中国科学院大学

University of Chinese Academy of Sciences

硕士学位论文

非线性实数可满足性问题的局部搜索算法

作者姓名：王忠汉

指导教师：张立军 研究员 中国科学院软件研究所

学位类别：工学硕士

学科专业：计算机科学与技术

培养单位：中国科学院软件研究所

2025 年 6 月

Local Search Algorithm for Nonlinear Real Satisfiability

A thesis submitted to

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Master of Engineering

in Computer Science and Technology

By

WANG Zhonghan

Supervisor: Professor ZHANG Lijun

Institute of Software, Chinese Academy of Sciences

June, 2025

中国科学院大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明或致谢。

作者签名：

日　　期：

中国科学院大学
学位论文授权使用声明

本人完全了解并同意遵守中国科学院有关保存和使用学位论文的规定，即中国科学院有权保留送交学位论文的副本，允许该论文被查阅，可以按照学术研究公开原则和保护知识产权的原则公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名：　　导师签名：

日　　期：　　日　　期：

摘 要

SMT 问题是形式化方法与软件工程领域涉及到的一类重要问题。相比于 SAT 问题（布尔可满足问题），SMT 问题可以看成是一阶逻辑的扩展，即给定特定理论下的约束，找到满足所有约束的一组解，或者证明不存在这样的解存在。SMT 问题被广泛应用于软硬件验证、程序分析、自动化推理等领域。SMT 问题视理论的不同可以分为不同理论，其中 NRA（非线性实数理论）指的是变量可以取实数值，约束可以是高次非线性多项式的一种理论。针对这类问题，目前的主流求解器 Z3、CVC5、Yices 等都提供了有效的求解方法。常见的搜索算法包括 CDCL（T）、MCSAT、增量线性化、变量替换以及基于区间算术的搜索算法等。然而，优于 NRA 问题解空间的复杂性以及高次多项式运算的复杂性，这类问题的求解仍然具有挑战性。

本文主要深入探讨 NRA 理论的解空间和求解上的难点，并提出用于求解此类复杂约束的高效的局部搜索算法。本文首先介绍 NRA 理论的解空间，并给出一些基本概念的定义，比如胞腔的划分等。接下来，本文提出目前的系统求解算法与局部搜索算法，其中的一些局部搜索算法并没有完全支持全部的 NRA 形式，并且在求解效率上也有一定的局限性。本工作基于此设计了基于边界的分数缓存机制，一方面可以减少搜索过程中的候选操作，另一方面可以有效地降低每次迭代的计算复杂度，并给出了实时更新的条件与具体实施算法。除此之外，NRA 问题具有不同于其他理论的无理数赋值特性，使得局部搜索的迭代效率极低。本文提出了等式松弛策略，可以在一定程度上延后无理数赋值，使用近似解暂时替代精确解。最后，我们也讨论了 NRA 问题独有的无单变量操作的问题，并给出多个变量移动的一种迭代策略。整体的算法实现还包括重启策略，可以在陷入局部最优的情况下跳出当前搜索空间并及时调整搜索区域。

根据上述算法，本文实现了一个名为 LS_NRA 的 SMT 求解工具，可以支持 NRA 理论的任何形式样例。我们在 SMT-LIB 上测试了工具的求解效果，包括一些来自程序验证、自动机理论以及生物网络的使用场景样例。实现结果表明，我们的求解算法和一些完备算法的求解相比具有竞争力，并在高次样例上表现非常好，打破了以往求解器求解个数零的记录。我们还探讨了不同求解器求解单个样例所需的求解时间，我们的算法在时间上可以媲美主流的求解器，可以解决 NRA、理论的实际问题。

关键词： SMT，非线性实数理论，局部搜索

Abstract

SMT problem is an important problem involved in the field of formal methods and software engineering. Compared with the SAT problem (boolean satisfiability problem), the SMT problem can be regarded as an extension of first-order logic, that is, given the constraints under a specific theory, find a set of solutions that satisfy all constraints, or prove that no such solution exists. SMT problems are widely used in software and hardware verification, program analysis, automated reasoning and other fields. SMT problems can be divided into different theories depending on the theory. Among them, NRA (nonlinear real arithmetic) refers to a theory in which variables can take real values and constraints can be high-order nonlinear polynomials. For this type of problem, current mainstream solvers including Z3, CVC5, Yices, all provide effective solutions. Common search algorithms include CDCL(T), MCSAT, incremental linearization, variable substitution, and search algorithms based on interval arithmetic. However, due to the high complexity of the solution space of the NRA problem and the computational complexity of high-order polynomial operations, the solution of this kind of problem is still challenging.

This paper mainly explores the solution space and searching difficulties of NRA theory, and proposes an efficient local search algorithm for solving such complex constraints. This paper first introduces the solution space of NRA theory and gives the definition of some basic concepts, such as the division of the sign-invariant cell. Next, this paper proposes the current system search algorithm and local search algorithm. Some of the local search algorithms do not fully support all NRA forms and have certain limitations in searching efficiency. Based on this, this work designs a boundary-based caching mechanism, which can reduce the candidate operations in the search process on the one hand, and effectively reduce the computational complexity of each iteration on the other hand, and also gives the conditions and specific implementation algorithms for real-time updates. In addition, the NRA problem has irrational number assignment characteristics that are different from other theories, which makes the iterative efficiency of local search extremely low. This paper proposes an equality relaxation strategy, which can postpone the irrational number assignment to a certain extent and use approximate solutions to temporarily replace the exact solution. Finally, we also discuss the problem of no single variable operation unique to the NRA problem, and give an iterative strategy for moving multiple variables. The overall algorithm implementation also includes a restart strategy, which can jump out of the current search space and adjust the search area in time when trapped in the local optimum.

Based on the above algorithm, this paper implements an SMT solver named LS_NRA,

which can support any form of instances of NRA theory. We tested the solving effect of the tools on SMT-LIB benchmark, including some usage scenario instances from program verification, automata theory, and even biological networks. The implementation results show that our solving algorithm is competitive with the performance of some complete algorithms, and performs very well on high-order instances, breaking the previous record of zero number of other solvers. We also explored the solution time required for different solvers to solve a single example. Our algorithm is comparable to mainstream solvers in terms of time, and can solve practical problems of NRA theory.

Keywords: SMT, Nonlinear Real Arithmetic, Local Search

目 录

图形列表

表格列表

第 1 章　绪论

本章节主要介绍本文工作的研究背景和意义。首先，本文对一些基本概念给出定义，包括 SMT 问题、非线性实数理论（NRA）、解空间以及符号一致胞腔。本文还会介绍目前的几种主流求解算法以及局限性，引出本文工作的研究动机。接着，本文会针对工作的几个创新点展开，详细阐述算法的设计。最后，本文将总结论文的整体结构。

1.1　研究背景及意义

随着信息技术的发展，软硬件系统的正确性和安全性日益成为人们关注的话题。现有的一些验证技术使用诸如模型检查、定理证明等手段将问题转化成为可满足性模理论 (Satisfiability Modulo Theories, SMT) 问题，并通过求解器求解。因此，SMT 求解算法的设计对工业生产、科学研究等领域具有重要意义。针对特定的约束类型，如何设计高效的求解算法，在短时间内求解更多的样例成为 SMT 研究的一个重点。基于此，研究一种高效求解 SMT 问题的算法具有很高的理论意义与实际价值。

1.1.1　可满足性模理论问题 (SMT Problem)

可满足性模理论问题 (SMT) 是一种在计算机科学中常见的问题类型，它组合了布尔可满足性问题 (SAT) 和一些理论约束。每一个约束一般可以表示为对变量集合取值范围的限制，当一组变量的赋值满足了所有约束时，当前赋值被称为 SMT 问题的一组解。SMT 求解可以理解为寻找一组解或者证明不存在解的过程。SMT 问题的一个挑战在于囊括理论的复杂性，比如算术理论、位向量理论等，不同理论需要的求解策略也不同。除此之外，SMT 问题涉及到的变量个数与约束个数十分庞大，解空间规模基本上呈指数关系，这也就造成了目前的求解器很难在短时间内处理十分庞大的 SMT 问题。

SMT 问题在很多领域有广泛应用，常用于符号执行[1,2]，程序验证[3,4], 程序生成[5]，自动机学习[6,7] 以及神经网络验证等[8–11]。非线性实数理论一般广泛应用于信息物理系统[12–14]，程序终止条件的秩函数生成[15,16]，非线性混成自动机的分析[17] 等。一般来说，这些问题会把需要验证的条件编码成为 SMT 问题，然后交给后端的求解器取处理。比如，在符号执行中约束包括程序的分支条件，需要对程序可能执行的每一条路径进行搜索，从而确保最终程序的状态符号认为的要求，这些条件最终被编码成为 SMT 问题中的约束，整个问题转化为全部约束是否可以同时满足，即 SMT 问题。

目前关于 SMT 求解的研究基本可以分为完备搜索算法和启发式搜索算法。完备搜索算法主要的思想是通过搜索 SMT 问题的一部分空间，然后在遇到冲突

的情况下通过推理和回溯完成对当前冲突区域的剪枝，以避免后续搜索遇到相同冲突，算法在所有变量得到赋值后结束。一般来说，完备算法因为其很强的推理能力成为主流算法，但整体推理的时间和空间复杂度较高，面对大规模的样例容易消耗过多的计算资源。启发式方法一般借助一些人工的启发式优化策略，比如引入随机等，来更好地完成整个搜索过程。启发式方法一般会针对特殊的约束类型采取不同的启发式策略，因此其求解效果可能会根据样例的形式不同而产生不同的效果。在处理大规模样例中，启发式效果对计算资源的消耗较少，有时会得到比完备算法更好的结果。本文接下来会重点概述几种不同算法的求解思路。

完备算法一般也成为系统搜索算法，可以同时处理约束可满足以及不可满足的情况。SMT 的完备算法一般包括 CDCL(T)[18] 和 MCSAT 算法[19,20]，其共同的思路是不断尝试新的赋值，在遇到冲突时通过冲突分析学习新的子句，通过不断试错缩小需要探索的解空间，直到最终找到满足所有约束的一组解，或者排除整个解空间从而证明原公式不可被满足。非线性实数理论一般需要通过柱形代数分解 (CAD)[21] 进行量词消去，从而学习到特定冲突下的新子句。这方面的研究包括应用 CAD 的变种[22]，对 CAD 投影设计启发式的变量顺序[23] 等。除了上述两种算法之外，近年来一些其他算法也在 SMT 求解上取得了不错的效果，包括增量线性化[24]，区间约束传播[25,26] 和亚热带方法[27,28]。这些方法一般作为求解器插件使用，可以在特定的样例进行快速处理。

近年来，一些基于优化方法常用来检测给定区域内是否存在符合多项式组的解，进而应用到了 SMT 问题上。其中，Cimatti 等人的工作[29] 首次应用全局优化的方法去寻找初始解，然后通过迭代寻找附近的可行解。Ni 等人的工作也使用了优化方法去寻找可行解[30]，然后通过解方程[31] 等手段求出一个精确解。

