利用Int-Splits进行搜索空间剪枝以加速QBF求解

Maximilian Heisinger 林茨
约翰·开普勒大学，奥地利
Irfansha Shaik 奥
胡斯大学，丹麦
Martina Seidl 林茨约翰·开普勒大学，奥地利
Jaco van de Pol
奥胡斯大学，丹麦

arXiv:2304.08308v1 [cs.LO] 17 Apr 2023

摘要

在许多量化布尔公式编码中，布尔变量序列表示整数变量的二进制表示。例如，有界模型检测中的状态标签或规划问题中的动作。通常不会用到所有可能的取值范围；例如，要表示六种不同状态，需要三个布尔变量，因此八种可能赋值中有两种与问题求解无关。由于量化布尔公式求解器对所处理的公式不具备任何领域专属知识，因此无法发现这一剪枝机会。

本文介绍了整数分割的概念，它向量化布尔公式求解器提供有关整数变量的领域专属信息。这对并行分治求解尤其有用，因为该方法会将搜索空间划分为可独立求解的子问题。借助这一技术，我们将生成的子问题数量从完整展开减少到仅包含所需的部分。随后，我们评估整数分割在本就适合分治的规划问题中能节省多少资源。为此，我们提供了一个参考实现，可将量化布尔公式划分为多个子问题（可选择使用或不使用整数分割），并合并求解结果。最后，我们提出一种基于注释的可选语法扩展，适用于 `(Q)DIMACS`，其中包含整数分割，并适合以原生方式向分治求解器提供建议的引导路径。

2012年ACM学科分类 计算理论 → 自动推理；计算方法 →搜索方法

关键词与短语 量化布尔公式、分治、搜索空间剪枝、位向量、整数分割

数字对象标识符 10.4230/LIPIcs.SAT.2023.23

1 引言

量化布尔公式（QBF）通过量词 $\forall, \exists$ 扩展了命题逻辑，其作用对象是布尔变量 [1]。因此，QBF 的判定问题（QSAT）是 PSPACE 完全的，使 QBF 成为形式化验证、综合或规划等诸多应用问题的合适框架 [13]。近年来，人们提出了功能强大的求解工具，即使搜索空间随变量数量呈指数级增长，它们也能为大型实例找到解。这些工具主要利用公式中的结构信息，采用先进的学习技术，并实现复杂的搜索策略（近期综述见 [1]）。目前，QBF 求解工具完全不考虑编码所对应的问题本身，也就是说，它们并不掌握原始问题解空间的任何领域专属信息。

例如，规划问题的解是一个用于到达预定义目标 [10] 的动作序列。在用命题逻辑对规划问题进行编码时，会执行基化步骤，将所有动作规则与具体的对象组合进行实例化。借助

© Maximilian Heisinger、Irfansha Shaik、Martina Seidl 和 Jaco van de Pol；依据知识共享许可协议 CC-BY 4.0 授权。第26届可满足性测试理论与应用国际会议（SAT 2023）。编辑：John Q. Open 和 Joan R. Access；文章编号：23；第 23:1–23:10 页 莱布尼茨信息学国际会议论文集 Schloss Dagstuhl – Leibniz-Zentrum für Informatik，德国 Dagstuhl Publishing
LIPICS

量词，规划问题和双人游戏的编码相比基于实例化命题的编码可紧凑一个数量级，从而降低求解器的内存需求 [12, 11]。在游戏编码中，前 n 个量词交替编码了为达到指定目标而要执行的动作或移动序列。每个量词块包含一组变量，这些变量合在一起表示相应步骤中要执行的动作。假设有 19 种不同的动作可供选择，那么每个量词块必须包含五个变量。由于可能的赋值有 32 种，其中 13 种与解无关。这些信息本可以显著减少搜索开销。然而，求解器迄今尚无法利用这些信息（或者说，尚未检测到这些信息）。

本文提出为量化布尔公式编码补充领域专属信息。具体而言，我们设计了一种格式，用于限制旨在表示整数的位集合的取值范围。随后，我们利用这些信息提高分治求解器的效率，并减少伪子问题的产生，即那些无法为最终解作出贡献的子问题。作为案例研究，我们考察了规划问题中的若干实例，以展示该方法的潜力。我们的参考实现已根据 MIT 许可证公开发布。

