB-ASIF for solving industrial instances is significantly higher than IPBMR, which effectively improves the effect of using the PB strategy for solving industrial instances. The third stage is to combine PB-ASIF with other excellent solvers, including CCEHC solver and SATLike3.0 solver, and then split the 300 seconds of the time limit, so that the two solvers take up a half time respectively. The results also show that PB-ASIF has strong complementarity with other local search algorithms, that it can improve the solution quality of other solvers.

Keywords combinatorial optimization; maximum satisfiability problem; incomplete algorithm; search information feedback; assignment approach

1 引 言

最大可满足性问题(Maximum Satisfiability problem, MaxSAT)是著名的可满足性问题(Satisfiability problem, SAT)的优化形式. SAT是第一个被证明为NP完全(NP-complete)的问题[1], MaxSAT问题是典型的NP难(NP-hard)问题. MaxSAT和SAT都是以命题逻辑合取范式(Conjunctive Normal Form, CNF)为研究对象,设置约束条件和求解目标形成的约束模型. SAT的求解目标是在约束条件下判定是否存在一个解使得给定的CNF满足,是一个约束判定模型. MaxSAT的求解目标是在约束条件下找到一个解使得给定的CNF满足的子句数最多,是一个约束优化模型.

CNF的基础元素是布尔变元,布尔变元通常用小写字母表示,如x、l. 变元有两种文字(Literal),正文字和负文字. 通常记变元x的正文字为x,负文字为$\neg x$. 文字的析取构成子句(Clause),子句的合取构成CNF. CNF中的子句约束条件不同,形成不同的MaxSAT算例类型. 在CNF中,要求某些子句在求解时必须被满足,这些子句称为硬子句(Hard

Clause)，其它子句在求解时可以不满足，这些子句称为软子句(Soft Clause). 若CNF中含有硬子句，则称该CNF为偏的(Partial)，否则称为非偏的(Non-partial). 给CNF中每个软子句添加正整数倍的权重(Weight)，若这些软子句的权重不全为1，则称该CNF是加权的(Weighted)，否则称为未加权的(Unweighted). 根据求解的CNF的不同属性，MaxSAT问题分成了不同的类别，包括Weighted Partial MaxSAT(WPM)、Partial MaxSAT(PM)等，一般不注明加权的即默认为未加权的，不注明偏的即默认为非偏的.

MaxSAT在学术研究领域和工业应用领域都展现出了重要的价值. MaxSAT问题的定义特别基础而且抽象，因此有极其强大的表达能力，很多学术界及工业界的问题都能转化为MaxSAT问题求解. 在学术研究领域，许多其它组合优化问题，如最大团问题[2]、相关聚类问题[3]、最小顶点覆盖问题[4]等，都能经过编码转化为MaxSAT问题，而且MaxSAT求解技术也能直接或者间接应用[5-7]在这些问题的求解上，促进了这些组合优化问题求解技术的发展与进步. 在工业应用领域，随着求解技术的不断进步，MaxSAT能高效地解决更多的工业问题. MaxSAT求解技术广泛应用在各行各业，应用领域包括电路设计自动化[8-9]、优化癌症治疗设计[10]、最小校正集计数问题[11]、企业对企业会议安排[12-13]、高校课表排班问题[14]、组缺陷测试[15]、检测硬件木马[16]、线性时序逻辑[17]等.

求解MaxSAT问题的算法分为两类，一类是完备(Complete)算法，这类算法要求返回可行解并保证其是最优解；另一类是非完备(Incomplete)算法，这类算法要求在规定求解时间或步数内返回质量最好的可行解. 这两类算法经过多年的发展都取得了很好的成绩. MaxSAT问题的非完备算法不需要保证可行解的最优性，因此理论上它能更快地找到一个质量更好的可行解. 非完备算法的发展可以根据问题自身特点进行策略研究[18]，也可以借鉴其它问题中的启发式策略[19-21]. 非完备算法在算例规模不大时能发挥很好的求解效果，能在短时间内找到更优质的解.

MaxSAT的非完备求解领域在国内外研究者们的共同努力下，近年来发展迅速[22]，出现了GRASP[23]、CCLS[24]、IPBMR[25]、Dist[26]、CCEHC[27]、Swcca[28]、SATLike[29-30]等不断优化的非完备求解器. MaxSAT非完备求解器在工业大算例上的表现在很长一段时间里都比不过完备算法. 工业算例不仅规模大而且结构更加复杂，不同类型的算例差异特别大，非完备算法在设计时很难兼顾到所有算例的特点. 因此，非完备算法在优化中小型算例的基础上也要根据大算例的特点进行技术改进. 以IPBMR为例，它提出了新的路径搜索类策略，即路径截断策略(Path Breaking，PB)，然后结合重启策略和扰动策略并进行调整形成有效的求解算法. IPBMR有效提升了非完备求解器对中小算例的求解效果，对比代表性的非完备求解器CCLS，求解效率有较大提升[25]. 对于工业算例的求解，IPBMR虽然有一定的效果，但对比完备求解器，其求解效率未能明显提升.

本文对IPBMR[25]求解工业算例存在的弱点进行分析，并针对其弱点提出了新的初始解构造算法，即基于搜索信息反馈的赋值算法(Assignment approach by Search Information Feedback，ASIF)，然后将其作为初始解构造子算法，结合路径截断策略形成新的算法，即基于搜索信息反馈策略的路径截断算法(Path-Breaking approach with ASIF strategies，PB-ASIF). 实验结果表明，PB-ASIF能有效改善IPBMR求解工业算例的效果，而且与其它求解器的联合求解实验也展示了与这些求解器的互补性，表明其具有改进其它求解器求解效果的能力.

2　求解工业算例的分析及改进思路

本节通过设计实验对IPBMR求解工业算例的细节进行分析，剖析其求解工业算例效果欠佳的原因，并针对该原因提出改进思路. 本节首先针对作为重启迭代算法的IPBMR的重启及迭代次数进行统计分析，展示其求解过于低效的搜索环节. 然后针对该低效环节对各个策略的耗时进行统计分析，确定造成求解过于低效的直接原因. 最后针对分析出的问题原因进行优化，提出相应的改进思路.

2.1　IPBMR求解工业算例的实验分析

从IPBMR论文[25]的实验部分可以看出，IPBMR求解MaxSAT Evaluation 2016(MSE2016)①工业算例有很大一部分无法找到可行解，或者找到的解的质量不高. 本小节对IPBMR求解MSE2016工业算例

① MaxSAT Evaluation 2016, http://www.maxsat.udl.cat/16, 2016

过程中的数据进行统计，从而展示其中的短板并分析原因．实验配置为Linux系统，处理器为Intel Xeon CPUs E5-2680@2.40 GHz，10 GB运行内存．实验算例集选取MSE2016的所有工业算例．如无特殊说明，本文所有实验采用同样的配置和算例集．

重启和迭代是IPBMR搜索中的主要框架，PB是框架中的主要搜索策略．首先对所有工业算例在300 s求解时间内的一些参数进行统计和分析，包括平均重启次数和重启的平均迭代PB搜索的次数，单次PB的初始平均候选变元数占比和单次PB搜索时间等．重启次数反映了迭代PB搜索的轮次，一轮迭代PB搜索包含多轮PB搜索，一轮PB搜索包含从初始解开始搜索到路径截断时的整个路径搜索的过程．平均迭代次数即每一轮迭代PB搜索内的平均PB搜索轮次．重启次数越多表示搜索的空间越大，搜索广度更强，迭代次数越多说明单次重启内需要多次PB搜索才能到达局部最优点．单次PB的初始候选变元数目占比是指候选变元数目占总变元数的比值，反映了PB搜索可能会经历的搜索路径长度占最长路径长度的比例，单次PB时间则反映了具体求解时间．

IPBMR的运行统计数据如表1所示，其中第一列是算例类型，算例包括MSE2016的不加权（Unweighted Instance，UI）、偏（Partial Instance，PI）、加权偏（Weighted Partial Instance，WPI）的工业算例集，第二列表示对应算例集中的总算例数．后五列数据分别是求出可行解的算例平均重启次数“avg. restart_fe”、未求出可行解的算例平均重启次数“avg. restart_unfe”、所有算例单轮重启内平均PB次数“avg. PB”、所有算例单次PB的初始候选变元数占总变元数比例均值“avg. cand_ratio”、总PB搜索次数大于1次的算例数、所有PB搜索次数大于1次的算例中单次PB路径搜索平均时间“avg. time_PB”．

