计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201000178

正则(3,4)-CNF公式的社区结构

何 彬 许道云
贵州大学计算机科学与技术学院 贵阳 550025
(2276197035@qq.com)

摘 要 通过构造适当的极小不可满足公式以实现在多项式时间内将3-CNF公式归约转换为一个正则(3,4)-CNF公式,转换后的公式与原公式具有相同的可满足性,同时公式的结构也发生相应的变化。图的社区结构反映了图的模块特性,文中将CNF公式转化为相应的图,研究公式图的模块特性与公式某些性质之间的关系。将归约前后的两类公式转换为相应的图并研究其模块特性,发现转换后得到的正则(3,4)-CNF公式具有较高的模块度。此外,在使用DPLL(Davis Putnam Logemann Loveland)算法求解CNF公式的过程中,发生冲突时利用冲突驱动子句学习策略,得到一个学习子句并将其添加到原公式中,使得原公式的模块度降低。研究发现:将DPLL算法与冲突驱动子句学习策略结合应用到正则(3,4)-CNF公式时,其学习子句所包含的绝大部分变元位于不同的社区中。

关键词:SAT问题;DPLL;正则(3,4)-CNF公式;社区结构;模块度
中图法分类号 TP301

Community Structure of Regular (3,4)-CNF Formula

HE Bin and XU Dao-yun
College of Computer Science and Technology,Guizhou University,Guiyang 550025,China

Abstract By constructing an appropriate minimum unsatisfiable formula,the 3-CNF formula can be reduced and converted into a regular (3,4)-CNF formula in polynomial time. The converted regular (3,4)-CNF formula has the same satisfiability as the original 3-CNF formula. The structure of the formula has also changed accordingly. The community structure of the graph reflects the modular characteristics of the graph. The CNF formula can be transformed into the corresponding graph to study the relationship between the modular characteristics of the formula graph and some properties of the formula. The two types of formulas before and after the reduction are converted into corresponding graphs,and the module characteristics are studied. It is found that the regular (3,4)-CNF formula obtained after conversion has a high modularity. In addition,in the process of using the DPLL (Davis Putnam Logemann Loveland) algorithm to solve the CNF formula,when a conflict is encountered,the conflict-driven clause learning strategy is used to obtain a learning clause and add it to the original formula,which reduces the modularity of the original formula. The research finds that when the DPLL algorithm combined with the conflict-driven clause learning strategy is applied to the regular (3,4)-CNF formula,most of the variables contained in the learning clause are located in different communities.

Keywords SAT problem,DPLL,Regular (3,4)-CNF formula,Community structure,Modularity

1 引言

可满足性问题(Satisfiability Problem,简称SAT问题)用于判定一个合取范式(Conjunction Normal Form,CNF)是否存在一组对变元的真值指派使得公式为真。SAT问题在数理逻辑和计算理论等领域有着重要的研究地位。在SAT问题的研究中,通常限制每个子句的长度为k,其判定问题对应称为k-SAT问题。2-SAT问题可在线性时间内求解[1],3-SAT问题是著名的NP完全问题[2]。正则(3,4)-CNF公式是一类特殊的3-CNF公式,其中每个子句的文字数恰好为3,每个变元的出现次数恰好为4。利用极小不可满足公式的临界特性,通过构造适当的极小不可满足公式,可以实现在多项式时间内将3-CNF公式归约转换为正则(3,4)-CNF公式,这说明(3,4)-SAT问题仍然保留NP完全性[3-4]。将一个随机3-CNF公式归约到正则(3,4)-SAT公式后,公式的结构会发生很大的改变。

很多复杂系统可以表示为网络,如文献引用网络、交通运输网络、电子邮件网络等。研究发现,某些网络中的结点具有社区结构(Community Structure),即网络中的结点可以划分为若干个子集,大部分边的两个结点都位于同一个子集中,少部分的边连接属于不同子集的两个结点。Girvan等提出了一种迭代算法来计算网络的社区结构[5-6],其最坏时间复杂度为

到稿日期:2020-10-29 返修日期:2021-02-03 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61762019)
This work was supported by the National Natural Science Foundation of China (61762019).
通信作者:许道云(dyxu@gzu.edu.cn)