2 预备知识

我们考虑如下形式的量化布尔公式：$Q.\varphi$，其为前束合取范式 (PCNF)，其中前缀 Q 形式为 $Q_1x_1, \ldots, Q_nx_n$，其中 $Q_i \in \{\forall, \exists\}$。矩阵 φ 是关于变量 $x_1, \ldots, x_n$ 的合取范式（CNF）命题公式。合取范式公式是子句的合取（$\wedge$）。子句 是文字的析取（$\vee$）。文字可以是变量 x、否定变量 $\neg x$，也可以是真值常量 $\top$（真）或 $\bot$（假）（可带否定）。对于文字 l，表达式 l 表示 x（若 $l = \neg x$），否则表示 $\neg x$。有时，我们将子句写成文字集合，将合取范式公式写成子句集合。

如果 φ 是一个合取范式公式，那么 $\varphi_{x\leftarrow t}$ 就是通过将 φ 中的 x 设置为值 $t \in \{0,1\}$而得到的合取范式公式。如果 $t = 0$，则将 x 设置为假；如果 $t = 1$，则将 x 设置为真。随后，根据命题逻辑的标准语义对公式进行化简。给定一个命题变量集合 X，赋值 $\sigma\colon X \to \{0,1\}$ 将 X 中的所有变量映射为真或假。我们用 φ_σ 表示将所有已由 σ赋值的变量进行设置后得到的公式。

量化布尔公式 ϕ 为真，当且仅当 $\nu(\phi) = 1$，其中将量化布尔公式映射到真值 $\{0,1\}$ 的解释函数 ν 定义如下：

- $\nu(\forall x\mathcal{Q}.\varphi) = 1$ 当且仅当 $\nu(\mathcal{Q}.\varphi_{x\leftarrow 0}) \cdot \nu(\mathcal{Q}.\varphi_{x\leftarrow 1}) = 1\nu$
- $(\exists x\mathcal{Q}.\varphi) = 1$ 当且仅当 $\nu(\mathcal{Q}.\varphi_{x\leftarrow 0}) + \nu(\mathcal{Q}.\varphi_{x\leftarrow 1}) \geq 1$

注意，我们假设量化布尔公式中的所有变量都经过量化，即我们只考虑闭合公式。此外，我们采用合取、析取、否定和真值常量的标准语义。例如，量化布尔公式 $\phi_1 = \forall x\exists y.((x \vee y) \wedge (\neg x \vee \neg y))$ 为真，而 $\phi_2 = \exists y\forall x.((x \vee y) \wedge (\neg x \vee \neg y))$ 为假。正如这个小例子所示，语义会根据前缀对变量施加顺序。

3 使用 Int-Split 的量化布尔公式

本节中，我们使用整数分割扩展量化布尔公式的语言。借助整数分割，我们向求解器提供领域特定知识，即指出哪些布尔变量集合表示某个特定范围内的整数。为此，我们必须重新定义量词的语法和语义，也就是为其补充有关整数域的信息。为便于说明，我们将布尔变量归组为位向量变量。位向量变量 $\vec{x}$

是一个命题变量元组（x_1、x_2、...、x_n）。在这种情况下，$\vec{x}$的大小为 $|\vec{x}|$ n。位向量是由位 {0, 1} 组成的元组。要为大小为 n的位向量中的所有位变量赋予真值，我们写作 $\vec{x} \leftarrow (t_1$、t_2、...、$t_n)$；或者，如果 σ 是 $\vec{x}$中出现的命题变量的一个赋值，则也可以写作 $\sigma(\vec{x})$。对于合取范式公式 φ，将 x_1 设为 $\sigma(x_1)$、将 x_2 设为 $\sigma(x_2)$、.....、将 x_n 设为 $\sigma(x_n)$后，得到公式$\varphi_{\sigma(\vec{x})}$。我们用 $\mathsf{S}(\vec{x})$ 表示 $\vec{x}$的所有可能赋值构成的集合，它基本上表示所有长度为 $|\vec{x}|$的位向量。若（t_1，...，t_n）是一个位向量，则其整数表示为$(t_1$，...，$t_n) = 2^{n-1} \cdot t_1 + \ldots + 2^0 \cdot t_n$。例如，若 $\vec{x} = (x_1$、$x_2)$，且 $\sigma(x_1) = 1$并且 $\sigma(x_2) = 0$，则$\sigma(\vec{x}) = (1, 0) = 2$。

▶ **定义 1**（带 Int-Split 的量化布尔公式的语法）。带整数分割的量化布尔公式具有如下形式$Q_1 (\vec{x_1})_{c_1} \ldots Q_n (\vec{x_n})_{c_n}.\varphi$，其中 $Q_i \in \{\forall, \exists\}$，且 $\vec{x_i}$是大小为 ≥ 1 的位向量变量，并且不存在同时出现在两个不同位向量变量中的布尔变量。约束 $c_i \in \{ <v_i, >v_i, =V, \top \}$ 限制量词 Q_i的定义域。在约束的定义中，$v_i \in \mathbb{N}^+$ 是一个常量，V 是位向量的集合。与之前一样，φ 是关于 $\vec{x_i}$中的布尔变量的合取范式命题公式。

