arXiv:2603.10944v1 [cs.DS] 11 Mar 2026

2-CNF的简单极小不可满足子集

OliverKullmann[1] 斯旺西大学计算机科学系，英国

Edward Clewer 独立研究者，英国

摘要

我们研究了2-CNF布尔公式的最小不可满足子集（极小不可满足子句集，MUSs），并以Abbasizanjani-Kullmann对极小不可满足2-CNF（2-MU）的分类为基础。首先，我们给出一种用于识别2-MU的线性时间过程。随后，我们研究寻找一个简单MUS的问题。一方面，我们扩展了Kleine Büning等人的结果；他们证明了判定是否存在亏值为1的MUS是NP 完全的。另一方面，我们证明，判定或寻找包含一个或两个单位子句（这类子句是特殊的亏值为1的MUS）的MUS可以在多项式时间内完成。最后，我们提出一种增量多项式时间算法，用于处理某类特殊的MUS，即至少包含一个单位子句的MUS。

最后，我们讨论主要的开放问题，以更深入地理解 2-CNF 中容易与困难的极小不可满足子句集的全貌。

2012 年 ACM 学科分类计算理论 → 逻辑；计算理论 →算法设计与分析

关键词与短语最小不可满足性，MUS，2-CNF，多项式时间

数字对象标识符10.4230/LIPIcs...

1 引言

一般而言，一个不可满足的合取范式公式包含许多不可满足性的“来源”，而主要方法是考察极小不可满足子集（简称 MUS）。MUS 是不可行性的根本原因，可应用于诊断、错误定位、产品配置、模型检测（CEGAR）和冗余消除。

关于所有最小不可满足子集的完全枚举，早期的一篇论文是 [16]。一项重要的里程碑成果见于 [15], ，该成果总结了该领域的大部分进展，并提出了 MARCO 框架。通用工具集见于 [4], ，而 [5]则介绍了一种用于近似计数的概率方法。

由于我们关注的是特殊输入类别 2-CNF，因此特殊类别的 MUS 枚举对我们来说十分重要。目前看来，关于这一问题仅有的研究针对 Horn CNF（每个子句中至多有一个正文字）。[18, 定理 5] 表明，对于 Horn CNF，可以在多项式延迟下进行 MUS 枚举。他们还对 Horn CNF 的一个特殊情形作了评论，值得在此讨论。具体而言，他们考虑了“核心 KB”，即仅含混合子句的 2-CNF，每个混合子句都包含一个正文字和一个负文字（这类子句总是可满足的）。在 [18, 页面 282]中，他们指出，通过枚举路径，核心 KB 可以在多项式延迟下找到极小公理集（有关路径枚举的最新论文，见 [19] ）。我们的情形更为复杂，因为我们考虑的是一般的 2-CNF，其中必须处理路径上出现互补文字的问题（“矛盾路径”）。因此，即使在最简单的情形下，即固定两个单位子句（我们称之为“族 I”），也需要更复杂的算法才能找到一个 MUS，而且目前尚不清楚能否在多项式延迟下进行枚举。如果放宽条件，允许 MUS 具有更一般的形式

[1] 通讯作者

© Oliver Kullmann 和 Edward Clewer；依据知识共享许可协议 CC-BY 4.0 授权莱布尼茨国际信息学会议论文集 Schloss Dagstuhl – 莱布尼茨信息学中心，Dagstuhl 出版社，德国

LIPICS

XX：2-CNF 的 MUS

（我们语境中的“族 II”），就会出现同一个 MUS 被不同路径重复枚举的问题，因此我们只能保证在增量多项式时间内完成枚举。 [3] 从实践角度研究 Horn CNF。

回到一般 CNF公式，另一种框架是寻找（单个）“简单”的 MUS，通常将“简单”理解为“短（或最短）”。这一方向较早的一篇论文是 [17], ，而近期一篇同时提出一般框架的文章是 [9]。

我们的目标是采用比单纯规模更智能的复杂度度量——一个 MUS 在结构上可能非常简单，但仍然很长。基于衡量极小不可满足子句集（极小不可满足 CNF公式）的复杂度指标“亏格”，我们考虑在这方面最简单的 MUS，即亏度为一的 MUS（也就是说，子句数恰好比变量数多一个）。此外，我们着眼于多项式时间的结果，因此聚焦于 2-CNF 中亏度为一的 MUS。我们在此指出，枚举任意 2-CNF 的 MUS 的复杂度仍悬而未决——枚举某个特殊类别的 MUS，可能比枚举一般类别更容易，也可能更难。

因此，我们在此结合了两条研究脉络，聚焦于亏格为一的情形以及 2-CNF。亏格为一的极小不可满足子句集通常简称为 MU(1)。 [7, 定理 14]证明，判断输入合取范式是否属于 MU(1) 可以在二次时间内完成（ [8] 将这一结果扩展到了 MU(k)，并实现了多项式时间算法）。基于 [1]， [2], 完整地分类了 2-CNF极小不可满足公式，称为 2-MU。我们只需要该分类中最简单的情形，即 2-MU 与 MU(1) 的交集。这类极小不可满足子句集分为四种类型，本文称之为“I–IV 类”。[2]

在介绍我们的结果之前，我们先提及“具有给定斜对称性的有向图”这一概念；它可以被视为从图论视角研究 2-CNF。[3] 这里的奠基性论文是 [10]。所谓“斜对称性”，类似于将文字视为顶点，并使取补成为反自同构；也就是说，弧 $a \to b$ 会映射为其逆否对应弧 $b \to a$。[10] 提出了弧正则路径（路径上不包含某条弧及其逆否对应弧），并证明关于这类路径的基本问题（包括寻找最短路径）都能在线性时间内解决。我们需要考虑顶点正则路径（路径上不包含某个文字及其补文字）。众所周知，可以通过将每个顶点拆分成两个顶点（分别对应入弧和出弧，并用一条新弧连接这两个新顶点），将顶点正则路径问题归约到该问题。

最后介绍我们的结果：
1. 第 4 节讨论 2-MU 的判定问题。基本算法（利用 2-SAT 在线性时间内完成判定）需要二次时间；我们在定理 4.5中给出一个线性时间算法，该算法基于定理 4.4中关于“受检奇异 DP归约”的一般结果。**2.** 第 5节基于定理 5.5、5.6中的结构性结果，针对亏度为一的最小不可满足子集提出更精细的困难性分析，并通过定理 5.4中的族 III 和 IV，更准确地指出复杂性的成因。**3.** 第 6 节展示如何找到一个包含某个单位子句的最小不可满足子集。定理 6.1 刻画了包含两个单位子句的情形；定理 6.4刻画了包含至少一个单位子句的情形；推论 6.7 表明，找到一个包含至少一个单位子句的最小不可满足子集（即属于族 I 或 II）可以在二次时间内完成。**4.** 最后，第 7 节讨论包含至少一个单位子句的极小不可满足子句集的枚举。对于

[2] 在 [2] 文中，四个族仅以其技术记号指代。本文提交至 SAT2026 后不久， [2] 的扩展版本将发布在 arXiv 上。不过，本文所需内容已由 [2]的第 4 节和附录 C 完全涵盖。 [3] 这一主题将在即将推出的 [2]扩展版中作更深入的讨论。

给定单位子句的情形，定理 7.3 证明了该算法的正确性，而定理 7.4 给出了其复杂度。
同样，定理 7.7 证明了处理所有单位子句的算法的正确性，而定理 7.8 给出了其复杂度。

2 预备知识

我们使用 $\mathbb{N}_0 = \{0, 1, 2, ...\}$。使用大 O 陈述复杂度结果时，如果该项变为零，则将其解释为线性时间（在通常的随机访问模型中，这是普通算法的最短运行时间）。

2.1 子句集

设 $\mathcal{VA}$ 为变量集，文字集 $= \mathcal{VA} \cup \{v : v \in \mathcal{VA}\}$ 为文字集，其中取补满足 $\overline{x} = x$ 对所有 $x \in$ 文字集。对于 $L \subseteq$ 文字集，我们使用 $\overline{L} := \{\overline{x} : x \in L\}$。$L$ 包含冲突当且仅当 $L \cap \overline{L} \neq \emptyset$（否则它是无冲突的）。

现在，子句 C 是一个有限的 $C \subseteq$ 文字集，其中 $C \cap \overline{C} = \emptyset$（$C$ 不包含冲突），所有子句的集合记为 $\mathcal{CL}$。最后，子句集是 $\mathcal{CL}$ 的有限子集，所有子句集的集合记为 $\mathcal{CLS}$。一种特殊子句是空子句 $\bot := \emptyset \in \mathcal{CL}$，一种特殊子句集是空子句集 $\top := \emptyset \in \mathcal{CLS}$。对于 $k \in \mathbb{N}_0$，我们定义 k–$\mathcal{CLS} := \{F \in \mathcal{CLS} \mid \forall C \in F : |C| \leq k\}$（所有子句的长度（或“宽度”）至多为 k）。特别地，我们使用 2–$\mathcal{CLS}^* := \{F \in 2\text{–}\mathcal{CLS} : \bot \notin F\}$。