Gao 等人引入 δ-完备 ($\delta - complete$) 决策程序的概念，并基于此设计了解决非线性约束的 dReal 求解器。与一般 SMT 求解器不同的是，dReal 支持对指数函数和三角函数的求解。其中 δ-完备包括 δ-满足 ($\delta - sat$) 和不可满足 (unsat)，通过松弛输入的公式来解决更宽泛问题的效果。本文的工作主要借鉴了这种松弛的想法来加速局部搜索的迭代。和 dReal 求解器不同的是，我们的算法最终仍然会返回一个严格满足所有约束的精确解。

局部搜索算法是本文工作的重点，也是近年来求解可满足样例的重点。局部搜索算法一般从一个完全赋值开始，针对当前尚且不可满足的约束设计操作，使用评价函数筛选合适的操作进行迭代，最终通过不断在邻域中搜索输出满足所有约束的一组赋值。其主要优点是对特定样例的求解效果很好，并且能够在很短的时间内找到足够好的一组解。主要缺点包括容易陷入局部最优、操作的设计和评价函数的设计较为困难等。局部搜索一般不可用于求解不可满足的样例。目前主流的局部搜索算法支持线性整数逻辑[32]、非线性整数逻辑[33]、多线性样例[34] 和部分多项式理论[35]。本文提出了第一个可以覆盖全部非线性实数理论的局部搜索算法。

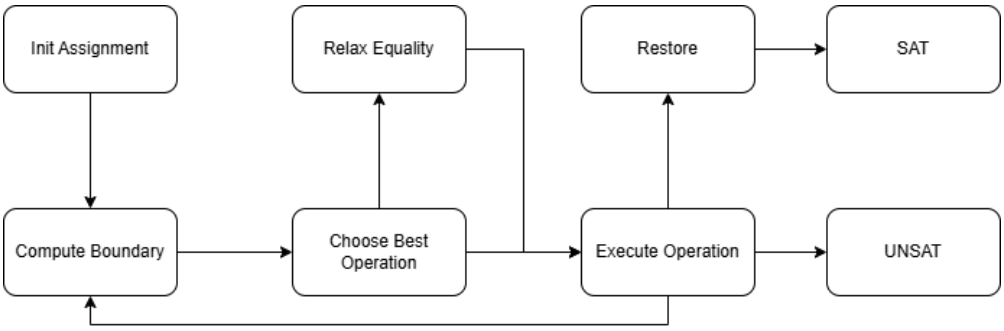

图 1.1 工具整体框架。

Figure 1.1 Overall framework of LS_NRA.

1.2　论文主要工作

本文重点关注非线性实数的 SMT 问题。本文主要讨论非线性实数问题求解的难点，并针对这些难点设计出合适的局部搜索算法，从而达到高效求解的效果。目前算法的主要难点是求解高次多项式约束需要太多求解时间，这些样例成为我们设计局部搜索算法的重点。

本工作主要在 SMT-LIB [36] 上进行试验，在可满足样例上超越了目前的主流搜索算法。算法的创新性上，本文主要考虑以下几个方面:

- 考虑通过设计更好地数据结构和迭代策略，针对实数问题的操作采样进行优化，以期望可以加速整体搜索过程；
- 针对非线性实数特有的无理数赋值问题，如何减少多项式计算上的时间消耗;
- 考虑非线性问题单变量无操作的情况，如何避免搜索陷入停滞的情况;

本文的局部搜索算法 **LS_NRA** 整体流程如图1.1所示。本工作主要包括以下几个贡献：

- 首先，本文通过分析非线性实数的解空间引入边界（boundary）数据结构，从而实现了可行域-分数对变量的缓存机制。本文还给出了邻居变量的定义以及边界的更新算法，从而可以保证算法的正确性以及数据结构的可复用性；
- 针对无理数赋值问题，本文借鉴 dReal 的做法，在强迫无理数赋值时引入等式松弛（relaxation）的概念，允许暂时的有理数赋值。在找到松弛解之后，本文给出了求解精确解的算法，保证了算法的正确性；
- 针对非线性问题独有的无单变量移动问题，本文给出了一个简单的迭代算法和前瞻策略 (look-ahead)，基本避免了算法停滞的现象；
- 本文增加了重启策略和预处理模块，相关工具在 SMT-LIb 上效果良好，可以在短时间内快速找到高次多项式的可满足赋值，打破了以往主流求解器的高次问题上的零求解。

1.3 论文组织

本文的后续章节按照以下方式组织：

第二章：介绍 SMT 问题的基本概念和解空间，并介绍目前主流的算法和研究现状。

第三章：介绍目前非线性实数理论的挑战和本文的设计思路及实现过程。

第四章：介绍预处理模块、重启策略和整体工具实现。

第五章：介绍实验设计和结果分析。

第六章：总结本文贡献，展望后续研究工作。

第 2 章　相关技术及研究现状

本章节主要介绍 SMT 和非线性实数理论的相关基本概念以及研究现状。首先，本文给出 SMT 问题和非线性实数理论的语法，然后给出解空间表达以帮助读者更好理解 SMT 问题的本质。紧接着我们给出 CDCL(T) 算法的结构、MCSAT 算法原理以及其他一些 SMT 求解器的介绍。然后我们给出近几年来局部搜索在 SMT 问题上的应用，包括线性整数逻辑、多线性逻辑和部分多项式理论等。最后，我们总结目前算法的缺点和非线性问题的挑战，并引出我们的工作带来的进展和突破。

2.1　可满足性模理论与非线性实数问题

首先，本文给出可满足行模理论 (Satisfiability Modulo Theories, SMT) 问题的一般定义，前置概念介绍如下：

定义 2.1 (变量). SMT 问题中的变量根据赋值要求分为布尔变量和算术变量。布尔变量是只能取布尔值 $(\top, \bot)$ 的变量，布尔变量集合一般用 $\boldsymbol{B}$ 表示。算术变量可以取整数或者实数值，一般用 $\boldsymbol{R}$ 表示。非线性实数问题中所有算术变量必须取实数值。

定义 2.2 (多项式约束). 非线性问题主要由多项式约束构成，多项式约束的形式为 $P(x) \sim 0$，其中 $P(x)$ 是一个多项式，$\sim$ 是一个关系符号，可以是 $=, <, >$ 中的一个。

Tseitin 编码保证了任意的逻辑约束可以在多项式时间内转化为合取范式 (CNF)。为方便说明，本文中所有 SMT 约束表达成为 CNF 形式，一些基本概念如下：

定义 2.3. 命题文字 命题文字 (literal) 是可满足问题中的基本结构，包括正文字 (p) 和负文字 $(\neg p)$。非线性实数理论的命题文字一般是多项式约束或布尔变量约束。正文字只有当对应的多项式约束满足时才满足，负文字只有当对应的多项式约束不满足时才满足。一个典型的非线性实数文字比如 $x^2 + y > 0$ 或者 $\neg(x + y^3 < 0)$。

定义 2.4. 子句 子句 (clause) 定义为文字的析取结构，一般记为 $C = l_1 \vee l_2 \vee \cdots \vee l_n$，其中 l_i 是一个文字。当子句包含的所有文字不满足时子句不满足，只要有一个文字被满足子句也被满足。**空子句**一般表示不包含任何文字的子句。比如，一个非线性实数理论的子句形如 $(x^2 + y > 0) \vee (x + y^3 < 0)$。

定义 2.5. 公式 公式 (formula) 定义为子句的合取结构，一般记为 $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$，其中 C_i 是一个子句。当公式包含的所有子句都满足时公式满足，只要有一个子句不满足公式就不满足。一个非线性实数理论的公式比如 $(x^2 + y > 0) \wedge (x + y^3 < 0)$。

定义 2.6. 赋值 赋值 (assignment) 指的是变量到布尔值或者实数的一种映射，布尔变量赋值为 $f : B \to \{\top, \bot\}$，算术变量赋值为 $f : R \to \mathbb{R}$。一个赋值 f 满足一个公式 φ，记作 $f \vDash \varphi$，当且仅当对于公式中的每一个子句 C_i，至少有一个文字 l_j 满足 $f \vDash l_j$。赋值可以分为部分赋值 (partial assignment) 和完全赋值 (full assignment)，部分赋值只对部分变量存在映射，完全赋值对所有变量都存在映射。赋值一般记为一组映射，比如 $\{x \mapsto 2, y \mapsto 3\}$。

定义 2.7. 评估 评估 (evaluation) 指的是给定赋值和一个文字，判断当前文字是否满足，记为 $eval(ass, l) \to \{\top, \bot\}$，其中 ass 是一个赋值，l 是一个文字。评估的结果为 $\top$ 表示文字满足，为 $\bot$ 表示文字不满足。比如 $eval(\{x \mapsto 2, y \mapsto 3\}, x^2 + y > 0) \to \top$。评估只有在文字包含的所有变量都有赋值时才有返回值。

定义 2.8. 可满足性模理论 (SMT) 可满足性模理论 (SMT) 指的是给定一个逻辑公式 φ，如果存在一组赋值 f 使得 $f \vDash \varphi$，则称 φ 是**可满足的 (satisfiable)**，否则称 φ 是**不可满足的 (unsatisfiable)**。SMT 问题的目标是找到一个可满足的赋值，即找到一个满足公式的赋值，或者证明赋值不可满足。

我们给出以下例子2.1进一步说明 SMT 问题的可满足性。

例 2.1. 给定 SMT 公式 $\varphi = (x^2 + y > 0) \wedge (x + y^3 < 0)$，我们可以找到一组赋值 $f = \{x \mapsto 2, y \mapsto -2\}$ 使得 $f \vDash \varphi$。因此我们称该问题是可满足的。给定 SMT 公式 $\varphi = (x^2 + y^2 < 0)$，我们在实数空间上找不到一组赋值满足当前约束，因此该问题是不可满足的。

逻辑公式化简：为方便说明，我们把所有的算术文字化简为以下几种形式 $\{p > 0, p \geq 0, p = 0\}$，其中 p 是多项式。具体的化简规则表述如下：

- $p < 0$ 化简为 $'(p) > 0$，其中 $'(p)$ 表示 p 的相反多项式。
- $p \leq 0$ 化简为 $'(p) \geq 0$，其中 $'(p)$ 表示 p 的相反多项式。
- $p \leq 0 \wedge p \geq 0$ 化简为 $p = 0$。

2.2 SMT(NRA) 的解空间

本小节主要探讨 SMT(NRA) 公式的解空间结构，以帮助读者更好理解 SMT 问题和搜索过程的本质。

2.2.1 多项式约束的解空间

定义 2.9. 解空间 解空间 (Solution Space) 指的是满足 SMT 公式的所有赋值的集合，一般用 S 表示。解空间是一个高维空间，每个维度对应一个变量的取值，每个点对应一个赋值。解空间的维度取决于变量的个数，解空间的大小取决于变量的取值范围。非线性实数的解空间是 R^n，其中 n 是变量的个数。

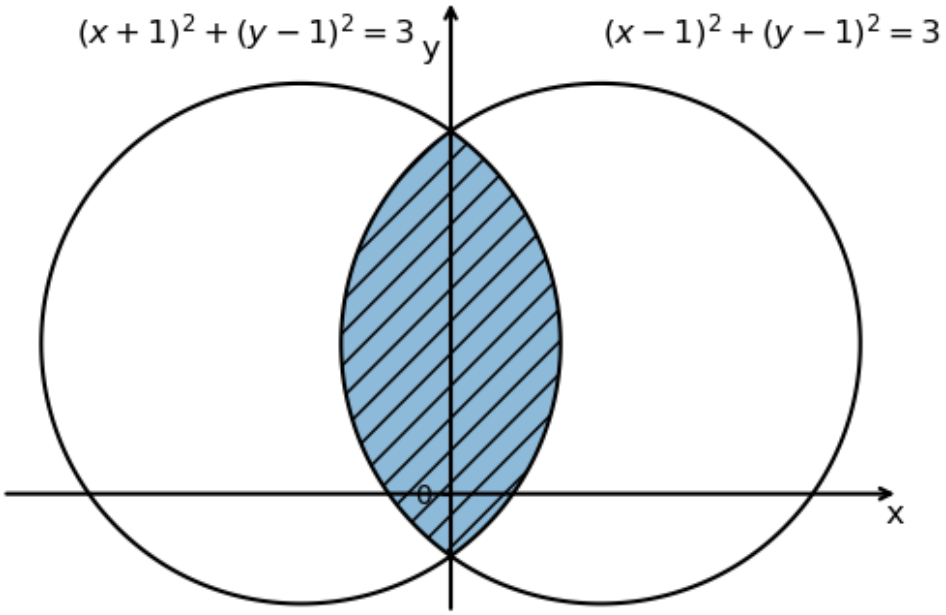

图 2.1 解空间示意图。

Figure 2.1 Solution Space Demo.

