针对线性实数算术和多线性实数算术的 SMT 局部搜索

Bohan Li，Shaowei Cai *计算机科学国家重点实验室、软件研究所、中国科学院，
计算机科学与技术学院、中国科学院大学中国北京{libh，caisw}@ios.ac.cn

arXiv:2303.06676v2 [cs.LO] 2 Aug 2023

摘要—**理论可满足性（SMT）在各个领域都有着重要应用。本文聚焦于量词自由的实数算术理论可满足性，称为SMT(RA)，其中包括线性和非线性实数算术理论。对于非线性实数算术理论，我们关注其一个重要片段，即原子约束为多线性的片段。我们提出了首个用于 SMT(RA) 的局部搜索算法，称为 LocalSMT(RA)，该算法基于两个新颖思想。首先，我们提出了一种基于区间的算子，通过考虑区间信息与传统局部搜索算子协同工作。此外，当根据评分函数无法区分各项操作时，我们提出了一种平局打破机制，以进一步评估这些操作。我们在来自 SMT-LIB 的基准测试上开展实验，以评估 LocalSMT(RA)。结果表明，LocalSMT(RA) 与当前最先进的SMT求解器具有竞争力，并且在多线性实例上表现尤为出色。**

I. 引言

理论可满足性（SMT）是一个关于判断一阶逻辑公式相对于特定背景理论是否可满足的问题。它已应用于多个领域，包括程序验证和终止性分析 [1], [2], 符号执行 [3] 以及测试用例生成 [4],等。

本文关注无量词实数算术理论，该理论由关于实数变量的多项式等式或不等式形式的原子约束构成。根据算术原子约束是否为线性约束，该理论可分为两类，即线性实数算术（LRA）和非线性实数算术（NRA）。对于NRA，本文研究其一个重要片段，即原子约束为多线性的情形。以LRA和NRA为背景理论的SMT问题，是确定相应原子约束的布尔组合是否可满足，分别称为SMT(LRA)和SMT(NRA)。一般而言，我们将实数算术理论上的SMT问题称为SMT(RA)。

A. 相关工作

求解 SMT(RA) 的主流方法是惰性方法 [5], [6], 也称为 DPLL(T) [7], ，该方法依赖于 SAT 求解器与理论求解器之间的交互。大多数支持实数算术理论的先进 SMT求解器主要基于惰性方法，包括 Z3 [8],Yices2 [9], SMT-RAT [10], cvc5 [11], OpenSMT [12]以及 MathSAT5 [13]。在 DPLL(T) 框架中，SMT公式通过将算术原子约束替换为新的布尔变量，被抽象为布尔公式。SAT 求解器用于处理布尔结构，同时调用理论求解器接收由 SAT 求解器确定的理论约束集合，并求解这些理论约束的合取，包括对赋值进行一致性检查和基于理论的推理。

惰性方法的研究工作主要致力于设计有效的判定过程，将其作为理论求解器来处理理论约束的合取。在 DPLL(T) 中集成的 LRA 核心推理模块，是一种专用于 SMT 求解的 单纯形算法变体，由 [14] 提出。求解 LRA 约束系统的另一种方法是 傅里叶-莫茨金变量消去法 [15], ，其性能通常不如单纯形算法。

对于非线性实数算术，柱状代数分解（CAD） [16] 是应用最广泛的判定过程。CAD 经过改进后被调整并嵌入 SMT-RAT 求解器中，作为理论求解器使用 [10] [17]。其他知名方法使用 Gr¨obner 基 [18]或符号条件的实现 [19]。不完备方法包括基于虚拟代换 [21],的理论求解器 [20] ， 以及 [23], [24] 提出的基于区间约束传播 [22]的技术。

此外，构造满足性（MCSat）演算也是求解 SMT(RA) 的一种高效框架。该方法最初在 [25] 中被提出，用于求解 SMT(LRA) ； 随后，Z3 [26] 在模型构造满足性演算框架中实例化了一种优雅的 CAD 方法，用于求解 SMT(NRA)。

局部搜索是一种不完备方法，在许多组合问题 [27] 中发挥着重要作用。它已成功应用于布尔可满足性（ SAT）问题 [28], [29], [30], [31], [32] ， 并且在某些类型的实例上可以与 CDCL求解器相媲美。

然而，针对 SMT 的局部搜索尚未受到多少关注。此前，人们曾探索将局部搜索求解器与理论求解器相结合来求解 SMT(LRA)：使用局部搜索 SAT 求解器 WalkSAT 求解 SMT公式的布尔骨架 [33]。最近，一种名为 LocalSMT 的局部搜索求解器被用于求解整数算术上的 SMT [34], [35]。此外，局部搜索

https://doi.org/

本文根据知识共享署名 4.0 国际许可协议获得许可

算法已应用于位向量[36], [37], [38]。然而，据我们所知，目前还没有用于实数算术上 SMT 的局部搜索算法。

B. 贡献

本文首次针对 SMT(RA) 设计了一种局部搜索算法，即 LocalSMT(RA)，其基于以下新颖策略。需要注意的是，LocalSMT(RA) 被实现为 LocalSMT [35], 的一个片段；LocalSMT 是我们专用于 SMT 的局部搜索求解器。

首先，我们提出了 基于区间的 算子，通过考虑区间信息来增强传统的局部搜索算子。具体而言，我们观察到，将实值变量赋予给定区间内的任意值，都会使数量相同的当前未满足子句变为满足。因此，基于区间的 算子会评估区间内的多个值，将其作为该操作的候选值，而不是仅将变量赋予一个固定值（例如满足约束的阈值）。

此外，我们观察到，在执行局部搜索时，经常会存在多个具有相同最佳得分的操作，因此提出了一种平局打破机制，以进一步区分这些操作。

我们在 2 个基准测试集上开展实验，以评估 LocalSMT(RA)，即来自 SMT-LIB 的 SMT(LRA) 和 SMT(NRA) 基准测试集。需要注意的是，我们排除了不可满足实例，并且仅考虑 SMT(NRA) 基准测试集中的多线性实例。根据 SMT-COMP 2022[1], ，我们将 LocalSMT(RA) 与相关逻辑（QF LRA、QF NRA）中的前 4 个 SMT求解器进行比较，但不包括组合式求解器和派生求解器。具体而言，对于 SMT(LRA)，我们将 LocalSMT(RA) 与 OpenSMT、Yices2、cvc5 和 Z3 进行比较；对于 SMT(NRA)，竞争者则为 Z3、cvc5、Yices2 和 SMT-RAT。实验结果表明，LocalSMT(RA) 具有竞争力，并且能够与最先进的 SMT求解器形成互补，尤其是在多线性实例上。此外，消融实验验证了我们所提出的新颖策略的有效性。

请注意，多线性实例相较而言难以由现有求解器解决。例如，根据 SMT-COMP 2022，Z3 可能是最擅长解决可满足 SMT(NRA) 实例的求解器，能够解决 90.5% 的 QF(NRA) 实例，而只能解决 77.5% 的多线性实例。然而，多线性实例适合局部搜索，因为没有高阶项时，可以高效地进行运算。

C. 论文组织结构

在第二节中介绍预备知识。在第三节中，我们提出一种新颖的基于区间的算子，通过考虑区间信息来丰富传统算子。在第四节中，提出一种平局打破机制，用于区分多个具有相同最佳得分的操作。基于这两种新颖策略，我们在第五节提出了用于 SMT(RA) 的局部搜索。第六节介绍实验。第七节给出结论和未来工作。

