arXiv:2303.06643v1 [cs.AI] 12 Mar 2023

使用 QBF 求解器进行一般布尔公式最小化*

Eduardo Cal`o[1][0000−0003−3881−8994] 和 Jordi Levy[2][0000−0001−5883−5746]

[1] 乌得勒支大学，荷兰
e.calo@uu.nlhttps://www.uu.nl/staff/ECalo[2]
IIIA，CSIC，西班牙
levy@iiia.csic.eshttps://www.iiia.csic.es/~levy

摘要。命题公式的最小化是逻辑中的经典问题，其首批算法至少可追溯到20 世纪 50 年代 Quine 和 Karnaugh 的研究。该领域以往的大多数研究都集中于求得合取范式（CNF）或析取范式（DNF）中的最小或近似最小公式，并将其应用于硬件设计。本文关注的是如何以任意格式求得一个等价公式，同时也允许使用原始公式中未出现的联结词。我们的主要动机是将最小化算法用于生成原始公式的自然语言翻译，因为使用更短的等价公式作为输入可能会带来更好的翻译。最近，Buchfuhrer 和 Umans 证明了该问题的判定版本是Σ_2^p-完全的。

我们分析了解决该问题的三种可能的（实用）方法。第一种是采用暴力搜索，按公式规模从小到大的顺序生成所有可能的公式，并通过测试所有可能的变量赋值，检查它们是否与原始公式等价。第二种是生成所有公式的 Tseitin 编码，并使用 SAT 求解器检查其与原始公式的等价性。第三种是将问题编码为量化布尔公式（QBF），并使用 QBF 求解器。结果表明，QBF 方法的表现远胜于另外两种方法。

关键词： SAT 求解器 · QBF 求解器 · 布尔公式最小化 · 自然语言处理

1 引言

复杂布尔表达式的最小化是逻辑学中一个由来已久的问题。最早的算法于20世纪50年代提出，例如 Quine 的研究，

*本项目获得了欧盟地平线 2020 研究与创新计划资助，该计划由玛丽·斯克沃多夫斯卡-居里资助协议（编号 860621）提供支持；同时还获得了 MICINN 项目 PROOFS（PID2019-109137GB-C21）的资助。

McCluskey [14,15,12], 和 Karnaugh [11]为随后数年的扩展与优化铺平了道路（例如 Petrick方法 [13], 和 Espresso启发式逻辑最小化器 [2],即）。这些研究侧重于在特定规范形式中获得最小的等价表示（例如合取范式（CNF）或析取范式（DNF）），并将研究范围限制在有限的一组联结词内。本文关注的是通用布尔公式最小化，不对输入公式或输出的形式作任何假设。事实上，我们的最小化方法允许输入和输出采用不同的联结词集合。

我们将布尔公式最小化（即，为给定公式寻找逻辑上等价且最短的公式）表述为量化布尔公式（QBF）可满足性问题，并设计了一种能够稳定地为给定公式找到最短等价形式的算法。我们将该算法与暴力基线以及一种基于可满足性问题（SAT）的方法进行比较。

动机。我们的工作有两个截然不同、相距甚远的动机。第一个动机与 [19]，中作者提出的 qbf2epr 工具有关，该工具可将量化布尔公式（QBF）转换为有效命题逻辑（EPR）中的公式。他们旨在为 EPR 生成基准测试，并比较 QBF 和 EPR 求解器。类似地，我们将公式最小化问题编码为 QBF，以生成供 QBF 求解器使用的基准测试，并比较 SAT 与 QBF 技术。自动演绎领域分为多个子领域（例如 SAT、QBF、SMT、最大可满足性（MaxSAT）和 EPR），分别致力于解决不同类别的问题：从 NP完全的 SAT，到 NEXPTIME完全的 EPR，再到 PSPACE完全的 QBF。每个子领域都有自己的竞赛和基准测试集。然而，许多在某一领域行之有效的思想（如 SAT 中的学习）已被引入其他领域。从这个意义上说，像我们这样可用两种不同技术解决的问题，有助于比较各领域所达到的成熟度。