例 2.2. 考虑逻辑公式 $F=(x-1)^2+(y-1)^2\leq 3\wedge(x+1)^2+(y-1)^2\leq 3$，构成的 R^2 空间图形如图2.1所示。两个子句分别表示两个圆的内部，逻辑公式 F 表示同时存在两个圆内部的区域，即图中阴影区域。任何存在于阴影区域内的点都满足逻辑公式 F，任何满足逻辑公式 F 的点都存在于阴影区域内。因此，解空间 S 是阴影区域的集合。

数学上，我们把这样的区域定义为胞腔。

定义 2.10. 胞腔 对于 R^n 空间上的多项式集合 Q，Q 的一个胞腔 (cell) 是每个多项式 $P\in Q$ 保持符号不变的 R^n 最大联通集合。对于任意的点 $a\in R^n$，如果 a 在 Q 的胞腔内，则 a 满足 Q 中的所有多项式，我们记这个胞腔为 $cell(Q,a)$。显然，胞腔是 R^n 空间的一个划分。

对于逻辑公式而言，因为胞腔内的任意一点对每个多项式 $P\in Q$ 保持符号不变，因此 SMT 公式对应的所有文字仍然保持布尔值不变，不会对 SMT 公示的满足或不满足造成任何影响。当找到其中任意一个满足逻辑公式的胞腔时，可以判定原公式可满足；当所有胞腔都被证明不可满足时，可以判定原公式不可满足。

给定逻辑公式，如何快速剔除不满足的胞腔并找到可满足的胞腔非常重要，因此现有的研究工作很多聚焦在更好地胞腔划分上。一般的胞腔划分是根据多项式的根和判别式等来判断的，从一个点得到胞腔的划分成为延拓。

定义 2.11. 延拓 假定 R^n 上的多项式集合 Q 和点 $a=(a_1,...,a_n)$。给定一个变量 $x_i(i=1,2,...,n)$,假定多项式 $\{q(a_1,...,a_{i-1},x_i,a_{i+1},...,a_n)|q(a_1,...,a_{i-1},x_i,a_{i+1},...,a_n)\not\equiv$

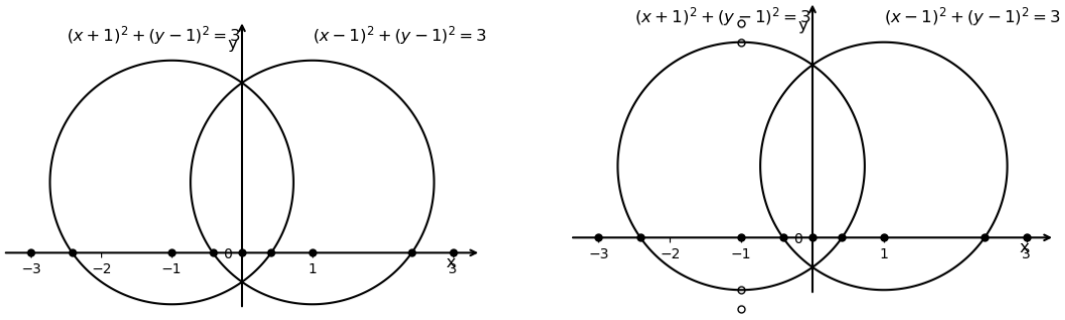

图 2.2 延拓点集示意图。

Figure 2.2 Expansion Demo.

$0, q \in Q\}$ 的实数根 $r_1 < r_2 < \cdots < r_s$。一个点 a 关于变量 x_i 在集合 Q 上的延拓定义为一个满足如下性质的点集 $\Lambda \subseteq R^n$：

- 点 $a \in \Lambda$ 并且对于 $1 \leq j \leq s$ 均存在点 $(a_1, ..., a_{i-1}, r_j, a_{i+1}, ..., a_n) \in \Lambda$。
- 对于任意点 $b = (b_1, ..., b_n) \in \Lambda$，对于 $j \in 1, ..., n$

i 有 $b_j = a_j$。

- 对于任意的区间 $I \in \{(-\infty, r_1), (r_1, r_2), \cdots, (r_{s-1}, r_s), (r_s, +\infty)\}$，都有唯一的 $b = (b_1, ..., b_n) \in \Lambda$ 满足 $b_i \in I$。

对于点集 $\{a^{(1)}, ..., a^{(m)}\} \subseteq R^n$，定义集合关于变量 x_i 在 Q 上的延拓时 $\bigcup_{j=1}^{m} \Lambda_j$，其中 Λ_j 是 $a^{(j)}$ 关于变量 x_i 的延拓。

例 2.3. 紧接着例子2.2，如图2.2所示，我们可以得到点 $(0,0)$ 关于 x 的延拓，即点集 $\{(-3,0), (-1-\sqrt{2},0), (-1,0), (1-\sqrt{2},0), (0,0), (\sqrt{2}-1,0), (1,0), (1+\sqrt{2},0), (3,0)\}$。这些点在 x 变量上分割了多项式的符号区间，从而划分了胞腔在 x 方向上的投影。右侧图的空心点构成了点 $(-1,0)$ 关于变量 y 的延拓，即点集 $\{(-1,-1), (-1, 1-\sqrt{3}), (-1, 1+\sqrt{3}), (-1,3)\}$，这些点在 y 方向上分开了多项式的符号区间，从而划分了胞腔在 y 方向上的投影。

点 a 关于变量 x 的延拓点集事实上是点 a 在方向 x 上相邻胞腔的采样点，如何能够对 R^n 空间上的所有胞腔进行采样是我们接下来的话题。为此，我们引入柱形延拓的概念。

定义 2.12. 柱形延拓 假设 R^n 上的多项式集合 Q 和点 a。给定一个变量顺序 $x_1 < x_2 < \cdots < x_n$，定义点 a 关于变量顺序在 Q 上的柱形延拓是 $\bigcup_{i=1}^{n} \Lambda_i$，其中 Λ_1 是 a 关于变量 x_1 在 Q 上的延拓，并且 Λ_{i+1} 是 Λ_i 关于变量 x_{i+1} 在 Q 上的延拓。我们把最终的结果 $\bigcup_{i=1}^{n} \Lambda_i$ 称为 Q 上的柱形延拓。

柱形延拓可以理解为给定变量顺序下多维的点集延拓，其目的是在 R^n 空间的每一个胞腔上都有一个采样点。

例 2.4. 紧接着例子2.3，如图2.3所示，图中所有的实心点和空心点共同构成了基于变量顺序 $x < y$ 的多项式 Q 的柱形延拓，达到了每个胞腔上均有一个采样点

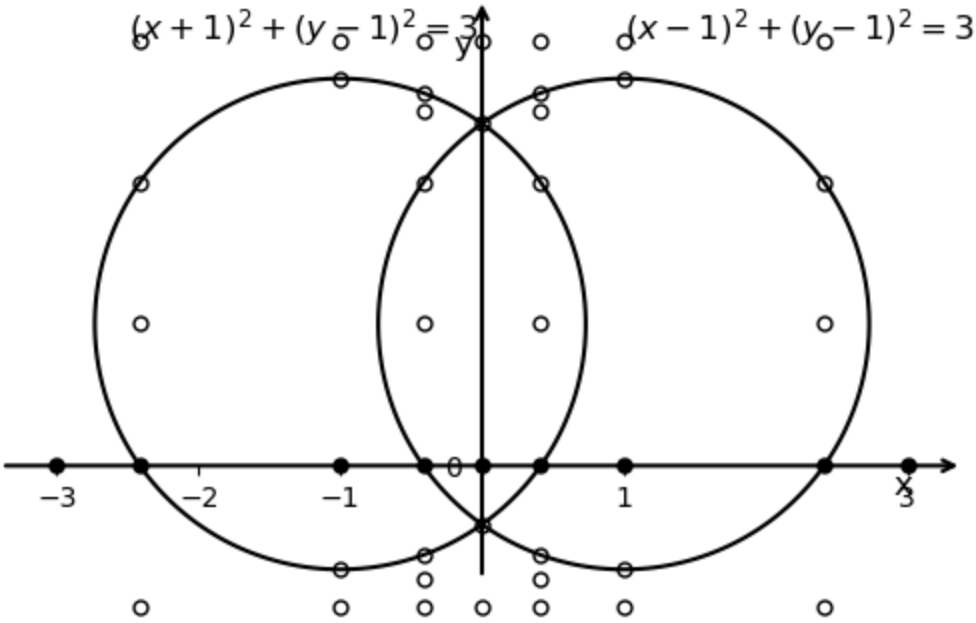

图 2.3 柱形延拓示意图。

Figure 2.3 Cylindrical Expansion Demo.

点坐标	P_1：$(x+1)^2+(y-1)^2=3$ 根	P_2：$(x-1)^2+(y-1)^2=3$ 根	根集合
$x \mapsto -3$	$\varnothing$	$\varnothing$	$\varnothing$
$x \mapsto -1-\sqrt{2}$	$\{0, 2\}$	$\varnothing$	$\{0, 2\}$
$x \mapsto -1$	$\{1-\sqrt{3}, 1+\sqrt{3}\}$	$\varnothing$	$\{1-\sqrt{3}, 1+\sqrt{3}\}$
$x \mapsto 1-\sqrt{2}$	{-0.63, 2.63}	$\{0, 2\}$	{-0.63, 0, 2, 2.63}
$x \mapsto 0$	$\{1-\sqrt{2}, 1+\sqrt{2}\}$	$\{1-\sqrt{2}, 1+\sqrt{2}\}$	$\{1-\sqrt{2}, 1+\sqrt{2}\}$
$x \mapsto \sqrt{2}-1$	$\{0, 2\}$	{-0.63, 2.63}	{-0.63, 0, 2, 2.63}
$x \mapsto 1$	$\varnothing$	$\{1-\sqrt{3}, 1+\sqrt{3}\}$	$\{1-\sqrt{3}, 1+\sqrt{3}\}$
$x \mapsto 1+\sqrt{2}$	$\varnothing$	$\{0, 2\}$	$\{0, 2\}$
$x \mapsto 3$	$\varnothing$	$\varnothing$	$\varnothing$

表 2.1 柱形延拓-区间计算。

Table 2.1 Demo of Computation of Cylindrical Expansion-Interval Splitting.

