第 21 卷 增刊
1998 年 8 月

计 算 机 学 报
CHINESE J. COMPUTERS

Vol. 21 Suppl.
Aug. 1998

用吴方法求解可满足性问题(Ⅱ)——实验研究

贺思敏　张　钹

(清华大学计算机科学与技术系　北京 100084)

(清华大学智能技术与系统国家重点实验室　北京 100084)

摘　要　本文使用随机 3-SAT 实例模型,对算法变换思想指导下设计的吴方法求解可满足性问题的算法进行了实验,并与语义归结、支持集归结和 DP 算法进行了对比.

关键词　算法设计,可满足性问题,吴方法,算法变换,归结法.

分类号: TP18

SOLVING SATISFIABILITY PROBLEM BY WU'S METHOD(Ⅱ) ——EXPERIMENTAL EVALUATION

HE Si-Min　ZHANG Bo

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

(*National Key Laboratory for Intelligent Technology and Systems, Tsinghua University, Beijing* 100084)

Abstract　Using random 3-SAT instance model, an experimental evaluation of Wu's method for solving SAT, and detailed comparisons with semantic resolution, set-of-support resolution and DP algorithm are presented in this paper, further elaborating the idea of algorithm transform.

Keywords　Algorithm design, satisfiability problem, Wu's method, algorithm transform, resolution method.

1 引　言

在"用吴方法求解可满足性问题(Ⅰ)——算法变换"一文(可以参见文献[1])中,我们在算法变换的思想指导下,研究了用吴方法求解可满足性问题的特点.通过建立吴方法的基本操作与子句间有限制的归结操作的对应,证明了吴方法求解可满足性问题基本上是一种以特征列计算为核心的有限制的子句归结过程,不仅使吴方法和归结法相互引入新的概念和认识,而且在算法实现时可以避免复杂的多项式计算,同时可以更好地利用问题特性和已有经验来计算吴方法的特征列,设计问题分解、变元定序的策略,以获得更高的效率.

本文介绍有关吴方法求解 SAT 问题的详细实验结果和分析.实验分三部分:第一部分是吴方法的实现,主要是通过实验决定采用何种变元序和分支策略最有效;第二部分是吴方法与归结法的比较,因为吴方法求解 SAT 问题与归结法有密切联系;第三部分是吴方法与 DP 算法的比较,因为 DP 算法是目前 SAT 问题完备算法中最有效的.

问题模型采用常用的均匀随机 3-SAT 实例模型,这是一种公认的易于生成而又非常难解的实例模型,常

本文 1996-05-07 收到,修改文 1998-05-04 收到.本课题得到国家自然科学基金、国家攀登计划和国家 863 高科技基金资助.**贺思敏**,男,1968 年生,获博士学位,主要研究领域为组合优化问题、局部搜索算法的实验设计与分析.**张　钹**,男,1935 年生,中国科学院院士,教授,博士生导师,主要研究领域为问题求解、人工神经网络等人工智能基础理论.

用于测试各种 SAT 问题算法[2]. 设要求生成一个 n 个变元 m 个子句的随机实例,生成方法是:从 n 个变元中随机选 3 个不同的变元,每个变元以 0.5 的概率变反后组成一个 3 个文字的子句,独立重复这一过程 m 次即得. 已有实验表明,$m/n \approx 4.3$ 时,生成的实例有 50% 可满足,这时实例最难[2].

实验用的计算机是 SGI-Elan4000 工作站. 所有程序均用 C 语言编写,并经最高级别的优化编译.

2　吴方法的实现

在我们定义的算法变换下[1],吴方法的实现是比较简单的. 子句多项式均按子句形式存放,子句用长度等于变元数、元素为 0,1,－1(分别表示相应变元在子句中不出现,以正文字出现,以负文字出现)的一维数组统一表示;特征列计算采用深度优先策略,仅处理子句多项式,基本运算按定理 2[1] 进行. 只有两个环节需要进一步通过实验确定:变元序和分支策略. 我们实验了 3 种变元序:强约束变元高序,强约束变元低序和随机序(参见文献[1]),2 种分支策略:高冲突变元优先分支和低冲突变元优先分支(参见文献[1]),共 6 种组合. 表 1 给出了这 6 种组合对变元数为 50,子句数为 215 的 50 个随机实例(其中 27 个可满足)的实验数据.