$O(n^3)$。Newman 等提出了图模块度(Modularity)的定义，模块度为落在同一模块内的边的比例与对这些边进行随机分配所得到的概率期望的差值，记为 Q[6]，用来定量分析网络的社区结构。根据定义，模块度为一个介于 0～1 之间的数，模块度越大表明社区结构越明显。

将 n 个结点划分为若干非空子集，记所有可能划分数为 S_n，S_n 的大小随着 n 的增加而呈指数级增长，网络最优模块度的计算是 NP 难的。Newman 等提出了一种基于模块度优化的贪心策略来计算网络的社区结构[7]，其最坏情况下的时间复杂度为 $O(n^2)$。Raghavan 等提出了一种具有线性时间复杂度的算法来计算网络的社区结构[8]，称为标签传播算法(Label Propagation Algorithm，LPA)，但该算法得到的划分结果精度损失较大。Vincent 等提出的图折叠算法(Graph Folding Algorithm，GFA)[9]在两个方面对标签传播算法进行了改进，提高了划分结果的精度。

根据变元交互图模型(Variable Incidence Graph Model)，可将一个 CNF 公式转化为一个对应的图[10]。文献[11]的结果表明，许多 SAT 实例对应的变元交互图可以分解为互相连接的社区，并且可以利用 SAT 实例的社区结构提高 SAT 求解器的性能。Ansótegui 等[12]将 SAT 实例表示为变元交互图来研究 SAT 公式的社区结构，结果表明，工业类 SAT 实例具有较高的模块度，而随机 SAT 实例没有这种性质。他们提出了一种利用 SAT 实例的社区结构来改进 SAT 求解算法的思想。Ansótegui 等的研究还表明[13]，通过冲突驱动子句学习策略(Conflict Driven Clause Learning，CDCL)得到的学习子句与公式的社区结构有关，学习子句的加入使得公式的模块度降低。Ansótegui 等认为 DPLL 算法在搜索整个公式的解空间时以社区为单位，从一个社区搜索到另外一个社区。大部分的冲突发生在不同社区之间，这些冲突产生的学习子句所包含的变元位于不同的社区之间，因此学习子句的加入使得公式的模块度降低。

在将 3-CNF 公式归约转换为正则(3,4)-CNF 公式的过程中，根据 3-CNF 公式中每一个变元的正负出现次数需引入多个变元集，这些变元构成 MU(2)公式和 MU(4)公式，以此加长归约过程中产生的 2-子句为 3-子句以及将变元在公式中的出现次数变为 4。每一步引入的若干变元构成的 MU 公式构成了类似社区的结构，运用 DPLL 算法求解社区内部的 MU 公式时发生冲突的概率非常小，冲突发生在社区之间的概率较大。通过实验发现：归约转换后的正则(3,4)-CNF 公式具有明显的社区结构(极高的模块度)，学习子句包含的变元所在的社区数量基本与学习子句长度相等。

2　图的社区结构

一个无向连接网络可以表示为 $G(V,W)$。V 是顶点集；边对应的权值函数 $W:V\times V\to R^+\cup\{0\}$，表示任意两个图结点 $x,y\in V$ 对应一个权值 $W(x,y)\geqslant 0$，$W(x,y)>0$ 表示 x,y 有边相连，否则 x,y 无边相连，且 $W(x,y)=W(y,x)$。在非带权网络中，边的权值总是 1。图结点 x 的度表示为 $\deg(x)=\sum_{y\in V}W(x,y)$。在具有明显社区结构的网络中，网络中的结点可以划分为若干社区，社区内部联系较为紧密，社区间通过很少的边相连。一个具有明显社区结构的网络如图 1 所示。

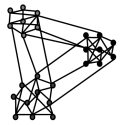

图 1　具有明显社区结构的网络实例

Fig. 1　Network instance with obvious community structure

给定一个图 $G(V,W)$ 以及对图结点集 V 的一个划分 $P=\{P_1,P_2,\cdots,P_n\}$($P_1\cup P_2\cup\cdots\cup P_n=V$；$P_i\cap P_j=\varnothing$($1\leqslant i\leqslant n$，$1\leqslant j\leqslant n$，$i\neq j$))，在此划分下图 G 的模块度为：

$$Q(G,P)=\sum_{p_i\in P}\frac{\sum_{x,y\in P_i}W(x,y)}{\sum_{x,y\in V}W(x,y)}-\left(\frac{\sum_{x\in P_i}\deg(x)}{\sum_{x\in V}\deg(x)}\right)^2$$