投影点	分割区间	延拓点集（采样点）
(-3, 0)	$(-\infty, \infty)$	(-3, 0)
$(-1-\sqrt{2}, 0)$	$(-\infty, 0)$	$(-1-\sqrt{2}, -1)$
	[0, 0]	$(-1-\sqrt{2}, 0)$
	(0, 2)	$(-1-\sqrt{2}, 1)$
	[2, 2]	$(-1-\sqrt{2}, 2)$
	(2, ∞)	$(-1-\sqrt{2}, 3)$
(-1, 0)	$(-\infty, 1-\sqrt{3})$	(-1, -1)
	$[1-\sqrt{3}, 1-\sqrt{3}]$	$(-1, 1-\sqrt{3})$
	$(1-\sqrt{3}, 1+\sqrt{3})$	(-1, 0)
	$[1+\sqrt{3}, 1+\sqrt{3}]$	$(-1, 1+\sqrt{3})$
	$(1+\sqrt{3}, \infty)$	(-1, 3)
$(1-\sqrt{2}, 0)$	$(-\infty, -0.63)$	$(1-\sqrt{2}, -1)$
	[-0.63, -0.63]	$(1-\sqrt{2}, -0.63)$
	(-0.63, 0)	$(1-\sqrt{2}, -0.5)$
	[0, 0]	$(1-\sqrt{2}, 0)$
	(0, 2)	$(1-\sqrt{2}, 1)$
	[2, 2]	$(1-\sqrt{2}, 2)$
	(2, 2.63)	$(1-\sqrt{2}, 2.5)$
	[2.63, 2.63]	$(1-\sqrt{2}, 2.63)$
	(2.63, ∞)	$(1-\sqrt{2}, 3)$
(0, 0)	$(-\infty, 1-\sqrt{2})$	$(0, -1)$
	$[1-\sqrt{2}, 1-\sqrt{2}]$	$(0, 1-\sqrt{2})$
	$(1-\sqrt{2}, 1+\sqrt{2})$	(0, 0)
	$[1+\sqrt{2}, 1+\sqrt{2}]$	$(0, 1+\sqrt{2})$
	$(1+\sqrt{2}, \infty)$	(0, 3)
$(\sqrt{2}-1, 0)$	$(-\infty, -0.63)$	$(\sqrt{2}-1, -1)$
	[-0.63, -0.63]	$(\sqrt{2}-1, -0.63)$
	(-0.63, 0)	$(\sqrt{2}-1, -0.5)$
	[0, 0]	$(\sqrt{2}-1, 0)$
	(0, 2)	$(\sqrt{2}-1, 1)$
	[2, 2]	$(\sqrt{2}-1, 2)$
	(2, 2.63)	$(\sqrt{2}-1, 2.5)$
	[2.63, 2.63]	$(\sqrt{2}-1, 2.63)$
	(2.63, ∞)	$(\sqrt{2}-1, 3)$
(1, 0)	$(-\infty, 1-\sqrt{3})$	$(1, -1)$
	$[1-\sqrt{3}, 1-\sqrt{3}]$	$(1, 1-\sqrt{3})$
	$(1-\sqrt{3}, 1+\sqrt{3})$	(1, 0)
	$[1+\sqrt{3}, 1+\sqrt{3}]$	$(1, 1+\sqrt{3})$
	$(1+\sqrt{3}, \infty)$	(1, 3)
$(1+\sqrt{2}, 0)$	$(-\infty, 0)$	$(1+\sqrt{2}, -1)$
	[0, 0]	$(1+\sqrt{2}, 0)$
	(0, 2)	$(1+\sqrt{2}, 1)$
	[2, 2]	$(1+\sqrt{2}, 2)$
	(2, ∞)	$(1+\sqrt{2}, 3)$
(3, 0)	$(-\infty, \infty)$	(3, 0)

表 2.2 柱形延拓-采样点生成。

Table 2.2 Demo of Computation of Cylindrical Expansion-Sampling Points.

的效果。具体的步骤是，首先点 $(0,0)$ 关于变量 x 形成延拓（实心点），然后每一个实心点关于变量 y 各自形成延拓（空心点），最终所有点的集合共同构成了柱形延拓。表格2.1展示了在实心点处根据多项式集合的根分割得到的区间。表格2.2展示了在每一个分割区间进行采样得到的结果，即最终的柱形延拓点集（空心点）。

定义 2.13. 柱形完备 对于 R^n 上的多项式集合 Q，给定一个变量顺序 $x_1 < x_2 < \cdots < x_n$，称 Q 对于变量顺序是柱形完备的，当任意的点 $a \in R^n$ 和其关于 Q 的柱形延拓 Λ，Q 的每一个胞腔都包含至少一个 Λ 的点。

具体的证明请参阅[21,37]。

2.2.2　量词消去与柱形代数分解

根据前文的讨论，我们引出基于多项式解空间的一种量词消去算法-柱形代数分解 (Cylindarical Algebraic Decomposition)。

定义 2.14. 量词消去 量词消去指的是给定一个带有量词的逻辑公式 φ，找到另一个不带有两次的逻辑公式 ψ，使得 φ 和 ψ 的逻辑等价，记为 $\varphi \Leftrightarrow \psi$。量词消去的目的是将逻辑公式转化为更易处理的形式，以更好地判断问题的可满足性。

例 2.5. 考虑多项式 $P_s(x,y) = s(x^2+y^2-1)+(1-s)(xy-1)$，和逻辑公式 $\varphi = \exists R \forall x, y[P_s(x,y) = 0 \Rightarrow x^2+y^2 \leq R^2]$。量词消去即求解当 s 取什么值的时候逻辑公式 φ 满足。通过量词消去工具，我们可以得到 $\varphi \Leftrightarrow s \leq -1 \wedge s > \frac{1}{3}$。

多项式理论常用的一种量词消去是柱形代数分解 (Cylindarical Algebraic Decomposition, CAD)。其核心仍然是把 R^n 空间划分成多个符号一致的胞腔，然后通过判断每个胞腔是否满足逻辑公式进而得到给定变量的赋值区间。如图2.4所示，给定变量顺序和多项式集合，柱形代数分解可以分为三个步骤：投影、实根隔离和提升，具体的步骤如下：

- **投影 (projection)：**从多项式集合开始，每次消去一个变量，生成新的多项式集合，新多项式继续消去变量直到只剩下一个变量为止，记为 $proj(P)$。
- **实根隔离 (root isolation)：**当投影只剩下一个变量时，对当前多项式集合求所有根，得到顺序排列的一组根，这些根表示了胞腔在当前变量上的分割。
- **提升 (lift)：**提升是对投影阶段得到的多项式集合进行采样。投影从单变量多项式开始，从采样点继续对上一个投影多项式进行采样，直到最终对 R^n 多项式采样，最终保证了每个胞腔得到了采样点。

投影算子 (projection operator) 指的是投影过程中对多项式集合的计算，比如 Collins 算子[37] 和 McCallum 算子[38]。后者因为其计算相对便捷，被广泛应用于量词消去工具和 SMT 求解器中。

McCallum 算子用到的计算包括结式、判别式和求根等操作。

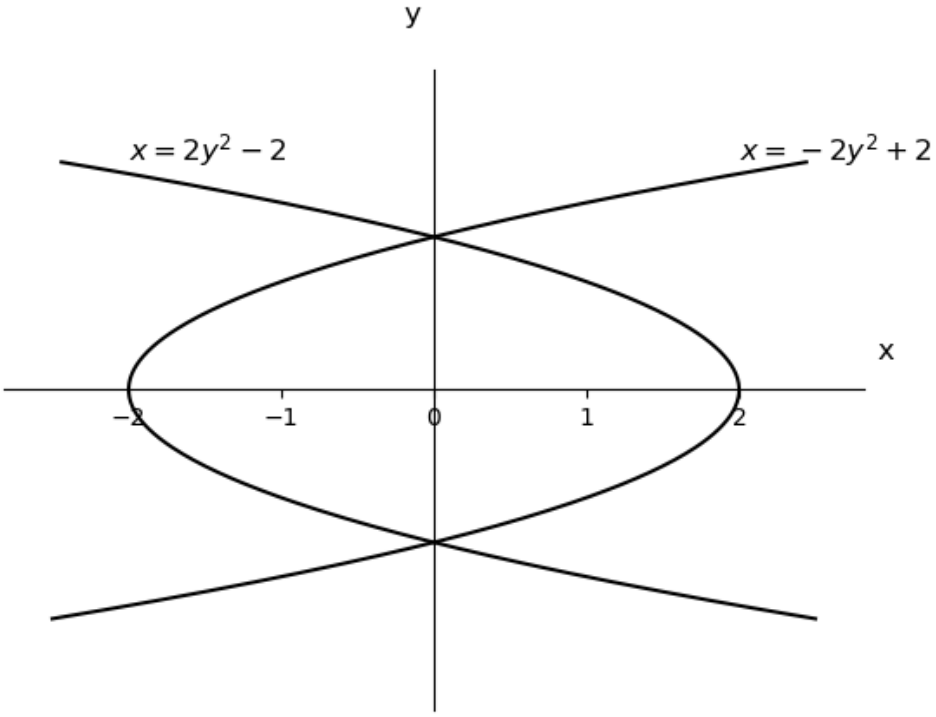

图 2.4 柱形代数分解步骤示意图。

Figure 2.4 Demo of Cylindarical Algebraic Decomposition Steps.

定义 2.15. 结式 (resultant) 对于两个多项式 $P_1, P_2 \in R[x_1, ..., x_n]$，假定

$$P_1 = a_m x_n^{d_m} + a_{m-1} x_n^{d_{m-1}} + \cdots + a_0,$$

$$P_2 = b_n x_n^{e_n} + b_{n-1} x_n^{e_{n-1}} + \cdots + b_0.$$

定义 P_1 和 P_2 的结式 $Res(P_1, P_2, x_n)$ 为：

$$Res(P_1, P_2, x_n) = \begin{vmatrix} a_m & a_{m-1} & \cdots & a_0 & & \\ & a_m & a_{m-1} & \cdots & a_0 & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & a_m & a_{m-1} & \cdots & a_0 \\ b_n & b_{n-1} & \cdots & b_0 & & \\ & b_n & b_{n-1} & \cdots & b_0 & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & b_n & b_{n-1} & \cdots & b_0 \end{vmatrix}$$

定义 2.16. 判别式 (discreminant)

对于多项式 $P \in R[x_1, ..., x_n]$，假定

$$P = a_m x_n^{d_m} + a_{m-1} x_n^{d_{m-1}} + \cdots + a_0.$$

分割区间	x 采样点	y 根集合	(x, y) 采样点	符号 (P_1, P_2)
$(-\infty, -2)$	$x \to -3$	$\{-\frac{\sqrt{10}}{2}, \frac{\sqrt{10}}{2}\}$	(-3, 2)	(-, +)
			$(-3, \frac{\sqrt{10}}{2})$	(-, 0)
			$(-3, 0)$	(-, -)
			$(-3, -\frac{\sqrt{10}}{2})$	(-, 0)
			$(-3, -2)$	(-, +)
$[-2, -2]$	$x \to -2$	$\{0. -\sqrt{2}, \sqrt{2}\}$	(-2, 2)	(-, +)
			$(-2, \sqrt{2})$	(-, 0)
			(-2, 1)	(-, -)
			(-2, 0)	(0, -)
			(-2, -1)	(-, -)
			$(-2, -\sqrt{2})$	(-, 0)
			(-2, -2)	(-, +)

To be Done

表 2.3 柱形代数分解-提升。

Table 2.3 Demo of Cylindrical Algebraic Decomposition - Lift.