表1 IPBMR运行统计数据

算例类型	总算例数	avg. restart_fe	avg. restart_unfe	avg. PB	avg. cand_ratio	PB＞1	avg. time_PB
UI	55	1	—	75.428571	0.351226	7	0.168676
PI	601	5955	888	296041.436	0.434784	508	0.138415
WPI	630	2979	4405	201388.266	0.420070	499	0.463492

从表1中可以看出，对于UI类算例，它的平均重启次数不大于1，说明搜索一直陷在迭代PB中，搜索广度极差．可以注意到UI算例集中总PB数大于1的算例只有7个，说明有很多算例陷入在首轮重启的第一个PB搜索里，即单次PB的搜索时间超过300 s．而PI和WPI中也有很多算例有这种情况，这个现象可以从avg. cand_ratio列的数据找到原因，这列展示的是所有算例单次PB的初始候选变元数占比均值．虽然三类算例的占比为百分之三四十，但这些工业算例的变元数大多在数千甚至数百万，因此单次PB要经历这么长的路径会非常耗时．而且由于PB每一步选取变元的策略要么是贪心选取，要么是依概率选取，但是都会对所有候选变元进行处理，候选变元过多也会造成单步耗时增加，以至于PB无法在较短时间发挥其搜索能力．需要说明的是，有极少算例由于预处理耗时太长没进入搜索阶段因而没有进行统计．

表1中还可以看到PI和WPI的平均重启次数达到了上千，但实际算例求解统计数据中的重启次数的数值差距很大，而且大多数的重启次数都很少．进一步地，分段统计了一定重启次数内的算例数，如表2所示．

表2 重启次数分段统计表

算例类型	＜10	＜100	＜1000	⩾1000
UI	54	0	0	0
PI	322	157	68	49
WPI	466	68	35	42

表2中将算例数按求解中的重启次数分四段进行了统计．可以看出，三类算例集中重启次数少于10次的算例超过了半数以上，这说明大多数的算例在求解时的搜索广度特别低，因此求解的效果很差．而这其中的原因还是由于单次PB的初始候选变元数目过大导致的搜索区域过于集中，且局部区域的搜索耗时过大导致全局的搜索没有时间去完成．

2.2 改进思路

首先，变元的选取策略需要重新设计与优化．工业大算例变元出现的平均次数比较少，而且影响力也较小，即翻转某个变元造成的不满足子句数的变化不会太大，其值不大于在子句中出现的次数．大部分的变元在子句中出现的次数都差不多，因此候选变元之间的重要性通常不会在单次路径截断搜索的

候选变元分数里体现出来，即大部分的候选变元分数差距都不大. 此时在单步选取较大分数的候选变元对解的优化效果不大，而且依概率选取造成的差距也不会太大，在整体求解中的重要性也特别有限.

其次，需要设计一个快速提升解的质量的策略. 由于变元之间的影响相对较小，候选变元一般是分数大于0的变元，翻转之后会使得不满足的子句减少. 候选变元越多说明当前解提升的空间越大，过多的候选变元说明当前解的质量不高. 设计合理的策略，快速降低候选变元数目，提升搜索空间的解的质量，节省时间使搜索更注重广度，这是提升搜索效率的关键.

设计一个全局搜索信息积累以指导搜索方向也是一个需要考虑的改进点. 正因为变元之间的影响较小，单步变元选择策略更需要全局信息的支持. 定义并设置一定的影响因素，利用全局信息对每一轮搜索的影响进行预测性分析，在每轮搜索之后，对实际造成的影响进行统计然后更新全局信息，这样形成一个有反馈机制的全局策略，能不断修正搜索目标，找到更好的解.

3　初始解构造算法

本节介绍提出的基于搜索信息反馈策略的初始解构造算法. 首先对信息反馈策略中用到的统计量进行定义，然后详细介绍使用这些量进行初始解构造的算法.

3.1　基本定义

定义1.　文字的不满足度. 是描述给定CNF中文字不满足概率的估计量，完全解中所有变元随机赋值的情形下，文字不满足度可视为其反文字出现次数的占比，文字x关于软子句的不满足度计算如式(1).

$$d_s(x)=\frac{nbs(\neg x)}{nbs(x)+nbs(\neg x)} \tag{1}$$

其中，$d_s(x)$是文字x关于软子句的不满足度，$nbs(x)$是该文字在软子句中出现的次数，$nbs(\neg x)$是该文字的反文字在软子句中出现的次数. 文字x关于硬子句的不满足度记为$d_h(x)$，其计算与式(1)类似，只是统计的文字出现次数是在硬子句中出现的次数.

文字的不满足度表示了文字不满足的可能性，不满足度越大表示该文字越可能被不满足. 文字被赋值可能的状态有满足或者不满足两种状态，文字满足度与不满足度之和为1，即文字的满足度在数值上等于其反文字的不满足度. 文字的满足度大于其不满足度表示若使该文字满足时可能造成的满足子句数比使该文字不满足时造成的满足子句数大.

定义2.　文字的联合不满足度. 是文字关于硬子句和软子句中不满足度的加权平均. 式(2)是文字x的联合不满足度的计算公式，其中$\alpha(x)$是变元的硬偏好量，$\alpha(x)\in[0,1]$.

$$d_l(x)=\alpha(x)\cdot d_h(x)+(1-\alpha(x))\cdot d_s(x) \tag{2}$$

文字x的联合不满足度记为$d_l(x)$，这个量将文字在硬子句和软子句中的不满足度结合了起来，变元的硬偏好量$\alpha(x)$用来调节联合不满足度中关于硬子句不满足度的比重，文字的联合不满足度用来表示文字的联合不满性.

定义3.　文字的贡献度. 是用来度量文字造成子句满足的总贡献的估计量. 给定算例φ和一个完整赋值S，造成子句满足的所有文字均分该子句的权重作为文字的贡献，文字在所有被包含子句中的贡献总和为该文字在赋值S下的贡献度. 记子句cls中满足的文字数为$nbsl(cls)$，权重为$w(cls)$，则文字x的贡献度记为$C_l(x)$，具体计算如式(3)所示.

$$C_l(x)=\sum_{x\in cls}\frac{w(cls)}{nbsl(cls)} \tag{3}$$

文字的贡献度分别针对软子句和硬子句进行计算，软子句权重是由算例给定的. 硬子句的权重设定只要保证所有硬子句的权重相同，本文将所有硬子句的权重设为1来进行计算.

定义4.　子句的不满足度. 子句不满足等价于子句中所有文字均不满足，由此定义子句cls的不满足度$d_c(cls)$为所有文字的联合不满足度的乘积，即式(4)，其中β是不满足系数，$\beta(cls)\in[0,2]$.

$$d_c(cls)=\beta(cls)\cdot\prod_{x\in cls}d_l(x) \tag{4}$$

定义5.　CNF的不满足度. 对于MaxSAT问题，要求满足给定算例φ中所有硬子句的情况下，满足尽量多的软子句. 由此在求解过程中要求不满足的硬子句减少到零，不满足的软子句尽量少. 基于这个目标，定义算例φ的不满足度$d_{CNF}(\varphi)$的计算公式如式(5)所示.

$$\begin{aligned}d_{CNF}(\varphi)&=\prod_{cls\in\varphi}d_c(cls)\\&=\prod_{cls\in\varphi}\Big(\beta(cls)\prod_{x\in cls}d_l(x)\Big)\end{aligned} \tag{5}$$

其中$d_{CNF}(\varphi)$的参数集记为θ,在求解过程中要求不满足子句出现的概率越小越好,因此求解目标函数设为$min\, d_{CNF}(\varphi|\theta)$,然后在求解过程中不断反馈按式(6)来更新参数集$\theta$以使得算例$\varphi$的不满足度减小.

$$\theta=\arg_{\theta}\min d_{CNF}(\varphi|\theta) \tag{6}$$

对于经过单子句传播和纯文字消解等预处理后的算例,$d_l(x)$的取值在区间(0, 1)中,因此$d_{CNF}(\varphi)$的取值也在区间(0, 1)中.对目标函数进行简单的变换,两边取对数不改变原函数单调性,记变换后结果为$f_{CNF}(\varphi)$,变换过程见式(7).经过变换之后,$f_{CNF}(\varphi)$的形式由$d_{CNF}(\varphi)$的连乘形式变成了连加的形式,这种变换有利于实际计算和编程实现.