图 G 的模块度为 G 中所有可能划分得到的模块度中的最大值，即 $Q(G)=\max\{Q(G,P)\mid P\}$。

在计算图模块度的诸多算法中，Raghavan 等[8]提出的标签传播算法具有线性时间复杂度。其基本思想为初始时为每个图结点赋一个唯一的标签，一般为一个整数，表示每个图结点为一个社区。若两个结点的标签值相同，则它们属于同一个社区。算法运行中不断执行迭代过程。

在一次迭代过程中，对于每个图结点 $i\in X$，找出其所有邻居结点构成集合 $Neighbors(i)$，计算 w'：

$$w'=\sum_{\substack{j\in Neighbors(i)\\ L[i]=L[j]}}W(i,j)$$

假设 $Neighbors(i)$ 中包括 s 个标签，将结点集合 $Neighbors(i)$ 根据标签值划分成 s 个子集 N_t($1\leqslant t\leqslant s$)，每个子集中的结点具有相同的标签值。对于每个子集 N_t，计算 $w_t=W(i,j)$($j\in N_t$)。找出 w_t 的最大值 $w_{\max}$。若 $w_{\max}>w'$，找出 $w_{\max}$ 对应的邻居结点子集 N'(若存在多个这样的子集，则随机选一个)。将集合 N' 对应的标签值赋给结点 i。

如果在某一次迭代过程中所有图结点 $i\in X$ 的标签值 $Label[i]$ 都没有发生改变，则停止迭代过程，算法结束，得到一个所有图结点的标签集合。标签相同的结点组成图结点的一个子集，所有子集构成图结点的一个划分。

虽然标签传播算法在时间复杂度上有着巨大的优势，但最终得到的结果与 Newman 等提出的贪心算法相比损失了很大的精度。图折叠算法在标签传播算法的基础上做了两点改进，如算法 1 所示。图折叠算法和标签传播算法都是根据贪心策略将一个结点从一个社区移动到另外一个社区。但在图折叠算法中，移动一个结点到其他社区时遵循图模块度增量最大原则。当移动任何结点都不能使图模块度增大时，图折叠算法执行折叠操作。图折叠方式如图 2 所示。

算法 1　Graph Folding Algorithm(GFA)

输入：图 G(V,W)

输出：图结点的 Label 数组 L

1. 初始化，L_1[i]=i(i∈X)；

2. L_2[i]=Labelling(G)；

3. whileModularity(G,L_1)<Modularity(G,L_2) do

　3.1. $L_1=L_2$；

　3.2. G=Folding(G,L_2)；

　3.3. L_2=Labelling(G)；

4. return L_1

Function Labelling(Graph G=(X,W))

1. 初始化：对于每个图结点 i∈X，L[i]←i；

2. for i∈X

2.1. 找出结点 i 的所有邻居节点的标签集合L_i；

2.2. for l∈L_i，如果将结点 i 的标签变为 l 之后能使得图模块度增大，则执行 L[i]←l；

3. 如果在步骤 2 中每个结点的标签都没有发生改变，return L，否则转向步骤 2。

Function Folding(Graph G=(X,W),Label L)

1. 定义一个新的顶点集X′，X′中的每一个结点 c 代表 G 中的一个社区，社区即标签相同的结点集合；

2. 定义一个权值函数$W'(c_1,c_2)=\sum_{\substack{i\in c_1,j\in c_2\\ L[i]=L[j]}} W(i,j)$；

3. return G′=(X′,W′)。

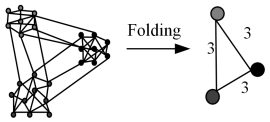

图 2 图折叠的社区合并

Fig. 2 Community aggregation of graph folding

3 CNF 公式的社区结构

一个 CNF 公式 F 是子句的合取 $F=[C_1\wedge C_2,\cdots,C_m]$，也可以表示为子句的集合 $F=\{C_1,C_2,\cdots,C_m\}$。每个子句是若干个文字的析取($C_i=L_1\vee L_2\vee\cdots\vee L_l$)或表示为文字的集合$C_i=\{L_1,L_2,\cdots,L_l\}$，$l$ 为该子句的长度。一个文字是任一变元 x 或为 x 的否定$\neg x$。

k-SAT 问题是 SAT 问题的子问题，对应的是一类具有特殊结构的 CNF 公式，即 k-CNF 公式。k-CNF 公式是指公式中每个子句包含的文字个数恰好为 k 的一类 CNF 公式。3-CNF 公式即每个子句中恰好有 3 个文字的 CNF 公式，若在 3-CNF 公式中固定每个变元出现的次数为 4，则可以得到正则(3,4)-CNF 公式。