通过 var : LIT $\to$VA，我们可以获得文字的底层变量；对于子句 C，我们使用 $\mathrm{var}(C) := \{\mathrm{var}(x) : x \in C\} \subseteq$ VA；对于子句集 F，则使用 $\mathrm{var}(F) := \bigcup_{C \in F} \mathrm{var}(C) \subseteq$ VA。借助 $\mathrm{lit}(F) := \mathrm{var}(F) \cup \overline{\mathrm{var}(F)}$，我们可以得到 F 的文字集；因此 $x \in \mathrm{lit}(F) \Leftrightarrow \{x, \overline{x}\} \cap \bigcup_{C \in F} C \neq \emptyset$。文字 $\overline{x}$ 在 F 中的度为 $\mathrm{ld}_F(x) := |\{C \in F : x \in C\}| \in \mathbb{N}_0$，变量 v 的度为 $\mathrm{vd}_F(v) := \mathrm{ld}_F(v) + \mathrm{ld}_F(\overline{v})$。对于 $k \in \mathbb{N}_0$，令 $\mathrm{var}_k(F) := \{v \in \mathrm{var}(F) : \mathrm{vd}_F(v) = k\}$ 表示度为 k 的变量集合，并令 $n_k(F) := |\mathrm{var}_k(F)| \in \mathbb{N}_0$ 表示其数量。类似地，我们使用 $\mathrm{lit}_k(F) := \{x \in \mathrm{lit}(F) : \mathrm{ld}_F(x) = k\}$ 和 $n'_k(F) := |\mathrm{lit}_k(F)| \in \mathbb{N}_0$。对于 $F \in$CLS，若变量 $v \in \mathrm{var}(F)$ 满足 $\mathrm{ld}_F(v) = 1$ 或 $\mathrm{ld}_F(v) \equiv 1$，则称其为奇异变量；所有奇异变量构成的集合为 $\mathrm{var_s}(F) = \{v \in \mathrm{var}(F) : \{v, \overline{v}\} \cap \mathrm{lit}_1(F) \neq \emptyset\}$。而若 v 满足 $\mathrm{ld}_F(v) = 1$ 且 $\mathrm{ld}_F(\overline{v}) = 1$，则称其为 1-奇异变量；所有 1-奇异变量构成的集合为 $\mathrm{var}_{1\mathrm{s}}(F) = \{v \in \mathrm{var}(F) : \{v, \overline{v}\} \subseteq \mathrm{lit}_1(F)\}$。

对于 $F \in$CLS，我们使用以下度量：$n(F) := |$变量数$F| \in \mathbb{N}_0$（变量数）$c(F) := |F| \in \mathbb{N}_0$（子句数）$u(F) := |\{C \in F : |C| = 1\}| \in \mathbb{N}_0$（单子句数）$\ell(F) := \sum_{C \in F} |C| \in \mathbb{N}_0$（我们的公式大小度量）。$\delta(F) := c(F) - n(F) \in \mathbb{Z}$（亏格）。

关于平凡情形，我们有 $n(F) = 0 \Leftrightarrow F \in \{\top, \{\bot\}\}$ 以及 $c(F) = 0 \Leftrightarrow \ell(F) = 0 \Leftrightarrow F = \top$。唯一非常规的度量是 $u(F)$，即单子句数。

语义由部分赋值给出，部分赋值是映射 $\varphi : V \to \{0, 1\}$，其中 $V \subseteq$ VA。对于 $v \in V$，我们定义 $\varphi(\overline{v}) = \varphi(v)$，因此 φ 定义在 $V \cup \overline{V}$ 上。若存在文字 $x \in C$ 使得 $\varphi(x) = 1$，则子句 C 被 φ 满足；而子句集若其中所有子句 $C \in F$ 均被 φ 满足，则称其被 φ 满足。所有可满足子句集的集合记为 SAT $\subseteq$ CLS，其中 $\top \in$ SAT；而 USAT := CLS \\ SAT 是所有不可满足子句集的集合。若一个子句集 F 是极小不可满足的（即为一个 MU），则它是不可满足的，但删除任意一个子句都会使其变得可满足；所有 MU 的集合定义为 MU := $\{F \in$ USAT $\mid \forall C \in F : F \setminus\setminus \{C\} \in$ SAT $\}$，其中 $\{\bot\} \in$ MU。并且 2–MU := MU $\cap$ 2–CLS。我们还使用 $\mathrm{MU}_{\delta=1} := \{F \in \mathrm{MU} : \delta(F) = 1\}$。回顾 $\delta(F) \geq 1$

对所有 $F \in \mathcal{MU}$; 参见 [11] 以获取更多背景信息。另一个（平凡的）事实是变量$_{1s}(F)$ $=$变量$_2(F)$对于 $F \in \mathcal{MU}$。

最后， $F \in \mathcal{CLS}$ 的一个最小不可满足子集是一个包含于 F 的极小不可满足子句集，而 F 的所有最小不可满足子集组成的集合为 $\mathrm{MU}(F) := \{F' \subseteq F\colon\ F' \in$ 极小不可满足子句集$\}$。我们用 $\mathrm{MU}_{C_1,\ldots,C_m}(F) := \{F' \in \mathrm{MU}(F) : C_1, \ldots, C_m \in F'\}$ 表示。

2.2 有向图

有向图 G 具有有限的顶点集 $V(G)$，而弧是不同顶点组成的有序对：$E(G) \subseteq \{(v,w) \in V(G)^2 : v \neq w\}$。我们用 G^{t} 表示通过反转弧的方向从 G 得到的有向图。有向图 G 的子有向图 G' 是一个有向图，其 $V(G') \subseteq V(G)$ 且 $E(G') \subseteq E(G)$。有向图 G 中的路径 P 是一个满足 $V(P) \neq \emptyset$ 的子有向图，其长度记为 $\mathrm{lgth}(P) := |E(P)| \geq 0$（即弧的数量；$\mathrm{lgth}(P) = |V(P)| - 1$）。路径 P中的首顶点和末顶点分别记为 $\mathrm{first}(P)$ 和 $\mathrm{last}(P) \in V(P)$。路径 P、P' 的拼接记作 $P; P'$，条件是 $\mathrm{last}(P) = \mathrm{first}(P')$ 且 $V(P) \cap V(P') = \{\mathrm{last}(P)\}$。作为一种特殊情况，将顶点 v 附加到路径 P（其中 $v \notin V(P)$）记作 $P; v$。对于集合 A，我们用 $G - A$ 表示从 G 中移除满足 $v \in A$ 的顶点 $v \in V(G)$ 及其关联弧后得到的子有向图 G。最后，关系 $x \xrightarrow{G} y$ 成立，当且仅当存在一条从 x 到 y、位于 G 中的路径。

2.3 带补图的有向图

文献中有一种具有给定斜对称性的有向图的概念，我们通过将文字作为顶点，并利用补文字来实现斜对称性，从而处理这种有向图。

▶**定义 2.1**。*A*带部分补的有向图是一个有向图 G，其顶点为文字，即 $V(G) \subseteq LIT$。所有带部分补的有向图的集合记为 *DPC*。带补图的有向图是一种带部分补的有向图，其顶点集在补文字运算下封闭（即 $V(G) = V(G)$），且其弧集在逆否命题运算下封闭，也就是说，对所有$(x,y) \in E(G)$，都有$(x,y)\colon = (y,x) \in E(G)$。所有带补图的有向图的集合记为 ~~*DC*~~。

对于 $G \in \mathcal{DPC}$ 中的弧集合 $A \subseteq E(G)$，记作

$$\mathcal{CL}(A) := \{\{\overline{x}, y\} : (x,y) \in A\} \in \mathcal{CLS}$$

我们记相应的子句集；因此 $\mathcal{CL}(E(\mathrm{idg}(F))) = F$. 对于单条弧 e，我们用 $\mathcal{CL}(e)$ 表示 $\mathcal{CL}(\{e\})$ 中的唯一元素。子句 $\mathcal{CL}(e) = \{x,y\}$ 对于 $e = (x,y)$ 是单子句，当且仅当 $y = x$；此时，弧 $e = (x,y)$ 称为一个**单位**。

▶ **定义 2.2**。考虑 $G \in DPC$。路径 P 称为正则路径（更准确地说是“顶点正则”，但本文不讨论弧正则路径），如果 $V(P)$ 无冲突。我们用 $\mathfrak{R}_{x,y}(G)$表示从 x 到 y的正则路径（作为子有向图）的集合。

▶ **定义 2.3**。考虑 $G \in DPC$。恰有一个冲突的路径 P，若去掉最后一个顶点后 P 是正则路径，则称为近正则路径。对于近正则路径 P，存在唯一的正则分解 $P = P_0; P_1$，其中 P_0 是正则路径或空有向图，而 P_1 是满足 $\mathrm{last}(P_1) = \mathrm{first}(P_1)$的路径。我们用 $\mathfrak{U}_x(G)$ 表示从 x 开始的 $\mathrm{idg}(F)$ 中的近正则路径集合（“*U*”表示“唯一冲突”）。

对于带补图的有向图中的一条路径 P，其长度为 $P \geq 1$，通过对所有弧取逆否命题并反转其顺序得到的**逆否命题**P，是一条从 $\mathrm{last}(P)$ 到 $\mathrm{first}(P)$ 的路径。

带补图的有向图的基本例子是**蕴含有向图** idg : 2–$\mathcal{CLS}^* \to \mathcal{DC}$，其中 idg($F$) 的顶点集为 $V(\mathrm{idg}(F)) = \mathrm{lit}(F)$，对于每个子句$\{x, y\} \in F$，都有弧 $(x, y), (y, x) \in E(\mathrm{idg}(F))$。因此，对于二元子句，我们得到两条弧；而对于单子句$\{x\}$，我们只得到一条弧，即 (x, x)。通过以各种形式构造蕴含有向图的子有向图，可以得到具有部分补 的有向图。