$$\begin{aligned} & f_{CNF}(\varphi) \\ &=\min\ln d_{CNF}(\varphi) \\ &=\min\ln\left(\prod\nolimits_{cls\in\varphi}\left(\beta(cls)\prod\nolimits_{x\in cls}d_l(x)\right)\right) \\ &=\min\sum\nolimits_{cls\in\varphi}\left(\ln\beta(cls)+\ln\prod\nolimits_{x\in cls}d_l(x)\right) \\ &=\min\sum\nolimits_{cls\in\varphi}\left(\ln\beta(cls)+\sum\nolimits_{x\in cls}\ln d_l(x)\right) \end{aligned} \tag{7}$$

3.2 初始解构造算法框架

本小节描述初始解构造算法ASIF.它是一个树形赋值算法,包含一个硬子句集处理子算法和一个软子句集处理子算法.两个子句处理子算法中的策略由2.1节所定义的不满足度等量构成,包含子句选择策略、变元选择策略以及多个反馈更新策略.

图1给出了ASIF算法的流程图;算法1的伪代码给出了ASIF算法的框架.

ASIF算法的输入参数包括算例φ、时间限制参数MAX_TIME以及参数集θ.θ包含文字联合不满足度中的硬偏好参数$\alpha(x)$的校正参数α_c,子句的不满足系数$\beta(cls)$的校正参数β_c等.ASIF算法输出为找到的可行解S.

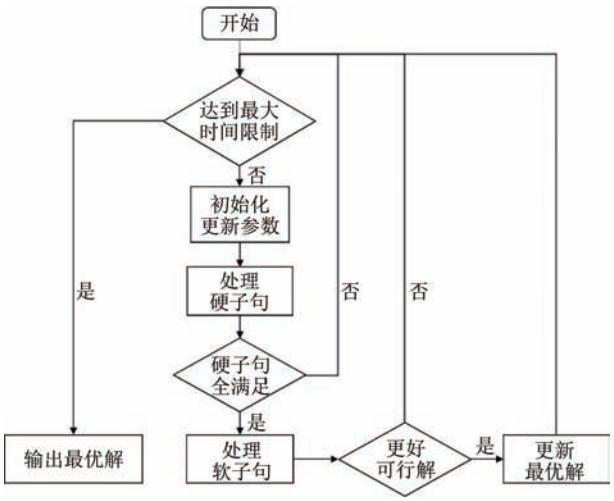

图1 ASIF算法的流程图

算法ASIF是一个树形赋值算法,是通过依次选择变元进行赋值的形式进行迭代直到找到一个完整可行解.算法的总体思想是从所有变元未赋值状态开始,每次按策略选择一个子句,然后从子句中选择一个文字令其满足,从而使得该子句满足.而且算法优先处理硬子句,变元赋值主要由硬子句处理和软子句处理两个子算法构成.

算法1. *ASIF*(φ, θ, *MAX_TIME*)

输入:MaxSAT算例φ,最大时间限制*MAX_TIME*,参数集θ;

输出:算例φ的可行解S.

1. 初始化$\alpha(x)$, $\beta(cls)$, *var_cht*, *cls_cht*;
2. *SAT_mark* ← 0;*feasible_mark* ← 0;
3. WHILE ! *MAX_TIME* DO
4. 初始化S';
5. *SAT_mark* ← *Cal_Hard_Clause*(φ, S', *var_cht*, *cls_cht*); *处理硬子句*\
6. IF *SAT_mark* THEN
7. *feasible_mark* ← *Cal_Soft_Clause*(φ, S', *var_cht*, *cls_cht*); *处理软子句*\
8. IF *feasible_mark* THEN
9. S ← S';
10. *Renew*($\alpha(x)$, $\beta(cls)$, *var_cht*, *cls_cht*);
11. RETURN S;

首先,算法ASIF进行初始化(第1~2步).包括变元的硬偏好值、子句不满足系数、变元特征值数组*var_cht*以及子句特征值数组*cls_cht*等量的初始化.这些量是硬子句和软子句处理子算法中子句选择和文字选择的数值依据,在2.3节详细介绍.

然后,算法进入迭代更新赋值部分.S'分步记录变元的赋值,初始时将S'中的变元赋值设置成未赋值标记值-1(第3~4步),变元被赋值后标记成0或1,因而在此过程中S'是一个部分赋值.ASIF先用子程序*Cal_Hard_Clause*处理算例φ中的硬子句集.将S'中的部分变元赋值以求使所有硬子句满足,选择变元及赋值的策略会考虑对软子句集的影响,在第2.3节详细介绍.当*Cal_Hard_Clause*程序处理完硬子句集(第5步),若出现硬子句不满足将返回值*SAT_mark*设为0,否则设为1.当*SAT_mark*为0时,用子算法*Renew*进行参数更新.当

*SAT_mark*为1时，用子程序*Cal_Soft_Clause*处理软子句.

子程序*Cal_Soft_Clause*处理完软子句，返回值*feasible_mark*记录处理结果. 若*feasible_mark*等于1，表示所有变元都已赋值，且解S'优于当前最优解S，将S更新成S'. 然后用*Renew*子算法进行参数更新，进入下一轮迭代(第6～10步). 当算法达到了最大限制时间，输出当前找到的最好解S(第11步).

3.3　硬子句集和软子句集处理子算法

硬子句和软子句处理算法主要是基于减小CNF不满足度的思想，尽量减小硬子句集和软子句集的不满足度. 本小节详细介绍ASIF算法中的硬子句集处理算法.

算法2.　*Cal_Hard_Clause*(φ, S', *var_cht*, *cls_cht*)

输入：MaxSAT算例φ，一个部分解S'，*var_cht*，*cls_cht*；

输出：*SAT_mark*.

```
1.  SAT_mark ← 1;
2.  WHILE存在未处理硬子句DO
3.    IF存在未处理单硬子句THEN
4.      CLS←不满足度最大的未处理硬单子句;
5.    ELSE
6.      CLS ←不满足度最大的未处理硬子句;
7.    LIT ←硬子句CLS中贡献度最大文字;
8.    assign_var(LIT);  \*赋值令该文字为真*\
9.    IF break_condition THEN
10.     SAT_mark ← 0;
11.     BREAK;
12.  RETURN SAT_mark;
```

算法2给出了硬子句集处理算法的伪代码. 输入参数包括算例φ、部分赋值S'以及变元特征值数组*var_cht*和子句特征值数组*cls_cht*. 变元特征数组*var_cht*包含常量——文字的不满足度，以及变量——文字的联合不满足度和文字的贡献度. 子句特征数组*cls_cht*包含变量——子句的不满足度，这些量的定义和计算方法在2.1节中已经进行了描述.

算法2中若使得所有硬子句满足则返回1，否则返回0. 对于给定的算例和当前传入的部分赋值S'，先将*SAT_mark*赋值为1. 算法先判断是否存在未处理的子句，按设定的规则选择一个未被满足的子句，若存在单子句则优先选取单子句. 然后在选出的子句中按设定的规则选择一个未赋值的文字，即自有文字，并令该文字满足. 接着判断停止条件是否达到，若未达到则继续下一轮，否则停止赋值，令*SAT_mark*为0并输出.

算法2中优先选择候选子句集中不满足度大的子句，这样能优先令该子句满足. 选取文字时优先选择贡献度最大的文字. 子句优先选取的是在动态阈值内候选子句的最优，即将阈值内的候选子句记录在列，并清除超出阈值的候选子句，动态更新阈值使候选子句数保持在一定的比例内，从而减少计算量.

算法2中停止条件*break_condition*指的是对当前不满足硬子句进行判断，若出现不满足硬子句，则停止条件满足. 此时继续赋值找不到可行解，算法会及时截断赋值操作，然后对当前轮赋值信息以及子句状态进行统计来更新参数.

软子句集的处理子算法和硬子句集处理子算法类似，如算法3所示，输入参数中的S'是经过硬子句处理后的部分赋值. 其输出参数为*feasible_mark*，用来表示是否找到更优可行解. 与处理硬子句程序不同的是，它在选子句时不优先处理单子句.