请注意，上述定义允许 $Q_i = Q_{i+1}$，且位向量变量的大小可以不同。如果此类位向量变量的大小为一，那么它就是一个简单的布尔变量。真值常量 $\top$ 表示对前一个位向量变量的取值范围没有限制。我们通常省略约束，而不写作 $\top$。此外，请注意，φ 是命题公式，而不是位向量公式。我们将 $Q(\vec{x})_c$ 称为带注释的量词。带整数分割的量化布尔公式的一个例子是

$$\exists(x_1, x_2)_{<3} \exists(y_1, y_2, y_3)_{=\{101,111\}} \forall(z_1, z_2)_{>2} \forall a \exists b.\varphi.$$

▶ **定义 2**（带整数分割的量化布尔公式的语义）。设 $\phi = Q\vec{x}_c Q.\varphi$ 是带整数分割、约束为 c 且 $Q \in \{\exists, \forall\}$ 的量化布尔公式。那么，当且仅当 $\nu(\phi) = 1$ 时，ϕ 为真，其中解释函数 ν 定义如下。

- $\nu(\exists\vec{x}_\top Q.\varphi) = 1$ 当且仅当 $\sum_{\sigma \in \mathsf{S}(\vec{x})} \nu(Q.\varphi_{\sigma(\vec{x})}) \geq 1$
- $\nu(\forall\vec{x}_\top Q.\varphi) = 1$ 当且仅当 $\prod_{\sigma \in \mathsf{S}(\vec{x})} \nu(Q.\varphi_{\sigma(\vec{x})}) = 1$
- $\nu(\exists\vec{x}_{\diamondsuit v} Q.\varphi) = 1$ 当且仅当 $\sum_{\sigma \in \mathsf{S}(\vec{x}), \sigma(\vec{x}) \diamondsuit v} \nu(Q.\varphi_{\sigma(\vec{x})}) \geq 1$ 其中 $\diamondsuit \in \{<, >\}$
- $\nu(\forall\vec{x}_{\diamondsuit v} Q.\varphi) = 1$ 当且仅当 $\prod_{\sigma \in \mathsf{S}(\vec{x}), \sigma(\vec{x}) \diamondsuit v} \nu(Q.\varphi_{\sigma(\vec{x})}) = 1$ 其中 $\diamondsuit \in \{<, >\}$
- $\nu(\exists\vec{x}_{=V} Q.\varphi) = 1$ 当且仅当 $\sum_{\sigma \in \mathsf{S}(\vec{x}) \ \sigma(\vec{x}) \in V} \nu(Q.\varphi_{\sigma(\vec{x})}) \geq 1$
- $\nu(\forall\vec{x}_{=V} Q.\varphi) = 1$ 当且仅当 $\prod_{\sigma \in \mathsf{S}(\vec{x}) \ \sigma(\vec{x}) \in V} \nu(Q.\varphi_{\sigma(\vec{x})}) = 1$

请注意，整数分割对应于有界量化，这可能会改变量化布尔公式的语义。但这并非我们的本意：我们将整数分割视为注释，即对量化布尔公式中原本就存在、但求解器未能识别的整数界限所作的提示；正因如此，搜索空间不必要地扩大了。用户（建模者）有责任提供正确的整数分割，具体规定如下假设：

▶ **假设 3** (整数分割的正确性)。给定一个带有 *Int-Split* 的量化布尔公式 ϕ，并令 ϕ' 为将 ϕ 中所有 *Int-Split* 约束替换为 $\top$ 后得到的量化布尔公式。ϕ 中的 *Int-Split* 是正确的 当且仅当 $\nu(\phi) = \nu(\phi')$。从现在起，我们假设指定的 *Int-Split* 始终正确。

借助（正确的）整数分割，可以大幅缩减搜索空间，因为我们现在能够指出某些变量赋值无法促成一个解，从而使某些展开变得不必要。请看图 1 中这个小例子，它将公式 $\forall\vec{x}_{<3} \exists\vec{y}_{<3} Q.\varphi$ 表示为一棵整数分割树。16 个分支中，求解时必须考虑其中 9 个，另外 7 个则可以忽略。因此，我们将展开区分为已计数展开和未计数展开，定义如下。