我们的第二个动机与自然语言处理中的一种算法最小化应用场景有关。理解逻辑形式体系的含义是许多学者的一项重要任务，但有时即使是经验丰富的逻辑学家，也可能难以解读复杂的公式。自然语言生成 [18,8], ，尤其是逻辑到文本生成方法 [17,4]„ 可用于简化逻辑公式，并将其转换为自然语言（NL）（如英语、普通话或韩语）中易于理解的文本，从而有效地向系统用户解释公式。例如，给定以下一阶公式：

$$\exists x(Problem(x) \wedge \forall y(Researcher(y) \rightarrow Interested(y, x)))$$

我们希望有一个系统，能够通过以下文本（或其他语义等价的文本）自动生成忠实且易于理解的解释：

这是每位研究者都会感兴趣的问题。

公式应具备哪些特征，才能成为逻辑到文本翻译系统的合适输入？一个值得考察的方面是长度。自 [10]以来，简洁性一直是语言学辩论的焦点。可以说，相较于较长的表达，人们应优先选择较短的表达，并避免不必要的冗长。这一原则或许也适用于逻辑公式。直观而言，与较长的等价逻辑公式相比，较短的公式更适合翻译成自然语言。本文专门探讨这一问题的逻辑层面。我们聚焦于命题逻辑——一种等价性可判定的形式系统，并将考察范围限定在公式的长度上，[3]只寻求给定公式最短的等价公式。

相关工作：公式最小化。 布尔公式最小化是多项式时间层级第二层中的一个自然优化问题 Σ_2^p 。事实上， [7] 利用该问题来引出多项式层级的定义。其判定版本可以表述为以下问题：给定一个布尔公式，证明存在一个（更小的）公式（由相同变量集合构成），使其在所有可能的变量赋值下都与给定公式具有相同的评测结果。输入中对两组量化变量以及公式求值时间的限制，证明了该问题属于 Σ_2^p 。正如我们将在第 2 节看到的，这对应于我们的暴力搜索算法。通常假设给定公式及其最小化结果都是形式相同的电路或公式。然而，在我们的实现中，我们允许使用不同的联结词集合。除了针对某些特定输入和输出形式的完备性证明外，一般形式的证明一直未能由研究人员解决，直到 [3] 证明了该问题是 Σ_2^p -完备的。

复杂布尔表达式的优化已在电子电路领域得到广泛研究。在这一领域，实际因素（即复杂电路占用物理空间，且实现时需要更多资源）使得寻找最优电路表示至关重要。著名的最小化方法包括奎因–麦克拉斯基算法 [14,15,12] 和卡诺图 [11]。在卡诺图中，真值表中的布尔结果会被转移到二维网格上，每个单元格的位置代表一种输入条件组合，而每个单元格的值则是相应的输出值。通过识别最优的 0 和 1 分组，可以得到规范形式中的各项，并据此写出最简表达式。奎因–麦克拉斯基算法会找出函数的所有主蕴含项，并利用图表找出：(i) 函数的本质主蕴含项；(ii) 覆盖该函数所必需的其他主蕴含项。该方法在功能上与卡诺图完全相同，但其表格形式使其更适合在计算机系统中使用。

然而，尽管相关研究由来已久，也有许多尝试扩展成熟的方法（例如， [22] 尝试在 Quine-

[3] 我们将长度定义为公式中包含的符号数量（即谓词和联结词，不包括括号）。

McCluskey 算法中实现异或运算符），大多数工作仍集中于有限的一组联结词和规范形式（例如合取范式或析取范式）。就我们的研究范围而言，我们需要一种更通用的方法，能够按需处理所有联结词。

量化布尔公式。量化布尔公式（QBF）是命题逻辑的一种扩展，允许使用全称量化和存在量化 [1]。量词的使用使其表达能力强于经典命题逻辑。如果 QBF ϕ 中出现的所有变量都已绑定，那么ϕ 称为闭合的。QBF 通常采用规范的前束合取范式（PCNF） $\phi = \exists\vec{x}\forall\vec{y}\exists\vec{z}\cdots\psi$，其中仅包含量词和绑定变量的部分称为前缀，其后是 ψ ，它是一个不含量词的布尔公式，由对称为矩阵的子句进行合取构成。