算法3.　*Cal_Soft_Clause*(φ, S', *var_cht*, *cls_cht*)

输入：MaxSAT算例φ，一个部分解S'，*var_cht*，*cls_cht*；

输出：*feasible_mark*.

```
1. feasible_mark ← 1;
2. WHILE存在未处理软子句DO
3.    CLS ←不满足度最大的未处理软子句;
4.    LIT ←软子句CLS中贡献度最大文字;
5.    assign_var(LIT); \*赋值令该文字为真*\
6.    IF break_condition THEN
7.      feasible_mark ← 0;
8.      BREAK;
9. RETURN feasible_mark ;
```

3.4　参数反馈更新策略

本小节通过介绍ASIF中的参数更新子程序*Renew*($\alpha(x)$, $\beta(cls)$, *var_cht*, *cls_cht*)来详细描述信息反馈更新策略，它是全局信息反馈更新策略的主体部分. 这里更新的参数包含每个变元的硬偏好量$\alpha(x)$、每个子句的不满足系数$\beta(cls)$，以及变元特征数组*var_cht*和子句特征数组*cls_cht*.

首先，介绍参数更新的思想. 变元的硬偏好量影响了其文字的联合不满足度对于硬子句的偏向，若硬子句不满足，则应该调整每个变元的硬偏好量$\alpha(x)$使变元赋值更注重硬子句的满足. 而文字的联合不满足度会通过2.1节中的定义4叠加到子句的不满足度上，相当于对子句的不满足度进行了微调. 每个子句的不满足系数$\beta(cls)$则是对子句不满足度的一个粗调，它在软硬子句的处理中直接乘以子句的不满足度，从而直接影响其值，若某个子句不

满足,则应该增大该子句的不满足系数,使得之后的子句选取中优先处理.文字的贡献度则是根据已处理子句的最终状态进行更新,子句最终状态的每个文字的贡献是确定的,此时文字对于当前解的贡献度也是确定的,因此将当前的贡献度按比例更新到全局贡献度中.

通过以上思路,算法中制定了详细的参数值反馈更新策略.首先介绍这些值的初始设置,每个变元的硬偏好量$\alpha(x)$是独立的,它反映的是对硬子句的偏向,因此在初始时都赋值为0.6,在后续算法中用校正参数α_c进行更新,且值域限制为$\alpha(x)\in[0.1, 0.9]$.根据2.1节中的定义,文字针对硬子句和软子句的不满足度是根据算例数据计算之后的常量,文字联合不满足度依据文字不满足度数值进行计算,也是每个文字分别进行计算,包括变元的正文字和负文字.然后再根据定义和文字不满足度的值将子句的不满足度$\beta(cls)$的值计算出来,每个子句的不满足度都有一个不满足系数用来粗调,其初始值为1,然后在后续算法中用校正参数β_c进行更新,在算法中的子句不满足度是乘以系数之后的值.

在硬子句和软子句处理算法中,选定子句后,选取贡献度$C_l(x)$最大的文字.文字的贡献度是软硬子句分开计算的,在初始时假设所有子句中所有文字都满足,从而计算出文字贡献度的初始值,这个值作为全局文字贡献度.在软硬子句处理过程中,选择最大贡献度的文字时,依据当前选定的子句中的文字的所有子句的情况计算出一个实时文字贡献度的值.去除包含该文字的原始子句中其它已经处理的文字,再按贡献度的定义进行计算.去除的规则按照文字赋值的消解规则进行,即原始子句中其它文字可能因为赋值而不满足,去除这类文字.部分子句因除本文字外的文字赋值而满足,则该子句不计入本文字的实时贡献度.然后计算加权贡献度$C_l(x)=\gamma_0\cdot C_g(x)+(1-\gamma_0)\cdot C_c(x)$,其中$C_g(x)$是全局贡献值,$C_c(x)$是实时贡献值,$\gamma_0$是值为固定参数.比较文字的$C_l(x)$来选取文字,这样能兼顾实时信息进行选择,而且此处计算不改变全局贡献值$C_g(x)$,只计算$C_l(x)$作为选取文字的参考权值,本文中令γ_0为0.7,使$C_l(x)$稍微偏向全局信息.

Renew($\alpha(x)$, $\beta(cls)$, *var_cht*, *cls_cht*)子程序是根据本轮赋值后的子句状态信息进行反馈更新.对于变元的硬偏好值$\alpha(x)$的更新,先针对每个变元分别统计包含该变元的不满足的软子句和硬子句数,并分别找出最大不满足子句数,然后通过不满足硬子句对$\alpha(x)$用式(8)进行更新.其中校正参数α_c值设置为0.95,*nb_maxunsath*是包含该变元的不满足硬子句数,*nb_maxunsath*是最大不满足硬子句数,$\alpha^{t-1}(x)$是当前$\alpha(x)$值,$\alpha^t(x)$是更新后$\alpha(x)$的值.若变元没有出现在不满足硬子句中或出现的不满足硬子句数比较少,则$\alpha(x)$会降低,否则$\alpha(x)$的值会增大.

$$\alpha^t(x)=\alpha^{t-1}(x)\cdot\left(\alpha_c+\frac{nb\_unsath}{2\cdot nb\_maxunsath}\right) \tag{8}$$

对于不满足软子句用式(9)进行更新,其中*nb_maxunsats*是包含该变元的不满足软子句数,*nb_maxunsats*是最大不满足软子句数.当*SAT_mark*值为1时,两个更新都进行;*SAT_mark*值为0时,只进行不满足硬子句的$\alpha(x)$更新,这样能在有硬子句不满足时尽量偏向该硬子句的处理.在更新时保证$\alpha(x)$的值在[0.1, 0.9]之间,超限制则设为0.1或者0.9直到更新方向往值域内.

$$\alpha^t(x)=\alpha^{t-1}(x)\cdot\left(1-\frac{nb\_unsats}{2\cdot nb\_maxunsats}\right) \tag{9}$$

子句的不满足系数$\beta(cls)$的更新也是基于不满足子句信息,即统计不满足子句信息,用式(10)进行更新.其中校正系数β_c值设置为0.3,$\beta(cls)$值域在求解时设置为[0.01, 2],$\beta^{t-1}(cls)$是当前$\beta(cls)$的值,$\beta^t(cls)$是更新后的值.当有硬子句不满足时只进行不满足硬子句的系数更新,当没有不满足硬子句时才对不满足软子句的系数进行更新.在更新过程中,若超值域限制,则设置为0.01或2,超最小限制达到5个就将所有硬子句或软子句系数进行平滑处理,即同时乘以5.这样能保证子句的不满足系数随每次赋值过程进行持续更新.

$$\beta^t(cls)=\beta^{t-1}(cls)\cdot\beta_c \tag{10}$$

更新完$\alpha(x)$和$\beta(cls)$后就能对文字的联合不满足度和子句不满足度进行更新.而文字的全局贡献度$C_g(x)$的更新则是根据满足子句中的情况进行更新,当有不满足硬子句时只更新文字的硬子句贡献度,否则更新文字的软子句贡献度.将满足的子句的权重按该子句中满足的文字数进行均分获得每个文字的满足贡献值,然后将贡献值累计到相应的文字,最后得到每个文字的后验贡献度值$C_p(x)$.全局贡献值$C_g(x)$按式(11)进行更新,其中$C_g^t(x)$是更新后的全局贡献度值,$C_g^{t-1}(x)$是原值,γ_1是固定

参数，本文中令其值为0.8.

$$C_g^t(x)=\gamma_1\cdot C_g^{t-1}(x)+(1-\gamma_1)\cdot C_p(x) \quad (11)$$

4　结合ASIF与PB策略求解算法

本节将ASIF结合PB策略形成非完备算法PB-ASIF，由ASIF构造初始解，将其输出的可行解作为PB初始解进行局部搜索. 算法PB-ASIF对比ASIF增加了当条件*condition0*满足时对θ的初始化，以及当条件*condition1*满足时进行PB搜索.