例 1 一个 CNF 公式实例为$f_1=[C_1\wedge C_2\wedge C_3\wedge C_4\wedge C_5\wedge C_6\wedge C_7\wedge C_8]$，其中$C_1=(\neg x_2\vee\neg x_3\vee\neg x_4\vee x_5)$，$C_2=(\neg x_1\vee\neg x_5\vee x_6)$，$C_3=(\neg x_5\vee x_7)$，$C_4=(x_1\vee\neg x_6\vee x_7)$，$C_5=(\neg x_1\vee\neg x_2\vee x_5)$，$C_6=(\neg x_1\vee\neg x_3\vee x_5)$，$C_7=(\neg x_1\vee\neg x_4\vee x_5)$，$C_8=(\neg x_1\vee x_2\vee x_3\vee x_4\vee x_5\vee\neg x_6)$。

根据变元交互图模型可以将一个给定的 CNF 公式转化为一个与之对应的图。在 CNF 公式的变元交互图中，图结点与公式变元一一对应，两个结点之间有边表示这两个变元出现在同一个子句中。一个子句产生$\binom{|c|}{2}$条边，$|c|$为子句长度。由于两个变元出现在同一个子句中的次数不固定，为了保存两个变元出现在同一子句中的次数信息，若两个变元 x，y 之间有边，则采取 $W(x,y)$与 x，y 出现在同一个子句中的次数与子句长度有关的做法。

给定一个 SAT 实例 Γ，变元集为 X，Γ 对应的变元交互图为 $G(X,W)$，权值函数 W 为：

$$W(x,y)=\sum_{\substack{c\in\Gamma\\ x,y\in c}}\frac{1}{\binom{|c|}{2}}$$

CNF 公式f_1对应的变元交互图如图 3 所示。利用图折叠算法对公式f_1对应的变元交互图进行社区划分，结果如图 4 所示。

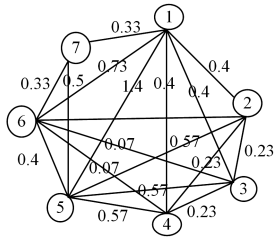

图 3 f_1对应的变元交互图

Fig. 3 Variable incidence graph of f_1

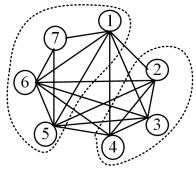

图 4 对图 3 进行社区划分的结果

Fig. 4 Community decomposition result in Fig. 3

4 CNF 公式的求解过程

求解 SAT 问题的算法分为完备的和非完备的算法。非完备算法大多数是基于局部搜索，在问题有解时快速找到一个解，但问题无解时不能证明其不可解。完备算法在问题无解时能证明其不可解。当前求解 SAT 问题的完备算法都是基于 DPLL 算法的，其核心思想如算法 2 所示。DPLL 算法的核心是通过布尔约束传播(Boolean Constraint Propagation，BCP)进行递归回溯，搜索整个解空间，此过程将会检查 CNF 公式的每一个可能赋值，直到找到公式的一个可满足解或者证明公式不可满足为止。DPLL 的搜索过程类似于一棵完全二叉树的遍历。随着变元数的增加，公式的解空间将会呈指数级增长，DPLL 算法可以避免不能满足公式的赋值，从而减小搜索树的大小。

算法 2 DPLL

输入：CNF 公式

输出：CNF 公式的一个解或 UNSAT

1. 在公式上执行 BCP 过程。

2. 如果公式的值为真，return true。

3. 如果公式的值为假，return false。

4. 如果公式的值未知：

4.1. 选择一个未被赋值的变元x′；

4.2. 将x′赋值为 true 或 false，return DPLL。

DPLL 算法求解f_1的过程如图 5 所示，其中实线表示对变元进行赋值，虚线表示布尔约束传播过程，圆结点表示发生了冲突，最右边的灰色路径表示得到最终解的步骤，分支表示回溯。