图 1 两层整数分割。虚线框表示未展开的存在扩展（$7/16 \approx 44\%$）。

▶ **定义4**。 设 $\Phi = Q.\varphi$ 是一个带有整数分割的量化布尔公式，其中包含一个带注释的量词$Q_i(\vec{x_i})_{c_i}$。再设 X 是出现在 $\vec{x_j}$ 中的布尔变量集合，且满足 $j \le i$。若赋值 $\sigma\colon X \to \{0,1\}$满足 $\vec{x}$，且 $c_i(\sigma(\vec{x}))$ 为真，则该赋值是 $\vec{x}$的一个已计数展开 *(AE)*。否则，σ 是 $\vec{x}$的一个未计入展开 *(UE)*。

因此， $\vec{x}_{<3}$ 和 $\exists\vec{y}_{<3}$ 各有三次计入展开和一次未计入展开。对于带注释的量词 $Q\vec{z}_\top$，AE 的数量为 $2^{|\vec{z}|}$，而存在扩展的数量为零。下文假设，对于每个带注释的量词 Q，若其有 s 个 AE，则 $s > 0$，因为否则可以将 Q 从前缀中移除。

4 整数分割与分而治之

4.1 背景

在分治（D&C）SAT 求解中， [3], 会选取一个分裂变量，并生成两个公式副本。一个副本中的分裂变量设为真（1），另一个副本中的分裂变量设为假（0）。重复此过程，直到达到给定深度 d，从而得到 2^d个待求解子问题。SAT 中可以自由选择决策变量，而对于包含 $Q_1 \neq Q_2$和 $Q_1, Q_2 \in \{\forall, \exists\}$的 QBF $Q_1 X Q_2 Y \mathcal{Q}.\varphi$，变量选择的范围则更受限制。只有最外层量词块中的变量 $Q_1 X$

可以作为分裂变量的候选；只有当 X 中的所有变量都已赋值后，才可以考虑 Y 中的变量。在 SAT 求解中，只要发现一个子问题可满足，就可以停止求解；而在 QBF 求解中，则需要考虑不同的量词类型，将各子问题的结果合并。

在 SAT 中，D&C 方法在一种称为立方与征服（C&C）[7]的变体中，已成功应用于棘手的组合型 SAT 问题。后者依靠先进的启发式方法来选择分裂变量，而 Jordan 等人 [8] 观察到，在量化布尔公式求解中，仅按照第一个量词块中的变量顺序进行处理就已经能够带来改进。Heisinger 等人 [5] 将组合求解与量化布尔公式的分治求解相结合，并展示了进一步的改进。在此基础上，有人通过命令行选项 `--intsplit` n 提出了整数分割的一种初步版本，该选项会针对前 $\lceil \log_2 n \rceil$个变量创建 n 个任务。与本文所介绍的方法相比，该方法与编码没有关联，可表达的约束更加受限，并且没有针对效率进行优化 η。使用 ParaQooba 开展的未发表实验表明，整数分割具有潜力（但也存在求解不完整的危险），这促成了本文的工作：对整数分割进行形式化、扩展，并开展深入评估。

4.2 将分治与 Int-Split 相结合

对于 d 个分裂变量， 2^d 个子问题必须由分治求解器生成；而整数分割有望大幅减少需要评估的子问题数量。设 $Q(\vec{x_1})_{c_1} \dots Q(\vec{x_n})_{c_n} \mathcal{Q}.\varphi$ 是一个包含整数分割的量化布尔公式，且 $|\vec{x_1}| + \ldots + |\vec{x_n}| = d$.[1] 此外，设 $s_i = |\{c_i(\sigma(\vec{x_i})) \mid \sigma \in S(\vec{x_i})\}|$，即 $\vec{x_i}$ 的 AE 数量。那么，$\vec{x_i}$ 的未计入展开数量为 $u_i = 2^{|\vec{x}|} - s_i$。使用整数分割时，无需考虑 2^d 个子问题，而只需考虑 $s_1 \cdot \ldots \cdot s_n$ 个问题。

▶ **示例 5**。考虑公式 $Q_1(\vec{x_1})_{<19} Q_2(\vec{x_2})_{<19} Q_2(\vec{x_2})_{<19} \mathcal{Q}.\varphi$，其中 $|\vec{x_i}| = 5$，分割深度为 15。在传统的分治求解中，将生成 $2^5 \cdot 2^5 \cdot 2^5 = 2^{15} = 32768$ 个子问题。相比之下，利用整数分割信息，每个 x_i 都有 19 个 AE（因此有 13 个存在扩展）。总共需要生成 $19 \cdot 19 \cdot 19 = 6859$ 个问题，占完整展开的 $\approx 21\%$。