3 2-MU 分类回顾

对于 $F \in$ 2–极小不可满足子句集，我们有以下基本事实，在本文余下部分将自由使用这些事实:

- 文字集(F) = 文字集$_1(F) \cup$ 文字集$_2(F)$，变量集(F) = 变量集$_2(F) \cup$ 变量集$_3(F) \cup$ 变量集$_4(F)$ (
- [12])。由此可得 $n'_1(F) + n'_2(F) = 2n(F)$， $n_2(F) + n_3(F) + n_4(F) = n(F)$，以及 $n'_1(F) = 2n_2(F) + n_3(F)$， $n'_2(F) = n_3(F) + 2n_4(F)$。此外，变量集$_s(F)$ = 变量集$_2(F) \cup$ 变量集$_3(F)$。
- $0 \le u(F) \le 2$ 且 $u(F) \ne 0 \Rightarrow \delta(F) = 1$ ([14, 引理 8])。

亏格 2–极小不可满足子句集 中亏格为 $k \ge 1$ 的非奇异元素为$\boldsymbol{k = 1}$ $\{\bot\}\boldsymbol{k \ge 2}\,\mathcal{B}$ $_k \doteq v_1 \leftrightarrow v_2 \leftrightarrow v_3 \leftrightarrow \ldots \leftrightarrow v_{k-1} \leftrightarrow v_k \leftrightarrow \neg v_1$，其中有 k 个变量和 $2k$ 个子句，由[12]和 [1, 推论 2]给出。

3.1 四个家族

报告 [2] 有效地分类了 2–极小不可满足子句集的同构类型，其中两个子句集F, F'同构，当且仅当 F可通过变量重命名（有$n(F)!$ 种选择）以及文字翻转（有 $2^{n(F)}$种选择）转化为 F'。本文仅需讨论亏格为 1 的情形，并且只需进行粗略分类，现对此作回顾。

四个 I–IV 类，分别记作类 $\mathfrak{F}_I$、 $\mathfrak{F}_{II}$、 $\mathfrak{F}_{III}$、 $\mathfrak{F}_{IV}$，构成 2–极小不可满足子句集亏格为 1 的子类的一个划分。在 [2] 中，这些族通过代表元给出，形成四个族 U^2_n、 U^1_n、 $_i$、 U^0_n、 $_i$、U^0_n、 $_x$、 $_y$，并配以具体参数，以便明确其同构类型；但本文关注的是这些族在同构下的闭包，从而得到上述各类。下面我们给出这四个族的主要特征，以及其元素的通用公式（可能包含子情况）。表示这些公式时，我们用记号“$x \to \ldots \to y$”表示形如 $a \to b$ 的蕴含链，其中 var(a) $\ne$ var(b)（因此，相应子句 {a、 b} 的长度为二）；此外，链中的内部变量（端点 x、 y 中的变量除外，即省略号所隐藏的变量）彼此不同，也不同于（同一公式中）其他此类链的内部变量。文字 x、 $y \in \in$文字集 LIT 任意选取，但须满足 var(x) $\ne$ var(y)，且构成 $F \in$ 2–极小不可满足子句集（我们无需限定为 $\delta(F) = 1$，但这总是由这些条件推出的）：

I $F \in \mathfrak{F}_I \Leftrightarrow n_3(F) = 0 \wedge n_4(F) = 0 \Leftrightarrow u(F) = 2$.

Ia $\boxed{\{x\}, \{\overline{x}\}}$

Ib $\boxed{\{x\}, \{y\}} + \boxed{x \to \ldots \to \overline{y}}$

II $F \in \mathfrak{F}_{II} \Leftrightarrow n_3(F) = 1 \Leftrightarrow n_3(F) = 1 \wedge n_4(F) = 0 \Leftrightarrow u(F) = 1$.

IIa $\boxed{\{x\}} + \boxed{x \to \ldots \to \overline{x}}$ $(\mathrm{var}_3(F) = \{\mathrm{var}(x)\})$

IIb $\boxed{\{x\}} + \boxed{x \to \ldots \to y \to \ldots \to \overline{y}}$ $(\mathrm{var}_3(F) = \{\mathrm{var}(y)\})$

III $F \in \mathfrak{F}_{III} \Leftrightarrow n_3(F) = 0 \wedge n_4(F) = 1$.

$\boxed{x \to \ldots \to \overline{x} \to \ldots \to x}$ $(\mathrm{var}_4(F) = \{\mathrm{var}(x)\})$

IV $F \in \mathfrak{F}_{IV} \Leftrightarrow n_3(F) = 2 \wedge n_4(F) = 0.$
$\boxed{x \to \ldots \to \overline{x} \to \ldots \to y \to \ldots \to \overline{y}}$ $(\mathrm{var}_3(F) = \{\mathrm{var}(x), \mathrm{var}(y)\})$

4 判定 2-MU

我们的第一个结果是，2-MU 可在线性时间内判定（而使用线性时间内对 2-CNF 进行 SAT 判定的平凡算法，则需要二次时间）。我们的主要工具是受检奇异 *DP* 归约，用于判定 2-MU 的算法会运行受检奇异 DP 归约；若归约成功，则检查最终结果（具体取决于亏格是 1 还是 ≥ 2）。奇异 DP 归约已在文献中用于化简极小不可满足子句集，参见例如 [13]，该文进行了基础性研究。 [13, 引理 9] 确实明确列出了判定一个子句集是否为极小不可满足子句集所需检查的条件，这正是“受检”sDP 归约（它可能失败）的依据：

▶ 定义 *4.1*。回顾一下，对于 $F \in$ *CLS*，若变量 $v \in$ *var(F)* 满足 $ld_F(v) = 1$ 或 $ld_F(v) = 1$，则称其为奇异变量。主子句是包含所选文字 v（分别是仅出现一次的文字 v）的子句 $C \in F$（当 $ld_F(v) = ld_F(v) = 1$ 时，才有选择余地）；而对于 $m \geq 0$，侧子句 D_1、……、$D_m \in F$ 是包含补文字 v（分别是 v）的子句（不重复计数）。若满足以下任一条件，则 v, F 的 *sDP* 检查失败：

(i) $m = 0$; *(ii)* 存在 $i \in \{1, \ldots, m\}$，使得 C 和 D_i 在除 v 以外的某个变量上发生冲突；*(iii)* 存在 $i, j \in \{1, \ldots, m\}$，$i \neq j$，使得 $D_i \setminus C = D_j \setminus C$；*(iv)* 存在 $i \in \{1, \ldots, m\}$ 和 $E \in F$，满足 $v \notin \mathrm{var}(E)$ 且 $D_i \setminus C = E \setminus C$.[4]

对于 $F \in$*CLS* 与 $v \in$*VA*，若 *sDP* 检查失败，则一次 经检查的 ***sDP*** 归约步骤 （*csDP* 归约）的结果为 *FAIL*，否则结果为

$$F' := (F \setminus \{C, D_1, \ldots, D_m\}) \cup \{(C \cup D_i) \setminus \{v, \overline{v}\} : i \in \{1, \ldots, m\}\} \in \mathcal{CLS}.$$

针对 $F \in$*CLS* 的 完整 ***csDP*** 归约 会（非确定性地）应用 *csDP* 归约，直到要么达到失败，此时结果为 *FAIL*，要么不再存在奇异变量，此时结果为最终的 $F' \in$ *CLS*。

csDP 归约的下述基本性质直接由 [13, 引理 9]得出（也很容易直接验证）：

▶ **引理 4.2.** 对于 $F \in \mathcal{CLS}$ 成立：
1. 若完整 *csDP* 归约失败，则 $F \notin$极小不可满足子句集。**2.** 否则，对于所得的 F'，有 $F \in$极小不可满足子句集 $\Leftrightarrow F' \in$极小不可满足子句集。

▶ **问题 4.3**。 对任意子句集 $F \in$*CLS* 计算完整的检查式 *sDP*-约简，或对 *MU* 计算完整的（未检查）*sDP* 约简，其时间上界为 $O(n(F)^2 \cdot c(F))$，即三次时间：每个步骤都会使变量数和子句数各减少1，更新文字度数可以在线性时间内完成；然而，子句长度的增长可能会导致每个步骤新增 $O(n \cdot c)$ 个文字出现位置。这个时间上界能否改进？或者能否通过强指数时间假设证明一个下界？

[4] 该条件略强于条件 3(c)，在 [13, 引理 9]中：在那里我们检查 C 与 D_i 的归结子句是否等于某个已有子句 E，而在这里我们检查某个已有子句 E 是否包含该归结子句。此处这样做是为了使各条件的表述风格保持一致。

对于 2-MU，我们有一个非常特殊的情形：每个文字至多出现两次（而且检查式 sDP 约简会保持这一性质）。我们考虑稍微一般一些的情形，即（保持）有界文字度，并证明在这种情况下，可以在线性时间内执行（完整的）检查式 sDP 约简：

▶ **定理 4.4**。 考虑 $k \in \mathbb{N}$ 以及 $C \subseteq k$–*CLS*，使得对于 $F \in C$，所有文字度数也都由 k界定，并且 C 在检查式 *sDP*-约简下保持稳定。于是，可以在线性时间内对 $F \in C$ 执行完整的检查式 *sDP*-约简。