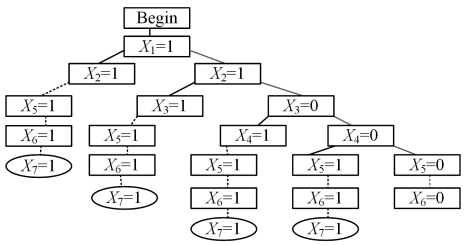

图 5 DPLL 算法求解f_1的过程

Fig. 5 DPLL process for solving f_1

DPLL 算法在搜索解空间时虽然能避免很多无效的搜索过程。但其在遇到冲突时没有从中得到任何有用的信息，不能避免很多重复的冲突。而且该算法只回溯到同一决策层，称为时序回溯。时序回溯使得算法仍然搜索相同的空间，导致算法在求解过程中将在一片注定会失败的解空间中浪费时间。

目前的完备求解器引入了冲突分析和子句学习策略(CDCL)[14]。引入该策略之后，DPLL 算法在每次遇到冲突时会对导致冲突的布尔约束传播过程进行分析，由此产生一个学习子句并将其添加到子句库中。通过学习子句，DPLL 过程回溯到其他决策层，称为非时序回溯，能避免对很大一部分解空间的重复搜索。

对于例 1 中的 CNF 公式 f_1，在对 x_2 赋值为 true 之后开始了一次 BCP 过程，最后产生冲突。根据 CDCL 策略对这次冲突进行分析可以得到一个学习子句 $C_9=(\neg x_1 \vee \neg x_5)$ 并将其加入 f_1 中。子句 C_9 加入之后，回溯到 C_9 中最近被赋值的变元所在的决策层，即 x_1 所在的决策层，开始布尔约束传播过程。引入冲突分析和子句学习策略之后 f_1 的求解过程如图 6 所示。

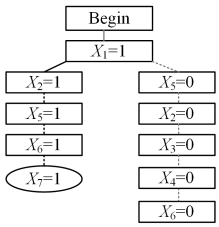

图 6　引入 CDCL 策略后 DPLL 算法求解 f_1 的过程

Fig. 6　DPLL process for solving f_1 after introducing CDCL strategy

5　实验结果

5.1　正则(3,4)-CNF 公式的模块度比较

本文实验中计算了 3 类 CNF 公式的模块度与社区数目，然后对实验结果进行了分析。3 类 CNF 公式分别是随机 3-CNF 公式、由随机 3-CNF 公式归约转换而来的正则(3,4)-CNF 公式以及随机正则(3,4)-CNF 公式。实验中随机 3-CNF 公式集由程序生成，生成过程中通过改变公式变元数以及子句数来得到变元数以及子句约束密度不同的随机 3-CNF 公式。再利用程序按照文献[4]中的规则将每一个随机 3-CNF 公式归约转换到正则(3,4)-CNF 公式，得到由随机 3-CNF 公式集归约转换而来的正则(3,4)-CNF 公式集。对于每一个由随机 3-CNF 公式归约转换而来的正则(3,4)-CNF 公式，利用程序生成一个子句数和变元数与其相等的随机正则(3,4)-CNF 公式，以此得到一个随机正则(3,4)-CNF 公式集。3 类 CNF 公式集合中的公式数目相同，一个随机 3-CNF 公式对应一个通过归约转换得到的正则(3,4)-CNF 公式，一个通过归约转换得到的正则(3,4)-CNF 公式对应一个子句数和变元数与其相同的随机正则(3,4)-CNF 公式。

产生随机 3-CNF 公式时令变元数分别为 100,500,1000,2000,5000，取子句约束密度分别为 3,4,5,6。实验结果如表 1 所列。在表 1 中，每一个变元数和一个子句约束密度对应 3 个公式的模块度(♯mod)和社区数目(♯com)，从上到下分别为随机 3-CNF 公式、随机正则(3,4)-CNF 公式以及由随机3-CNF 公式归约转换得到的正则(3,4)-CNF，两类正则(3,4)-CNF 具有相同的变元数和子句数。