当一个前缀具有 $Q(\vec{x_1})_{c_1} \dots Q(\vec{x_n})_{c_n} \mathcal{Q}.\varphi$ 形式，且 $Q \in \{\forall, \exists\}$，但分割深度为 $d < |\vec{x_1}| + \ldots + |\vec{x_n}|$ 时，应选择剪枝潜力最大的展开进行展开。为此，我们引入 int-split 效率这一概念，以便重新排列同类的带注释的量词。

▶ **定义 6** (Int-Split 效率)。设 $\Phi = Q.\varphi$ 是一个含有整数分割的量化布尔公式，其中包含一个带注释的量词 $Q(\vec{x})_c$，它具有 s 个 *AE* 和 u 个存在扩展。那么 $Q(\vec{x})_c$ 的 int-split 效率 η 为 $\eta = \frac{u}{s}$。

回顾一下，对于每个带注释的量词 $Q(\vec{x})_c$，若其具有 s 个 AE，则有 $s > 0$。例如，$Q((x_1, x_2))_{<3}$ 具有 $\eta = \frac{1}{3}$，而 $\neg Q(\vec{y})_\top$ 具有 $\eta = 0$。

5 参考实现

本节将介绍我们的参考实现，用于测试在分治语境中采用整数分割的可行性。首先，我们提出一种基于注释的向前和向后兼容格式

[1] $|\vec{x_1}| + \ldots + |\vec{x_{n-1}}| < d < |\vec{x_1}| + \ldots + |\vec{x_n}|$，整数分割仅应用于第一个 $n-1$ 位向量。

扩展，以适用于 (Q)DIMACS 文件格式。[2] 随后，我们介绍如何在带注释的量词所施加的约束下，将问题分割为子问题。最后，我们说明如何合并子问题的结果，从而得到原始公式的解。我们的参考实现采用 MIT 许可证，并已公开发布于

github.com/maximaximal/qdimacs-splitter

它使用 Rust 编写，并且对于不同公式大小都具有很强的可扩展性。读取具有相同量化类型的整数分割时，它会按效率降序对其排序，并在每个带注释的量词内局部进行排序 η。它沿着已排序的前缀扩展公式，生成原始公式的多个副本，并将假设作为单子句附加。如果被扩展的变量由全称量词量化，则在副本中将其量词改为存在量词，以便应用布尔约束传播。每个副本都会保存为单独的文件，文件名以原始公式命名，并在前面加上相应副本的索引。

5.1 (Q)DIMACS 格式扩展

我们在 (Q)DIMACS 公式中以注释形式标注整数分割，以保持与 (Q)DIMACS 格式的兼容性。扩展后的语法见附录 A。整数分割注释与前缀在量词块方面必须匹配，但量词块内的顺序可以不同。请注意，若不满足上述正确性假设，量化布尔公式的语义可能会发生变化。目前，用户有责任正确使用这些注释。在我们的实现中，一个带注释的量词可以包含多个以分号分隔的约束。只要至少有一个约束得到满足，就会进行扩展。如果能够唯一确定 AE 的数量 s，则 $\vec{x}$ 可以留空，表示需要考虑前缀中的下一个 $\lceil \log_2 s \rceil$ 个变量。生成 QDIMACS 时必须提供前缀；生成 DIMACS 时，所有 $\vec{x}$ 都必须显式给出，且不允许提供前缀。

5.2 公式分割

为了计算所需的分割，我们的工具会跟踪待分割的布尔变量数量，为这些变量的所有可能赋值创建生成器，遍历生成器，并检查每一种可能的赋值是否都能满足所有纳入考虑的带注释的量词中的至少一个约束。分割深度参数 `--depth` 定义了待考虑分割的变量数量上限，并尽可能多地进行（int-）分割。对于给定的分割深度 d，最多会创建 2^d 个文件，每个文件的内容都与原始公式相同，唯一的区别是会追加用于表示假设的单位子句。由于分割和合并与求解过程本身（可能是并行的）相互独立，我们将这一 D&C 变体称为离线分割。

5.3 结果合并

离线分割使得可以使用诸如 Simsala [4], 之类的并行执行工具，同时还允许按顺序运行所有展开，仅模拟并行执行。求解器运行结束后，我们的合并器会读取原始公式以及已在内存中展开的 n 带注释的量词。随后，它会读取所有日志，并将文件映射为 $r_n = \prod^n s_i$结果代码与运行时间的元组（r_j^n，t_j^n），其中包含 $1 \leq j \leq r_n$。这些 r_n 元组会被归约