证明。 使用标准数据结构，我们可以首先在线性时间内计算该数组；对于每个文字，该数组给出它在子句中的出现位置；同时还可以计算文字度数数组。因而，找出所有（初始的）奇异变量也可以在线性时间内完成。现在，对于每个奇异变量，执行检查（使用文字—子句出现列表；注意，代价最高的步骤，即检查定义4.1中的条件 (iv)，耗时 $O(k^3)$），并在肯定的情况下用其归结式替换主子句和侧子句，同时更新文字度数（以及确定并找出新的奇异变量），所有这些都可以在常数时间内完成（在 k中）。 ◀

▶ **定理 4.5**。 对于 $F \in 2$–*CLS*，是否满足“$F \in MU$？”可以在线性时间内判定。

证明。 根据第 3 节，我们可以应用定理 4.4 （带有 $k = 2$），并在线性时间内计算 F 的完整检查式 sDP-约简，得到 F'。如果归约失败，或者 $k := \delta(F) \leq 0$，那么 $F \notin$ 极小不可满足子句集。否则 $F \in$ 极小不可满足子句集 当且仅当 $F' = \{\bot\}$ 对 $k = 1$成立，或者 F' 与 $\mathcal{B}_k$同构。这些检查也可以在线性时间内完成。 ◀

▶ **问题 4.6**。 对于 $F \in 2$–*CLS*，使用平凡算法可以在二次时间内找到 极小不可满足子句集 (F) 中的一个元素：

要求： $F \in USAT$确保： $F' \in$ 极小不可满足子句集(F)函数 FI

$ND(F \in 2\text{-}CLS)$ ▷ 确实适用于所有 $F \in CLS$

$F' \leftarrow F$对于 $C \in F$，如果 $F' \setminus \{C\} \in USAT$ </style>则返回 $' \leftarrow F_{'} \setminus \{C\}\ F'$

是否存在渐近意义上更快的算法？既然我们可以在线性时间内判定 *2-MU*（定理4.5），那么我们甚至能否在线性时间内找到一个 *2-MU*？

5 寻找简单 MUS 的 NP 完全性

现在，我们开始研究 2-CNF 的“简单”最小不可满足子集 F。考虑极小不可满足子句集的复杂度参数亏度，最简单的 F MUS 亏度为一。 [6] 证明了，判定是否存在亏度为一的 MUS，也就是 $\mathrm{MU}(F) \cap \mathcal{MU}_{\delta=1} \neq \emptyset$是否成立，是 NP 完全的；这一结论基于如下问题的 NP 完全性：判定有向图中给定的两对顶点之间是否各存在一条顶点不相交路径。现在我们可以直接证明这一点（不必像 [6]那样使用“一次读归结”），直接利用族III、IV（回顾小节 3.1）包含两条“独立”路径这一事实。这也使我们能够更准确地指出困难性的来源；稍后我们将看到，只包含一条“独立”路径的 I 类、II 类可以高效地找到。

我们首先考虑具有固定顶点 s、t 的特殊有向图（为方便起见，将变量用作顶点，但排除用于构造族III极小不可满足子集的特殊变量 x_0）：

▶**定义 5.1**。 考虑三个互不相同的变量 s、t、$x_0 \in VA$（此后固定不变）。***st*-有向图** 是一个有向图 G，满足 $V(G) \subseteq VA$、$x_0 \notin V(G)$，且 s、$t \in V(G)$，使得$(s, t) \notin E(G)$ 且 $(t, s) \notin E(G)$。

形式上，st-有向图是带部分补的有向图，但这里我们只考虑正文字（即变量），而且 st-有向图不含负文字。

▶ **定义5.2** ([6, 定理4。1])。 考虑一个 *st*-有向图 G。子句集 $\mathrm{FC}(G) \in 2\text{-}\mathcal{CLS}$ （不含单子句或空子句）定义如下：

1. 令 $t:\ V(G) \to (V(G) \setminus \{s,t\}) \cup \{x_0, \overline{x_0}\}$ 定义为

$$t(v) := \begin{cases} x_0 & \text{if } v = s \\ \overline{x_0} & \text{if } v = t \\ v & \text{otherwise} \end{cases}.$$

2. 此外，我们用 $t((a,b))$ 表示 $a, b \in V(G)$ 表示子句 $\{t(a)\ t(b)\}$。
3. 现在 $\mathrm{FC}(G)$: $= \{t(e):\ e \in E(G)\}$。

我们有 var(FC(G)) $\subseteq (V(G) \setminus \{s,t\}) \cup \{x_0\}$。弧（$s, t$）会产生单子句 $\{x_0\}$，而弧（t, s）会产生 $\{x_0\}$——我们排除了这两条弧，以免处理这些平凡情形。根据 [6, 定义4.2]进一步展开：

▶**定义5.3**。 一个***C-DPP*** 实例 是一个 *st*-有向图 G:

- G 有特殊闭游走 当且仅当存在一条路径 P_1 从 s 到 t 以及一条路径 P_2 从 t 到 s。如果存在这样的路径，满足 $V(P_1) \cap V(P_2) = \{s,t\}$，那么我们称 G 具有特殊圈。这里，“DPP”代表“不相交路径问题”，“C”代表“循环”。示例见附录A。

我们现在可以证明 $\mathrm{FC}(G)$ 将特殊圈中的两条特殊路径编码为构成 III 类极小不可满足子句集的两条蕴含链（回顾第 3.1小节）：

▶**定理 5.4**。 考虑一个 *C-DPP* 实例 G:
1. 若 s, t 中的两个顶点并非同时具有入弧和出弧，则 *FC(G)* 可满足。*2.* 每个 $F' \in$ *MU(FC(G))* 都包含变量 x_0 的四次出现（即 $vd_{F'}(x_0) = 4$）。*3.* G 有特殊闭游走，当且仅当 *FC(G)* 不可满足。*4.* G 有特殊圈，当且仅当 $MU(FC(G)) \cap \mathfrak{F}_{III} \neq \emptyset$。*5.* 若 G 没有特殊圈，则对于每个 $F' \in MU(FC(G))$，都有 $\delta(F') \geq 2$。

证明。 对于第 1 部分，以下给出四种情形下的满足赋值，使用 $V:=V(G) \setminus \{s,t\}$，并将 V 的所有元素都映射为所示的同一值，同时指出 $\mathrm{FC}(G)$ 中对于任意 $v \in V$ 的赋值会违反的缺失子句：

- 没有从 s 出发的弧: $x_0 \mapsto 1, V \mapsto 0$; 违反了 $x_0 \to v \doteq \{x_0,\ v\}$; 没有指向 s 的弧: $x_0 \mapsto 0, V \mapsto 1$; 违反了 $v \to x_0 \doteq \{x_0,\ v\}$; 没有从 t 出发的弧: $x_0 \mapsto 0, V \mapsto 0$; 违反了 $x_0 \to v \doteq \{x_0,\ v\}$; 没有指向 t的弧: $x_0 \mapsto 1, V \mapsto 1$; 违反了 $v \to x_0 \doteq \{x_0,\ v\}$。

第2部分由第1部分以及以下事实推出：全部四种（被违反的）子句类型都必须出现。对于第3部分，请回忆，任意 $F \in 2\text{-CLS}^*$ 当且仅当 idg(F) 具有一条矛盾游走时才是不可满足的（即在蕴含有向图中，从某个文字 x 到 x 并返回的一条游走）。如果 G 中存在一条特殊闭游走，那么这会立即在 idg(FC(G)) 中产生一条从 x_0 到 x_0 并返回的矛盾游走，因此 FC(G) 是不可满足的。反过来，根据第1部分，idg(FC(G)) 中的每条矛盾游走都必须包含 x_0 和 x_0，因而包含一条矛盾游走P_1，该游走经由 P_2从 x_0 到 x_0 并返回。如果 P_1 或 P_2 包含某个取反文字 v，则

$v \in V$，那么所有内部顶点都必须属于此类型，因而逆否命题 P_1 分别为P_2 仅由来自 V 的内部顶点组成，于是我们得到所要求的特殊闭游走。

第4部分直接由对族III极小不可满足子集（第 3.1 小节）的刻画得出，即它们是蕴含链 $x \to \ldots \to x \to \ldots \to x$，其中点状部分中的文字都具有作为基础的新变量（再次使用上述论证，我们总能使P_1、 P_2 只包含正文字）。最后，对于第5部分，假设 G 不具有特殊圈，并考虑 $F' \in \mathrm{MU}(\mathrm{FC}(G))$。如果 $\delta(F') < 2$，那么确实有 $\delta(F') = 1$，因此 F' 必须属于 I–IV 类之一。现在，I 类、II 类被排除，因为 F' 不具有单子句；而 IV 类被排除，因为 F' 必须包含一个度为 4 的变量（第2部分）。最后，III 类由第4部分排除，因此必有 $\delta(F') \geq 2$。◀

我们得到如下推广： [6, 推论 4。3]（判定一个 2-CNF 是否包含亏格为 1 的 MUS 是 NP 完全的）：

▶ **定理 5.5**。 考虑任意一个类 $\mathfrak{F}_{III} \subseteq \mathfrak{C} \subseteq MU_{\delta=1}$。判定 $F \in 2\text{–}CLS$ 是否具有属于 $\mathfrak{C}$ 的 *MUS*，即判定 $\mathrm{MU}(F) \cap \mathfrak{C} \neq \emptyset$ 是否成立，是 *NP* 完全的。