[1]https://smt-comp.github.io/2022/

II. 预备知识

A. 基本定义

$monomial$ 是形如 $x_1^{e_1}...x_m^{e_m}$的表达式，其中$m > 0$、 x_i 是变量，e_i 是指数；对所有$i \in \{1...m\}$， $e_i > 0$ 成立，并且对所有 i, $j \in \{1...m\}$， $i \neq j$， $x_i \neq x_j$成立。若 $m = 1$ 且 $e_1 = 1$，则单项式是线性的。

$polynomial$ 是单项式的线性组合，即形如 $\sum_i a_i m_i$ 的算术表达式，其中 a_i 是系数， m_i 是单项式。如果其所有单项式在多项式中都是线性的，表示 $polynomial$可以写成 $\sum_i a_i x_i$，那么它就是 $linear$，否则就是非线性的。非线性多项式的一种特殊情况是$multi-linear$ 多项式，其中所有变量的最高指数均为 1，表示每个单项式的形式为 $x_1...x_m$。

定义 *1*： 实数算术理论的原子约束是多项式不等式和等式，其形式为 $\sum_i a_i m_i \bowtie k$，其中 $\bowtie \in \{=, \leq, <, \geq, >\}$、 m_i 是由实值变量组成的单项式，k 和 a_i 是有理常数。

实数算术理论上的 SMT问题公式记为 SMT(RA)，是原子约束和命题变量的布尔组合，其中实值变量集和命题变量集分别记为 X 和 P。线性实数算术（LRA）理论和非线性实数算术（NRA）理论上的 SMT问题分别记为 SMT(LRA) 和 SMT(NRA)。对于 NRA，本文仅考虑一个重要的片段，其中原子约束中的多项式是多线性的，本文将其记为*MRA* 。

例 *1*： 设 $X = \{x_1, x_2, x_3, x_4, x_5\}$ 和 $P = \{p_1, p_2\}$ 分别表示整数值变量集和命题变量集。一个典型的 SMT(LRA) 公式 F_{LRA} 和 SMT(MRA) 公式 F_{MRA} 如下所示:

F_{LRA}: $(p_1 \vee (x_1 + 2x_2 \leq 2)) \wedge (p_2 \vee (3x_3 + 4x_4 = 2) \vee (-x_2 - x_3 < 3))$

F_{MRA}: $(p_1 \vee (x_1 x_2 \leq 2)) \wedge (p_2 \vee (3x_3 x_4 + 4x_4 = 2) \vee (-x_2 - x_3 < 3))$

在实数算术理论中，正的无穷小实数记作 δ。

文字是原子约束或命题变量，或者它们的否定。 $clause$ 是一个文字集合的析取，而合取范式 (CNF)公式是一个子句集合的合取。对于 SMT(RA) 公式 F，赋值 α 是一个映射 $X \to R$和 $P \to \{false, true\}$，而 $\alpha(x)$表示变量 x 在 α下的取值。完全赋值 是一个为每个变量都赋予一个值的映射。在给定赋值下，若文字的取值为真，则该文字为真；否则为假。若一个子句至少包含一个真文字，则该子句满足 ；若其中所有文字均为假，则该子句不满足 。当且仅当一个完全赋值满足某个 SMT(RA) 公式的所有子句时，它才是该公式的解 。

B. 局部搜索

当对 SMT问题执行局部搜索时，搜索空间由所有完全赋值构成，每个完全赋值

都代表一个候选解。通常，局部搜索算法从一个完全赋值开始，并通过修改变量的值反复更新该赋值，以寻找一个解。

给定公式 F，赋值 α 的代价记为 $cost(\alpha)$，即在 α 下不满足的子句数量。在使用子句加权技术的动态局部搜索算法中，[39]、[30]、$cost(\alpha)$ 表示赋值 α 下所有不满足子句的总权重（权重根据 PAWS 方案计算，详见第 V-B 节）。

局部搜索算法的一个关键组成部分是算子集合，它们定义了如何修改当前解。当通过指定要操作的变量及要赋予的值来实例化一个算子时，就得到一个操作。将变量 x 赋值为 v 的操作记为 $op(x, v)$。文献 [34] 中针对线性整数算术 SMT 提出的关键移动算子定义如下。

定义 *2*：关键移动算子记为 $cm(x, \ell)$，它将整数变量 x 赋值为使文字 ℓ 为真的阈值，其中 ℓ 是包含 x 的不满足文字。

例如，给定一个不满足的文字 $\ell = (x + 1 \leq 0)$，其中 x 当前赋值为 0，相应的操作 $cm(x, \ell)$ 会将 x 赋值为 -1。

局部搜索算法通常根据某种评分函数，从候选操作中选择一个操作。给定一个公式和一个赋值 α，操作 op 最常用的评分函数定义为

$$score(op) = cost(\alpha') - cost(\alpha)$$

其中，α' 是对 α应用 op 后得到的赋值。若操作 op 满足 $score(op) > 0$，则称其为 $decreasing$。

另一种用于评估操作的属性是其***make*** 值。

定义 *3*：给定一个操作 op，其 make 值记为 $make(op)$，表示执行 op后将变为满足的未满足子句数量。

III. 基于区间的操作

关键移动 通过修改假文字中的一个变量使其变为真，从而满足被违反的子句。该算子仍可用于 SMT(RA) 的上下文中，也用于我们的算法。然而，在实数算术的上下文中应用关键移动算子时，会产生精度问题。回顾一下，我们需要计算使文字变为真的阈值；在求解严格不等式时，并不存在阈值。相反，该值取决于我们希望保持的精度。在本节中，我们针对 SMT(RA) 提出一种算子，该算子考虑区间信息，并且比关键移动更加灵活。

A. 满足域

关于实值变量的线性或多线性不等式，一个重要事实是：当其中除一个变量外的所有变量均已固定时，对于系数不为 0 的剩余变量，存在一个定义域，使得将该变量赋值为该定义域中的任意值都能使不等式成立。因此，给定一个形式为原子约束的假文字 ℓ 以及其中的一个变量 x，只需将 x 赋值为相应定义域中的任意值即可使其满足，该定义域称为满足域。例如，考虑文字 ℓ：$(x - y > 4)$，当前赋值为 $\alpha = \{x = 0, y = 0\}$，那么显然，将 x 赋值为 $(4, +\infty)$ 中的任意值都能使不等式满足，因此满足域 为 $(4, +\infty)$。

我们进一步将满足域 的定义扩展到子句层面，定义如下。

定义 *4*：给定一个赋值 α，对于假文字 ℓ 以及出现在 ℓ 中的变量 x，x **的满足域** ℓ **对于文字** 为 $SD_l(x, \ell)$：将 x 赋给 $v\}$ 后，$= \{v | \ell$ 变为真；对于被违反的子句 c 以及 c 中的变量 x，x **的满足域** c **对于子句** 为 $SD_c(x, c) = \bigcup_{\ell \in c} SD_l(x, \ell)$。