QBF 可满足性问题 [9] 是指：对于给定的 QBFϕ，确定是否存在一个针对自由变量的赋值，使得 ϕ 在该赋值下取值为真。因此， ϕ 为真，当且仅当存在一个针对 $\vec{x}$的真值赋值，使得对于 $\vec{y}$的所有真值赋值，都存在一个针对$\vec{z}$的真值赋值，.....使得 ψ为真。随着时间推移，人们开发了多种 QBF 求解器[4]，QBF 技术的应用范围从人工智能到规划 [5,6,20]。QBF 求解器仅用于实例化最外层的存在量化变量 $\vec{x}$；对于其他变量，其实例化取决于先前的全称变量 $\vec{z} = f(\vec{y})$。在这项工作中，我们利用 QBF 对布尔最小化问题进行编码和求解。

论文结构。本文其余部分安排如下：第 2 节介绍我们在实验中采用的算法，以及我们开发的 QBF 编码。第 3 节介绍我们开展的实验，对上述三种方法进行比较，并展示实验结果。第 4 节探讨未来可能的研究方向。

2 算法

在实验中，我们分析了三种算法，分别称为暴力搜索、基于 SAT 和基于 QBF。

蛮力算法（见算法 1）就是我们在第 1 节中提到的、用于证明公式最小化属于 Σ_2^p 的算法。当且仅当 $\phi \leftrightarrow \psi$ 是重言式时，两个公式 ϕ 和 ψ 等价。与 TAUT 一样，公式等价问题是 CoNP 完全。不过，考虑到我们会测试所有小于 ϕ 的公式 ψ 之间的等价性，调用 equivalent(ϕ, ψ) 的平均耗时，与将 ψ 视为一个小于 ϕ 的随机公式时相同。因此，该函数调用所需的平均时间仅为 $\mathcal{O}(|\phi|)$。请注意，在这种情况下，一半的调用在检查一个赋值后结束， $1/4$ 在检查两个赋值后结束，依此类推。因此，平均而言，每次调用会检查 $\sum_{i=1}^{2^{|\phi|}} i\frac{1}{2^i} < 2$ 个赋值。

[4] http://www.qbflib.org

算法 1： 蛮力算法

输入： ϕ
输出： 最小等价公式 ψ

```
函数 等价性(φ, ψ):
    对于每个 Assignment I : Var(φ) → {0,1} 执行
        如果 I(φ) ≠ I(ψ) 则
            返回 true
    返回 false
函数main(φ):
    对 i = 1,..,|φ| 中的每一项执行
        对 formula s.t. |ψ| = i and Var(ψ) ⊆ Var(φ) 中的每一项执行
            如果 equivalent(φ, ψ) 则
                返回 ψ
```

我们还可以如下估算对该函数的调用次数。使用 k 个二元符号，可以构造的大小为 n 的不同完整树的数量为 k^n。如果树可以采用任意形状，计算就会更复杂。令 $\mathcal{C}$ 为可能的二元符号集合（因此不考虑非），令 $\mathcal{V}$ 为可能的叶节点集合。具有 m 个二元节点和 $m+1$ 个叶节点的树形数量由递推式 $f(m) = \sum_{i=0}^{m-1} f(i)\ f(m-i-1)$ 给出，该递推式定义了卡特兰数 C_m。不同树的数量为 $C_m \left|\mathcal{C}\right|^m \left|\mathcal{V}\right|^{m+1}$。使用斯特林近似，可以将其近似为 $\frac{4^m}{\sqrt{\pi} m^{3/2}} \left|\mathcal{C}\right|^m \left|\mathcal{V}\right|^{m+1}$。将树的大小 $n = 2m+1$ 作为变量时，这相当于调用等价性函数 $\mathcal{O}((4|\mathcal{C}||\mathcal{V}|)^{n/2}/n^{3/2})$ 次。

第二种算法（见算法 2）基于使用 SAT 求解器，以及对我们要证明等价的两个公式进行 Tseitin 编码。给定两个公式 ϕ、ψ，我们可以在线性时间内 $|\phi| + |\psi|$ 找到一个合取范式公式 Γ，使得这两个公式等价，当且仅当 Γ 不可 满足。乍看之下，使用 SAT 求解器检查一个平均只需线性时间的性质似乎并不合理。然而，实验表明，实践中相较于蛮力算法，我们仍能获得一定收益（见第 3 节）。