定义 P 关于变量 x_n 的判别式 $Disc(P, x_n)$ 为：

$$Disc(P, x_n) = \frac{(-1)^{\frac{m(m-1)}{2}}}{a_m} Res(f, \frac{\partial f}{\partial x_n}, x_n).$$

定义 2.17. McCallum 投影算子 (McCallum Projection Operator)

假定 $F = \{f_1, f_2, \ldots, f_m\}$ 是一组 R^n 上的多项式。McCallum 投影算子 $proj(F, x_i)$ 是从 F 到 R^{n-1} 上的多项式集合 $proj_m(F)$ 的一组映射，包含以下元素:

- F 中每个多项式的系数
- F 中每个多项式关于变量 x_n 的判别式
- F 中每两个不同多项式 f_i, f_j 关于变量 x_n 的结式

实际计算中，多项式集合的元素通过因式分解化为最简系数的形式，对结果不产生影响。

例 2.6. 我们给出 CAD 算法的一个例子。假设 $F = \{P_1 : x-2y^2+2, P_2 : x+2y^2-2\}$，我们规定变量顺序为 $y < x$，按步骤计算如下：

1. **投影**：计算对于变量 y 的投影 $proj(F, y)$
- 化简后系数多项式为 $\{x + 2, x - 2\}$
- 化简后判别式为 $\{x + 2, x - 2\}$
- 化简后结式为 $\{x\}$。

得到投影多项式 $proj(F, y) = \{x, x - 2, x + 2\}$

2. **实根隔离**：我们对 $proj(F, y)$ 的每个多项式求根，得到根集合 $\{-2, 0, 2\}$。

3. **提升**：我们对根集合分得的区间进行提升，进一步采样所有胞腔，结果如表格2.3所示。

2.3　非线性实数理论求解现状

2.3.1　CDCL(T) 算法

CDCL(T) (Conflict Driven Clause Learning for Theories) 算法主要是在 SAT 问题的 CDCL 搜索框架上增加了理论求解器 (Theory Solver)，主要的搜索思路包括

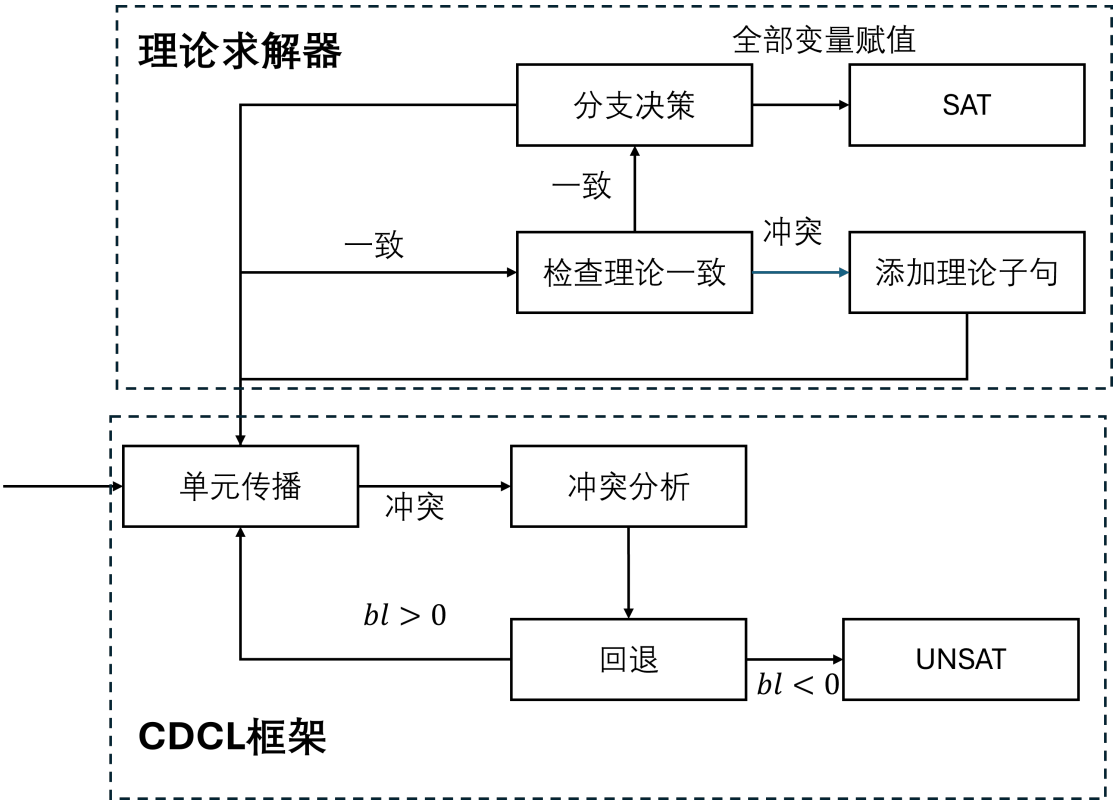

图 2.5 CDCL(T) 框架示意图。

Figure 2.5 Demo of CDCL(T) Framework.

单元传播、分支决策、冲突分析及回退等 CDCL 常见技术。理论求解器的作用是对给定的布尔骨架赋值进行推理，当全部布尔骨架得到赋值时通过理论分析返回问题可满足或者报告冲突。具体框架请参见图2.5。

非线性实数的理论求解器主要目的是判断多项式不等式集合的一致性，主要方法可以分为以下几种：

- **区间约束传播 (interval constraint propagation, ICP):** 通过变量或者单项式的赋值区间推导出多项式的赋值区间，根据这些区间快速判断一些常见的不一致性。
- **虚拟替代 (virtual substitution, VS):** 通过替换多项式中的变量，将多项式转化为更简单的形式，进而判断问题的可满足性。
- **柱形代数分解 (cylindrical algebraic decomposition, CAD):** 如前文所述，通过把 R^n 空间划分为多个符号一致的胞腔，逐一排查的方式来推出问题的可满足性。
- **增量线性化 (incremental linearization):** 把一些多项式线性化，然后使用线性求解器检测问题可满足性，一般是不完备的。

2.3.2 MCSAT/NLSAT 方法

MCSAT(Model-Constructing Satisfiability) 和 NLSAT(NonLinear Satisfiability) 是另一种具有 CDCL 结构的算法，核心的思想仍然是决策赋值、冲突分析和回

退。与 CDCL(T) 不同的是，MCSAT/NLSAT 算法加入了基于可行域的算术变量推理，允许在脱离布尔骨架的情况下直接对算术变量进行赋值，一些基本概念介绍如下：

- **决策层数 (level)：**指文字层面的决策层数，包括布尔文字和算术文字。
- **决策阶段 (stage):** 指算术变量赋值次数。
- **日志 (trace):** 一种记录算法更新的线性结构，包括可行域的更新、决策变量的赋值等。
- **算术传播 (arithmetic propagation):** 基于可行域的文字层面的传播，可以直接推断某些文字的可满足性。

其中，MCSAT/NLSAT 最关键的步骤就是根据当前变量和文字的可行域关系进行算术文字赋值，假设算术变量的可行域是 $curr\_set$，文字的可行域是 lit_set，算术传播分为以下几种情况：

- $lit\_set = \varnothing$：任何赋值都不能使文字满足，判断文字为 $\bot$。比如当 $x \mapsto -1$ 时，文字 $y^2 \leq x$ 不可满足。
- $lit\_set = R$: 任何赋值都可以使文字满足，直接赋值文字为 $\top$。比如文字 $x^2 \geq -1$ 一定满足。
- $curr\_set \subseteq lit\_set$: 当前文字的可行域包括了算术变量的可行域，因此可以直接判断文字满足。比如当 $x \in [-2, 2]$ 时，文字 $x^2 \leq 10$ 一定满足。
- $curr\_set \cap lit\_set = \varnothing$：任何可行域内的赋值都不会让文字满足，因此直接赋值文字为 $\bot$。比如当 $x \in [-2, 2]$ 时，文字 $x^2 \geq 10$ 不可满足。

例2.7给出了一个 MCSAT/NLSAT 搜索的具体例子。

例 2.7. 考虑公式 $F = (P_1 : (x-2)^2 + y^2 \leq 4) \wedge (P_2 : x - y \geq 0)$。假设我们的搜索顺序是 $x < y$，算法的搜索过程如下：

1. **决策算术变量：**赋值 $x \mapsto 5$。
2. **可行域计算：**计算 P_1 关于变量 y 的可行域是 $\varnothing$，发现冲突。
3. **冲突分析：**对公式 $\exists y.(x-2)^2 + y^2 \leq 4$ 进行量词消去，通过柱形代数分解得到实根 $\{0, 4\}$，生成新的学习子句 $x \leq 4$。
4. **回退：**回退到 x 未赋值的状态。
5. **决策算术变量：**重新赋值 $x \mapsto 2$。
6. **可行域计算：**计算 P_1 关于变量 y 的可行域是 $[-2, 2]$，P_2 关于变量 y 的可行域是 $(-\infty, 2]$，取交集得到可行域是 $[-2, 2]$。
7. **决策算术变量：**赋值 $y \mapsto 0$。所有变量得到赋值，返回 SAT。

2.3.3　局部搜索算法现状

局部搜索算法是一种启发式的非完备算法，它的主要思想是把 SMT 问题看作是一个优化问题，通过迭代操作找到最终的可行解。一般来说，局部搜索只能用来求解可满足问题，而不能证明问题的不可满足性。我们首先给出基本概念的定义如下：

定义 2.18. 状态 状态是指问题的一个解，比如变量的一组赋值。

定义 2.19. 操作 操作是每一次迭代需要做的状态上的改变，SMT 问题的状态是修改变量的赋值。假设移动前后的状态为 S 和 S'，操操作时 op，记为 $S \xrightarrow{op} S'$。

定义 2.20. 打分函数 打分函数是一个状态到实数的函数，用来评估状态的好坏。一般认为离目标越远的状态分数越低。操作的打分函数被定义为状态改变前后分数的差值。

定义 2.21. 邻域 邻域表示的是状态空间的邻居。假设当前状态为 S，邻域是指所有通过一次操作 op 可以到达的状态的集合，即 $N(S) = \{S' : S \xrightarrow{op} S'\}$。

假设状态空间 S 为所有候选解构成的解空间，局部搜索的具体求解过程可以概括如下：

1. **初始化**：随机生成一个初始状态 $s_0 \in S$。一般来说，初始状态可以启发式生成，尽量减小约束的复杂度。

2. 定义操作 op，同时定义邻域函数 $N : S \to 2^S$，表示从状态 S 可以到达的状态集合。这样，我们找到了从初始状态可以到达的一系列候选状态 $N(s_0)$