[2] http://www.qbflib.org/qdimacs.html

为 $r_{n-1} = \prod^{n-1} s_i$ 元组：如果$Q_n = \exists$，则取最大的结果值和最小的时间值；如果 $Q_n = \forall$，则取最小的结果值和最大的时间值。该过程会不断重复，直到新列表中只剩下一个结果，该结果代表树的根节点，其值为最终结果，以及使用 r_n 个处理器的虚拟并行求解器的墙钟时间，或采用最优决策启发式的顺序求解器的运行时间。因此，我们的合并器适用于决策启发式研究，因为它生成的结果表为改进启发式函数提供了效率上界，且不受可用物理核心数量或并行编程本身所带来困难的限制。将子问题映射到结果和运行时间的内部表示可被视为并行执行证书。

6 评估

我们扩展了 [5] 中使用的规划基准测试，加入整数分割，并将结果与未使用整数分割的相同公式进行比较。所有子公式均在双路 AMD EPYC 7313 @ 16x 3.7 GHz 机器组成的集群上运行，超时设为 3700 s，每个任务配备 8 GB 内存。我们使用 QBF 求解器 CAQE [9] 进行基准测试，并使用 Blo- qqer [2]，对分割后的公式进行预处理，因为这对所选公式最有利。Hex 和 GTTT 公式的前缀形式均为 $\exists(x_1x_2x_3x_4x_5)_{<19}\forall(x_6x_7x_8x_9x_{10})_{<19}\exists(x_{11}x_{12}x_{13}x_{14}x_{15})_{<19}\ldots$。

6.1 CPU 时间比较

我们的实验比较了求解若干公式所需的 CPU 时间，分别考察使用和不使用整数分割的情况。我们在公式中加入整数分割定义，并使用 $d = 10$ 进行分割。这样最多会产生 $2^{10} = 1024$ 个子问题。根据公式中的整数分割定义，启用整数分割后进行分割所产生的子问题数量会呈指数级减少。我们在有机合成（取自 [12]）的规划问题提升式 QBF 编码，以及双人位置游戏 Hex 和广义井字棋（取自 [11]）的提升式 QBF 编码上进行了测试。图 2 的结果符合预期：使用整数分割时，CPU 时间更少。

为进行公平比较，我们仅纳入最大深度 d等于使用 d进行分割时整数分割所覆盖的变量数量的结果。这意味着，采用整数分割时的 AE 数量与不采用整数分割时相同，只有存在扩展会进一步增加 CPU 时间。否则，整数分割的结果将覆盖更少的 AE，且采用整数分割时的有效分割深度会低于不采用时。表 1 的最后一行展示了这一点；保留该行是为了说明这一效果。

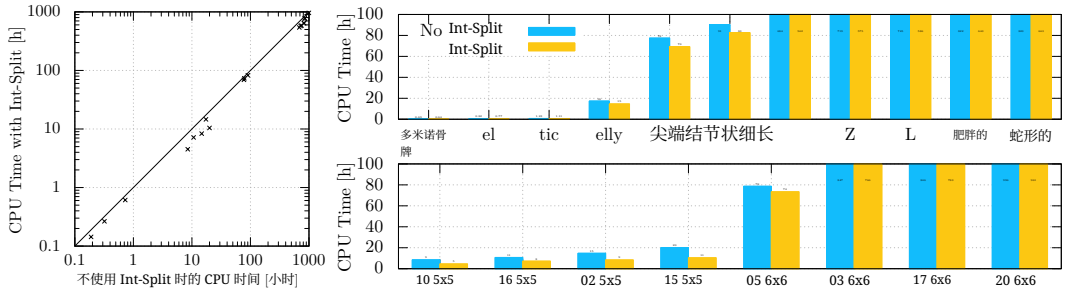

图 2 采用和不采用整数分割处理同一公式所需的 CPU 时间。数据包含可满足、不可满足和超时（均为 $t > 500$）的结果。纵轴按已求解实例的运行时间调整。上方柱状图显示 GTTT 的结果，下方显示 Hex 的结果。