证明。 给定一个 C-DPP 实例 G 作为输入，我们将其转换为 $\mathrm{FC}(G)$，该过程可在多项式时间内（实际上是线性时间内）完成。现在， G 具有特殊圈，当且仅当 $\mathrm{FC}(G) \in \mathfrak{C}$，根据定理 5.4的第 4和第 5部分。该断言现由如下判定问题的 NP 完全性得出：判定一个 C-DPP 实例是否具有特殊圈（[6, 引理 4。1]）。◀

通过一些小的修改，我们可以证明，IV 类同样难以寻找：

▶ **定理 5.6**。 考虑任意类 $\mathfrak{F}_{IV} \subseteq \mathfrak{C} \subseteq MU_{\delta=1}$。判定 $F \in 2\text{–}CLS$ 是否具有属于 $\mathfrak{C}$ 的 *MUS*，即判定 $\mathrm{MU}(F) \cap \mathfrak{C} \neq \emptyset$ 是否成立，是 *NP* 完全的。

证明。 引入一个新变量 y_0，并使用修改后的转换 $\mathrm{FC}'(G)$，该转换添加新的二元子句 $x_0 \to y_0$；对于出弧 (t, v)，使用子句 $y_0 \to v$（而不是 $x_0 \to v$）；对于入弧 (v, s)，使用子句 $v \to y_0$（而不是 $v \to x_0$）。

利用定理 5.4，我们得到 FC(*G*) 的每个 MUS 都包含类型为 $x_0 \to v$、 $v \to x_0$、 $v \to x_0$、 $x_0 \to v$ 的四个子句；而现在， FC′(*G*) 的每个 MUS 都包含 $x_0 \to y_0$（从而也产生逆否弧 $y_0 \to x_0$），以及类型为 $x_0 \to v$、 $v \to y_0$、 $v \to x_0$、 $y_0 \to v$ 的四个子句。因此， x_0 正出现一次、负出现两次，而 y_0 正出现两次、负出现一次，也就是说，我们有两个度为 3 的变量。通过与定理 5.4 类似的证明，我们现在得到： G 具有特殊圈，当且仅当 MU(FC′(*G*)) $\cap \mathfrak{F}_{IV} \neq \emptyset$；如果 G没有特殊圈，那么对于每个 $F' \in \mathrm{MU}(\mathrm{FC}'(G))$，都有 $\delta(F') \geq 2$。因此，我们可以调整定理 5.5 的证明。◀

有关 $\mathrm{FC}'(G)$ 的翻译示例，请参阅第 A 节。

6 寻找最简单的极小不可满足子句集

在上一节中，我们已经看到，判断一个 2-CNF 是否存在属于族 III 或族 IV 的某个最小不可满足子集是 NP 完全问题。现在我们转向族 I（两个单位子句；请记住，2-MU 至多包含两个单位子句）和族 II（一个单位子句），并说明如何高效地找到它们。

6.1 两个单位子句

通过蕴含有向图中从 x 到 y 的正则路径（记作 $\mathfrak{R}_{x,\ y}$；回想 − 1 定义 2.2）我们可以有效地处理包含单位子句 $\{x\}$、$\{y\}$：

▶ 定理 *6.1*。考虑 $F \in 2\text{-}CLS^*$ 以及 $\{x\}$、$\{y\} \in F$、$\{x\} \neq \{y\}$。则映射 $P \in \mathfrak{R}_{x,y}(idg(F)) \mapsto F(P) := CL(E(P)) \cup \{\{x\},\ \{y\}\} \subseteq F$ 是从 $\mathfrak{R}_{x,y}(idg(F))$ 到 $MU_{\{x\},\{y\}}(F)$ 的双射。

证明。第 3.1 小节对族 I 的刻画表明，族 I 恰好处理两个单位子句的情形，并且从 x 到 y 的正则路径 P（注意，由于正则性，P 不包含单位弧）通过映射 $P \mapsto F(P)$ 恰好覆盖 $\mathrm{MU}_{\{x\},\ \{y\}}(F)$ 中的所有元素。还需证明不存在重复覆盖，因此考虑两条路径 P、$P\prime \in \mathfrak{R}_{x,\ y}(\mathrm{idg}(F))$，其中 $F(P) = F(P\prime)$，但 $P \neq P\prime$。从 P 中的第一条弧开始，我们可以通过归纳法证明 $P' = P$ 成立，但 $\mathrm{first}(P) = \mathrm{last}(P) = y \neq x$，矛盾。◀

▶ **推论 6.2**。$F \in 2\text{-}\mathcal{CLS}^*$ 有一个包含两个不同单子句的最小不可满足子集 $\{x\}$，$\{y\} \in F$ 当且仅当其蕴含有向图中存在一条正则路径 F 从 x 到 y。该条件可通过 *[10]* 中的算法在线性时间内检验（且在肯定情况下可提取出该最小不可满足子集）。此外，也可以在线性时间内找到这样一个最短的最小不可满足子集，其中“最短”可解释为使 $n(F\prime)$ $c(F')$ $\ell(F\prime)$ 最小，这几种解释彼此等价。

遍历所有单位子句对，可得：

▶ **推论 6.3**。判断 $F \in 2\text{-}CLS^*$ 是否存在一个恰好包含两个单子句的最小不可满足子集（即属于族 *I* 的最小不可满足子集），可在时间 $O(u(F)^2\ell(F))$ 内完成，因此对于输入而言可在三次时间内完成（且在肯定情况下可提取出该最小不可满足子集）。此外，也可以在相同的时间界内找到这样一个最短的最小不可满足子集。

6.2 一个单位子句

通过以 x（记作 $\mathfrak{U}_x$；回顾定义 2.3）开头的近正则路径，我们可以有效地处理包含 $\{x\}$ 的极小不可满足子句集（不过需要考虑重复项）

这里我们使用从 x 到 x 的路径 P（始终指给定蕴含有向图中的路径）的以下简单性质：首先注意到 $\mathrm{lgth}(P) \geq 1$；$\mathrm{lgth}(P) = 1$ 当且仅当 P 仅由一个单位组成，对应于单位子句 x。现在有一个重要而显然的观察：逆否命题 P 也是一条从 x 到 x 的路径，其中 $P = P$ 当且仅当 $\mathrm{lgth}(P) = 1$。

▶ **定理 6.4**。考虑 $F \in 2\text{-}CLS^*$ 和 $\{x\} \in F$。定义映射 $P \in \mathfrak{U}_x(\mathrm{idg}(F)) \mapsto F(P) := CL(E(P)) \cup \{\{x\}\} \subseteq F$。

1. 映射 $P \mapsto F(P)$ 是从 $\mathfrak{U}_x$ 到 $MU\{x\}(F)$ 的满射。*2.* 回想一下，原像 $F^{-1}(F')$ 是所有满足 $F(P) = F'$ 的 $P \in \mathfrak{U}_x$ 的集合；这里我们只称之为“原像”。*3.* $F' \in MU\{x\}(F)$ 的原像唯一，当且仅当 F' 恰好包含两个单子句（即属于族 *I*）。这些 F' 恰好是那些 $P \in \mathfrak{U}_x$ 对应的 $F(P)$，其中 P 中的最后一条弧是单位（因而对应第二个单子句 $\{y\}$；如果移除这条最后的弧，那么我们恰好处于定理 *6.1* 的情形，从而得到 $R_{x,y}$ 中的一个元素）。*4.* 对于所有恰好包含一个单子句（即 $\{x\} \in F'$）的 $F' \in MU\{x\}(F)$，恰有两个原像：*a.* 考虑某个满足 $F(P) = F'$ 的 $P \in \mathfrak{U}_x$。

b. 如果 $last(P) \equiv x$（注意 $lgth(P) \geq 2$），那么 F' 属于 *IIa* 类，并且有 $P \neq P$，满足 $P \in \mathfrak{U}_x$ 和 $F(P) = F(P)$，而 $\overline{F'}$ 的原像是 $\{P, P\}$。*c.* 否则，F' 属于 *IIb* 族。考虑正则分解 P_0；$P_1 = P$（定义 *2.3*）。令 $P' := P_0$；P_1。我们有 $P' \neq P$，且 F' 的原像为 $\{P, P'\}$。

证明。第 3.1 小节中对 I 类、II 类的刻画表明，包含 $\{x\}$ 的极小不可满足子句集恰好由近正则路径 $P \in \mathfrak{U}_x(\mathrm{idg}(F))$ 覆盖，该路径通过映射$P \mapsto F(P)$得到；这说明了第1部分。接下来的断言，即第3部分，由定理 6.1（以及第4部分中的断言，即那里总是至少有两个不同的原像）得出。剩下要证明第4部分。对于情形 4b，依照定理 6.1 中相同的论证可知，唯一的另一个原像是 P。最后，对于情形 4c，P_0 的第一条弧是唯一的，因为 x 不可能位于该路径上（否则就构成了 IIa 类，其同构类型不同）。由此通过归纳法可知，P_0 的其他弧也是唯一的。而 P_1 的唯一替代项是 P_1，这一点同样可由定理 6.1 中的论证得出。◀

▶ **推论 6.5.** $F \in 2\text{-}CLS^*$ 存在一个包含单子句 $\{x\} \in F$ 的最小不可满足子集，当且仅当 F 的蕴含有向图中存在一条从 x 到 x的路径。该条件可在线性时间内检验（若条件成立，还可提取出该最小不可满足子集）。