第三种算法（参见算法 3）基于 QBF 求解器。这里，我们不再测试每一种可能的最小公式 ψ，而是测试每一种可能的深度 δ。这是一项重大改进，因为需要尝试的深度数量是线性的，而公式候选项的数量则是指数级的。其次，我们不对候选项进行 Tseitin 编码，而是计算候选项的方案。原始公式与该方案之间的等价性可以编码为一个具有三次量词交替的 QBF 公式：$\exists \vec{x}.\forall \vec{y}.\exists \vec{z}.\Gamma$。在算法 3 中，这三组变量分别表示为 $\exists^{(1)}$、$\forall^{(1)}$ 和 $\exists^{(2)}$，各个变量则分别命名为 x、y 和 z。如果 QBF 公式为真，我们得到的变量 $x \in \exists^{(1)}$ 的取值将编码出最小公式。请注意，QBF 求解器只提供最外层存在量化变量的实例化，因为其他存在量化变量的取值取决于更外层的全称量化变量。

算法 2： 基于 SAT 的算法

```
输入： φ
输出： 一个最小等价公式 ψ
函数 Tseitin 变换(φ, x)：
    如果 φ = φ1 ∧ φ2 则
        y1, y2 := freshvars()
        返回 tseitin(φ1, y1) ∪ tseitin(φ2, y2) ∪ CNF({x ↔ y1 ∧ y2})
    ... /* 其他联结词或变量同理 */
函数 等价性(φ, ψ)：
    x1, x2 := freshvars()
    Γ := tseitin(φ, x1) ∪ tseitin(ψ, x2) ∪ CNF({¬(x1 ↔ x2)})
    返回 SAT(Γ) ≠ satisfiable
函数main(φ)：
    对 i = 1,...,|φ| 中的每一项执行
        对 formula s.t. |ψ| = i and Var(ψ) ⊆ Var(φ) 中的每一项执行
            如果 equivalent(φ, ψ) 则
                返回 ψ
```

基本上，对于方案中的每个节点 i 以及每个真值常量、原始公式中的变量 y 或联结词 c，我们都有一个变量 x_c^i：当位置 i 处是联结词 c（分别对应变量 y 或常量）时，该变量的取值为真。约束 CNF（$\{x_{false} + \sum_{y \in \forall^{(1)}} x_y + \sum_{c \in \mathcal{C}} x_c = 1\}$）确保其中恰有一个变量的取值为真。$\forall^{(1)}$ 中的变量就是原始公式中的变量集合。对于这些变量的所有赋值，如果原始公式和方案的评测结果相同，那么原始公式与方案就等价。$z^i \in \exists^{(2)}$ 中的变量编码方案或原始公式的 Tseitin 编码在位置 i 处每个可能子公式的真值。QBF 公式中的子句编码了以下形式的限制

$$
\begin{aligned}
&x^i_{\text{false}} \to \neg z^i \\
&x^i_y \to (z^i \leftrightarrow y) \\
&x^i_c \to (z^i \leftrightarrow z^{i \cdot 1}\, c\, z^{i \cdot 2})
\end{aligned}
$$

这些约束的预期含义是：如果方案在位置 i 处的常量为假，那么子方案的求值结果为假；如果此处是原始变量 $y \in \forall^{(1)}$，那么其求值结果为 y；如果此处是联结词 $c \in \mathcal{C}$，那么子方案的值就等于该联结词 c 对左子节点 i 的求值结果 $z^{i \cdot 1}$ 和右子节点的求值结果 $z^{i \cdot 2}$ 进行运算所得的值。