表 1　吴方法的各种实现(变元数=50,子句数=215,实例数=50(其中可满足实例数=27))

		强约束变元低序			随机序			强约束变元高序		
		SAT	UNSAT	TOTAL	SAT	UNSAT	TOTAL	SAT	UNSAT	TOTAL
低冲突变元优先分支	时间	0	1	0	1	6	4	29	109	66
	节点数	17	28	23	36	113	72	328	999	637
	基本运算	35778	70909	51938	121738	421027	259411	1811622	6617467	4022311
	A	0	0	0	0	0	0	0	0	0
	B	0	0	0	0	0	0	0	0	0
	C	0	0	0	0	0	0	0	0	0
高冲突变元优先分支	时间	0	2	1	3	10	6	22	61	40
	节点数	65	141	100	144	433	277	364	904	612
	基本运算	67684	156358	108474	241661	716021	459867	1413003	3813508	2517235
	A	0	0	0	0	0	0	0	0	0
	B	0	0	0	0	0	0	0	0	0
	C	0	0	0	0	0	0	0	0	0

注 1:SAT,UNSAT,TOTAL 分别表示可满足实例,不可满足实例,全体实例的平均数据;

注 2:时间单位为秒,0 表示小于 0.5 秒;

注 3:节点数即计算特征列的次数,参见文献[1];

注 4:基本运算是指定理 2[1] 中情况 2,3,4 的总计数;

注 5:A,B,C 为第一个特征列的信息,其中 A 表示第一个特征列有使其初式不为 0 的零点的实例个数,B 表示第一个特征列矛盾的实例个数,C 表示第一个特征列既不矛盾又没有使其初式不为 0 的零点时,特征列中对应单元子句的子句多项式的个数.

从表 1 的各项数据可以看出:

(1) 变元序对算法效率影响非常大. 其中,“强约束变元低序”各项指标一致优于“随机序”,“随机序”各项指标一致优于“强约束变元高序”,而且差异是很大的. 这对吴方法求解一般问题时的变元定序也有启发,即在定义某种约束后,将约束较强的变元序定得低些可能效果好一些.

(2) 分支策略对求解效率影响不如变元序大,相对优劣与变元序有关. 分支策略必须配合变元序才能取得较好的效果.

(3) 第一个特征列基本上没有可以马上利用的信息,进一步进行问题分解是必要的.

因此,本文的吴方法实现采用“强约束变元低序”的变元序和“低冲突变元优先分支”的分支策略. 我们进一步测试了吴方法在更大规模问题上的性能,发现也是在 $m/n \approx 4.3$ 附近比较难,其余较易. 表 2 给出了变元数为 100,子句数分别为 400、430、500,各 100 个实例的包括有关指标平均、最小、最大值的实验数据.

表 2 吴方法的性能(变元数=100,子句数= 400,430,500,实例数各 100)

		实例数	时 间			节点数			基本运算		
			最小	平均	最大	最小	平均	最大	最小	平均	最大
子句数=400	SAT	99	0	25	171	8	144	857	34546	952470	6335061
	UNSAT	1	125	125	125	591	591	591	4703445	4703445	4703445
	TOTAL	100	0	26	171	8	149	857	34546	989980	6335061
子句数=430	SAT	54	1	38	355	11	165	1133	58457	1394844	12629841
	UNSAT	46	21	83	237	87	313	841	807036	2997091	8661354
	TOTAL	100	1	59	355	11	233	1133	58457	2131877	12629841
子句数=500	SAT	1	4	4	4	15	15	15	150987	150987	150987
	UNSAT	99	8	40	138	21	84	403	305857	1430169	4971739
	TOTAL	100	4	40	138	15	83	403	150987	1417377	4971739

注:数据说明见表 1 注.

3 吴方法与归结法的比较

文献[1]分析表明吴方法求解 SAT 问题与归结法有密切联系,因此我们对吴方法和归结法进行了比较.我们选择了支持集归结和语义归结作为比较对象.