由于假文字 ℓ 采用线性或多线性不等式的形式，$SD_l(x, \ell)$ 的形式必为 $(-\infty, u]$ 或 $[l, \infty)$。因此，作为 $SD_l(x, \ell)$ 的并集，$SD_c(x, c)$ 可能包含 $(-\infty, u]$)，其上界定义为 $UB(x, c) = u$；也可能包含 ($[l, \infty)$，其下界定义为 $LB(x, c) = l$；或者同时包含这两类区间。为简单起见，区间 $(-\infty, u)$ 或 (l, ∞) 分别记作 $(-\infty, u - \delta]$ 或 $[l + \delta, \infty)$。

示例 2：给定子句 $c = \ell_1 \vee \ell_2 \vee \ell_3 = (a - b > 4) \vee (2a - b \geq 7) \vee (2a - c \leq -5)$ 以及当前赋值 $\alpha = \{a = 0, b = 0, c = 0\}$，对于变量 a，三个文字的满足域分别为 $SD_l(a, \ell_1) = [4 + \delta, \infty)$、$SD_l(a, \ell_2) = [3.5, \infty)$ 和 $SD_l(a, \ell_3) = (-\infty, -2.5]$)。子句 c 的满足域为 $SD_c(a, c) = (-\infty, -2.5] \cup [3.5, \infty)$，相应的上界和下界分别为 $UB(a, c) = -2.5$ 和 $LB(a, c) = 3.5$。

B. 等制造区间

根据变量针对子句的满足域，我们观察到，将变量赋值为给定区间内任意值的操作，都会满足相同数量的未满足子句，也就是说，它们具有相同的***make*** 值。我们将这样的区间称为等变化值区间。

定义 *5*：给定一个 SMT(RA)公式 F 及其变量的赋值 α，对于变量 x，**等变化值区间**是一个极大区间 I，使得所有操作 $op(x, v)$（其中$v \in I$）都具有相同的 make 值。

对于某个变量，我们可以将 $(-\infty, +\infty)$ 划分为若干个等变化值区间。

例 *3*：考虑公式 F：$c_1 \wedge c_2$，在当前赋值下，两个子句都不满足，且变量 a 出现在两个子句中。假设 $SD_c(a, c_1) = [3, +\infty)$ 且 $SD_c(a, c_2) = [5, +\infty)$，那么我们可以将 $(-\infty, +\infty)$ 划分为三个区间：$(-\infty, 3)$、$[3, 5)$ 和 $[5, +\infty)$。将 a 赋值为 $(-\infty, 3)$ 内任意值的操作，其 make 值为 0；将 a 赋值为 $[3, 5)$ 内某个值的操作，其 make 值为 1；而对应于 $[5, \infty)$ 的操作，其 make 值为 2。

因此，我们可以通过考虑区间信息来丰富传统的 关键移动 算子。其直观思路是寻找

等制造区间，然后将此类区间中的多个值视为操作未来值的选项，而不是仅考虑阈值。

我们关注至少出现在一个未满足子句中的变量。这里我们描述一种过程，将 $(-\infty, +\infty)$ 划分为这些变量的等变化值区间。

- 首先，我们遍历未满足子句。对于每个未满足子句 c，我们针对其中的每个实值变量 x，计算其对应于 c 的满足域 c $SD_c(x, c)$，以及上界 $UB(x, c)$ 和下界 $LB(x, c)$（如果存在）。
- 然后，对于出现在未满足子句中的每个实值变量 x，将其所有 UBs 按升序排列，同时将 LBs 按降序排列。排序后，这些界分别标记为 $UB^1(x) \ldots UB^n(x)$ 和 $LB^1(x), \ldots LB^m(x)$，其中 $UB^n(x)$ 和 $LB^m(x)$ 分别表示 x 的 UB 的最大值和 LB 的最小值 [2]。为便于描述，我们记 $UB^0(x)$ 为 $= -\infty$，记 $LB^0(x)$ 为 $= \infty$。这些界按以下顺序列出：$UB^0(x) < UB^1(x) < \ldots < UB^n(x) < LB^m(x) < \ldots < LB^1(x) < LB^0(x)$。
- 最后，对于每个变量 x，我们得到一个区间划分 $IP(x) = \bigcup_{0<i\leq n}\{(UB^{i-1}(x)\ UB^i(x)]\} \cup (UB^n, LB^m) \cup \bigcup_{0<j\leq m}\{[LB^j(x)\ LB^{j-1}(x))\}$

形式化地，给定一个实变量 x 和一个来自$IP(x)$的区间 I，$\forall v_1, v_2 \in I$, $make(op(x, v_1)) = make(op(x, v_2))$。稍微滥用一下记号，对于一个来自 $IP(x)$ 的区间 I，我们将其 *make* 值定义为任意满足 $v \in I$ 的操作$op(x, v)$ 的 make 值。注意，$IP(x)$ 中除 (UB^n, LB^m) 外的所有区间的 make 值均为正，后者的 make 值为 0。

示例 4：给定一个公式 F： $c_1 \wedge c_2 = (a - b > 4 \vee 2a - b \geq 7 \vee 2a - c \leq -5) \wedge (a - c \geq 2 \vee a - d \leq -1)$，当前赋值为 $\alpha = \{a = 0, b = 0, c = 0\}$。对于变量 a，$SD_c(a, c_1) = (-\infty, -2.5] \cup [3.5, \infty)$ 且$SD_c(a, c_2) = (-\infty, -1] \cup [2, \infty)$。于是，$UB^0(a) = -\infty$，$UB^1(a) = UB(a, c_1) = -2.5$，$UB^2(a) = UB(a, c_2) = -1$，$LB^2(a) = LB(a, c_2) = 2$，$LB^1(a) = LB(a, c_1) = 3.5$ 以及$LB^0(a) = \infty$。

因此，x 的区间划分为 $IP(x) = I_1 \cup I_2 \cup I_3 \cup I_4 \cup I_5$=$(-\infty, -2.5] \cup (-2.5, -1] \cup (-1,2) \cup [2, 3.5) \cup [3.5, +\infty)$，如图1所示。对于相对于 x 的这些区间，其 make 值依次为 2、1、0、1、2。

C. 操作的候选值

由于将变量 x 赋值为等变化值区间中的任意值，都会满足相同数量的被违反子句，因此，选定一个等变化值区间后，我们可以考虑该区间内更多的值作为操作未来取值的选项，而不局限于阈值。

[2]注意，$UB^n(x) < LB^m(x)$，因为否则当前赋值要么位于区间 $[LB^m(x)\ \infty)$ 中，要么位于 $(-\infty, UB^n(x)]$ 中。假设对应于区间 $[LB^m(x)\ \infty)$ 和 $(-\infty, UB^n(x)]$ 的被违反的子句分别为c_1 和 c_2，如果当前赋值位于第一个或第二个区间中，那么根据定义，c_1 或 c_2 中的一个已经满足，这与 UB^n 和 LB^m的定义相矛盾。

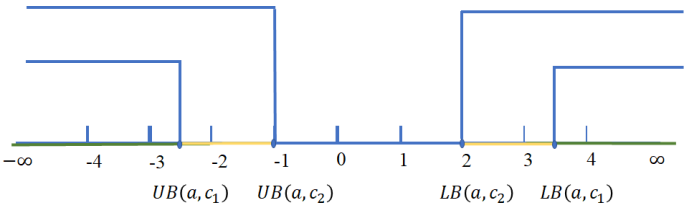


图1：区间示例

候选未来值的动机基于以下三个直觉：首先，我们必须限制给定区间中的值。此外，我们希望相应值的分母相对较小，以避免探索复杂的搜索空间，下一节将对此进行解释。最后，该值应易于计算，以提高算法的效率。