表 1　3 类 CNF 公式模块度的比较

Table 1　Comparison of modularity of three types of CNF formulas

	100		500		1000		2000		5000	
	♯mod	♯com	♯mod	♯com	♯mod	♯com	♯mod	♯com	♯mod	♯com
3	0.212230	6	0.227828	12	0.231167	12	0.226560	14	0.215551	15
	0.390424	58	0.476493	16	0.581465	5	0.676781	5	0.796568	9
	0.981916	225	0.987318	502	0.988770	643	0.989737	941	0.990511	1449
4	0.179456	6	0.186395	10	0.184496	10	0.200354	8	0.177334	13
	0.390354	62	0.513649	16	0.625427	4	0.725391	6	0.822851	11
	0.982674	256	0.987692	579	0.988780	778	0.989583	1187	0.990307	1626
5	0.151040	6	0.164499	9	0.166979	11	0.176332	8	0.149877	15
	0.390212	65	0.547804	9	0.662350	5	0.752618	7	0.840208	13
	0.983562	284	0.988040	641	0.988966	905	0.989580	1317	0.990214	1955
6	0.142660	7	0.152373	10	0.162208	7	0.157535	8	0.162717	7
	0.391743	59	0.586811	5	0.690367	6	0.774692	8	0.852771	12
	0.984425	306	0.988191	698	0.989015	1048	0.989633	1393	0.99019	2277

根据实验结果可以看出，随机 3-CNF 公式的模块度较低，大部分小于 0.2，随着子句约束密度的增大，公式模块度与社区数目呈降低趋势，该实验结果与文献[13]中的结果一致。根据文献[13]中的结果可以发现，公式的社区数目与公式的模块度无直接关系。

与原 3-CNF 公式相比，归约转换得到的正则(3,4)-CNF 的模块度非常高，原因是在归约转换到正则(3,4)-CNF 的过程中，原 3-CNF 公式的每一个变元都对应一个基本的归约步骤，该步骤会引入多个变元集构造极小不可满足公式。在同一步骤中引入的若干变元集中，变元联系较为紧密，出现在同一子句中的概率较大。对于不同步骤中引入的变元集，变元出现在同一子句的概率较小。因此，变元间构成了明显的社区结构。

在变元数相同的情况下，归约转换得到的正则(3,4)-CNF 公式的模块度要比随机正则(3,4)-CNF 公式的模块度高。随着变元数的增加，随机正则(3,4)-CNF 公式的模块度增大，且与归约转换得到的正则(3,4)-CNF 公式的模块度的差值不断缩小。尽管当变元数较大时随机正则(3,4)-CNF 公式具有较高的模块度，但与归约转换得到的正则(3,4)-CNF 公式相比，其社区数目远小于后者。这是由于在随机正则(3,4)-CNF 公式中，变元在一个子句中的出现是随机的，每一个变元都有可能与任意一个其他变元出现在同一个子句中。

5.2　正则(3,4)-CNF 公式的学习子句与其社区结构的关系

在使用 DPLL 算法结合冲突驱动子句学习策略求解 CNF 公式时，对于产生的每一个学习子句 C^l，令 ls 为 C^l 的长

度，lc 为C'所包含的变元所在的社区数目，定义 $cs=lc/ls$。cs 值是一个大于 0 且小于或等于 1 的数。若 $cs=1$，则该学习子句的每一个变元都位于不同的社区；若 $cs=1/ls$，则该学习子句的所有变元都位于同一个社区。定义 $AVEcs$ 为求解一个公式的过程中产生的所有学习子句的平均 cs 值，即 $AVEcs=\sum_{i=1}^{conf} cs_i/conf$，$conf$ 为冲突次数。选取 5 个 3-CNF 公式实例，对比归约转换前后 3-CNF 公式与正则(3,4)-CNF 公式的 $AVEcs$ 值，实验结果如表 2 所列。

表 2 归约转换前后公式的 $AVEcs$ 值比较

Table 2 Comparison of $AVEcs$ values of formaulas before and after transformtion

	Ins1	Ins2	Ins3	Ins4	Ins5
3-CNF	0.589656	0.673294	0.780905	0.226161	0.493986
(3,4)-CNF	0.991365	0.988194	0.994928	0.959339	0.982789