[5] 如果使用的 QBF 求解器无法提供这些实例化结果，我们就无法计算出最小等价公式。

算法 3： 基于 QBF 的算法

输入： ϕ
输出： 一个最小等价公式 ψ

函数 方案(δ, z)**：**
  $x_{false} := freshvar(\exists^{(1)})$
  $\Gamma := CNF(\{x_{false} \rightarrow \neg z\})$
  对每个 $y \in \forall^{(1)}$ **执行**
    $x_y := freshvar(\exists^{(1)})$
    $\Gamma := \Gamma \cup CNF(\{x_y \rightarrow (z \leftrightarrow y\})$
  如果 $\delta > 0$**，则**
    $z_1, z_2 := freshvar(\exists^{(2)})$
    $\Gamma := \Gamma \cup scheme(\delta - 1,\ z_1) \cup scheme(\delta - 1,\ z_2)$
    对每个 $c \in \mathcal{C}$ **执行**
      $x_c := freshvar(\exists^{(1)})$
      $\Gamma := \Gamma \cup CNF(\{x_c \rightarrow (z \leftrightarrow z_1\ c\ z_2)\})$
    返回 $\Gamma \cup CNF\ (\{x_{false} + \sum_{y \in \forall\ (1)}\ x_y + \sum_{c \in \mathcal{C}} x_c = 1\})$

函数 等价性$(\phi,\ \delta)$**：**
  $z_1, z_2 := freshvars(\exists^{(2)})$
  $\forall^{(1)} := Vars(\phi)$
  $\Gamma := tseitin(\phi,\ z_1) \cup scheme(\delta,\ z_2) \cup CNF(\{z_1 \leftrightarrow z_2\})$
  返回 $QBF(\Gamma) = true$

函数`main`(ϕ)**：**
  对 $\delta = 1,.., depth(\phi)$ **执行**
    对每个 $formula\ s.t.\ depth(\psi)\ = i\ and\ Var(\psi)\ \subseteq Var(\phi)$ **执行**
      如果 $equivalent(\phi, \delta)$ **那么**
        返回 ψ

我们得到的量化布尔公式的规模为 $\mathcal{O}\left(2^{depth(\phi)} \cdot \left(|\mathcal{C}| + |Vars\left(\phi\right)|\right)\right)$。假设原始公式是平衡的，实际中 $2^{depth(\phi)} \approx |\phi|$。因此，我们可以将其视为一种多项式编码。还要注意，对于大多数联结词，我们并未利用其交换律和结合律。

当我们在方案的某个节点中放入一个非时，只会使用它的一个子节点；而放入一个变量时，则不会使用任何子节点。为了避免 QBF 求解器进行无用搜索，我们可以通过添加约束 $CNF(\{x^i_{\neg} \rightarrow x^{i \cdot 2}_{dummy}\})$ 和 $CNF(\{(x^i_y \vee x^i_{dummy} \vee x^i_{false}) \rightarrow (x^{i \cdot 1}_{dummy} \wedge x^{i \cdot 2}_{dummy})\})$，强制将所有这些无用节点固定为一个哑值。

在这种方法中，我们只限制方案的深度。如果还想限制其大小，可以添加某种基数约束的编码，以限制方案中不同于虚拟节点的节点数量：
$\sum_i \neg x^i_{dummy} \leq size\ bound$

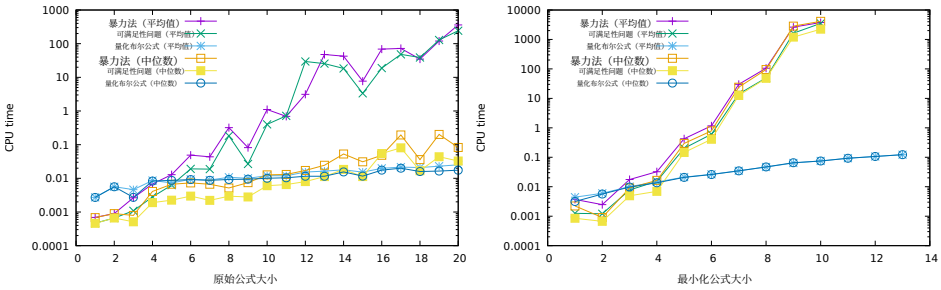


图 1. 三种算法所需时间的平均值和中位数，分别对应原始公式（左）和最终最小化公式（右）的规模。

3 实验与结果

我们对所提出的算法进行了一些实验。这三种算法均使用 Python 3 实现，并已公开发布于 https://gitlab.nl4xai.eu/eduardo.calo/QBF-boolean-minimization。对于基于 SAT 的算`python-sat`[6] 法，我们使用 Python 模块作为 SAT 求解器。对于基于 QBF 的算法，我们使用`CAQE` [16,21] QBF 求解器，不过也可以使用任何其他接受 QDIMACS 标准[7] 输入和输出的 QBF 求解器。