算法PB与IPBMR中的PB算法相同. 图2给出了PB算法的流程图. 对于给定的起始解，将所有变元作为候选变元，在每一步选取并翻转候选变元，然后判断截断条件是否达到，若达到则输出找到的最优解. 变元选择优先选取分数（翻转该变元后满足子句的净增量）为正的变元，以概率P选取分数最大的正分数变元，以概率$1-P$对所有正分数变元依概率选取. P是给定的概率，对于加权算例为0.2，加权偏算例为0.99. 正分数变元集合记为$PosVars$，每个正分数变元的分数记为$score(x)$，定义函数$f(x)=score^2(x)$，正分数变元的概率定义为$\frac{f(x)}{\sum_{y\in posVars}f(y)}$. PB算法中截断条件为$\alpha\cdot lastPos\leqslant|totalLoss|$，其中$\alpha=3$.

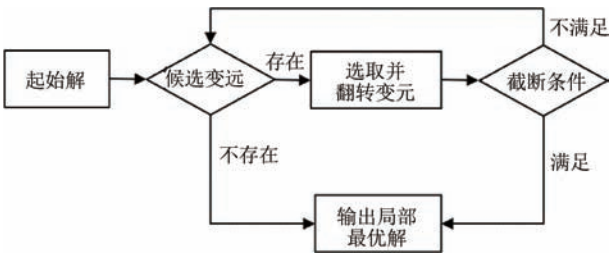

图2　PB算法流程图

算法4给出了PB-ASIF算法的框架. 其中条件*condition1*是指初次进行PB或解的质量可以优化的情况. 而条件*condition0*是指解的质量经过100轮仍没有被优化的情况. 对θ的初始化则包含两个选择，其一是对子句不满足系数进行初始化，其二是对文字的不满足度进行初始化，算法中随机选择其中一个进行初始化.

算法4.　*PB-ASIF*(φ, θ, *MAX_TIME*)

输入：MaxSAT算例φ，最大时间限制*MAX_TIME*，参数集θ；

输出：算例φ的可行解S.

```
1.  初始化 α(x), β(cls), var_cht, cls_cht;
2.  SAT_mark ← 0; feasible_mark ← 0;
3.  WHILE ! MAX_TIME DO
4.    初始化 S';
5.    IF condition0 THEN initialize θ;
                                   \*条件0满足*\
6     SAT_mark ← Cal_Hard_Clause(φ, S', var_cht,
      cls_cht);                    \*处理硬子句*\
7.    IF SAT_mark THEN
8.      feasible_mark ← Cal_Soft_Clause(φ, S',
      var_cht, cls_cht);           \*处理软子句*\
9.      IF feasible_mark THEN
10.       S ← S';
11.       WHILE condition1DO PB(φ, S, α, P)
                                   \*条件1满足进行PB搜索*\
12.   Renew(α(x), β(cls), var_cht, cls_cht);
13. RETURN S;
```

子句不满足系数的初始化是将所有子句的不满足系数值重新设置为1. 文字不满足度更新则是将原来ASIF中固定不变的文字不满足度进行更新，以初始解构造过程中的文字贡献度为基础，每个文字x的软子句不满足度$d_s(x)$初始化公式为式(12). 其中的$C_g(x)$即为文字的全局贡献度，文字的硬子句不满足度$d_h(x)$的计算与$d_s(x)$相同，只是全局贡献度使用针对硬子句的值.

$$d_s(x)=\frac{C_g(x)}{C_g(x)+C_g(x)} \quad (12)$$

子句不满足系数是子句不满足度的粗调，易陷入局部陷阱，因此对它进行初始化有益于跳出局部陷阱. 文字的不满足度初始以文字在算例集中出现频次为基础，随着求解过程中信息的积累，用文字的贡献度进行替换，这一过程相当于后验更新，更具有全局性.

5　实验设计与结果分析

本节通过设计实验分别展示ASIF与PB-ASIF的求解效果，主要通过与IPBMR进行工业算例求解的对比实验，展示新算法对求解效果的提升.

5.1　实验配置与算例集

本节实验配置和算例集与1.1中相同，即实验配置为：Linux系统下，处理器为Intel Xeon CPUs E5-2680@2.40G Hz，10 GB运行内存. 实验算例集

选取MSE2016的所有工业算例.

5.2 ASIF与IPBMR对比实验分析

本小节对ASIF的性能进行了测试,以展示其发掘初始解的效果.主要从两个方面进行实验统计,首先是它发掘可行解的速度,其次是其发掘的初始解的质量.这里通过与IPBMR的实验对比来显示ASIF的求解效果.对于初始解构造算法ASIF,在求解过程中按算法流程第一次构造出一个可行解即停止并输出,记录其解和求解时间,限制求解最大时间300 s.对于IPBMR则是以最大求解时间300 s为限,记录求解过程中的最优解以及首次发现最优解的时间.

表3展示了ASIF与IPBMR的实验对比结果.其中"#solv"列表示求得可行解的算例数,"#win"列表示输出的可行解更优的算例数,即输出的可行解满足的软子句权重之和更大的算例数,"平均时间"列表示输出可行解的算例的平均求解时间,"ASIF/IPBMR"列表示这两个算法求解的相应数据的比值,"Total"行给出了在所有算例上的综合对比结果.实验结果表明,ASIF在求解可行解数量和质量更优解数量上都比IPBMR要多,特别是在PI和WPI类型的算例上,这是因为ASIF算法中具有对硬软子句分别处理的组件与机制,在含有硬子句的MaxSAT算例上有更明显的效果.且这个统计数据对比的是ASIF的初始可行解和IPBMR的最优解,这说明ASIF找到的大多数初始可行解的质量和数量都比IPBMR的最优解还要好.通过"ASIF/IPBMR"列的数据比值更能清晰地展示这一点.此外,通过求解时间的结果可以看到,ASIF发掘初始可行解的平均时间是很短的,对比IPBMR的求解时间差距很大.此处平均值不能反映整体的求解时间对比,实际上大部分算例的求解时间差距更大.

表3 ASIF与IPBMR实验对比结果

算例类型	算例总数	ASIF			IPBMR			ASIF/IPBMR		
		#solv	#win	平均时间/s	#solv	#win	平均时间/s	#solv	#win	平均时间/s
UI	55	55	31	3.53	54	24	297.75	1.02	1.29	0.01
PI	601	375	323	9.16	102	63	106.74	3.68	5.13	0.09
WPI	630	312	240	27.21	167	164	145.54	1.87	1.46	0.19
Total	1286	742	594	16.33	323	251	158.73	2.30	2.37	0.10

图3展示了ASIF与IPBMR求解三类算例集的所有算例求解时间的详细对比.从左到右分别是UI、PI、WPI三个算例集的对比图,其中每个点代表一个算例.点的横坐标表示IPBMR求解该算例时的计算时间,纵坐标表示ASIF求解该算例时的计算时间.有些算例仅有一个求解器求解出,或两个求解器均未求解出.为了将所有算例的求解情况展示在图中,将每个未解出的算例的求解时间记为310 s,对应图3横纵坐标中的310 s刻度.横轴310 s刻度处纵坐标小于300 s的点表示IPBMR未求出可行解而ASIF求出可行解的算例,这些点的纵坐标值是ASIF的求解时间;纵轴310 s刻度处横坐标小于300 s的点表示ASIF未求出可行解而IPBMR求出可行解的算例,这些点的横坐标值是IPBMR的求解时间.从图3中可以看到,三个子图中的点大多集中在横轴右端附近,这说明在单个算例的求解时间对比上,ASIF求解时间明显小很多.并且可以看出有很多算例IPBMR在300秒内无法求出可行解,而ASIF能够用较短的

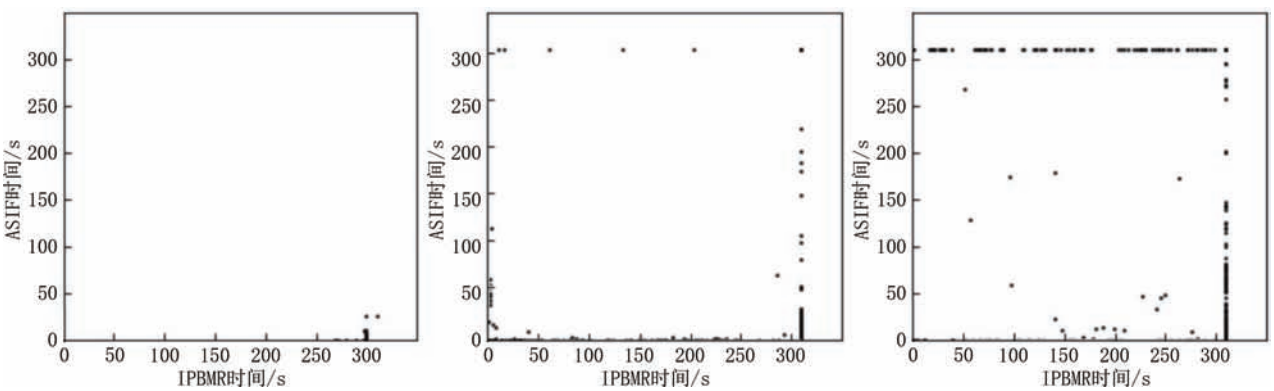

图3 ASIF与IPBMR求解时间对比图

时间得到可行解. 结合图3中ASIF求解更优解的数量和求解初始可行解的时间对比上可以看出，ASIF构造优质初始可行解具有很大的优势.