证明。 如果存在 $F' \in$ 极小不可满足子句集$_{\{x\}}(F)$，那么存在一条路径 $P \in \mathfrak{U}_x$，满足 $F(P) = F'$。如果 $\mathrm{last}(P) \neq x$，那么 P 是从 x 到 y 的一条路径与从 y到 y的一条路径的拼接：该拼接 P；P_0 是从 x到 x的一条路径。反过来，如果存在一条从 x 到 x的路径，取到首次发生冲突为止的初始子路径，即可得到 $\mathfrak{U}_x$的一个元素。◀

只需搜索一条从 x到 y的正则路径（因此 $\mathrm{var}(y) \neq \mathrm{var}(x)$），即可找到族 IIa 中的一个（确切）元素；对这样的 y，满足 $(y, x) \in E(\overline{\mathrm{idg}}(F))$（即$\{y, x\} \in F$）。

▶**问题 6.6**。 我们能否利用 $\{x\}$高效地找到一个最短极小不可满足子句集？与推论*6.2*不同，由于路径集合 $\mathfrak{U}_x$ 的开放性，这里的问题似乎很困难。

如果在推论 6.5 中，我们想测试是否存在恰好包含一个单子句的最小不可满足子集（即 $\{x\}$），也就是说，测试是否存在族 II 中的元素，那么我们可以通过从 F 中移除所有其他单子句来进行预处理。另一方面，遍历所有单子句可得：

▶ **推论 6.7**。 是否存在包含至少一个单子句的 $F \in 2\text{-}CLS^*$ 极小不可满足子句集（即属于族 *I* 或族 *II* 的极小不可满足子句集），可以在 $O(u(F)\ell(F)$ 时间内判定，因此可在输入规模的二次时间内完成（若答案为是，还可提取该极小不可满足子句集）。同样的方法也适用于判定（并找出）恰好包含一个单子句的极小不可满足子句集（族 *II*）。

7 含一个单位子句的极小不可满足子句集的枚举

根据定理 6.4，我们可以高效地枚举至少包含一个单位子句的极小不可满足子句集。主要过程如下：

$$\mathrm{enum}(F \in 2\text{-}\mathcal{CLS}^*, \{x\} \in F, L \in \mathcal{LO}(\mathrm{lit}(F))),$$

对于集合 X，记 $\mathcal{LO}(X)$ 为 X 的线性序集合。文字集 F 上的序 $L \in \mathcal{LO}(\mathrm{lit}(F))$ 首先为 $\mathrm{idg}(F)$ 的路径建立一个线性序，然后为$\bigcup_{\{x\} \in F} \mathrm{MU}_{\{x\}}(F)$ 建立一个线性序；后者是 F 中所有至少包含一个单位子句的极小不可满足子句集的集合，具体如下：

▶ **定义 7.1.** 考虑 $F \in 2\text{-}\mathcal{CLS}^*$ 和 $L \in \mathcal{LO}(\mathrm{lit}(F))$。

1. 对于路径 P, P'，我们使用基于 L 的字典序，也就是说，$P <_L P'$ 若将路径视为 *lit(F)* 上的词，对于从左起第一个 i 使得 P 与 P' 不同的索引，我们有 P_i 按照 L 排在 P'_i 之前。这在所有路径上定义了一个严格线性序。*2.* 对于 $F \in \bigcup_{\{x\}\in F}$ 极小不可满足子句集$_{\{x\}}(F)$，至多有两个（不同的）文字 x, x' 满足 $F \in$ 极小不可满足子句集$_{\{x\}}(F) \cap$ 极小不可满足子句集$_{\{x'\}}(F)$，于是我们令 $x(F)$ 为按照 L 较小的那个。*3.* 此外，根据定理 *6.4*，原像中至多有两条路径 $P, P' \in \mathfrak{U}_{x(F)}$，于是我们令 $P(F)$ 为按照 $<_L$ 较小的那条。*4.* 最后，对于 $F, F' \in \bigcup_{\{x\}\in F}$ 极小不可满足子句集$_{\{x\}}(F)$，我们令 $F <_L F'$ 当且仅当 $P(F) <_L P(F')$。我们将路径及含有单子句的极小不可满足子句集上的该序称为 *L-pathlex* 序。

现在，enum(F，x，L)-过程通过对 idg(F) 进行递归重复（并带守卫）的深度优先搜索遍历（实际输出发生在深度优先搜索过程中），按 L-pathlex 顺序枚举 MU$_{\{x\}}(F)$：

算法 1 按 L-pathlex 顺序枚举 MU$_{\{x\}}(F)$

```
1: 过程 ENUM(F ∈ 2-CLS*, {x} ∈ F, L ∈ LO(lit(F))        ▷ 全局变量： x, L
2:     G ← idg(F)                                        ▷全局变量： G
3:     if x --G--> x̄ 然后
4:         深度优先搜索((∅, ∅) x)
```

注意，若要仅枚举不使用除 $\{x\}$ 以外任何单子句的极小不可满足子句集，只需从 F 中删除所有其他单子句。现在，实际执行工作的递归过程如下（回忆定理 $\mathfrak{U}_x$ 中的 $\mathfrak{U}_x$ 定理 6.4）：

算法 2 用于按 L-路径字典序枚举 $P \in \mathfrak{U}_x(G)$ 并传递给 OUTPUT 的深度优先搜索过程

```
1: 过程 深度优先搜索(P, y)
2:     P ← P; y
3:     V ← V（P）                                        ▷ 只是一个缩写
4:     如果 V 包含冲突 则
5:         输出(P)
6:     else
7:         R ← 的出邻居集合 z 属于 y，在 G − V中，且带有 z --G−V--> x̄。
8:         ▷ R 的计算可以通过从顶点 x 开始进行一次广度优先搜索来完成 ◁
           在反向有向图 （G − V（P））ᵗ。
9:         对于每个顶点 z ∈ R，按顺序 L 执行
10:            深度优先搜索(z, P)
```

通过直接检查代码，我们可以证明：

▶ **引理 7.2**。对过程的所有调用 深度优先搜索(P, y) (在算法 *2* 中给出），即算法 *1* 中的初始调用，第 *4* 行，以及算法 *2* 第 *10* 行中的递归调用，都满足关于两个参数 P, y 的以下不变式：

1. P 是 G的无冲突子有向图，它要么为空，要么是一条路径。**2.** $y \in V(G) \setminus V(P)$，满足：如果 P不为空：$(\mathrm{last}(P)\ y) \in E(G)$; $y \xrightarrow{G-V(P)} x$。

因此对于 $P' := P; y$（对应于第 *2.2* 行）存在 $P'' \in \mathfrak{U}_x(G)$ 扩展 P'（可能为 $P'' = P'$）。

最后是实际输出：

算法3 用于路径 $P \in \mathfrak{U}_x$ 的输出过程

```
1: 过程 输出(P)
2:     如果 P 的最后一条弧是一个单位 则
3:         打印(P)                                          ▷族 I
4:     else
5:         令 P = P_0；P_1 为 P 的正则分解。
6:         P' ← P_0；P_1
7:         如果 P <_L P'，那么
8:             打印(P)                                      ▷ 族 II
1: 过程 打印(P)
2:     打印 CNF 头部： n = lgth(P)， c = lgth(P) + 1
3:     打印 {x}
4:     对于 (y, z) ∈ E(P) 执行
5:         如果 (y, z) 是单位 则
6:             打印 {z}                                     ▷ 仅适用于族 I
7:         else
8:             打印 {ȳ, z}
```

▶ **定理 7.3.** 算法 *1*，实现 枚举$(F, \{x\}\ L)$，确实枚举了集合极小不可满足子句集$_{\{x\}}(F)$ 按 L-路径字典序。

证明。首先需要证明，在深度优先搜索算法 2 中，对于调用 output(P)（第 5 行），路径 P 恰好是 $\mathfrak{U}_x$ 的元素，并且它们按 L-路径字典序排列。P 始终是 $\mathfrak{U}_x$ 的一个元素，这由引理 7.2 可知。我们不会遗漏任何路径，这一点可通过归纳法轻易证明：从 x （第 1.4 行）开始，然后始终考虑所有出邻居 z （第 2.7 行），仅使用能够到达 x（不经过已 _ 访问顶点）这一必要条件来剪除分支。最后，由于按照顺序 L 遍历所有出邻居（第 2.9 行），我们遵循 L-路径字典序。

剩下需要证明的是，通过 P 得到的极小不可满足子句集的输出，由 输出和 打印在算法 3中的输出是正确的；根据定理 6.4的详细陈述，这一点可立即得出。◀

对于底层路径的枚举，我们可以实现多项式延迟；但对于极小不可满足子句集的枚举，只能实现增量多项式时间，因为可能会连续出现很长一段“静默路径”（即 $P >_L P'$中测试的否定情形输出第 3.7 行）：

▶ 定理 *7.4*。算法 *1* 按照 L-路径字典序枚举 $P \in \mathfrak{U}_x$*(idg(F))*，并将其输出到第 *2.5* 行的 *output(P)* 中；找到第一个元素以及此后每个元素均需 $O(n(F)\ell(F))$ 时间（二次时间），而最终确定不存在更多元素则需 $O(\ell(F))$ 时间。空间使用为 $O(\ell(F))$（注意，路径不会被存储）。

对于按 L-路径字典序进行的 $F' \in$极小不可满足子句集$_{\{x\}}(F)$ 枚举，每个 F'都有一条或两条对应的路径 P，其输出为 F'；其中只打印第一条路径，忽略第二条路径。因此，已找到 $m \in \mathbb{N}_0$ 个最小不可满足子集 F' 时，找到下一个 F'（或确定不存在下一个）所需时间为 $O((m+1)\cdot n(F)\ell(F))$。