支持集归结是 1965 年 Wos 等提出的[3],其思想是:用归结法证明定理 $A_1 \wedge A_2 \wedge \cdots \wedge A_m \rightarrow B$ 实际上是证明 $A_1 \wedge A_2 \wedge \cdots \wedge A_m \wedge \overline{B}$ 不可满足,而矛盾不大可能存在于前提中,因为前提通常是可满足的,因此应尽量避免在$\{A_1, A_2, \cdots, A_m\}$的子句中归结.一般地,子句集 S 的一个子集 T 称为 S 的支持集,如果 $S-T$ 是可满足的.支持集归结就是归结过程中只选取不同时属于 $S-T$ 的子句进行归结.支持集归结是完备的,而且是目前归结定理证明中效率最高的方法[4],这是我们选择它与吴方法进行比较的原因.

语义归结是 1967 年 Slagle 提出的[5],对子句间的归结做了两点限制,在命题逻辑中表述为:(1) 用子句集 S 的一个解释 I(即各命题变元的一组 0,1 赋值)将 S 分成两部分,S 中在 I 下为真的子句组成 S_1,S 中在 I 下为假的子句组成 S_2,规定归结只能在 S_1 和 S_2 的子句之间进行,S_1 内或 S_2 内的子句间不允许进行归结;(2) 规定命题变元的一个序,S_1 和 S_2 的子句进行归结时,归结变元必须是 S_2 的子句中序最高的变元.语义归结是完备的,支持集归结可以看成是语义归结的特例;此外,语义归结的第 2 个限制与吴方法极为相似,因此我们也把语义归结与吴方法进行了比较.

支持集归结的实现主要需解决支持集的确定问题.定理证明中支持集常由结论取反构成,而 SAT 问题是判定形式,没有结论部分可以利用.我们采用两种方法确定支持集:一种方法是随机生成一个解释,初始子句集中在此解释下为假的子句构成支持集,我们称之为"随机生成解释";另一种方法是用局部搜索方法寻找一个较好的解释,使初始子句集中在此解释下为假的子句构成的支持集尽可能小,我们称之为"局部搜索解释".

语义归结的实现需要解决两个问题:划分子句集的解释和变元序.划分子句集的解释的确定我们采用了和确定支持集相同的两种方法,即随机生成一个解释,或由局部搜索方法寻找一个使初始子句集中在此解释下为假的子句尽可能少的解释.变元序的确定我们采用了吴方法实验的 3 种变元序:强约束变元低序,随机序和强约束变元高序.

把局部搜索方法应用到归结法中是一个新的尝试.近几年来,应用局部搜索方法求解 SAT 问题取得了很大进展[2],其特点是:如果实例可满足,则局部搜索方法一般可以非常高效地找到一个解;如果实例不可满足,则局部搜索方法无法给出任何明确的结论,但是一般可以找到一个解释,使在此解释下为假的子句数达到极少.如何利用局部搜索给出的信息加速对不可满足实例的证明一直没有很好的结果.我们把局部搜索用到归结法求解 SAT 问题上,一个目的是在把局部搜索应用到一阶谓词逻辑归结定理证明之前,先在命题逻辑归结定理证明中实验其可行性;对本文来讲,更重要的目的是提高归结法的效率,尽可能用最好的归结法与吴方法进行比较.

我们实现了以上 6 种语义归结和 2 种支持集归结,以便选出最好的一种与吴方法进行比较.程序实现时,

对所有子句均建立索引表,使各变元通过索引表可以方便地访问变元以正文字或负文字出现的子句,从而使归结操作避免不必要的查找;采用单元子句优先归结的策略以提高归结效率;归结生成的子句进行包容检查之后再加入子句集,这一过程虽然最为费时,但如果不做包容检查,则归结法很快就会因生成大量无用子句而陷于瘫痪[4];子句表最大容量定为 10000 个子句,超过这个限度还没有结果即宣布该归结法失败. 表 3.1,表 3.2 给出了有关数据.

表 3.1　归结法实现(变元数=30,子句数=129,实例数=50(其中可满足实例数=25))

		语义归结						支持集归结	
		强约束变元低序		随机序		强约束变元高序			
		SAT	UNSAT	SAT	UNSAT	SAT	UNSAT	SAT	UNSAT
随机生成解释	解出的实例数	25	25	9	5	1	0	0	0
	时间	20	20	67	60	100			
	归结操作	23602	23708	63240	51730	101113			
	包容过程调用	20103	20249	56145	44295	87632			
	包容操作	5266310	4932749	17038533	14793143	30461750			
	子句表中子句数	3183	4281	7128	7906	8434			
	初始不满足子句数	17	15	15	13	10			
局部搜索解释	解出的实例数	25	25	25	9	25	0	25	0
	时间	0	8	0	32	0		0	
	归结操作	0	10804	0	27696	0		0	
	包容过程调用	0	9124	0	24139	0		0	
	包容操作	0	1798941	0	5613506	0		0	
	子句表中子句数	129	3104	129	6543	129		129	
	初始不满足子句数	0	1	0	1	0		0	