5.3　PB-ASIF与IPBMR对比实验分析

本小节通过设计实验来对PB-ASIF的求解工业算例的能力进行检测，同时将PB-ASIF与IPBMR的结果进行对比. 实验算例集和图示标注含义与4.2小节相同，求解时间均限定为300 s.

表4给出了PB-ASIF与IPBMR的实验结果，可以看到，PB-ASIF求解的算例数在三类工业算例集中都超过IPBMR，特别是PI和WPI类算例集的求解数量有很大提升，前者求解数量分别是后者的3.85倍和1.93倍. 而在解的质量对比上区别更加明显，PI算例集的求解更优解的数量PB-ASIF是IPBMR的7.25倍，在UI和WPI算例集上分别是1.29和1.81倍.

表4　PB-ASIF与IPBMR实验对比结果

算例类型	算例总数	PB-ASIF			IPBMR			PB-ASIF/IPBMR		
		#solv	#win	平均时间/s	#solv	#win	平均时间/s	#solv	#win	平均时间/s
UI	55	55	31	34.15	54	24	297.75	1.02	1.29	0.11
PI	601	393	348	76.89	102	48	106.74	3.85	7.25	0.72
WPI	630	322	284	89.39	167	157	145.54	1.93	1.81	0.61
Total	1286	770	663	79.06	323	229	158.73	2.38	2.90	0.50

求解时间的对比也显示了两个求解器的区别. 三类算例集上PB-ASIF的平均求解时间都小于IPBMR，而且在UI算例集上，PB-ASIF的平均求解时间只是IPBMR的十分之一左右，另两个算例集上的平均求解时间也几乎减半. 为了清晰显示每个算例求解时间的差异，本次实验仍然将求解时间分算例集作了对比图，如图4所示，横纵坐标分别是PB-ASIF与IPBMR的求解时间，为了将所有算例求解情况展示出来，将每个未求解出的算例的求解时间记为310 s. 从图4可以看出，PB-ASIF的求解时间比IPBMR还是有很大优势的，但不像ASIF那样明显，这也说明PB-ASIF算法中的每个策略都在发挥作用，即在ASIF给出初始解后，PB搜索也进行了优化，并且占用了一定的求解时间，而且通过表4的数据可以看出，在同样的求解时间内，PB-ASIF的求解效率比IPBMR要高很多.

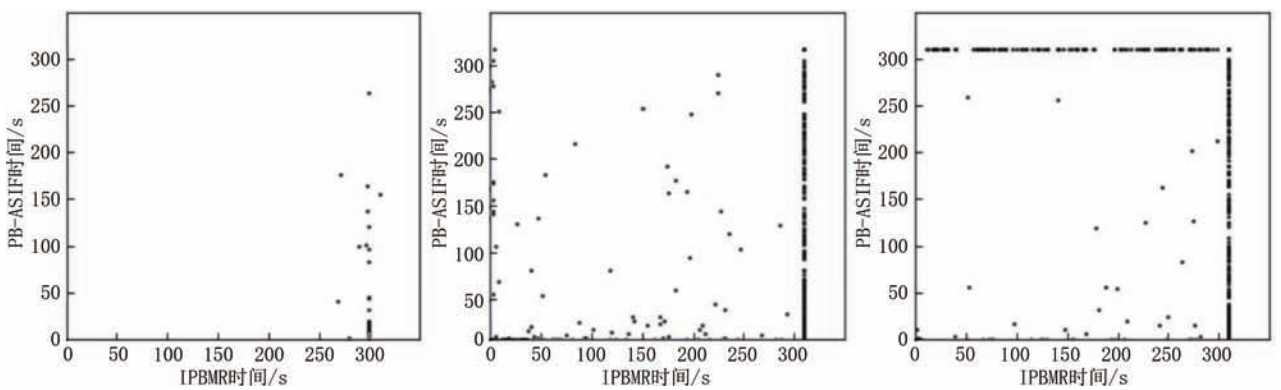

图4　PB－ASIF与IPBMR求解时间对比图

而且通过表3和表4的实验结果对比也可以看出，在300 s的求解时间限制内，PB-ASIF算法的求解算例数量相对ASIF求解初始可行解的数量上也有提升，这说明结合PB和ASIF能发挥更好的求解效果. 表4说明PB-ASIF明显地提升了基于PB策略求解器IPBMR的求解效果.

进一步地，为了展示PB策略在算法PB-ASIF中的效果，这里将PB-ASIF中的PB策略去除，再进行300 s求解测试，记该版本为noPB-ASIF. 与ASIF不同的是，noPB-ASIF中有子句不满足系数更新和文字不满足度更新. 表5给出了PB-ASIF与noPB-ASIF的实验对比结果. 从结果中可以看出PB-ASIF有一定的优势，在求解数量和解的优度方面基本都超过了noPB-ASIF，除了WPI中更优解数量上稍微少一些. 然而，在平均求解时间上PB-ASIF并不占优，这说明加入PB策略确实提升了算法的求解能力，但也增加了求解时间，这也符合普遍的时间和效果的关系.

表5 PB-ASIF与noPB-ASIF实验对比结果

算例类型	算例总数	PB-ASIF			noPB-ASIF			PB-ASIF/ noPB-ASIF		
		#solv	#win	平均时间/s	#solv	#win	平均时间/s	#solv	#win	平均时间/s
UI	55	55	29	34.15	55	26	22.55	1	1.12	1.51
PI	601	393	271	76.89	385	207	46.03	1.02	1.31	1.67
WPI	630	322	199	89.39	317	213	64.91	1.01	0.93	1.38
Total	1286	770	499	79.06	757	446	52.23	1.02	1.12	1.51

此外，从图3和图4的结果中还可以看出，对于UI、PI类型的算例，IPBMR能够求出可行解，而ASIF或PB-ASIF不能求出可行解的算例非常少；对于WPI类型的算例，IPBMR能够求出可行解，而ASIF或PB-ASIF不能求出可行解的算例数量则有许多，这说明加权的WPI算例相比于不加权的UI、PI算例的结构类型更加多样和复杂，某种算法、策略和技术在WPI类型的算例上难以全方面的超越另一种方法.

5.4 与其它求解器联合求解实验分析

本小节将PB-ASIF与两个代表性的针对工业算例的求解器进行实验对比分析，展示PB-ASIF对于其它求解器的互补性. 这里选取的求解器为MSE2016中的CCEHC和MSE2019[①]中的SATLike3.0. 这两个求解器都是针对工业算例求解进行设计的非完备求解器，分别在当时的竞赛中取得了很好的成绩.

CCEHC求解器是Chuan Luo、Shaowei Cai等人在MSE2016竞赛中提交的，相关算法在2017年发表的论文[27]中有详细介绍. 这个算法是在CCLS的基础上针对工业算例的求解进行改进而来，它是应用格局检测策略的局部搜索算法. 在CCLS的格局检测基础上，CCEHC针对硬子句设置了格局策略，并添加了硬子句权重设置与更新策略，有效改进了针对工业算例的求解效果.