3. 定义打分函数 $f : S \to R$，表示状态的好坏。我们贪心地从候选状态中找到分数最好的一个进行迭代。

4. 设置停止条件，比如最大迭代次数、最大限制时间，或者已经找到问题的最优解。

5. 迭代搜索 2-3 步，直到满足停止条件 4。

对于 SMT 问题而言，工作[33] 给出了一种实例化的概念：

- 初始解：一般通过变量的上下界来选取，比如当变量 v 的上下界为 $[-2, 3]$ 时，可以选择初始状态为 $v \mapsto 0$。
- 单变量关键操作：SMT 问题的操作定义为让一个约束满足的赋值变量进行变化。比如对于约束 $x + y \geq 2$，当前状态为 $\{x \mapsto 1, y \mapsto 0\}$，则我们有以下两种操作：
  - 保持变量 x 不变，增加变量 y 的值，使约束满足，比如 $y \mapsto 1$。
  - 保持变量 y 不变，增加变量 x 的值，使约束满足，比如 $x \mapsto 2$。
- 邻域：邻域是所有通过一次操作可以到达的状态的集合 $\{S' : S \xrightarrow{op} S'\}$，单变量关键操作的邻域是只有一个变量不同的状态集合，即 $N(S) = \{N(S) : \exists v \in V : S(v) \neq S'(v) \wedge \forall v' \in V/\{v\} : S(v) = S'(v)\}$。

工作[33] 为求解算术理论的局部搜索算法给出了一个基本的框架，如算法1所示。

我们以一个具体的例子2.8进行说明。

例 2.8. 假设当前 SMT 约束 $F = \{C_1 : x + y \leq 2, C_2 : 2x + 3y \geq 5\}$，我们的目标是找到一个满足约束的解。我们可以通过局部搜索算法进行求解，具体步骤如下：

算法 1 Basic local search algorithm

输入: A set of clauses F

输出: An assignment that satisfy F, or failure

```
1: Initialize assignment randomly;
2: while ⊤ do
3:     if all clauses are satisfied then return current assignment;
4:     end if
5:     if time or step limit reached then return failure;
6:     end if
7:     var, new_val, score ← bestcriticalmove;
8:     if score > 0 then Perform move, assign v ↦ new_val;
9:     else
10:        Update clause weight according to PAWS scheme;
11:        cls ← randomunsatisfiedclause;
12:        var, new_val, score ← bestmove(cls);
13:        if score ≠ −∞ then Perform move, assign v ↦ new_val;
14:        end if
15:        if no move performed previously then
16:            Randomly change assignment of some variables;
17:        end if
18:    end if
19: end while
```

1. **初始解生成：**初始赋值 $\{x \mapsto 0, y \mapsto 0\}$。当前赋值满足约束 C_1，但不满足约束 C_2。

2. **生成操作：**考虑未满足约束 C_2，我们可以固定变量 x 或 y 的赋值，比如赋值 op_1 ：$\{x \mapsto 0, y \mapsto 2\}$，或者 op_2 ：$\{x \mapsto 3, y \mapsto 0\}$。

3. **打分函数：**打分函数设计为移动前后约束权重的差值，操作 op_1 可以满足两种约束，而操作 op_2 在满足约束 C_2 的同时破坏了约束 C_1。

4. **移动操作：**我们选择操作 op_1，移动后赋值为 $\{x \mapsto 0, y \mapsto 2\}$。

5. **算法停止：**当所有约束都满足时，算法停止，返回当前赋值。

2.4　本章小结

本章节主要介绍了 SMT 非线性算术问题的相关概念和研究现状。首先，我们给出逻辑公式的形式化定义，包括文字、子句、赋值等基本概念，然后引出非线性多项式及对应的可满足问题。然后，我们讨论了 SMT 问题的解空间，并讨

论了柱形代数分解求得解空间胞腔的方法。最后，我们介绍了目前的几种求解方法，包括 CDCL(T)、MCSAT 和局部搜索算法。这些算法在实际应用中有着不同的优势和劣势，可以根据具体问题的特点选择合适的算法进行求解。

第 3 章　基于可行域的胞腔跳跃缓存机制

本章首先总结了前述工作的胞腔跳跃算法，并通过例子给出当前算法带来的操作冗余性。接着，本章给出一种更合理的基于变量可行域的胞腔跳跃操作，并且针对每次迭代中可行域的重复计算问题给出数据结构的形式化表示。具体而言，本章引入边界（boundary）的概念来刻画几何上胞腔之间的界限，从而可以进一步计算变量不同赋值对分数的影响。最后，本章介绍了算法迭代中边界的实时迭代算法，具体表现为当邻居变量发生移动后，所在子句的所有算术变量需要重新计算边界，其余子句的边界不受影响。最后，本文给出局部搜索算法 LS_NRA 中操作选择与执行部分，并详细展示一个 SMT 约束如何被求解的。

3.1　胞腔跳跃操作 (Cell-Jump)

变量的操作移动是局部搜索算法的核心概念，即每一步如何更改变量的赋值，以期望在搜索空间中找到更好的解。在布尔可满足问题（SAT 问题）中，变量的操作被定义为翻转（Flip），即将变量的赋值从真（True）改为假（False）或反之。SMT 问题因为其赋值的多样性（整数或实数）使得变量的移动操作更为复杂。一个经典的操作是关键移动（critical move）[32]，即使得文字不等式恰好达到满足状态时对应的整数赋值操作。工作 [34] 将关键移动拓展到了实数操作上，即允许变量赋值为实数。对于多项式理论的关键移动，工作 [35] 借助胞腔的概念首次提出胞腔条约操作 (Cell-Jump)，具体定义如下：

定义 3.1 (平行坐标轴的胞腔跳跃 Cell-Jump Parallel to Axis). 假设当前赋值为 $\alpha = \{x_1 \mapsto a_1, x_2 \mapsto a_2, \cdots, x_n \mapsto a_n\}$。$l$ 是赋值 α 下未满足的多项式不等式，假定其不等式符号为 $<$ 或者 $>$。

- 假定 l 具有形式 $p(x) < 0$。对于每一个单变量多项式 $p(a_1, \cdots, a_{i-1}, x_i, a_{i+1}, \cdots, a_n)$ 具有负值采样点的变量 x_i 来说，存在一个胞腔跳跃操作，记为 $cjump(c_i, l)$，即赋值 x_i 到离 a_i 最近的负值采样点。
- 假定 l 具有形式 $p(x) > 0$。对于每一个单变量多项式 $p(a_1, \cdots, a_{i-1}, x_i, a_{i+1}, \cdots, a_n)$ 具有正值采样点的变量 x_i 来说，存在一个胞腔跳跃操作，记为 $cjump(c_i, l)$，即赋值 x_i 到离 a_i 最近的正值采样点。

如果把赋值看作是 R^n 空间中的一个点，那么前面的胞腔跳跃操作就是沿着平行坐标轴的直线 $(a_1, \cdots, a_{i-1}, R, a_{i+1}, \cdots, a_n)$ 移动到另一个点。

定义 3.2. 沿着固定直线的胞腔跳跃 Cell-Jump Along Fixed Straight Line 假设当前赋值为 $\alpha = \{x_1 \mapsto a_1, x_2 \mapsto a_2, \cdots, x_n \mapsto a_n\}$。$l$ 是赋值 α 下未满足的多项式不等式，假定其具有形式 $p(x) > 0$ 或 $p(x) < 0$。给定一个方向向量 $dir = (d_1, \cdots, d_n)$。引入新变量 t，对于多项式 $p(x)$ 中的每一个变量 x_i，用 $a_i + d_i t$ 来替换得到新的多

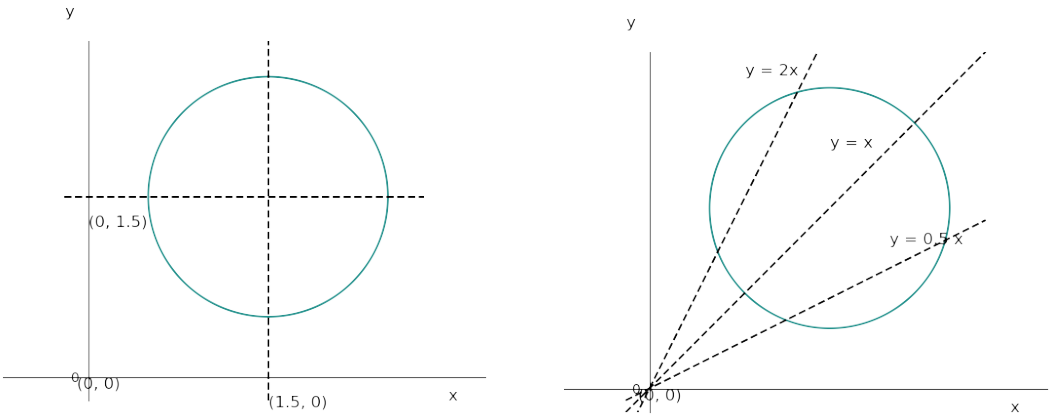

图 3.1 胞腔跳跃操作示意图。

Figure 3.1 Demo of cell-jump operations.

项式 $p^*(t)$。如果 l 不等式符号为 $<$ 并且多项式 $p^*(t)$ 具有负值采样点，那么定义胞腔跳跃操作 $cjump(dir, l)$ 为将赋值 α 沿着方向 dir 移动到离原点最近的负值采样点。如果 l 不等式符号为 $>$ 并且多项式 $p^*(t)$ 具有正值采样点，那么定义胞腔跳跃操作 $cjump(dir, l)$ 为将赋值 α 沿着方向 dir 移动到离原点最近的正值采样点。假设采样点为 t^*，则更改后的赋值为 $\alpha' = \{x_1 \mapsto a_1 + d_1 t^*, \cdots, x_n \mapsto a_n + d_n t^*\}$。

例 3.1. 如图3.1所示，给定公式 $F = \{(x-1.5)^2 + (y-1.5)^2 < 1\}$，赋值 $\alpha_1 = \{x \mapsto 1.5, y \mapsto 0\}, \alpha_2 = \{x \mapsto 0, y \mapsto 1.5\}, \alpha_3 = \{x \mapsto 0, y \mapsto 0\}$。$F$ 在赋值 $\alpha_1, \alpha_2, \alpha_3$ 下均不满足。

如左图所示，当局部搜索算法选择初始赋值为 α_1 时，通过平行 y 轴的胞腔跳跃可以得到操作 $y \mapsto 1.5$ 使得不等式满足；同理当初始赋值为 α_2 时，通过平行 x 轴的胞腔跳跃可以得到操作 $x \mapsto 1.5$ 使得不等式满足。当局部搜索算法选择初始赋值为 α_3 时，任何平行坐标轴的移动都不会使不等式满足，因此算法存在停滞的风险。