表 3.2　支持集归结(变元数=12,子句数=60,实例数=50(其中可满足实例数=24))

		解出的实例数	时间	归结操作	包容过程调用	包容操作	子句表中子句数	初始不满足子句数
随机生成解释	SAT	24	1	11227	8441	702798	1889	7
	UNSAT	26	2	12867	9659	945420	2202	7
局部搜索解释	SAT	24	0	0	0	0	60	0
	UNSAT	26	2	9845	7461	700371	1864	1

注 1:表中所给出的数据是归结法在 10000 个子句内得出解答的实例的平均数据;

注 2:归结操作计数中不包括单元归结操作;

注 3:包容过程调用不记单元归结中的包容过程调用;

注 4:包容操作是指包容过程中的基本操作两子句间的包容检查,也不记单元归结中的包容操作. 因为包容操作约占 85% 的运行时间,而我们不排除有更有效的包容操作实现方法,为慎重起见,我们同时给出"包容过程调用"和"包容操作"两个数据,相当于给出了包容基本操作的下界和上界;

注 5:子句表中子句数是指不被当时子句表中已有子句包容的归结式子句数和初始子句集中子句数之和;

注 6:由于支持集归结对变元数 30、子句数 129 的实例无有效结果,我们又对它测试了变元数 12、子句数 60 的实例;

注 7:对可满足实例,用局部搜索方法寻找划分子句集的解释(对语义归结)或尽可能小的支持集(对支持集归结)时已经给出实例的解,因而不必再进行归结过程. 但切记:这不是归结法的威力,而是局部搜索方法的威力. 在用吴方法之前用局部搜索方法做预处理会有同样结果. 因此,应注意的是局部搜索方法对归结法证明不可满足实例的作用.

从表 3.1,表 3.2 的各项数据可以看出:

(1) 语义归结要比支持集归结好很多. 这可能是命题逻辑特有的,也可能与我们的问题模型有关.

(2) 语义归结中,变元序的作用非常大,其中,"强约束变元低序"各项指标一致优于"随机序","随机序"各项指标一致优于"强约束变元高序",而且差异是很大的. 这与吴方法的实验结果相同.

(3) 局部搜索方法的引入对可满足实例的效果是可以预料到的,重要的是它对归结法证明不可满足实例

的作用非常显著,使语义归结各项指标改善了 100%,支持集归结各项指标改善了 20%.因此,把局部搜索方法引入一阶谓词逻辑归结定理证明,在子句集的 H 解释空间中进行局部搜索,可能对支持集归结中支持集的选择、语义归结中划分子句集的解释的选择、线性归结中顶子句的选择这些直接影响相应归结方法效率的关键环节有较大帮助,是一个值得研究的有希望的方向.

我们比较了吴方法与强约束变元低序下的语义归结.尽管归结法的两大基本操作归结和包容的计数中不包括单元归结中的有关操作,而吴方法的基本操作计数中包含了所有操作;此外,局部搜索方法的应用使语义归结的效率提高了一倍;但无论从基本操作计数还是从时间上看,吴方法的优势是显而易见的,而且优势很大.具体数据见表 4.

表 4 吴方法与语义归结的比较(变元数=30,子句数=129,实例数=50(其中可满足实例数=25))

	可满足实例		不可满足实例	
	时间	基本操作	时间	基本操作
语义归结(强约束变元低序,随机生成解释)	20	归结操作 = 23602 包容过程调用 = 20103 包容操作 = 5266310	20	归结操作 = 23708 包容过程调用 = 20249 包容操作 = 4932749
吴方法	0	6701	0	8980
语义归结(强约束变元低序,局部搜索解释)	0	0	8	归结操作 = 10804 包容过程调用 = 9124 包容操作 = 1798941

注 1:本实验与表 3 是同一组实例;

注 2:有关基本操作的说明参见表 1 和表 3 的注.