在本研究中，我们只考虑 make 值为正的区间，因此省略区间 (UB^n, LB^m)。因此，待考虑的区间形式为 $(UB^{i-1}(x)\ UB^i(x)]$ 或 $[LB^i(x)\ LB^{i-1}(x))$。对于这样的区间，我们考虑以下操作取值：

- 将 x 赋值为阈值 $UB^i(x)$ 或 $LB^i(x)$。
- 将 x 赋值为区间的中位数，即 $(UB^{i-1}(x) + UB^i(x))/2$ 或 $(LB^i(x) + LB^{i-1}(x))/2$（当其中一个边界为 ∞ 或 $-\infty$时，将不考虑中位数）。
- 如果区间 $(UB^{i-1}(x)\ UB^i(x))$ 或区间 $(LB^i(x), LB^{i-1}(x))$ 中存在整数，则将 x 赋值为相应开区间中的最大或最小整数；否则，假设该开区间可以写成 $(\frac{a}{b}, \frac{c}{d})$，则将 x赋值为 $\frac{a+c}{b+d}$。

第一种选择与关键移动相同，因此关键移动 可以被视为我们基于区间的运算符的特例。第二种选择是在区间中进行简单选择。第三种选择旨在寻找一个受限于开区间且分母较小的有理数值，并且易于计算。

请注意，基于以下原因，计算导致评分最大局部下降的操作十分困难，甚至不切实际：一个操作由要修改的变量以及要赋予的未来值组成。然而，变量可以被赋予任意实数，这是无穷尽的。尽管可以将其缩减为可枚举区间，但枚举所有可能的操作仍然过于耗时。此外，一个文字 ℓ 包含不同的变量，修改其中一个变量会影响所有其他包含该变量的文字（不仅是 ℓ）。因此，要计算某个操作的评分，我们需要遍历所有包含待修改变量的文字，这十分耗时。这两个原因使得计算出导致评分最大局部下降的操作成为一个非常耗时的过程。

因此，我们的算法不会尝试计算此类运算，而是从基于这些候选值采样得到的候选运算中选择得分较高的运算。在

未来的工作中，我们将通过考虑区间内更多分母较小的随机值，进一步丰富采样得到的候选值。

IV. 一种平局打破机制

我们注意到，在局部搜索过程中，通常会存在多个操作具有相同的最优 $score$ ，因此，平局打破对于引导搜索同样十分重要。

为验证我们的观察结果，我们在100个随机选取的实例上进行了预实验。对于每个实例，我们执行一个简单的局部搜索算法，在10000次迭代中选择具有最优 $score$ 的操作，并统计找到具有相同最优 $score$的 k 个操作的步骤数，记为 $step(k)$。

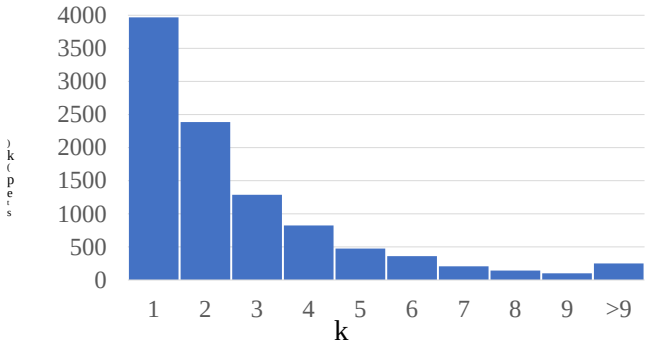


图2：平均 $step(k)$分布

如图2所示，具有相同最优 $score$ 且操作数超过一个的步骤占总步骤数的61.2%。因此，需要采用破平启发式来进一步区分这些具有相同最优 $score$的操作。

首先，我们认为，将实值变量赋予分母较大的值，可能会使算法进入更复杂的搜索空间，其中变量被赋予具有极大分母的实数，从而导致更复杂的计算并可能产生错误。这是因为在迭代过程中，局部搜索求解器需要对大数进行因式分解并进行数值近似，以处理这些值。因此，我们更倾向于选择将变量赋予分母较小的值的操作。

此外，我们认为，将变量赋为绝对值较大的数可能导致赋值结果异常庞大，从而使算法偏离寻找可行解的目标。因此，我们倾向于执行将变量赋为绝对值较小数值的操作。

基于上述观察和直觉，我们提出了一种操作选择规则，具体如下。

选择规则：选择具有最大$score$的操作；若出现并列，则优先选择将相应变量赋值为分母最小的值的操作。若仍然并列，则选择将绝对值最小的变量进行赋值的操作。

V. LOCALSMT(RA) 算法

我们的局部搜索算法采用双模式框架，在实数模式和布尔模式之间切换。该框架已用于整数算术理论的局部搜索算法 LS-LIA [34]。

A. 局部搜索框架

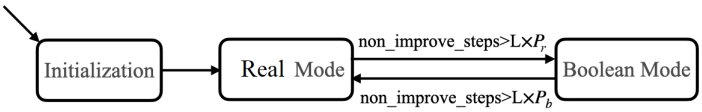


图3：一种 SMT 局部搜索框架

LocalSMT(RA) 从一个任意模型开始运行局部搜索算法，其中所有实值变量均赋值为 0，所有布尔变量均赋值为 false。如图3所示，初始化后，算法在实数模式和布尔模式之间切换。在每种模式下，都会选择一个对应数据类型变量的操作来修改当前解。当当前模式的非改进步数（记为$non\ improve\ steps$）达到阈值时，这两种模式便会相互切换。算法未能找到更优解时，$non_improve_steps$加 1；否则将其重置为 0。布尔模式和实数模式的阈值分别设为 $L \times P_b$ 和 $L \times P_r$，其中 P_b 和 P_r 分别表示未满足子句中布尔文字和实变量文字占所有文字的比例， L为一个参数。

B. 实数和布尔模式下的局部搜索

LocalSMT(RA) 的实数模式算法如算法1所述：如果当前赋值 α 满足给定公式 F，则找到解（第2行）。该算法根据**选择规则**尝试寻找一个下降的 基于区间的操作（第3–4行）。

如果不存在这样的下降操作，则表明算法陷入了局部最优。我们首先根据纯加法加权方案（PAWS）的概率版本 [39][30] 更新子句权重（第6行），然后将 K 个基于区间的操作随机采样到集合 Set_{op} 中（第7行），其中 K 是一个相对较小的参数。根据**选择规则**，从 Set_{op} 中选取最佳操作（第8行）。注意，由于基于区间的 操作至少可以满足一个子句，因此从少量随机采样的 基于区间的操作中选取最佳操作，可以视为一种多样化操作。

PAWS方案的概率版本运行如下。开始时，每个子句的权重都初始化为 1。在局部搜索过程中，以 $1-sp$ 的概率将每个被违反的子句的权重增加 1；以 sp 的概率，将每个权重大于 1 的满足子句的权重减少 1。

对于聚焦于由布尔变量组成的子公式的布尔模式，LocalSMT(RA) 的工作方式与 LS-LIA 的布尔模式相同。通过将

算法1： LocalSMT(RA)的实数模式

输入： 公式 F

输出： F的满足赋值 α，或“未知”

```
当 非改进步数 ≤ L × P_r 时执行
    如果 α 满足 F，则 返回 α;
    如果 ∃ 下降型基于区间的操作那么
        op: = 根据以下内容选择此类操作
          选择规则;
    else
        根据 PAWS方案更新子句权重方案;
        随机抽样 K 个基于区间的操作进入集合 Set_op;
        op:= 在 Set_op 中选出的最佳操作
          通过选择规则;
    α := α 执行了 op ;
返回“未知”;
```