对于 $s = 1, \ldots, 20$ 中的每个规模，我们生成 100 个规模为 s 的随机公式，并使用三种算法对其进行最小化。我们确保所有语法上不同的公式以相同的概率生成。不过，我们不考虑联结词的交换律和结合律，也不考虑其他公式等价关系。规模为 s 的公式由一组 $\sqrt{s}$变量[8] 和联结词 $\mathcal{C} = \{$非,与,俄勒冈州$\}$ 生成；最小化则在包含联结词 $\mathcal{C}' = \{$非, 与, 俄勒冈州,蕴含$\}$ 的公式中进行。

我们使用一个包含 11 个计算节点的集群，每个节点配备 2 个英特尔至强处理器，每个处理器有 $2.2GHz$ 个核心，并配备 $92GB$ 随机存取存储器。我们将超时设为 20，$000s$。对于某些实例，暴力法和基于 SAT 的算法会达到超时：$s = 15$，18，19，20。计算平均时间和中位数时间时不考虑这些数值。因此，平均值和中位数异常偏低。

图 1（左）展示了每种算法所需 CPU 时间的平均值和中位数（取对数）随输入公式大小的变化情况。我们可以清楚地看到，基于 QBF 的算法优于另外两种算法；后两者似乎需要随输入规模增长而耗费指数时间。我们还观察到，基于 SAT 的算法始终优于暴力搜索算法（在对数坐标轴上，曲线间距恒定意味着性能提升了一个恒定倍数）。这相当令人惊讶，因为，

[6] https://github.com/pysathq/pysat [7] http://www.qbflib.org/qdimacs.html [8] 使用 s/c 或 s^c 变量似乎不会对结果产生显著影响。

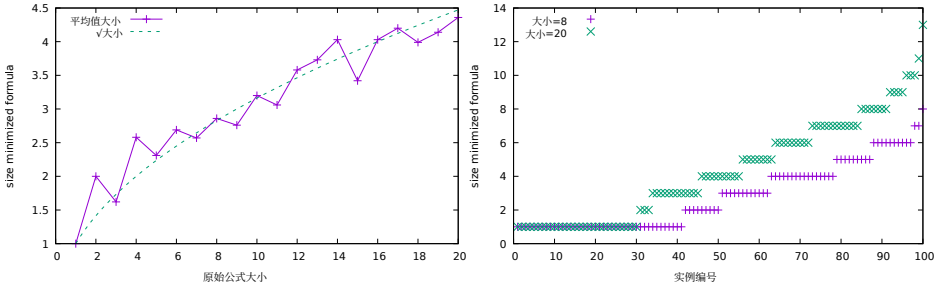


图 2. 最小化公式的平均大小与原始公式大小之间的关系（左），以及原始大小为 8 和 20 的公式的最小化大小分布（右）。

正如我们在第 2 节中提到的，计算公式等价性所需的平均时间为线性时间。另一个值得注意的现象是，在暴力搜索和基于 SAT 的算法中，平均时间与中位数时间有显著差异。正如下文将详细讨论的，这是因为各个实例所需时间存在显著差异。这种现象也会导致平均时间的数值波动。我们可以得出结论：尽管对于大多数实例（以中位数为准），三种算法都能在不到一秒的时间内完成公式最小化，但对于小于 20 的规模，只需少数几个实例，就会使暴力搜索和基于 SAT 的算法在规模约为 20 时平均耗时约为 $1h$ 。

在图 1（右）中，我们展示了 CPU 时间（取对数后）的平均值和中位数随所得最小公式大小变化的情况。这里，平均时间与中位数时间之间的差异较小。因此，我们可以得出结论，输出大小决定了算法所需的时间。同样明显的是，基于 QBF 的算法优于另外两种算法。我们仍然观察到，中位数时间小于平均时间，这表明仍存在显著的变异性。值得注意的是，运行时间取决于公式大小的奇偶性：偶数大小的公式比奇数大小的公式更容易处理。原因可能是，除了否定之外，其余联结词都是二元的。