根据表 2 的结果可以看出，在求解过程中每一个归约转换后得到的正则(3,4)-CNF 的 $AVEcs$ 值非常接近于 1。这说明在求解此类公式时冲突基本上发生在变元社区之间，而社区内部很少发生冲突。社区内部变元组成很多 MU(2)和 MU(4)公式，这些公式的引入没有增加求解难度，故没有发生冲突，这也从另一个角度说明了归约转换得到的正则(3,4)-CNF 公式保留了原 3-CNF 公式的求解难度。

结束语 与随机 3-CNF 公式相比，归约转换得到的正则(3,4)-CNF 除了变元数与子句数极大增加外，公式的结构也发生了很大的变化。本文利用图社区结构与模块度的概念，描述了这两类公式的差异。此外，文中研究了归约转换前后的公式在求解过程中产生的学习子句与公式的社区结构的关系。根据文献[13]的实验结果，当随机 3-CNF 公式的子句约束密度较小时，其模块度较大；若固定子句约束密度，变元数目增幅较大时，公式的模块度呈降低趋势。随机正则(3,4)-CNF 公式的子句约束密度固定为 4/3，属于较低的值，但当其变元数增加时，其模块度也呈增长趋势，本文没有对这一现象做详细的讨论研究，这是本文的不足之处。随着变元数的不同，随机正则(3,4)-CNF 公式的模块性差异很大，下一步研究内容为正则$(3,k)$-CNF 公式的模块度与其求解难度的关系，以及利用 CNF 公式的社区结构对学习子句进行删除。

参 考 文 献

[1] ASPVALL B, PLASS M F, TARJAN R E, et al. A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas[J]. Information Processing Letters, 1979, 8(3): 121-123.

[2] COOK S A. The complexity of theorem proving procedures [C]// Proceeding of the 3rd Annual ACM Symposium on Theory of Computating. 1971: 151-158.

[3] XU D Y. Application of minimal unsatisfiable formulas to polynomially reduction for formulas[J]. Journal of Software, 2006, 17(5): 1204-1212.

[4] XU D Y, WANG X F. A Regular NP-Complete Problem and Its Inapproximability[J]. Journal of Frontiers of Computer Science & Technology, 2013, 7(8): 691-697.

[5] GIRVAN M, NEWMAN M E J. Community Structure in Social and Biological Networks[J]. PNAS, 2002, 99(12): 7821-7826.

[6] NEWMAN M E J, GIRVAN M, et al. Finding and evaluating community structure in networks[J]. Physical Review E, 2004 (69): 17-32.

[7] NEWMAN M E J. Fast algorithm for detecting community structure in networks[J]. Phys. Rev. E Stat. Nonlin. Soft. Matter. Phys., 2004(69): 66-133.

[8] RAGHAVAN U N, ALBERT R, KUMARA S. Near linear time algorithm to detect community structures in large-scale networks[J]. Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics, 2004(76): 36-106.

[9] VINCENT D B, JEAN-LOUP G, RENAUD L, et al. Fast unfolding of communities in large networks[J]. arXiv: 0803.0476.

[10] SINZ C. Visualizing the internal structure of SAT instances preliminary report[C]// Sat-the Seventh International Conference on Theory & Applications of Satisfiability Testing. DBLP, 2004.

[11] BIERE A, SINZ C. Decomposing sat problems into connected components[J]. JSAT, 2006, 2(1-4): 201-208.

[12] ANSÓTEGUI C, BONET M L, GIRÁLDEZ-CRU J, et al. Community Structure in Industrial SAT Instances[J]. Journal of Artificial Intelligence Research, 2019(66): 443-472.

[13] ANSÓTEGUI C, GIRÁLDEZ-CRU J, LEVY J. The Community Structure of SAT Formulas[C]// International Conference on Theory and Applications of Satisfiability Testing. Berlin, Heidelberg: Springer, 2012(7317): 410-443.

[14] SILVA J P M, LYNCE I, MALIK S. Conflict-Driven Clause Learning SAT Solvers[J]. Frontiers in Artificial Intelligence & Applications, 2009, 185(4): 131-153.

HE Bin, born in 1995, postgraduate. His main research interests include algorithm design and analysis.

XU Dao-yun, born in 1959, professor, Ph.D supervisor, is a senior member of China Computer Federation. His main research interests include computability and computational complexity.