右图表示了基于固定直线的胞腔跳跃操作，定义三组方向向量分别为 $dir_1 = \{1, 2\}, dir_2 = \{1, 1\}, dir_3 = \{2, 1\}$，对应的胞腔跳跃操作分别为 $cjump(F, dir_1)$, $cjump(F, dir_2), cjump(F, dir_3)$，替换后的多项式和可行域（保留三位小数）分别为：

$$
\begin{aligned}
F_1 &= \{(t-1.5)^2 + (2t-1.5)^2 < 1\} \to (0.568, 1.232), \\
F_2 &= \{(t-1.5)^2 + (t-1.5)^2 < 1\} \to (0.793, 2.207), \\
F_3 &= \{(2t-1.5)^2 + (t-1.5)^2 < 1\} \to (0.568, 1.232).
\end{aligned}
$$

三个多项式的可行域均不为空，因此可以选取合适的值 t 同时移动变量 x 和 y 使得约束满足。

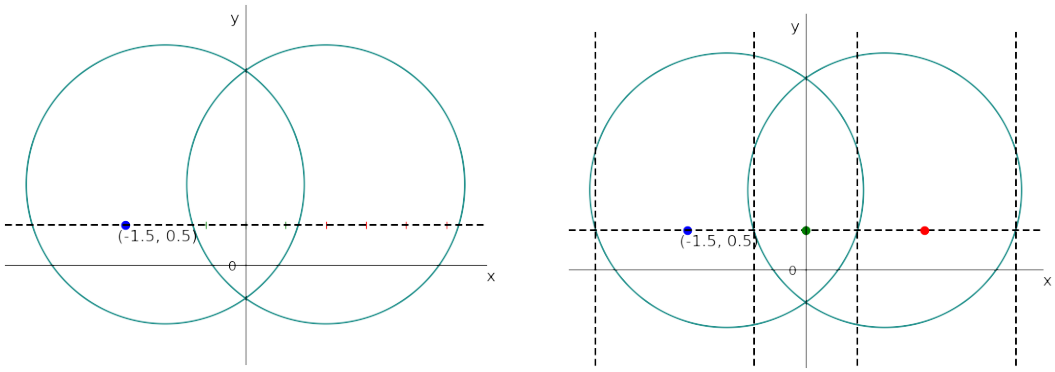

图 3.2 基于变量级别可行域的胞腔跳跃操作。

Figure 3.2 Cell-jump operation based on variable-level feasible-set.

3.2　基于变量级别可行域的操作缓存机制

前述工作的胞腔跳跃操作都是基于单一子句进行的，通过收集所有不满足子句的操作最终决定最优的操作进行执行。但是，在实际的局部搜索迭代中，一个变量可能同时出现在多个子句中，因此在每次迭代中都需要重复计算变量的可行域。除此之外，基于单一子句的胞腔操作容易造成操作的冗余，从而减慢整体迭代效率。我们以例3.2进行说明。

例 3.2. 考虑逻辑公式 $F = \{P_1 : (x-1)^2+(y-1)^2 \leq 3, P_2 : (x+1)^2+(y-1)^2 \leq 3\}$，假设当前赋值为 $\alpha : \{x \mapsto -1.5, y \mapsto 0.5\}$。可知当前赋值仅满足 P_1，不满足 P_2。我们考虑平行于 x 轴的胞腔跳跃操作 $cjump(P_1, x)$，即图3.2中虚线所示。

传统计算方法 (图3.2左图)：

- 我们首先计算变量 x 对约束 P_1 的可行域 $(-0.658, 2.658)$。
- 根绝可行域我们进行采样，比如选取 7 个采样点得到 $\{-0.5, 0, 0.5, 1, 1.5, 2, 2.5\}$，如图3.2左图中短竖线所示。
- 我们对这 5 个采样点逐一计算分数，选取其中分数最大的操作，比如 $\{x \mapsto 0\}$。
- 移动后的赋值 $\{x \mapsto 0, y \mapsto 0.5\}$ 满足约束 P_1 和 P_2，找到可行解，算法停止。

这种方法带来了两个潜在的问题：

- 因为目前的胞腔跳跃操作 $cjump(P_1, x)$ 是基于单一约束设计的，因此选择的采样点有可能同时破坏当前已经满足的约束。虽然我们可以通过打分函数进行筛选，但更好的方式是选择操作的时候进行简单判断。图3.2左图中，红色采样点表示破坏了原先 P_2 的可满足状态，而绿色采样点表示保持了 P_2 的可满足状态。
- 在每次迭代中，我们只会根据当前约束的可行域进行采样，因此可能造成整体 R^n 空间上同类操作的冗余，从而在接下来的打分函数计算环节降低了算法的效率。比如在图3.2左图中，红色点同属一个胞腔，绿色点同属另一个胞腔。

定义 3.3. 同类操作 (Similar Operations) 对于多项式约束 P，给定两个胞腔跳跃操作 $cjump_1(P,x)$ 和 $cjump_2(P,x)$，他们分别将变量 x 移动到不同的赋值。如果两个移动后的赋值 α_1 和 α_2 在解空间上存在于同一个胞腔中，我们定义两个操作是同类操作，记作 $cjump_1(P,x) \sim cjump_2(P,x)$。

从图中可以看出，更好地刻画胞腔跳跃操作需要引入变量在多个子句的可行域概念，总和所有可行域后才能可以得到可行域-分数的对应关系。我们把改进的计算方法描述如下：

改进计算方法 (图3.2右图)：

- 首先计算变量 x 对两个约束 P_1 和 P_2 各自的可行域: $(-0.658, 2.658)$ 和 $(-2.658, 0.658)$。
- 综合两个可行域,我们可以得到 R 上的一个划分:$(-2.658, -0.658), (-0.658, 0.658), (0.658, 2.658)$。
- 对于每一个部分，我们只需要选择一个采样点即可，然后计算当前操作的得分，选择分数最佳的操作。如图3.2右图中，我们选择了红色 $(0, 0.5)$ 和绿色 $(1.5, 0.5)$ 两个采样点，前者同时满足两个约束，而后者仅仅满足 P_1 约束。
- 检查移动后的赋值 α ：$\{x \mapsto 0, y \mapsto 0.5\}$，找到可行解，算法停止。

比较两种算法，可以看出传统算法采样在单一子句可行域计算时进行，也仅仅维护了赋值-分数的对应关系；而改进后的算法通过综合多个子句的可行域，维护了 R 上每一个赋值和分数的映射关系，可以非常方便计算出对应的操作分数。除此之外，因为传统算法提前采样的特点，产生了大量的冗余操作，改进后的算法因为直接在胞腔层面进行了维护，因此采样点的选择更具有合理性。

我们把 R 上胞腔的分割定义为**边界 (boundary)**，具体定义如下：

定义 3.4. 边界 (boundary) 我们定义**边界 (boundary)** 为一种四元组数据结构 $< val, is\_open, is\_make, cid >$，其中 val 是实数，表示边界的值，is_open 是布尔值，表示边界是否为开区间，is_make 是布尔值吗，表示边界处分数递增还是递减，cid 表示该处边界对应的子句编号。我们定义边界之间的排序为：首先 val 值小的排在前面，如果 val 相同，is_open 为假的排在前面，这样我们可以得到一组按照实数自然排序的边界集合 (boundary set)。

需要注意的是，边界仅仅刻画变量从边界左侧移动到边界右侧对应的约束变化情况（可满足到不可满足，不可满足到可满足），如果要计算对应胞腔的具体分数，需要从边界处采样点进行计算。我们使用例3.3进行说明。

例 3.3. 仍然考虑例3.2中的约束和赋值，约束 P_1 和 P_2 的可行域是：$(-0.658, 2.658)$ 和 $(-2.658, 0.658)$。在边界处，变量从左向右移动会导致相应约束的满足状态发

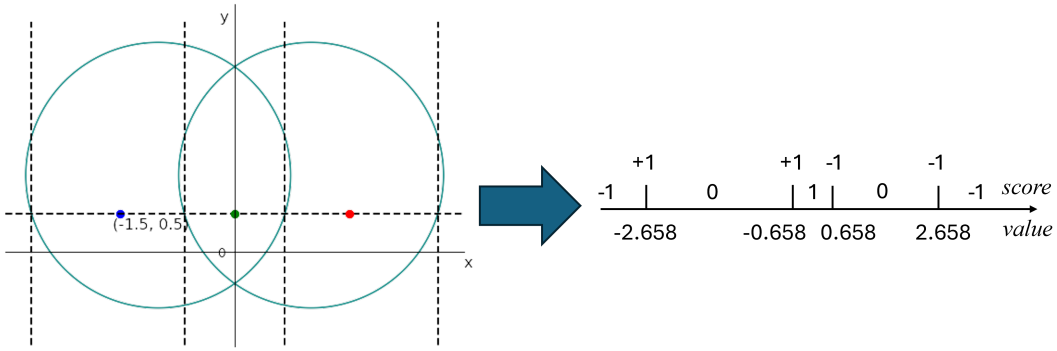

图 3.3 边界计算示意图。

Figure 3.3 Demo of Coomputation of Boundaries.

生变化，假设每个约束的权重为 1，可以计算得到 R 上的四个边界如下：

$$Boundary_1 :< -2.658, \mathsf{T}, \mathsf{T}, 2 >$$
$$Boundary_2 :< -0.658, \mathsf{T}, \mathsf{T}, 1 >$$
$$Boundary_3 :< 0.658, \bot, \bot, 1 >$$
$$Boundary_4 :< 2.658, \bot, \bot, 2 >$$

四次边界处的分数变化均为 1。为了求得赋值和分数的对应关系，我们需要在每个分割的胞腔处采样，然后计算分数。一种更简单的办法是计算某一个胞腔处的分数，然后根据边界的分数变化计算其他胞腔处的分数。可以注意到，当变量 x 移动到和原来赋值所在的胞腔时（比如例子中的 $(-2.658, -0.658)$），对应的分数一定为 0。为了计算的方便性，我们引入**起始分数 (starting score)** 的概念来表示数轴最左侧代表的分数。

定义 3.5. 起始分数 (starting score) 我们定义**起始分数 (starting score)** 为一个实数值，表示数轴上最左侧的分数，即变量移动到 $-\infty$ 时对应的分数。

在本例子中，最左侧的赋值会破坏原先已经满足的 P_2 约束，因此分数为-1。在每次边界处我们增加或减少对应的分数，因此可以得到图3.3右侧的结果。边界和分数的计算展示在算法2中。

在局部搜索算法中，每一次迭代只会改变一个变量的赋值，因此绝大多数约束的可满足状态和可行域不会发生变化，一个可行的优化是增量式计算变量的边界集合。具体而言，当一个变量的赋值发生改变时，只有其所在的约束会发生相应的赋值变化，只有那些约束所包含的变量的边界需要更新。为方便说明，我们定义邻居变量的概念如下：

定义 3.6. 邻居变量 (Neighbor Variables) 对于变量 x，其所在的约束集合记为 C_x，那么 x 的邻居变量集合 N_x 定义为 $N_x = \{y | \exists c \in C_x, y \in Var(c) \wedge y \neq x\}$。

算法 2 Computation of best critical move and score

输入: variable x, boundary set B and starting score s_0

输出: Best critical move range *val* and score *score*

```
1: score ← −∞;
2: s ← s_0;
3: for each boundary b in B do
4:     s ← s + s_change;
5:     if s > score then
6:         score ← s;
7:         val ← b.val;
8:     end if
9: end forreturn val and score
```