4 吴方法与 DP 算法的比较

DP 算法是目前求解 SAT 问题的完备算法中效率最高的一个,因此我们有必要把吴方法与 DP 算法进行比较,以全面评价吴方法的效率.

DP 算法经过几年来的集中研究,算法实现得比较好,其中分支变元的选取是关键.我们采用约束强的变元优先分支,变元约束强弱用变元以正文字出现的子句数和以负文字出现的子句数之乘积来估计,乘积越大,约束越强.吴方法的分支策略就是借鉴了 DP 算法的这一分支策略并利用特征列的信息而设计的.与吴方法有所不同的是,DP 算法在搜索树的各个节点上将动态调整变元序,而不是在搜索前一次定序后不再变化.表 5给出了 DP 算法和吴方法对变元数为 100、子句数为 430 的 100 个随机实例的实验数据.

表 5 吴方法与 DP 算法比较(变元数=100,子句数=430,实例数=100(其中可满足实例数=54))

	可满足实例		不可满足实例		全体实例	
	时间	节点数	时间	节点数	时间	节点数
DP 算法	0	155	0	471	0	301
吴方法	38	165	83	313	59	233

注:本组实例与表 2 中同一规模的那组实例是相同的.

从表 5 的数据看,吴方法的搜索树要比 DP 算法的搜索树小,但总的效率有很大差距.

我们进一步实验了把吴方法的特征列计算作为节点操作嵌入 DP 算法的想法,即在 DP 算法搜索树的每个节点上做完单元归结后,若既没发现矛盾也没找到解,则计算当前节点子问题(仍是子句集)的特征列,其中变元序在节点处按当前子问题强约束变元低序的策略动态确定.如果特征列计算发现矛盾或者找到解,则该节点计算可以比 DP 算法提前结束.如果特征列计算既没发现矛盾也没找到解,则继续按 DP 算法求解.表 6 对变元数为 100、子句数为 430 的 100 个随机实例实验了这个想法.

表 6　把吴方法的特征列计算嵌入 DP 算法,并与吴方法和 DP 算法比较

(变元数=100,子句数=430,实例数=100(其中可满足实例数=54)

	可满足实例		不可满足实例		全体实例	
	时间	节点数	时间	节点数	时间	节点数
DP 算法	0	155	0	471	0	301
吴方法	38	165	83	313	59	233
把吴方法的特征列计算嵌入 DP 算法	6	50	19	124	12	84

注:本实验与表 5 是同一组实例.

从表 6 的数据看,把吴方法的特征列计算嵌入 DP 算法后,节点数和时间均比吴方法要好;但与 DP 算法相比,尽管节点数下降很大,总的效率仍有差距.

提高吴方法的实现效率,关键在于提高特征列的计算效率. 目前看来特征列的计算还有可能进一步减少无用操作,主要是改变特征列计算时"牵一发而动全身"的特点,即在基列有所变化时,并不一定要重新计算所有多项式对此基列的余式,当然这需要进行必要的标记以及进一步的实验.

5　结　论

本文通过对吴方法求解 SAT 问题的实验研究,进一步示范了算法变换的应用及其价值. 吴方法实现中变元序的选择、分支策略、特征列计算嵌入 DP 算法的节点,以及语义归结和支持集归结中利用局部搜索选择解释等等,都表现出算法变换带给我们的启发和效率. 从实验结果看,对于求解 SAT 问题,吴方法比归结法的效率要高,但是比 DP 算法还有很大差距. 不过应当清楚,如果没有算法变换,吴方法的实现效率之低是不难想象的. 除了算法变换,文中提出的在自动推理中应用局部搜索的思想也是值得进一步探讨的.

参　考　文　献

1　贺思敏. 可满足性问题的算法设计与分析[博士学位论文]. 清华大学计算机科学与技术系, 北京, 1997

2　Selman B, Levesque H, Mitchell D. A new method for solving hard satisfiability problems. In: Proc AAAI-92, San Jose, California, 1992. 440－446

3　Wos L, Carson D, Robinson G. Efficiency and completeness of the set-of-support strategy in theorem proving. *J ACM*, 1965, 12:536－541

4　Wos L. Automated reasoning: 33 basic research problems. Prentice-Hall, 1988

5　Slagle J. Automatic theorem proving with renamable and semantic resolution. *J ACM*, 1967, 14:687－697