证明。 首先我们考虑路径的枚举 $P \in \mathfrak{U}_x(\mathrm{idg}(F))$。根据引理 7.2 我们可知每次（递归）调用 深度优先搜索 至少产生一条路径。由于 深度优先搜索 的内部运行时间为 $O(\ell(F))$，且一条路径的最大深度为 $2n(F)$，我们得到运行时间

$O(n(F)\ell(F))$ 用于寻找下一条路径，而最后阶段（判定不再存在进一步的路径）要么只是回溯到根节点，要么只是第 1.3 行中的初始测试，这只需线性时间。空间使用与普通深度优先搜索相同（当不复制路径 P，而是通过全局栈实现时）。

输出下一个最小不可满足子集的最坏延迟，是指对于所有 m 个先前输出，我们都先遇到它们的（静默）兄弟节点，因此搜索下一条路径至多会被调用 $m+1$ 次。◀

▶ **问题 7.5. </style}能否以多项式延迟枚举极小不可满足子句集**（F）</style? 一旦知道其中一条底层路径，我们也就知道其兄弟节点′，但这是"事后"才知道的，而且似乎很难利用这一点来进行打印，从而避免长串的"静默路径"。

▶**问题 7.6.** 我们采用最简单的回溯算法的一种改编，称为 *T-DFS*在 *[19]*；更复杂的算法似乎不会改变渐近界，但可能对实际应用很重要。

最后，通过遍历所有单子句，我们可以枚举所有包含某个单子句的极小不可满足子句集：

算法 4 按 $\bigcup_{\{x\}\in F}$ 极小不可满足子句集$_{\{x\}}(F)$ 的 L-路径字典序进行枚举

```
过程 枚举(F ∈ 2–CLS*, L ∈ LO(lit(F)))
对于 {x} ∈ F，按顺序 L 执行
    枚举 (F,{x} L)
    F ← F |{{x}}
```

▶ **定理 7.7。** 算法 *4* 按 $\bigcup_{\{x\}\in F}$ 极小不可满足子句集$_{\{x\}}(F)$ 的 L-路径字典序进行枚举。

证明。对于不同的单子句 $\{x\}$、$\{y\}$，$\in F$我们有 $\mathrm{MU}_{\{x\},\{y\}}(F) = \mathrm{MU}_{\{y\},\{x\}}(F) \subseteq \mathrm{MU}_{\{x\}}(F) \cap \mathrm{MU}_{\{y\}}(F)$，因此我们需要避免使用 $\{x\}$ 和 $\{y\}$ 两次输出极小不可满足子句集。根据定义 7.1 第 2 项，我们必须根据 L 选取 x、y 中较小者，这发生在第 4.2 行，同时在第 4.4 行将其消去。现在由定理 7.3 可得结论。◀

▶ 定理 *7.8*。算法 *4* 按 *L-pathlex* 顺序枚举 $P \in \bigcup_{\{x\}\in F} \mathfrak{U}_x(idg(F))$，并按照第 *2.5* 行的规定输出到 *output(P)*，找到第一个元素以及之后每个元素所需的时间为 $O(n(F)\ell(F))$；最终确定不存在更多元素所需的时间为 $O(u(F)\ell(F))$。空间使用为 $O(\ell(F))$。

对于按 *L-pathlex* 顺序对 $F' \in \bigcup_{\{x\}\in F}$ 极小不可满足子句集$_{\{x\}}(F)$ 进行的枚举，假设我们目前已经找到 $m \in \mathbb{N}_0$ 个极小不可满足子句集，那么找到下一个 F'（或确定不存在下一个）可在时间 $O((m+1)\cdot n(F)\ell(F))$内完成。

证明。 结论由定理 7.4推出，同时注意到对于给定的单位子句$\{x\}$不可满足性只需线性时间。◀

8 总结与展望

我们首先展示了如何在线性时间内进行 2-MU 判定（第 4 节），其依据是奇异 DP 约简（sDP）；问题 4.3 询问 sDP 的一般复杂度如何。随后，我们开始研究 2-CNF 的 MUS。问题 4.6 询问能否在线性时间内找到某个 MUS。接着，在第 5 节中，我们转向困难性结果，证明判定是否存在 III 类或 IV 类最小不可满足子公式是 NP 完全的。I 类和 II 类

MUS（分别含有两个和一个单位子句）则可以高效地找到，如第 6 节所示。问题6.6 探讨寻找最短 II 类最小不可满足子集的问题（而 I 类最小不可满足子集可以高效找到）。最后，我们证明 I 类和 II 类可以一起在增量多项式时间内枚举；问题 7.5询问能否实现多项式延迟，而问题 7.6 则简要讨论更实用的算法。

一个有趣的开放研究方向是，建立一幅更完整的**复杂度图谱**，用于寻找 2-CNF 的 MUS。关于进一步高效枚举极小不可满足子句集的问题，同样也需要更完整的映射。

▶**猜想8.1**。 具有给定斜对称性的有向图中的正则路径可以通过多项式延迟进行枚举。

如果这一猜想成立，那么我们就可以通过多项式延迟枚举所有包含两个给定单子句的极小不可满足子句集。

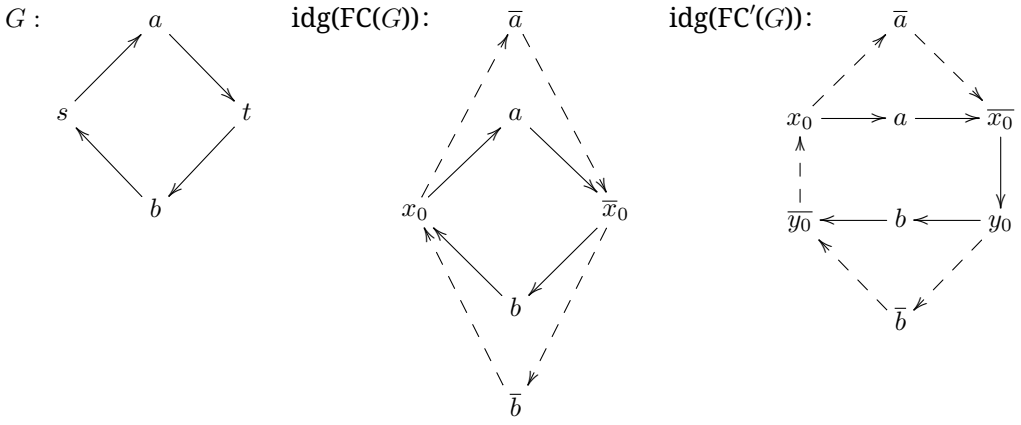
参考文献

1 Hoda Abbasizanjani 和 Oliver Kullmann。极小不可满足性与极小强连通有向图。载于 Olaf Beyersdorff 和 Christoph M. Wintersteiger 编，《可满足性测试理论与应用——SAT 2018》，《计算机科学讲义》第10929卷，第329–345页。施普林格，2018年。 doi:10.1007/978-3-319-94144-8_20。2 Hoda Abbasizanjani 和 Oliver Kullmann。极小不可满足2-CNF公式的分类。技术报告 arXiv:2003.03639v2 [cs.DM]，arXiv，2020年5月。网址： http://arxiv.org/abs/2003.03639。3 M. Fareed Arif、Carlos Mencía 和 Joao Marques-Silva。利用 Horn公式高效枚举最小不可满足子集及其在公理定位中的应用。载于 Marijn H. Heule 和 Sean Weaver 编，《可满足性测试理论与应用——SAT 2015》，《计算机科学讲义》第9340卷，第324–342页。施普林格，2015年。 doi:10.1007/978-3-319-24318-4_24。4 Jaroslav Bendík 和 Ivana Černá。MUST：最小不可满足子集枚举工具。载于 Armin Biere 和 David Parker 编，《系统构造与分析工具及算法》（TACAS 2020），LNCS第12078卷，第135–152页，2020年。 doi:10.1007/978-3-030-45190-5_8。5 Jaroslav Bendík 和 Kuldeep Meel。最小不可满足子集计数。载于 Alexandra Silva 和 K. Rustan M. Leino 编，《计算机辅助验证》（CAV 2021），《计算机科学讲义》（LNCS）第12760卷，第313–336页，2021年7月。 doi:10.1007/978-3-030-81688-9_15。6 Hans Kleine Büning、Piotr Wojciechowski 和 K. Subramani。在2CNF子句系统中寻找只读一次归结反驳。《理论计算机科学》，729：42–56，2018年6月。doi:10.1016/j.tcs.2018.02.002。7 Gennady Davydov、Inna Davydova 和 Hans Kleine Büning。针对CNF子类的最小不可满足性问题的高效算法。《数学与人工智能年刊》，23(3-4)：229–245，1998年。 doi:10.1023/A:1018924526592。8 Herbert Fleischner、Oliver Kullmann 和 Stefan Szeider。多项式时间识别具有固定子句—变量差值的最小不可满足公式。《理论计算机科学》，289(1)：503–516，2002年11月。 doi:10.1016/S0304-3975(01)00337-1。9 Emilio Gamba、Bart Bogaerts 和 Tias Guns。通过不可满足子集优化高效解释约束满足问题。《人工智能研究杂志》，78：709–746，2023年。网址： https://arxiv.org/abs/2105.11763, doi:10.1613/JAIR.1.14260。10 Andrew V. Goldberg 和 Alexander V. Karzanov。斜对称图中的路径问题。《组合数学》，16：353–382，1996年。 doi:10.1007/BF01261321。11 Hans Kleine Büning 和 Oliver Kullmann。最小不可满足性与自足赋值。载于 Armin Biere、Marijn Heule、Hans van Maaren 和 Toby Walsh 编，《可满足性手册》，