SATLike3.0[30]是SATLike系列求解器的3.0版本，SATLike是Zhendong Lei、Shaowei Cai等人[29]提出的针对加权偏MaxSAT求解的局部搜索算法. 它定义了硬子句和软子句权重设置和更新策略，使得求解过程中能综合考虑软硬子句的状态. 该权重策略将两类子句放在一起而不是分开处理，同时根据搜索中的子句状态调整子句权重，从而调整子句在搜索中的重要性. SATLike中还设置了BMS的翻转变元选取策略，利用统计学原理使得选取候选变元的计算量减少而又尽量保证优度. SATLike3.0在原始版本上对初始解构造和软子句初始权重进行了细节优化，也有一定的改进效果.

本小节针对PB-ASIF的特征，设置了一定的结合策略，将PB-ASIF与CCEHC和SATLike3.0两个求解器进行简单的联合. 通过与原始求解器的对比，展示PB-ASIF与它们的求解互补性.

PB-ASIF在处理完所有硬子句后，对软子句的处理实际上设置了一个不满足软子句数的上界，当达到上界时会截断赋值过程，然后通过不满足子句的状态来更新参数，从而优化赋值过程. 利用这个特点，这里使用其它求解器进行一定时间的求解，若能输出不满足软子句数，则将其作为PB-ASIF处理软子句的初始上界. 这样既能利用其它求解器的求解特性，又能体现PB-ASIF的求解特性，从而展示PB-ASIF与其他求解器的互补性.

将300 s的求解时间分为两段，第一段为其它求解器的求解时间，第二段是PB-ASIF的求解时间. 为了更全面的实验，第一段其它求解器的求解时间设置了三个值，分别为10 s、60 s和150 s. 较小的求解时间可能使得其它求解器无法有效输出上界，因此三个时间段的设置能保证更全面地展示联合求解的效果.

由于CCEHC是2016年的竞赛求解器，这里算例集选取的是MSE2016的工业算例集. 这里将CCEHC求解器单独求解300 s的实验结果与联合求解器求解300 s的实验结果进行对比. 联合求解过程中，CCEHC求解一段时间后，其中一部分算例未输出可行解，另一部分算例输出可行解，这两种情况都有可能被PB-ASIF继续求解而改进. CCEHC未输出可行解的算例经过PB-ASIF求解，有一部分算例可以输出可行解. CCEHC输出可行解的算例经过PB-ASIF进行求解，有一部分能找到更好的解.

表6给出了求解器CCEHC与联合求解器的对比结果. 表6中“#win”表示联合求解器求解出比CCEHC更优解的数量，“#win0”和“#win1”列则是将这些更优解的数量分类统计的结果. “#win0”表

① MSE2019, https://maxsat-evaluations.github.io/2019, 2019

表6　PB-ASIF与CCEHC联合求解实验对比结果

算例类型	算例总数	10 s			60 s			150 s		
		#win	#win0	#win1	#win	#win0	#win1	#win	#win0	#win1
UI	55	39	0	39	39	0	39	44	0	44
PI	601	60	26	34	79	25	54	107	25	82
WPI	630	76	13	63	119	11	108	189	7	182
Total	1286	175	39	136	237	36	201	340	32	308

示联合求解器的更优解中，前半段时间内CCEHC未输出可行解，由PB-ASIF继续求解后输出可行解的算例数量."#win1"表示前半段时间内CCEHC输出可行解的算例中，由PB-ASIF继续求解后输出更优可行解的算例数量. 从表6中"#win"列的结果可以看出，随着分配给CCEHC求解时间的不断增加，联合求解的更优解数量在逐渐增大，而且更优解的数量相对于总算例数比较可观. 再通过"#win0"和"#win1"列的结果可以看出，这些更优解大部分是在CCEHC输出可行解后经过PB-ASIF继续求解从而得到更优的解，还有一小部分是CCEHC未输出可行解，经过PB-ASIF求解后得到可行解. 表6的实验结果说明，PB-ASIF能对CCEHC求解工业算例时输出的可行解进行有效的优化，还能求解出一部分CCEHC无法求出可行解的算例.

SATLike3.0是MSE2019的求解器，PB-ASIF与SATLike3.0联合求解实验选取MSE2019的算例集. MSE2019算例集分为不加权(简记UW)和加权(简记W)两类，而且算例基本是工业算例或复杂算例. 将SATLike3.0单独求解300 s的实验结果与联合求解300 s的实验结果进行对比. 联合求解部分同样将300 s的时间分为两段，首先用SATLike3.0求解一段时间，再用PB-ASIF求解剩下一段时间.

表7给出了实验对比结果. 同表6的标记类似，"#win"表示联合求解器求解出比SATLike3.0更优解的数量，"#win0"和"#win1"列则是将这些更优解的数量分类统计的结果. 可以看出，对于不加权类算例集，联合求解器的优化效果一般，对于加权类算例集，联合求解器表现出了明显的优化效果. 这再次说明了加权类算例的结构多样性，使得同种技术在不同加权类算例上的性能差异较大.

表7　PB-ASIF与SATLike3.0联合求解实验对比结果

算例类型	算例总数	10 s			60 s			150 s		
		#win	#win0	#win1	#win	#win0	#win1	#win	#win0	#win1
UW	299	0	0	0	1	0	1	0	0	0
W	297	30	1	29	56	1	55	81	1	80
Total	596	30	1	29	57	1	56	81	1	80

联合求解器对于未加权算例的优化并不明显. 通过具体实验数据分析可知，由于PB-ASIF对于软子句集的处理是基于权重以及变元贡献度的信息的反馈更新，而实验选取的不加权算例在这两个量以及相关的量上的反应并不明显. 因此需要针对这些不加权算例的特点改进当前的信息反馈策略，使得信息反馈能更具效果.

联合求解器对于加权类算例集求解更优解的数量随着SATLike3.0求解时间的增加而明显增加. 通过详细实验数据分析可知，随着SATLike3.0求解时间的增加，输出可行解的数量也不断增加，

这使得后半段PB-ASIF求解能对更多可行解进行优化. 实验对比是与SATLike3.0单独求解300秒的结果进行对比，这充分表明PB-ASIF与SATLike3.0求解加权类算例有很强的互补性，PB-ASIF能有效优化SATLike3.0对于加权类算例的求解.

由于PB-ASIF是以构造型赋值策略为主体的算法，与CCEHC或SATLike3.0等局部搜索类算法的框架不同，因此在方法上具有互补性. 而且通过4.2与4.3节的实验分析可以看到，ASIF和PB-ASIF的求解时间相对较短，这使得它们具有高效的优化求解潜力. 值得进一步研究的是，将PB-ASIF或ASIF直接与高效的局部搜索类求解算法进行结合，利用其中的全局信息反馈策略对局部搜索类算法的求解信息进行统计，然后对局部搜索类算法进行优化，将有效改进局部搜索类算法的求解效率.

5.5 小结与讨论

MaxSAT的局部搜索算法经过数十年的发展，出现了许多高效的策略，例如子句加权技术、配置检测技术、BMS采样技术等等，这些策略被广泛应用于近年来提出的优秀局部搜索算法中，例如Dist、CCEHC、SATLike以及SATlike3.0. MaxSAT问题不同类型算例的结构区别很大，这一特点导致一项技术难以在所有类型的算例上都有良好的性能. 本文针对MaxSAT问题的工业算例，同时也是局部搜索算法最不擅长求解的一类算例，创新地提出了一种搜索信息反馈策略，该策略以及文中设计的算法在近年的公认算例集上均有良好表现，特别是与其它先进的求解器具有良好的互补性. 搜索信息反馈策略的提出拓展了MaxSAT求解的研究思路，即在MaxSAT问题求解中进行系统地搜索知识积累与指导求解.

6 结 论

本文针对MaxSAT工业算例的求解，提出了新的初始解构造算法，即基于搜索信息反馈的赋值算法ASIF，然后结合路径截断策略提出了基于搜索信息反馈策略的路径截断算法PB-ASIF. 实验结果表明，PB-ASIF在工业算例的求解上显著超过未使用ASIF的对比算法IPBMR，充分显示了ASIF对于初始解构造的强大能力. 同时，与其它求解器的联合求解结果表明，PB-ASIF与其它求解器有较强的互补性，有提升其它求解器求解效果的能力.