布尔模式与实数模式结合起来，我们提出了用于 SMT(RA) 的局部搜索算法，记作 LocalSMT(RA)。

VI. 实验

我们开展了实验，以评估 LocalSMT(RA) 在 SMT-LIB 的 2 组基准测试上的表现，并将其与最先进的 SMT 求解器和局部搜索求解器进行比较。此外，我们还进行了消融实验，以分析所提策略的有效性。

A. 实验准备

实现： LocalSMT(RA) 使用 C++ 实现，并通过 g++ 编译，采用“-O3”选项。LS-LRA 有 3 个参数： L 用于切换阶段， K 用于设置采样操作的数量， sp （平滑概率）用于 PAWS 方案。参数根据文献 [34], [30] 中的建议以及我们对 20% 抽样实例进行的初步实验进行调优。初步实验表明，在相当大的范围内，LocalSMT(RA) 对参数设置并不敏感。所有基准测试均采用以下参数： $L = 20$、 $K = 3$、 $sp = 0.0003$ 。在我们的实现中，为避免陷入局部最优，LocalSMT(RA) 每 500000 次迭代重启一次。我们将无穷小实数 δ 固定为min（$1\,256$， $\frac{1}{c_{max}}$），其中 c_{max} 表示公式中系数的最大绝对值。

请注意，与此前针对线性整数算术 SMT 的局部搜索方法相比， [34],我们的求解器能够处理公式的任意布尔结构，包括“ite”运算符，其实现采用了 Tseitin编码 [40]。

尽管任何含有高次幂的公式都可以通过引入新变量重写为多线性公式（例如， $x^2 > 4$ 可以重写为（$x = y$） $\wedge$ （$x * y > 4$）），但在这种情况下，由于二者之间的关系，我们无法高效地找到 x 的正确可行解变量。在我们的实现中，我们通过消除形如 $x = (a * y)$的方程来简化公式，其中 x 和 y是实值变量，而 a 是系数。具体而言，我们会将 x 替换为（$a * y$），简化后，仅保留仍然是多线性的实例。

竞争者： 在 SMT(LRA) 的背景下，我们将 LocalSMT(RA) 与根据 SMT-COMP 2022 评选出的排名前 4 位的先进 SMT求解器进行比较，分别为 OpenSMT（版本 2.4.2）[3]、Yices2（版本 2.6.2）[4]、cvc5（版本 1.0.0）[5]和 Z3（版本 4.8.17） [6]。而在 SMT(NRA) 的背景下，排名前 4 位的竞争者如下：cvc5（版本 1.0.0）、Yices2（版本 2.6.2）、Z3（版本 4.8.17）和 SMT-RAT-MCSAT（版本 22.06） [7]。所有竞争者的二进制文件均从其官方网站下载。请注意，组合求解器和派生求解器不在比较范围内。

基准测试： 实验基于 SMT-LIB 中的两组基准测试进行。

- SMTLIB-LRA：该基准测试集包含来自 SMT-LIB[8] 的 SMT(LRA) 实例。由于 LocalSMT(RA) 是一个不完备求解器，因此排除不可满足实例，最终得到包含 1044 个未知和可满足实例的基准测试集。
- SMTLIB-MRA：该基准测试集包含来自 SMT-LIB[9] 的、具有多线性原子约束的 SMT(NRA) 实例。同样排除不可满足实例，最终得到包含 979 个未知和可满足实例的基准测试集。

实验设置： 所有实验均在一台配备 AMD EPYC 7763 64 核 2.45GHz 处理器和 2048G 内存、运行 Ubuntu 20.04.4 系统的服务器上进行。对于两组基准测试中的每个实例，每个求解器均执行一次，截止时间设为 1200 秒（与 SMT-COMP 中相同）；两组基准测试包含了足够多的实例（SMTLIB-LRA 为 1044 个，SMTLIB-MRA 为 979 个）。我们比较算法找到模型的实例数量（“#solved”）以及运行时间。表格中的粗体数值表示“#solved”最多的求解器。

B. SMTLIB-LRA 基准测试结果

1）与 *DPLL(T)*求解器的比较：如表 I 所示，LocalSMT(RA) 可以求解 1044 个实例中的 900 个，具有竞争力，但仍无法与其竞争者相媲美。我们还在图 4 中对 LocalSMT(RA) 与各竞争者在 SMTLIB-LRA 实例上的运行时间进行了比较，结果表明 LocalSMT(RA) 与这些竞争者具有互补性。

LocalSMT(RA) 无法与其 DPLL(T) 竞争者匹敌的一个解释是，SMTLIB-LRA 中 54.5% 的实例包含布尔变量，而 LocalSMT(RA) 的布尔模式不擅长利用布尔变量之间的关系

[3]https://github.com/usi-verification-and-security/opensmt [4] https://yices.csl.sri.com [5]https://cvc5.github.io/ [6] https://github.com/Z3Prover/z3/ [7]https://github.com/ths-rwth/smtrat [8] https://clc-gitlab.cs.uiowa.edu:2443/SMT-LIB-benchmarks/QF LRA [9] https://clc-gitlab.cs.uiowa.edu:2443/SMT-LIB-benchmarks/QF NRA

表 I：SMTLIB-LRA 实例的结果

	实例数	cvc5	Yices	Z3	OpenSMT	LocalSMT(RA)
2017-Heizmann	8	4	3	4	4	**7**
2019-ezsmt	84	61	61	53	**62**	35
检查	1	1	1	1	1	1
DTP-调度	91	91	91	91	91	91
LassoRanker	271	232	**265**	256	262	240
latendresse	16	9	**12**	1	10	0
Meti-Tarski	338	338	338	338	338	338
MIPLIB	22	14	**15**	**15**	**15**	4
sal	11	11	11	11	11	11
sc	108	108	108	108	108	108
TM	24	**24**	**24**	**24**	**24**	11
热带矩阵	10	1	**6**	4	**6**	0
tta	24	24	24	24	24	24
uart	36	**36**	**36**	**36**	**36**	30
总计	1044	954	**995**	966	992	900

表 II：与来自 SMTLIB-LRA 实例上的 WalkSMT 的比较

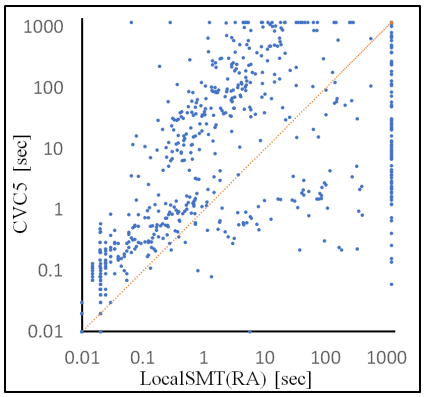
	sc	uart	sal	TM	tta	MIPLIB	总计
#实例	108	36	11	24	24	22	225
WalkSMT UBCSAT	78	**35**	**10**	14	9	**3**	149
WalkSMT UBCSAT++	63	14	8	**19**	10	2	116
LocalSMT(RA)	**106**	29	8	9	**24**	**3**	**179**