公式	#v	#c	s	u	$d_{\mathrm{w/}}$	$d_{\mathrm{w/o}}$	$\#_{\mathrm{w/}}$	$\#_{\mathrm{w/o}}$	η	$t_{\mathrm{w/}}$ [h]	$t_{\mathrm{w/o}}$ [h]	$\frac{t_{\mathrm{w/}}}{t_{\mathrm{w/o}}}$	$\frac{t_{\mathrm{seq}}}{t_{\mathrm{par}}}$
Hex 10 5x5-13	1582	5531	19	13	10	10	361	1024	0.68	4.51	8.49	0.53	47.7
Hex 02 5x5-13	1570	5437	19	13	10	10	361	1024	0.68	8.33	14.7	0.57	32.9
Hex 16 5x5-13	1550	5437	18	14	10	10	324	1024	0.78	7.17	10.7	0.67	39.1
GTTT Domino	943	3075	25	7	10	10	625	1024	0.28	0.144	0.189	0.76	11.6
GTTT El	1180	3951	25	7	10	10	625	1024	0.28	0.265	0.320	0.83	10.4
GTTT Elly	1465	5024	25	7	10	10	625	1024	0.28	14.6	17.4	0.84	8.59
GTTT Tic	1155	3848	25	7	10	10	625	1024	0.28	0.612	0.726	0.84	9.47
GTTT Tippy	1409	4799	25	7	10	10	625	1024	0.28	69.2	77.4	0.89	7.72
GTTT Knobby	1411	4808	25	7	10	10	625	1024	0.28	82.6	90.4	0.91	-
Hex 05 6x6-13	1812	5437	31	1	10	10	961	1024	0.032	73.4	78.6	0.93	2.47
Hex 15 5x5-15	1602	5339	16	0	8	10	256	1024	0	10.4	20.0	0.52	30.7

表 1 所选基准测试结果，不含超时项，突出显示了采用（w/）和不采用（w/o）整数分割时求解结果的差异。所有公式的量词交替形式均为 $\exists\forall\ldots$。GTTT 和 Hex 15 5x5-15 有 15 次交替，其他 Hex 公式有 13 次。若无法在给定深度 $d = 10$内再进行整数分割，则 $d_{\mathrm{w/}}$ 低于 $d_{\mathrm{w/o}}$。 t 是 CPU 时间总和。# 是生成的子问题总数。 u越小，$t_{\mathrm{w/}}$ $t_{\mathrm{w/o}}$ 越接近 1。最右列 t_{seq} t_{par} 表示最优并行求解相对于顺序求解该公式的加速比（见第 5.3 小节）。采用和不采用整数分割时，这些数值相同，但整数分割能大幅减少所需资源（CPU 时间）。

6.2 墙钟时间比较

我们还计算了上述基准测试的墙钟时间。与按顺序求解完整公式相比，采用整数分割所观察到的加速比高于不采用整数分割时使用 ParaQooba 所观察到的加速比，因为我们可以将分割深度加倍。采用离线分割时，加速比最高可达 47.7，而在线拆分求解器 ParaQooba（在单个 16 核节点上）的最高加速比为 18.09。与 ParaQooba 不同，我们的拆分器和合并器在集群上运行单个任务，因此相比大型并行基准测试，可以更高效地进行调度。当前，离线分割在找到解后不会中止求解过程，不过，将此功能集成到运行时环境中可以通过脚本实现（例如使用 Slurm 的 `scancel` 命令）。

7 结论

我们为前束合取范式中的量化布尔公式提出了一种语言扩展，以便支持利用有界量词表达领域特定知识的求解器。尽管我们已清楚展示整数分割的潜力，但这项工作也提出了若干值得未来研究探讨的问题。目前，提供正确整数分割的责任在于用户。一个问题是如何高效验证整数分割是否正确（通过完整验证或测试方法）。另一个问题是，在什么情况下可以自动推断整数分割。在这两种情况下，工具都能保证假设 3。

未来工作的其他方向包括改进量化布尔公式预处理器和后端量化布尔公式求解器，使其也能利用整数分割信息。此外，通过限制部分变量的界限，可以轻松利用整数分割实现某些形式的对称性约简。最后，如果整个工具链中的所有工具都能识别整数分割，假设 3 就不再必要。在这种情况下，可以将整数分割视为量化布尔公式编码的组成部分，并简化量化布尔公式的矩阵（即从编码中移除界限检查）。