《人工智能与应用前沿》第336卷，第14章，第571–633页。IOS出版社，第二版，2021年。在线版本。 doi:10.3233/FAIA200997。12 Hans Kleine Büning 和 Xishun Zhao。最小不可满足性：研究成果与开放问题。技术报告 tr-ri-02-230，《计算机科学系列》，帕德博恩大学，帕德博恩大学数学与计算机科学系，2002年。 http://wwwcs.uni-paderborn.de/cs/ag-klbue/de/research/MinUnsat/index.html。13 Oliver Kullmann 和 Xishun Zhao。关于极小不可满足子句集的 Davis-Putnam 归约。《理论计算机科学》，492：70–87，2013年6月。 doi:10.1016/j.tcs.2013.04.020。14 Paolo Liberatore。逻辑冗余 II：2CNF与Horn命题公式。《人工智能》，172(2-3)：265–299，2008年2月。 doi:10.1016/j.artint.2007.06.003。15 Mark H. Liffiton、Alessandro Previti、Ammar Malik 和 Joao Marques-Silva。快速、灵活的最小不可满足子集枚举。《约束》，21(2)：223–250，2016年4月。 doi:10.1007/s10601-015-9183-0。16 Mark H. Liffiton 和 Karem A. Sakallah。关于寻找所有极小不可满足子公式。载于 Fahiem Bacchus 和 Toby Walsh 编，《2005年可满足性测试理论与应用》，《计算机科学讲义》第3569卷，第173–186页，柏林，2005年。施普林格。 doi:10.1007/11499107_13。17 Yoonna Oh、Maher N. Mneimneh、Zaher S. Andraus、Karem A. Sakallah 和 Igor L. Markov。AMUSE：极小不可满足子公式提取器。载于第41届设计自动化年会（DAC）论文集，第518–523页，2004年。 doi:10.1145/996566.996710。18 Rafael Peñaloza 和 Barış Sertkaya。EL 系列描述逻辑中公理定位的复杂度。载于第十二届知识表示与推理原理国际会议论文集，KR'10，第280–289页。AAAI出版社，2010年。网址：http://dl.acm.org/citation.cfm?id=3031748.3031785。19 You Peng、Ying Zhang、Xue min Lin、Wenjie Zhang、Lu Qin 和 Jingren Zhou。跳数约束的 s-t 简单路径枚举：致力于弥合理论与实践。《VLDB基金会会刊》，13(4)：463–476，2019年12月。 doi:10.14778/3372716.3372720。

A 圈问题翻译示例

一个带有特殊圈的 st-有向图简单示例 G （回顾定义5.3），以及它的翻译FC(G) （定义5.2），再加上定理FC$'(G)$ 证明中的变体5.6；对应于给定子句的弧用实线绘制，而其逆否命题用虚线绘制：

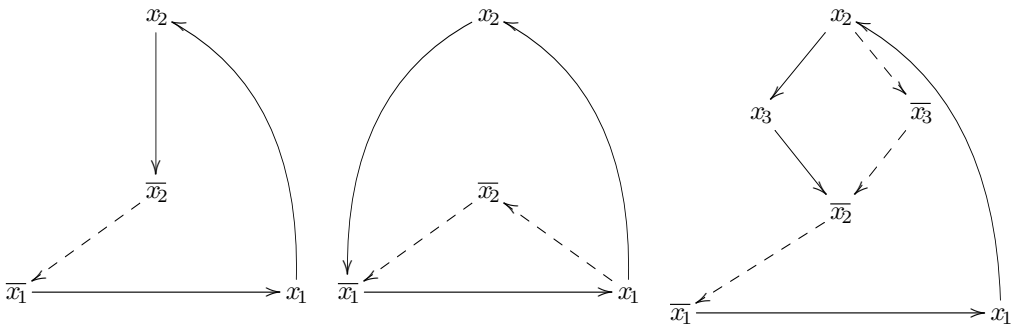


FC(G): $x_0 \to a$, $a \to \overline{x}_0$, $\overline{x}_0 \to b$, $b \to x_0$; $n = 3$, $c = 4$FC$'(G)$: $x_0 \to a$, $a \to x_0$, $x_0 \to y_0$, $y_0 \to b$, $b \to y_0$; ~~$n = 4$~~, $c = 5$.

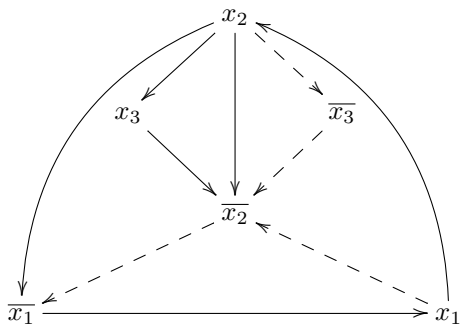
B 最小不可满足子集枚举算法示例

我们在三个简单 MUS 的并集上演示算法 1 ：
1. $\mathrm{U}_2^2 = \{\{x_1\}, \{x_1, \overline{x_2}\}, \{x_2\}\}$ 2. $\mathrm{U}_{12,1} = \{\{x_1\}, \{x_1, x_2\}, \{x_1, \overline{x_2}\}\}$ 3. $\mathrm{U}_{13,2} = \{\{x_1\}, \{\overline{x_1}, \overline{x_2}\}, \{x_2, x_3\}, \{\overline{x_2}, \overline{x_3}\}\}$.

它们的蕴含有向图为



它们分别属于族 I、IIa、IIb。它们并集的蕴含有向图是这些有向图的并集：



运行算法 1 处理 $x = x_1$ 和 $L : (x_1 < x_1 < x_2 < \overline{x_2} < x_3 < \overline{x_3})$ 会生成表 1所示的执行轨迹。该表仅列出算法 2 和 3中的非平凡事件。步骤 5、8 和 12 产生了三个不同的极小不可满足子句集，而步骤 16 和 20 是字典序更大的兄弟路径，行 3.7 中的守卫会抑制这些路径的输出。列 y 和 R分别记录当前的深度优先搜索文字（或者在冲突和输出时，记录 P 上的最后一个文字）以及在行 2.7处计算的值。

XX：18 个 2-CNF 的极小不可满足子句集

表 1 对 F、$\{x_1\}$ L 执行枚举的执行轨迹，其中 $F = U_2^2 \cup U_{2,1}^1 \cup U_{3,2}^1$。 P记录文字序列。

事件	算法参考
初始化调用	算法 1，第 4 行调用算法 2
深度优先搜索(P, y)	算法 2，第 7–10 行：扩展 P，计算 R，对 y 递归
冲突	算法 2，第 5 行：$V(P)$ 包含互补文字
输出	算法 3，族 I/IIa/IIb 类分支（包括第 3.7 行的守卫）
输出（静默）	算法 3，守卫行 7 对字典序兄弟节点失败

步骤	事件	y	R	P	注释
1	初始化调用	x_1	$(x_2, \overline{x_2})$	(x_1)	
2	深度优先搜索(P, y)	x_2	$(\overline{x_1}, \overline{x_2}, x_3, \overline{x_3})$	(x_1, x_2)	
3	深度优先搜索(P, y)	$\overline{x_1}$	–	$(x_1, x_2, \overline{x_1})$	
4	冲突	$\overline{x_1}$	–	$(x_1, x_2, \overline{x_1})$	
5	输出	$\overline{x_1}$	–	$(x_1, x_2, \overline{x_1})$	IIa 类
6	深度优先搜索(P, y)	$\overline{x_2}$	–	$(x_1, x_2, \overline{x_2})$	
7	冲突	$\overline{x_2}$	–	$(x_1, x_2, \overline{x_2})$	
8	输出	$\overline{x_2}$	–	$(x_1, x_2, \overline{x_2})$	族 I
9	深度优先搜索(P, y)	x_3	$(\overline{x_2})$	(x_1, x_2, x_3)	
10	深度优先搜索(P, y)	$\overline{x_2}$	–	$(x_1, x_2, x_3, \overline{x_2})$	
11	冲突	$\overline{x_2}$	–	$(x_1, x_2, x_3, \overline{x_2})$	
12	输出	$\overline{x_2}$	–	$(x_1, x_2, \overline{x_3}, \overline{x_2})$	IIb 族
13	深度优先搜索(P, y)	$\overline{x_3}$	$(\overline{x_2})$	(x_1, x_2, x_3)	
14	深度优先搜索(P, y)	$\overline{x_2}$	–	$(x_1, x_2, x_3, \overline{x_2})$	
15	冲突	$\overline{x_2}$	–	$(x_1, x_2, x_3, \overline{x_2})$	
16	输出（静默）	x_2	–	$(x_1, x_2, \overline{x_3}, \overline{x_2})$	步骤 12 的兄弟节点；守卫抑制打印
17	深度优先搜索(P, y)	$\overline{x_2}$	(x_1)	$(x_1, \overline{x_2})$	
18	深度优先搜索(P, y)	$\overline{x_1}$	–	$(x_1, \overline{x_2}, \overline{x_1})$	
19	冲突	$\overline{x_1}$	–	$(x_1, \overline{x_2}, \overline{x_1})$	
20	输出（静默）	x_1	–	$(x_1, \overline{x_2}, \overline{x_1})$	步骤5的同级项；守卫会抑制打印