这与此前针对线性整数算术中的 SMT 的局部搜索类似 [34]。LocalSMT(RA) 在这组基准测试中表现不佳的另一个可能原因是，我们的算法不具备概率近似完备性（PAC）意义上的完备性，因为基于区间的操作的候选值选择可能会遗漏满足解，即没有考虑变量取值的情况。

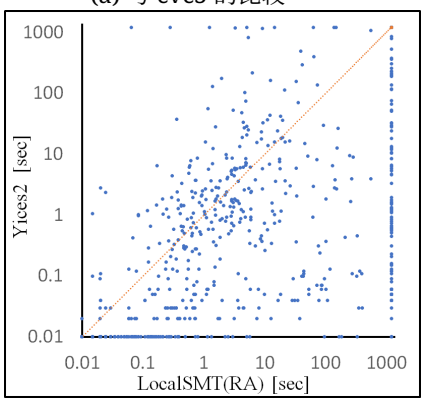
2）与局部搜索求解器的比较：我们将 LocalSMT(RA) 与先前专用于 SMT(LRA) [33], 的局部搜索求解器 WalkSMT 进行比较。我们采用 WalkSMT 的两个最佳配置，即“WalkSMT UBCSAT”和“WalkSMT UBCSAT++”进行比较。由于 WalkSMT 仅支持较早版本的 SMT-LIB 标准，而该版本现已弃用，因此我们使用其论文 [33],中的相同实验设置开展实验，其中截止时间设为 600 秒，并将结果与原论文中给出的结果进行比较。结果如表 II 所示，表明 LocalSMT(RA) 能够显著优于 WalkSMT 的两种配置，尤其是在“SC”类型上。

C. SMTLIB-MRA 基准测试结果

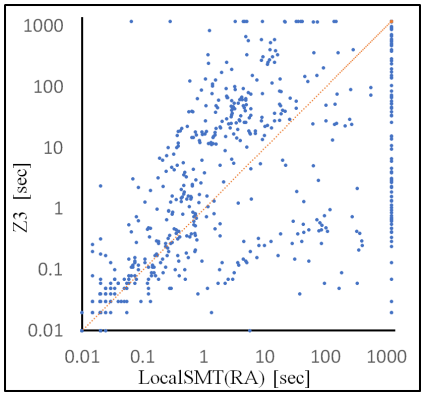
在该基准测试集上，LocalSMT(RA) 能够解决比所有竞争者更多的多线性实例（979 个实例中解决了 891 个），如表 III 所示。此外，LocalSMT(RA) 还能在该基准测试集中独自解决 28 个实例。图 5 展示了 LocalSMT(RA) 与其竞争者之间的时间对比，证实 LocalSMT(RA) 在 SMTLIB-MRA 中比所有竞争者更高效。如表 IV 所示，LocalSMT(RA) 在较小截止时间下表现更佳。具体而言，Lo-



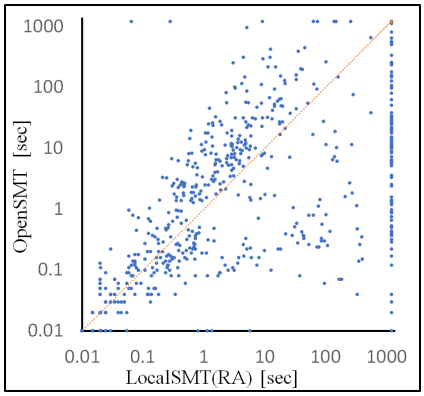
(a) 与 cvc5 的比较



(b) 与 Yices2 的比较



(c) 与 Z3 的比较



(d) 与 OpenSMT 的比较

图 4：SMTLIB-LRA 实例的运行时间比较

表 III：SMTLIB-MRA 中实例的结果

	实例数	cvc5	Yices	Z3	SMT-RAT	LocalSMT(RA)
20170501-Heizmann	51	1	0	4	0	**17**
20180501-Economics	28	28	28	28	28	28
2019-ezsmt	32	31	**32**	**32**	21	28
20220314-Uncu	12	12	12	12	12	12
LassoRanker	347	**312**	124	199	0	297
Meti-Tarski	423	423	423	423	423	423
UltimateAutomizer	48	34	39	46	18	**48**
zankl	38	24	25	28	30	**38**
总计	979	865	683	772	532	**891**

表 IV：SMTLIB-MRA 实例在不同截止时间下的结果

截止时间	cvc5	Yices	Z3	SMT-RAT	LocalSMT(RA)
1s	535	567	595	495	**733**
5s	607	609	674	507	**796**
10s	649	623	714	508	**813**

calSMT(RA) 在截止时间分别为 1 秒、5 秒和 10 秒时，比表现最佳的竞争者多解决了 138 个、122 个和 99 个实例。

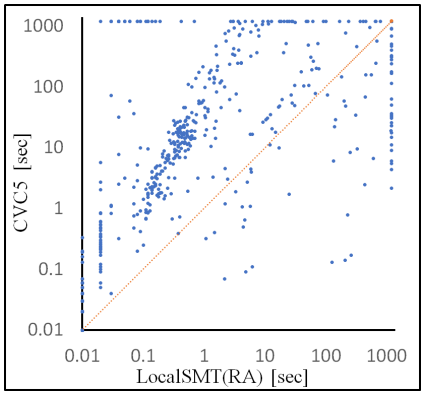
LocalSMT(RA) 在“zankl”和“UltimateAutomizer”类型的实例上表现尤为出色，这些工业实例是在软件验证的背景下生成的。在这些类型的实例上，LocalSMT(RA) 可以解决所有实例，性能超过所有竞争者。此外，LocalSMT(RA) 还能独家解决“20170501-Heizmann”类型中的 13 个实例，该类型实现了不变量的基于约束的综合[41]。

LocalSMT(RA) 在 SMTLIB-MRA 上具有优势的原因如下。与 LRA 不同，NRA约束的理论求解器需要进行复杂计算，这会降低这些竞争者的性能；而 LocalSMT(RA) 可以轻松确定多线性约束中的运算，从而能够高效地探索搜索空间。此外，在 SMTLIB-MRA 基准测试中，3.4% 的实例包含布尔变量。相比之下，SMTLIB-LRA 中对应的比例为 54.5%，SMTLIB-MRA 具有更简单的布尔结构。

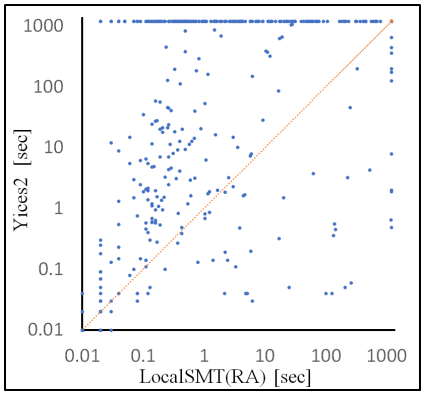
D. 所提策略的有效性

为分析 Lo-calSMT(RA) 中各策略的有效性，我们修改 LocalSMT(RA)，得到如下 3 个替代版本。

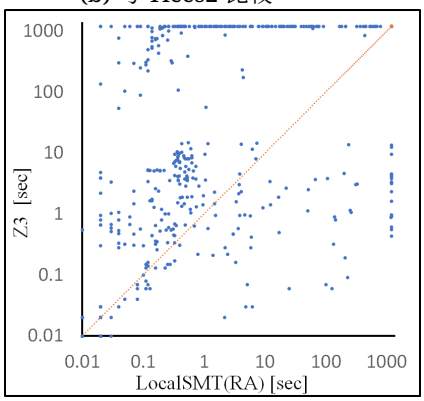
- 为分析基于区间的运算符的有效性，我们通过将该运算符替换为传统的 cm 运算符来修改 LocalSMT(RA)，从而得到版本 v_cm。
- 为分析平局打破机制的有效性，我们对 LocalSMT(RA) 进行修改：基于 $score$ 评估操作，而不考虑**选择规则**，即随机选择具有最大 $score$ 的操作，从而得到版本 v_score。