未来的工作将从以下三个方面展开. 其一，将PB-ASIF中的全局求解信息反馈策略进一步优化，使其能更准确有效地实时反馈并指导求解；其二，设计能接入该信息反馈策略的局部搜索策略，使得局部搜索的搜索信息也能进行积累并反馈，从而形成更加高效的求解MaxSAT问题的非完备算法；其三，将ASIF或PB-ASIF与其它局部搜索算法直接结合形成新的求解器，更有效利用ASIF或PB-ASIF与其它求解器的互补性，进一步提升MaxSAT工业算例的求解质量.

参 考 文 献

[1] Cook S A. The complexity of theorem-proving procedures// Proceedings of the Third Annual ACM Symposium on Theory of Computing. Shaker Heights, USA, 1971: 151-158

[2] Zuckerman D. Linear degree extractors and the inapproximability of max clique and chromatic number//Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing. Seattle, USA, 2006: 681-690

[3] Bansal N, Blum A, Chawla S. Correlation clustering. Machine Learning, 2004, 56(1-3): 89-113

[4] Dinur I, Safra S. On the hardness of approximating minimum vertex cover. Annals of Mathematics, 2005, 162(1): 439-485

[5] Li C, Quan Z. An efficient branch-and-bound algorithm based on maxsat for the maximum clique problem//Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence. Atlanta, USA, 2010: 128-133

[6] Fang Z, Li C, Xu K. An exact algorithm based on maxsat reasoning for the maximum weight clique problem. Journal of Artificial Intelligence Research, 2016, 55: 799-833

[7] Berg J, Järvisalo M. Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. Artificial Intelligence, 2017, 244 : 110-142

[8] Chen Y, Safarpour S, Veneris A G, et al. Spatial and temporal design debug using partial maxsat//Proceedings of the Nineteenth ACM Great Lakes Symposium on VLSI. Boston Area, USA, 2009: 345-350

[9] Chen Y, Safarpour S, Marques-Silva J, et al. Automated design debugging with maximum satisfiability. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2010, 29(11): 1804-1817

[10] Lin P C K, Khatri S P. Application of Max-SAT-based ATPG to optimal cancer therapy design. BMC genomics, 2012, 13(6): 1-10

[11] Morgado A, Liffiton M H, Marques-Silva J. MaxSAT-based MCS enumeration//Proceedings of the Eighth International Haifa Verification Conference. Haifa, Israel, 2012: 86-101

[12] Bofill M, Garcia M, Suy J, et al. MaxSAT-based scheduling of B2B meetings//Proceedings of the International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Barcelona, Spain, 2015: 65-73

[13] Bofill M, Giráldez-Cru J, Suy J, et al. A study on implied constraints in a MaxSAT approach to B2B problems// Proceedings of the Twenty-Second International Conference of the Catalan Association for Artificial Intelligence. Mallorca, Spain, 2019: 183-192

[14] Demirovic E, Musliu N. MaxSAT-based large neighborhood search for high school timetabling. Computers & Operations Research, 2017, 78: 172-180

[15] Ciampiconi L, Ghosh B, Scarlett J, et al. A MaxSAT-based framework for group testing//Proceedings of the AAAI Conference on Artificial Intelligence. New York, USA, 2020, 34(06): 10144-10152

[16] Saikko P, Malone B M, Järvisalo M. MaxSAT-based cutting planes for learning graphical models//Proceedings of the International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Barcelona, Spain, 2015: 347-356

[17] Gaglione J R, Neider D, Roy R, et al. Learning linear temporal

properties from noisy data: a MaxSAT-based approach//Proceedings of the International Symposium on Automated Technology for Verification and Analysis. Gold Coast, Australia, 2021: 74-90

[18] Liu Y, Li C, He K. Improving lower bounds in MAXSAT complete algorithm based optimizing inconsistent set. Chinese Journal of Computers, 2013, 36(10): 2087-2095.
(刘燕丽,李初民,何琨.基于优化冲突集提高下界的MAXSAT完备算法.计算机学报,2013,36(10):2087-2095)

[19] Zheng Y, Xue J, Ling H. Combinatorial optimization problem reduction and algorithm derivation. Journal of Software, 2011, 22(9): 1985-1993
(郑宇军,薛锦云,凌海风.组合优化问题简约与算法推演.软件学报,2011,22(9):1985-1993)

[20] Zhang X, Li Z. Research on feature selection algorithms based on natural evolution strategy. Journal of Software, 2020, 31(12): 3733-3752
(张鑫,李占山.自然进化策略的特征选择算法研究.软件学报,2020,31(12):3733-3752)

[21] Yin M, Zhou J, Sun J, et al. Heuristic survey propagation algorithm for solving QBF problem. Journal of Software, 2011, 22(7): 1538-155
(殷明浩,周俊萍,孙吉贵等.求解QBF问题的启发式调查传播算法.软件学报,2011,22(7):1538-155)

[22] He K, Zheng J. Survey on algorithms for the maximum satisfiability problem. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2022, 50(2): 82-95
(何琨,郑迥之.最大可满足性问题的算法研究综述.华中科技大学学报(自然科学版),2022,50(2):82-95)

[23] Festa P, Pardalos P M, Pitsoulis L S, et al. GRASP with path relinking for the weighted MAXSAT problem. ACM Journal of Experimental Algorithmics, 2007, 11: 2-4

[24] Luo C, Cai S, Wu W, et al. CCLS: an efficient local search algorithm for weighted maximum satisfiability. IEEE Transactions on Computers, 2014, 64(7): 1830-1843

[25] Xu Z, He K, Li C. An iterative Path-Breaking approach with mutation and restart strategies for the MAX-SAT problem. Computers & Operations Research, 2019, 104: 49-58

[26] Cai S, Luo C, Thornton J, et al. Tailoring local search for partial MaxSAT//Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. Quebec City, Canada, 2014: 2623-2629

[27] Luo C, Cai S, Su K, et al. CCEHC: an efficient local search algorithm for weighted partial maximum satisfiability. Artificial Intelligence, 2017, 243: 26-44

[28] Cai S, Su K. Configuration checking with aspiration in local search for SAT//Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence and the Twenty-Fourth Innovative Applications of Artificial Intelligence Conference. Toronto, Canada, 2012: 434-440

[29] Lei Z, Cai S. Solving (weighted) partial MaxSAT by dynamic local search for SAT//Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. Stockholm, Sweden, 2018: 1346-1352

[30] Lei Z. Novel algorithms of maximum satisfiability and its extensions. Beijing: University of Chinese Academy of Sciences, 2021
(雷震东.最大可满足性及其拓展问题的求解.北京:中国科学院大学,2021)

XU Zhen-Xing, Ph. D. His research interests include NP-hard problem, combinatorial optimization.

HE Kun, Ph. D., professor. Her research interests include machine learning, adversarial learning, combinatorial optimization, graph data mining.

LI Chu-Min, Ph. D., professor. His research interests include NP-hard problem, combinatorial optimization.

LIU Yan-Li, Ph. D., associate professor. Her research interests include NP-hard problem, combinatorial optimization.

ZHENG Jiong-Zhi, Ph. D. candidate. His research interests include NP-hard problem, combinatorial optimization.

Background

Maximum Satisfiability problem (MaxSAT) is an important generalization of a classic NP-complete problem, Satisfiability problem (SAT), in theoretical computer science. MaxSAT aims to find a complete assignment of truth values to maximize the total weight of satisfied soft clauses, with the constraint that all the hard clauses (if any) must be satisfied. MaxSAT problem can describe many practical problems, such as circuit design, maximum clique, timetabling, planning, etc.

Solvers for MaxSAT can be divided into complete and incomplete. The performance of incomplete MaxSAT solvers in large-scale industrial instances is usually worse than that of

complete solvers for a long time. Industrial instances are not only large in scale but also more complex in structure. The differences between different types of instances are particularly large, and it is difficult for incomplete solvers to take into account the characteristics of all instances when designing. Therefore, incomplete solvers should be technically improved based on the characteristics of large-scale instances on the basis of optimizing small and medium-sized instances.

This paper analyzes the weakness of an incomplete MaxSAT solver, IPBMR, in solving industrial instances, and proposes a new initial solution construction algorithm called Assignment approach by Search Information Feedback (ASIF). Then, ASIF is used to combine with the path-breaking strategy to form a new algorithm called Path-Breaking approach with ASIF strategies (PB-ASIF). The experimental results show that PB-ASIF can effectively improve the performance of IPBMR in solving industrial instances and has a good complementarity with other effective incomplete solvers.