在图 2（左）中，我们展示了最小化公式的平均大小如何随原始公式大小的增加而增长。我们观察到，其增长趋势接近原始公式大小的平方根。回想一下，我们生成了大小为 s 、含有 $\sqrt{s}$ 个变量的随机公式。有趣的是，我们发现奇数大小的公式比偶数大小的公式简化得更多，尽管原因尚不明确。在图 2（右）中，我们展示了最小化公式大小的分布（对应原始公式大小为 20 和 8 的情况）。

如上所述，暴力搜索算法和基于 SAT 的算法的 CPU 时间存在显著差异。在图 3 中，我们按 CPU 时间降序排列各个实例，并在双对数坐标轴上展示 100 个实例所需的时间。可以看到，这种表示近似为一条直线（由于超时，顶部被截断），且随着规模增大，斜率（为负）绝对值也随之增大。这意味着这些算法的 CPU 时间

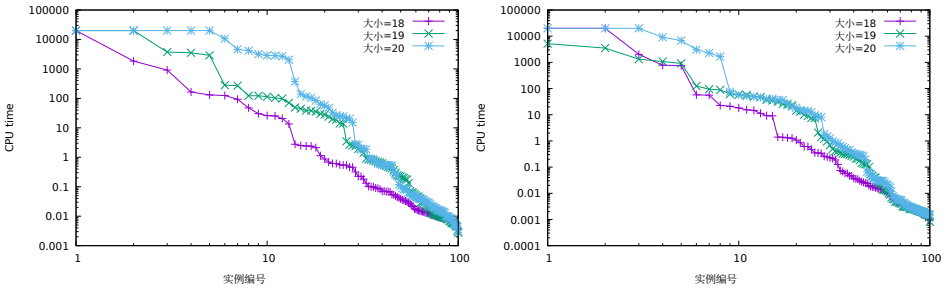


图 3. 使用蛮力算法（左）和基于 SAT 的算法（右）求解的实例的耗时分布。

服从幂律概率分布，少数实例所需的时间占据了大部分平均耗时。在这种情况下，标准做法是采用某种重启策略，或对算法进行随机化。在我们的例子中，可以随机排列最小公式的候选项。不过，由于我们希望得到最小等价公式，因此不能随机改变我们尝试的公式规模顺序。

4 结论与未来工作

本文分析了三种用于一般布尔公式最小化的算法在实践中的应用：一种简单算法，用于证明该问题属于 Σ_2^p；一种利用 SAT 求解器检查公式等价性的算法；以及一种对公式方案进行 Tseitin 编码并使用 QBF 求解器的算法。我们表明，第三种算法的表现明显优于另外两种。因此，使用 QBF 求解器代表了布尔最小化问题的最先进水平。

我们的实验仅限于布尔公式。这项工作的首个自然延伸方向，是探究这种或类似方法能否扩展到其他（表达能力更强的）形式体系，例如一阶逻辑（FOL）。这将引出一系列有趣的研究问题，因为在一阶逻辑中，等价性是不可判定的。改用量化布尔公式方法可能并不可行；不过，半暴力搜索方法（例如使用一阶定理证明器）或许能够奏效。

参考文献

1. Beyersdorff, O., Janota, M., Lonsing, F., Seidl, M.：量化布尔公式。载于：Biere, A., Heule, M., van Maaren, H., Walsh, T.（编），《可满足性手册》（第二版），人工智能与应用前沿，第 336 卷，第 1177–1221 页。IOS Press（2021）。https://doi.org/10.3233/FAIA201015