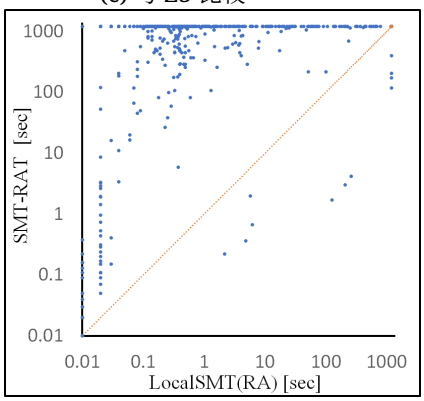
(a) 与 cvc5 比较



(b) 与 Yices2 比较

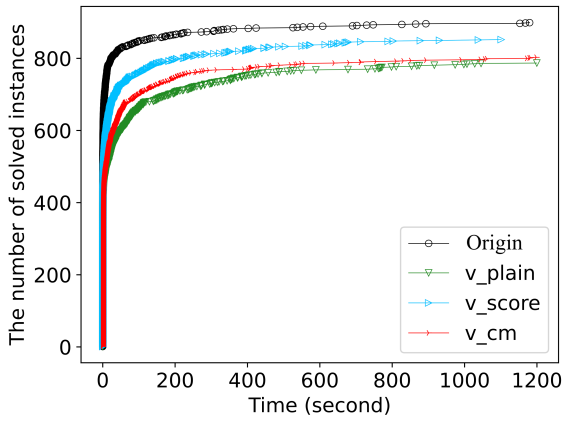


(c) 与 Z3 比较

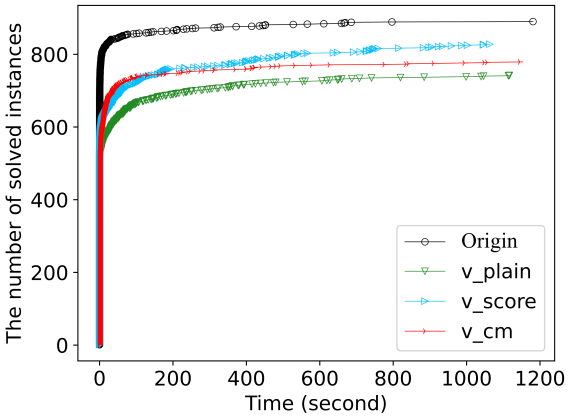


（d）与 SMT-RAT 比较

图 5：SMTLIB-MRA 实例的运行时间比较



(a) SMTLIB-LRA 上的比较



(b) SMTLIB-MRA 上的比较

图 6：运行时间分布比较

- 我们还实现了一个普通版本，该版本既不采用基于区间的运算符，也不采用平局打破机制，记为 v_plain。

原始版本的 LocalSMT(RA)（记为 $Origin$）与这些修改后的版本在两类基准测试上进行了比较。LocalSMT(RA) 及其修改版本的运行时间分布如图 6 所示，这证实了我们所提出策略的有效性。

VII. 结论与未来工作

本文基于以下新颖思想，提出了首个用于实数算术上的 SMT 的局部搜索算法。首先，通过考虑区间信息，提出了基于区间的 操作，以丰富传统的关键移动算子。此外，还提出了一种平局打破 机制，用于区分具有相同最佳 $score$ 的操作。在 SMT-LIB 上进行的实验表明，我们的求解器具有竞争力，并且能够与最先进的 SMT求解器形成互补，尤其适用于多线性实例。

未来的研究方向是将 LocalSMT(RA) 扩展至支持所有 SMT(NRA) 实例，并将我们的局部搜索算法与 DPLL(T)SMT求解器进行深度结合，从而形成一种能够充分发挥各自优势的混合求解器。此外，我们将通过考虑更多分母较小的随机值，丰富基于区间的运算所采样的候选值。

VIII. 致谢

本工作得到国家自然科学基金项目62122078的资助。

参考文献

[1] S. Lahiri 和 S. Qadeer，“回到未来：使用 SMT求解器重新审视精确的程序验证”，*ACM SIGPLAN Notices*，第 43 卷第 1 期，第 171–182 页，2008 年。[2] N. S. Bjørner、K. L. McMillan 和 A. Rybalchenko，“程序验证作为理论可满足性。”*SMT@ IJCAR*，第 20 卷，11，2012 年 3 页。–。[3] R. Baldoni、E. Coppa、D. C. D’elia、C. Demetrescu 和 I. Finocchi，“符号执行技术综述”，*ACM Computing Surveys (CSUR)*，第 51 卷第 3 期，第 1–39 页，2018 年。[4] J. Peleska、E. Vorobev 和 F. Lapschies，“结合 SMT求解和抽象解释的自动化测试用例生成”，载于 *NASA Formal Methods:Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings 3*。Springer，2011 年，第 298–312 页。[5] R. Sebastiani，“惰性理论可满足性”，*JSAT*，第 3 卷第 3-4 期，第 141–224 页，2007 年。[6] C. Barrett 和 C. Tinelli，“理论可满足性”，载于 *Handbook of model checking*。Springer，2018 年，第 305–343 页。[7] R. Nieuwenhuis、A. Oliveras 和 C. Tinelli，“求解 SAT 与理论模可满足性：从抽象的 Davis–Putnam–Logemann–Loveland过程到 DPLL (T)”，*Journal of the ACM*，第 53 卷第 6 期，第 937–977 页，2006 年。[8] L. De Moura 和 N. Bjørner，“Z3：一种高效的 SMT求解器”，载于 *ToolsandAlgorithmsfortheConstruction andAnalysisof Systems:14thInternational Conference, TACAS2008, Held as Partof the JointEuropean Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings 14*。Springer，2008 年，第 337–340 页。[9] B. Dutertre，“Yices 2.2”，载于*Computer Aided Verification: 26th Interna-tional Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings 26*。Springer，2014 年，第 737–744 页。[10] F. Corzilius、U. Loup、S. Junges 和 E. ´Abrah´am，“SMT-RAT：符合 SMT 标准的非线性实数算术工具箱：（工具介绍）”，载于 *Theory and Applications of Satisfiability Testing–SAT 2012: 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings 15*。Springer，2012 年，第 442–448 页。[11] H. Barbosa、C. Barrett、M. Brain、G. Kremer、H. Lachnitt、M. Mann、A. Mohamed、M. Mohamed、A. Niemetz、A. N¨otzli *et al.*，“cvc5：一种多用途且具备工业级性能的 SMT求解器”，载于 *Tools and Algorithms fortheConstruction andAnalysis ofSystems: 28th International Conference,TACAS 2022,Heldas Partof the European Joint Conferences on Theoryand Practice of Software, ETAPS 2022, 7, 2022,* 22022，415 442*Munich, Germany, April –Proceedings, Part I*。Springer，第 – 页。[12] R. Bruttomesso、E. Pek、N. Sharygina 和 A. Tsitovich，“OpenSMT 求解器。”载于 *TACAS*，第 6015 卷。Springer，2010 年，第 150–153 页。[13] A. Cimatti、A. Griggio、B. J. Schaafsma 和 R. Sebastiani，“MathSAT5 SMT 求解器”，载于 *Tools and Algorithms for the ConstructionandAnalysis ofSystems: 19th International Conference, TACAS 2013, Heldas Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings 19*。Springer，2013 年，第 93–107 页。[14] B. Dutertre 和 L. De Moura，“用于 DPLL (T) 的快速线性算术求解器”，载于 *Computer Aided Verification: 18thInternational Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006. Proceedings 18*。Springer，2006 年，第 81–94 页。[15] N. Bjørner，“线性量词消去作为一种抽象判定过程”，载于 *Automated Reasoning: 5th International Joint Conference, IJ- CAR 2010, Edinburgh, UK, July 16-19, 2010. Proceedings 5*。Springer，2010 年，第 316–330 页。[16] G. E. Collins，“通过柱状代数分解对实闭域进行量词消去”，载于 *Automata Theory and Formal Languages: 2ndGI Conference Kaiserslautern, May 20–23, 1975*。Springer，1975 年，第 134–183 页。