数据可用性

用于对所述软件进行基准测试的数据（包括源代码）已通过 Zenodo 上的公开存档，以宽松许可证永久提供。本文所示图表的原始源数据也一并提供 [6]。

参考文献

1 Olaf Beyersdorff、Mikolás Janota、Florian Lonsing 和 Martina Seidl。量化布尔公式。载于《可满足性手册》，《人工智能与应用前沿》第336卷，第1177–1221页。IOS出版社，2021。doi:10.3233/FAIA201015。2 Armin Biere、Florian Lonsing 和 Martina Seidl。QBF 的阻塞子句消除。载于第23届自动演绎国际会议（CADE）论文集，《计算机科学讲义》第6803卷，第101–115页。Springer，2011。doi:10.1007/978-3-642-22438-6_10。3 Ludovic Le Frioux、Souheib Baarir、Julien Sopena 和 Fabrice Kordon。基于 painless 框架的模块化高效分治SAT求解器。载于《系统构造与分析的工具和算法》——第25届国际会议（TACAS），2019年，捷克共和国布拉格，2019年4月6日至11日，会议论文集第一部分，《计算机科学讲义》第11427卷，第135–151页。Springer，2019。doi:10.1007/978-3-030-17462-0_8。4 Maximilian Heisinger 和 Martina Seidl。可复现的集群（较少）加速分析。RRRR2022，2022。网址：https://qcomp.org/rrrr/2022/。5 Maximilian Heisinger、Martina Seidl 和 Armin Biere。ParaQooba：用于并行与分布式QBF求解的快速灵活框架。载于《系统构造与分析的工具和算法》——第29届国际会议（TACAS 2023），法国巴黎，2023年4月24日至27日，会议论文集第二部分，《计算机科学讲义》。Springer，2023。6 Maximilian Heisinger、Irfansha Shaik、Martina Seidl 和 Jaco van de Pol。论文《利用 Int-Splits 进行搜索空间剪枝以加速 QBF 求解》的制品，2023 年 3 月。doi:10.5281/zenodo.7753526。7 Marijn Heule、Oliver Kullmann、Siert Wieringa 和 Armin Biere。立方与征服：利用前瞻引导CDCL SAT求解器。载于第7届硬件与软件：验证与测试国际会议（HVC）论文集，《计算机科学讲义》第7261卷，第50–65页。Springer，2011。doi:10.1007/978-3-642-34188-5_8。8 Charles Jordan、Lukasz Kaiser、Florian Lonsing 和 Martina Seidl。MPIDepQBF：迈向无需知识共享的并行QBF求解。载于第17届可满足性测试理论与应用国际会议（SAT）论文集，《计算机科学讲义》第8561卷，第430–437页。Springer，2014。doi:10.1007/978-3-319-09284-3_32。9 Markus N. Rabe 和 Leander Tentrup。CAQE：可认证QBF求解器。载于计算机辅助设计中的形式化方法国际会议（FMCAD）论文集，第136–143页。IEEE，2015。10 Jussi Rintanen。规划与SAT。载于《可满足性手册》，《人工智能与应用前沿》第336卷，第765–789页。IOS出版社，2021。doi:10.3233/FAIA201015。11 Irfansha Shaik、Valentin Mayer-Eichberger、Jaco van de Pol 和 Abdallah Saffidine。位置游戏QBF编码中的隐式状态与目标（扩展版），2023。doi:10.48550/ARXIV.2301.07345。12 Irfansha Shaik 和 Jaco van de Pol。无需基化的经典规划QBF表达。载于第三十二届国际自动规划与调度会议（ICAPS 2022）论文集，新加坡（线上），2022年6月13日至24日，第329–337页。AAAI出版社，2022。网址：https://ojs.aaai.org/index.php/ICAPS/article/view/19817。13 Ankit Shukla、Armin Biere、Luca Pulina 和 Martina Seidl。量化布尔公式应用综述。载于第31届IEEE人工智能工具国际会议（ICTAI）论文集，第78–84页。IEEE，2019。doi:10.1109/ICTAI.2019.00020。

23:10 使用整数分割进行搜索空间剪枝，以加快 QBF 求解

我们的 (Q)DIMACS 语法扩展的语法

前导部分和 Int-Split 入口点

〈输入〉::=〈前导部分〉 [〈前缀〉]〈矩阵〉 EOF〈前导部分〉::
=〈注释或分割〉*〈问题行〉〈注释或分割〉::= ‘c’〈文本〉EOL
| ‘cs’〈分割〉 EOL〈问题行〉::= ‘p cnf’〈问题编号〉〈问题编
号〉 EOL

我们的 Int-Split 语法扩展

〈*split*〉::= int [〈*int-split-vars*〉]〈*int-split-list*〉〈
int-split-vars〉::= ‘[’〈*pnum*〉+‘]’〈*int-split-list*〉::=〈
int-split〉 [‘;’〈*int-split-list*〉]〈*int-split*〉::= ‘<’〈*pnum*〉
| ‘>’〈*pnum*〉| ‘=’‘{’〈位模式〉+ ‘}’

可选量词前缀

〈前缀〉::=〈量词集〉+〈量词集〉::=〈量词〉〈正数〉+‘
0’ EOL〈量词〉::= ‘e’ | ‘a’

子句矩阵

〈矩阵〉::=〈子句〉*〈子句〉::=〈文字〉*‘0’
EOL〈 文字 〉::=〈 数字 〉

定义

〈文本〉::= 由非特殊 *ASCII* 字符组成的序列〈数字〉::= 非 *0* 的 *32* 位有符号整数〈正数〉::= 大于 *0* 的 *32* 位有符号整数〈位模式〉::= 长度最多为 *32* 位、由 *0* 和 *1* 组成且不含空格的模式