2. Brayton, R.K., Hachtel, G.D., Hemachandra, L.A., Newton, A.R., Sangiovanni-Vincentelli, A.L.M.：使用 espresso 的逻辑最小化策略比较：用于分区逻辑最小化的 APL 程序包。载于：国际电路与系统研讨会论文集。第 42–48 页（1982） 3. Buchfuhrer, D., Umans, C.：布尔公式最小化的复杂性。《计算机与系统科学杂志》**77**(1), 142–153（2011）。https://doi.org/https://doi.org/10.1016/j.jcss.2010.06.011, 纪念 Karp 获得京都奖 4. Cal`o, E., van der Werf, E., Gatt, A., van Deemter, K.：增强并评测逻辑到文本生成的语法框架方法。载于：第二届自然语言生成、评测与指标研讨会（GEM）论文集。第 148–171 页。计算语言学协会，阿布扎比，阿拉伯联合酋长国（线上线下结合）（2022 年 12 月） 5. Cashmore, M., Fox, M.：作为 QBF 的规划。载于：国际自动规划与调度会议博士生论坛（ICAPS 2010）（2010） 6. Diptarama, R.Y., Shinohara, A.：广义井字棋的 QBF 编码。载于：第四届量化布尔公式（QBF）国际研讨会，与第十九届可满足性测试理论与应用国际会议（SAT）同期举行，法国波尔多。第 14–26 页（2016） 7. Garey, M.R., Johnson, D.S.：《计算机与难解性》，第 174 卷。Freeman，旧金山（1979） 8. Gatt, A., Krahmer, E.：自然语言生成最新进展综述：核心任务、应用与评测。《人工智能研究杂志》**61**, 65–170（2018） 9. Giunchiglia, E., Marin, P., Narizzano, M.：量化布尔公式推理。载于：《可满足性手册》，第 761–780 页。IOS Press（2009） 10. Grice, H.P.：逻辑与会话。载于：言语行为，第 41–58 页。Brill（1975） 11. Karnaugh, M.：用于组合逻辑电路综合的图解法。《美国电气工程师学会汇刊，第一部分：通信与电子学》**72**(5), 593–599（1953）。https://doi.org/10 6371932 12 1109 1953 . /TCE. . . McCluskey, E.J.：布尔函数最小化。《贝尔系统技术杂志》**35**(6), 1417–1444（1956）。https://doi.org/10.1002/j.1538-7305.1956.tb03835.x 13. Petrick, S.R.：从主蕴含项集合直接确定布尔函数的无冗余形式。空军剑桥研究中心技术报告，第 56–110 页（1956） 14. Quine, W.V.：简化真值函数的问题。《美国数学月刊》**59**(8), 521–531（1952）。https://doi.org/10 11988183 15 1080 00029890 1952 . / . . . Quine, W.V.：一种简化真值函数的方法。《美国数学月刊》**62**11988710 16 9 627 631 1955 10 1080 00029890 1955 2015 ( ), – ( ). https://doi.org/ . / . . . Rabe, M.N., Tentrup, L.：CAQE：一种可验证 QBF 求解器。载于：计算机辅助设计形式化方法（FMCAD）。第 136–143 页（2015）。https://doi.org/10 7542263 17 1109 2015 . /FMCAD. . . Ranta, A.：语言与逻辑之间的转换：何者容易，何者困难。载于：国际自动演绎会议论文集。第 5–25 页。Springer（2011） 18. Reiter, E., Dale, R.：自然语言生成系统构建。《自然语言处理研究》，剑桥大学出版社（2000）。https://doi.org/10.1017/CBO9780511519857

19. Seidl, M., Lonsing, F., Biere, A.: qbf2epr：一种从量化布尔公式生成有效命题逻辑公式的工具。载于：Fontaine, P., Schmidt, R.A., Schulz, S.（编）PAAR-2012. 第三届自动推理实践专题研讨会。EPiC计算系列，第21卷，第139–148页。EasyChair（2013）。https://doi.org/10.29007/2b5d 20. Shukla, A., Biere, A., Pulina, L., Seidl, M.: 量化布尔公式的应用综述。载于：2019年IEEE第31届人工智能工具国际会议（ICTAI）。第78–84页。IEEE，美国俄勒冈州波特兰（2019年11月）。https://doi.org/10 00020 21 1109 2019 . /ICTAI. . . Tentrup, L.: CAQE and QuAbS：基于抽象的QBF求解器。可满足性、布尔建模与计算期刊**11**，155–210（2019年9月）。https://doi.org/10 190121 22 3233. /SAT . Turton, B.C.: 将奎因-麦克拉斯基方法扩展至异或逻辑综合。IEEE教育汇刊**39**(1)，81–85（1996年2月）。https://doi.org/10.1109/13.485236