[17] U. Loup、K. Scheibler、F. Corzilius、E. ´Abrah´am 和 B. Becker，“区间约束传播与柱状代数分解的共生”，载于 自动演绎—CADE-24：第 24 届自动演绎国际会议，美国纽约州普莱西德湖，2013 年 6 月 9 日至 14 日 24 2013，193 207 - 。论文集。Springer，第 – 页。[18] S. Junges、U. Loup、F. Corzilius 和 E. ´Abrah´am，“关于实数上的理论模可满足性求解中的 Gr¨obner 基”，载于 代数信息学：第 5 届国际会议，CAI 2013，法国波克罗勒岛，2013 年 9 月 3 日至 6 日。论文集 5。Springer，2013 年，第 186–198 页。[19] S. Basu，“实代数几何中的算法：综述”，arXiv 预印本 arXiv:1409.1534，2014 年。[20] F. Corzilius 和 E. ´Abrah´am，“ 用于 SMT求解的虚拟代换”，载于 计算理论基础：第 18 届国际研讨会，FCT 2011，挪威奥斯陆，2011 年 8 月 22 日至 25 日。论文集 18。Springer，2011 年，第 360–371 页。[21] V. Weispfenning，“实代数的量词消去——二次情形及其推广”，《工程、通信与计算中的应用代数》，第 8 卷，第 85–101 页，1997 年。[22] P. Van Hentenryck、D. McAllester 和 D. Kapur，“使用分支剪枝方法求解多项式系统”，SIAM 数值分析杂志，第 34 卷第 2 期，第 797–827 页，1997 年。[23] S. Gao、S. Kong 和 E. M. Clarke，“dreal：一种用于实数上非线性理论的 SMT求解器”，载于 自动演绎—CADE-24：第 24 届自动演绎国际会议，美国纽约州普莱西德湖，2013 年 6 月 9 日至 14 日。论文集 24。Springer，2013 年，第 208–214 页。[24] S. Schupp、E. ´Abrah´am、P. Rossmanith 和 D.-I. U. Loup，“SMT 兼容判定过程中的区间约束传播”，硕士论文，亚琛工业大学，2013 年。[25] D. Jovanovic、C. Barrett 和 L. De Moura，“模型构造可满足性演算的设计与实现”，载于 2013 年计算机辅助设计中的形式化方法。电气电子工程师学会，2013 年，第 173–180 页。[26] D. Jovanovi´c 和 L. De Moura，“求解非线性算术”，《ACM计算机代数通讯》，第 46 卷，105，2013 年 3 4，104，编号 / 页码 –。[27] H. H. Hoos 和 T. St¨utzle，《随机局部搜索：基础与应用》，2004 年。[28] C. M. Li 和 Y. Li，“可满足性局部搜索中的满足与证伪”，载于 SAT 2012 论文集，2012 年，第 477–478 页。[29] A. Balint 和 U. Sch¨oning，“为随机局部搜索选择概率分布以及增加与删除的作用”，载于 SAT 2012 论文集，2012 年，第 16–29 页。[30] S. Cai 和 K. Su，“带配置检查和子分数的布尔可满足性局部搜索”，《人工智能》，第 204 卷，98，2013 年 75 页。–。[31] S. Cai、C. Luo 和 K. Su，“CCAnr：一种基于配置检查的非随机可满足性局部搜索求解器”，载于 SAT 2015 论文集，2015 年，第 1–8 页。–。[32] A. Biere，“Splatz、Lingeling、Plingeling、Treengeling、YalSAT 参加 SAT竞赛2016，45，2016 2016，44”，SAT竞赛论文集，第 – 页。[33] A. Griggio、Q.-S. Phan、R. Sebastiani 和 S. Tomasi，“用于 SMT 的随机局部搜索：结合理论求解器与 WalkSAT”，载于 组合系统前沿国际研讨会。Springer，2011 年，第 163–178 页。[34] S. Cai、B. Li 和 X. Zhang，“在线性整数算术上的局部搜索”，载于 国际计算机辅助验证会议。Springer，2022 年，第 227–248 页。[35] ——，“整数算术理论模可满足性的局部搜索”，《ACM计算逻辑汇刊》，2022 年。[36] A. Niemetz、M. Preiner 和 A. Biere，“用于位精确推理的基于传播的局部搜索”，《Formal Methods Syst. Des.》，第 51 卷第 3 期，第 608–636 页，2017 年。 [在线]。可用地址：https: //doi.org/ 10 1007 10703 017 0295 6 。 /s - - - A. Niemetz 和 M. Preiner，“用于更精确位推理的基于三值传播的局部搜索”，载于 2020 年计算机辅助设计中的形式化方法，FMCAD 2020，以色列海法，2020 年 9 月 21 日至 24 日。电气电子工程师学会，2020 年，第 214–224 页。 [在线]。可用地址：https://doi.org/ 10 34727 2020 978 3 85448 042 6 29 。 / /isbn。- - - - [38] A. Frohlich 、 A. Biere 、 C. M. Wintersteiger 和 Y. Hamadi，“ 理论模可满足性的随机局部搜索”，载于第二十九届 AAAI人工智能会议论文集，2015 年 1 月 25 日至 30 日，美国得克萨斯州奥斯汀，美国，B. Bonet 和 S. Koenig，编者。AAAI出版社，2015年，第1136–1143页。 [在线]。获取地址：http://www.aaai.org/ocs/index.php/AAAI/AAAI 15 9896 /paper/view/ J. [39] Thornton、D. N. Pham、S. Bain 和 V. F. Jr.，《SAT 的加法与乘法子句加权》，载于*2004*年*AAAI*会议论文集，2004年，第191–196页。[40] S. D. Prestwich，《合取范式编码》。《可满足性手册》，第185卷，第75–97页，2009年。[41] M. A. Col´on、S. Sankaranarayanan 和 H. B. Sipma，《使用非线性约束求解进行线性不变量生成》，载于国际计算机辅助验证会议。Springer，2003年，第420–432页。