在具体实现中，我们的算法为每一个变量维护一个边界集合 $S_v x$，当一次迭代带来变量 v 赋值发生变化后，我们重新计算其包含的每个约束 c 对邻居变量 v' 的新边界，然后更新到其边界集合 $S_{v'}$ 中。在所有边界更新完毕后，我们对每个邻居变量 v' 重新计算其最佳的边界和分数，以方便下一次算法迭代。算法3展示了伪代码。

下面的例子3.4展示了边界计算的更新过程。

例 3.4. 考虑约束集合 $F = \{P_1 : x^2 + y^2 \leq 1, P_2 : x + y < 1, P_3 : x + z > 0\}$。当前赋值为 $\alpha : \{x \mapsto 1, y \mapsto 1, z \mapsto 1\}$，所有约束权重为 1。则当前状态每个约束关于变量 x 的可行域和边界计算如下：

$$x^2 + y^2 \leq 1 \rightarrow x \in [0, 0] \qquad \text{Boundary: } \{< 0, \bot, \top, 1 >, < 0, \top, \bot, 1 >\}$$
$$x + y < 1 \rightarrow x \in (-\infty, 0) \qquad \text{Boundary: } \{< 0, \bot, \bot, 1 >\}$$
$$x + z > 1 \rightarrow x \in (-1, +\infty) \qquad \text{Boundary: } \{< -1, \top, \top, 1 >\}$$

排序后的边界集合为：

$$\{< -1, \top, \top, 1 >, , < 0, \bot, \top, 1 >, < 0, \top, \bot, 1 >, < 0, \bot, \bot, 1 >\}$$

当某次迭代发生赋值 $y \mapsto -2$ 时，约束 $x + y < 1$ 有不满足状态变为可满足状态。由于变量 y 和 z 不为邻居变量（没有共同存在的约束），因此我们无需更改变量 z 的边界。对于前两个同时存在变量 x 和 y 的子句，我们需要重新计算关于变量 x 的边界信息。

$$x^2 + y^2 \leq 1 \rightarrow x \in \varnothing \qquad \text{Boundary: } \varnothing$$
$$x + y < 1 \rightarrow x \in (-\infty, 3) \qquad \text{Boundary: } \{< 3, \bot, \bot, 1 >\}$$

算法 3 Incremental computation of make-break move and scores

输入: Variable v that is modified

更新: Make-break move and scores

```
1:  V ← ∅;
2:  for each clause cls that contains v do
3:      for variable v′ in cls do
4:          V ← V ∪ {v}′;
5:          boundary bd ← compute boundary of v′ with respect to cls;
6:          add bd to boundary set S_{v′};
7:      end for
8:  end for
9:  for variable v′ in V do
10:     recompute best critical move and score for v′;          ▷ algorithm 2
11: end for
```

排序后的边界集合为：

$$\{< -1, \top, \top, 1 >, < 3, \bot, \bot, 1 >\}$$

除了本章节提到的设计之外，边界信息的维护仍让存在以下几种优化空间：

- **更复杂的数据结构：**除了使用边界（线性数据结构）维护变量赋值-分数关系之外，也可以使用二叉树等顺序存储的数据结构。一般来说，当变量的边界个数很小时（SMT-LIB 中大多数样例），现行的数据结构已经足够高效。对于更多操作，二叉搜索树也许更为高效。
- **迭代优化：**在每次迭代中，由于我们只会从未满足子句中选择操作，因此我们可以只更新那些出现在未满足子句中的变量的边界信息，而不是所有的邻居变量。除此之外，我们可以使用更懒惰的办法，比如为特定需要更新的约束打标记，然后在其不可满足时才更新边界信息。

第 4 章　等式松弛机制

本章节主要介绍局部搜索算法在 LS_NRA 处理等式约束时的松弛技术，主要可以分为两个阶段：松弛阶段和恢复阶段。首先，本章节考虑到局部搜索迭代算法的效率问题，引入赋值复杂度的概念，并提出非线性实数理论独有的无理数赋值挑战。紧接着，本文提出一种等式松弛方法来暂时扩大约束的可行域，从而保证了有理数赋值，使得算法可以找到附近的近似解。然后，本文探讨了算法最后的恢复阶段，即近似解如何恢复为原问题的精确解，并提出了具体的懒惰版本的实现方法。最后，本文介绍了其他的一些可行的改进策略。

4.1　代数数赋值的复杂度

在 SMT 问题的四种主流算术理论（线性整数理论、线性实数理论、非线性整数理论、非线性实数理论）中，非线性实数理论（NRA）因为其存在高次多项式约束与潜在的无理数赋值问题而最难处理。这些情况基本发生在等式约束上，例如 $x^2 + y^2 = 2$ 等。对于局部搜索迭代而言，尽管在计算机中可以使用代数数表示无理数，但无理数赋值问题带来的仍是多项式评估的效率低下，进而影响到实根隔离与计算可行域等操作。除此之外，一些分母较大的代数数仍然会影响计算速度。

在参考文献[34,35] 中，以往的工作都尽量避免了代数数引发的计算问题，因此将非线性问题限定为多线性约束 (multilinear) 或至少包含一个线性项的等式约束上。其中，工作[34] 考虑了有理数赋值的分母大小问题，并将其作为操作得分相同时的打破平均策略。

我们的工作考虑了非线性实数的全部测试样例，一个关键的优化是尽量减少复杂赋值的出现频率。为此，我们首先定义赋值之间的复杂度关系如下：

定义 4.1. 赋值复杂度 (Complexity of values) 我们定义代数数上的偏序 $\prec_c$ 如下。$x \prec_c y$ 当且仅当以下任何一种情况成立：

- x 和 y 都是有理数，且 x 的分母小于 y 的分母。
- x 是有理数，而 y 是无理数。

当 $x \prec_c y$ 或者 $y \prec_c x$ 均不成立时，我们认为 x 和 y 的复杂度相当，记为 $x \sim_c y$。

4.2　松弛机制

我们将等式的松弛机制简单描述如下：每次当等式或者不等式约束迫使某一个变量必须赋值为一个相对复杂的代数数时，我们会暂时松弛这些约束，使用复杂度相对低的松弛解，然后以松弛的形式继续局部搜索的迭代过程。在具体实现上，我们引入以下两个参数来设置算法门槛：

- ϵ_v: 根据定义4.1衡量表示代数数复杂度，取值为 10^{-4}。
- ϵ_p: 多项式约束松弛的程度，见定义4.2，取值为 10^{-4}。

需要注意的是，在非线性实数理论中，无理数赋值并不完全由等式约束所要求，可能是多个形如 $p \geq 0$ 或 $p \leq 0$ 的约束交集所决定。因此，本文实际上讨论的是严格多项式的松弛问题。具体的松弛机制定义如下：

定义 4.2. 约束松弛 (Relaxation of constraints)

- 如果约束形如 $p = 0$，将其松弛为 $p < \epsilon_p$ 和 $p > -\epsilon_p$。
- 如果约束形如 $p \geq 0$，将其松弛为 $p > -\epsilon_p$。同样的，如果约束形如 $p \leq 0$，将其松弛为 $p < \epsilon_p$。

在局部搜索迭代中，当我们计算某个变量 v 的分数时，如果最优的分数来自于一个点区间，我们会记录这个点区间对应的边界子句标号。如果变量 v 在后续的迭代中被选中，并且其赋值比之前的赋值和 ϵ_v 的复杂度高，那么记录的等式约束和严格不等式约束都会被松弛。也就是说，我们的松弛机制是懒惰的，只有当无理数赋值十分必要时才会进行，而非直接在预处理阶段进行。当松弛之后，局部搜索算法继续迭代，并且文字和多项式的评估完全按照松弛后的形式进行。

4.3 恢复机制

当局部搜索算法找到了一个松弛形式下的“可行解”之后，这个解被称为**近似解**。事实上，这并不能确保精确解的存在，鉴于 SMT 问题的要求，我们仍然需要找到原始状态下的附近精确解。本工作主要尝试两种方法。

第一种方法是对于松弛状态下的约束进行启发式分析，尝试找到一种可以满足所有等式的变量信息。整体的分析步骤如下：

1. 如果任意一个变量目前赋值为 0，将其代入到所有的约束中，用来化简多项式的复杂项。

2. 删除所有形如 $p \cdot x + q = 0$ 等式约束中的变量 x，其中 p 在当前赋值下的评估不接近 0。

3. 最后，我们迭代地寻找只出现在一个等式约束的变量。我们将变量与对应的等式约束相关联，然后在迭代中忽略该等式（因为该等式的可满足性可以由这个变量直接决定）。

当步骤 3 中不存在等式约束时，我们尝试在上述步骤中以倒序等顺序求解变量。我们首先考虑步骤 3 中的变量-等式关联，利用这种等式约束直接求解这些变量的赋值。然后我们在步骤 2 中求解其余变量的赋值。最后我们检查所有子句的可满足性。如果等式 3 中仍然存在未解决的等式约束，或者目前的赋值仍然没有满足所有约束，我们使用以下的第二种方法求解。

第二种方法使用一种简化版本的局部搜索算法来尝试将近似解移动到最终的精确解。首先，我们把所有松弛的约束恢复为起始状态，然后继续在算术变量

算法 4 Relaxation of Equalities

Input: A set of clauses F

Output: An assignment of variables that satisfy F, or failure

```
Initialize assignment to variables;
while ⊤ do
    if all clauses satisfied then
        success ←find exact solution by analyzing structures;
        if success then return success with assignment;
        else
            Restore relaxed constraints to original form;
            success ←limited local search;
            if success then return success with assignment;
            else
                Perform major restart;
            end if
        end if
    end if
    if time or step limit reached then return failure;
    end if
    if no improvement for certain steps then
        Perform minor restart;
    end if
    Proceed algorithm as Algorithm 1;
end while
return failure
```

上运行局部搜索，直到找到了一组精确解或者局部搜索没有改进为止。相比于主流程中的局部搜索，恢复阶段的局部搜索算法是一种局限的版本，因为我们舍弃了布尔变量的迭代，以及一些随即步骤的发生。并且在实际运行中，近似解往往已经满足了绝大多数约束，因此受限版本的局部搜索算法只聚焦于几个为满足的等式约束，迭代地速度和运行效率要快很多。

算法4展示了加入了松弛机制后的局部搜索算法。和以往算法的主要区别在于，当一个变量的赋值复杂度超过了 ϵ_v 时，等式约束会被松弛。当所有的约束都被满足时（找到了近似解），根据松弛约束的结构尝试找到附近的一个精确解。如果此方法失败，我们尝试首先版本的局部搜索，直到找到一个精确解或者局部搜索没有改进为止。如果以上尝试均失败，尝试使用一种新的赋值方式重启。

实际上，我们在实验中发现两种寻找精确解的启发式方法在不同场景下有

不同的用处。第一种基于等式约束结构的方法在涉及到线性方程时表现很好。第二章方法能够很好地处理变量存在多个赋值区域的情况，但是仍然很难保证同时满足所有的不等式约束。事实上，应用很多先进的方法来精确求解等式约束十分有前景[29,31]。但是，本文提出的方法仍然可以简单地拓展到更复杂的约束中，并且独立地验证精确解的存在。

第 5 章　LS_NRA 的工具实现与实验结果

本章节主要介绍了 LS_NRA 的整体框架和算法细节。除了前面两章介绍的胞腔跳跃缓存机制和等式松弛机制，我们还为非线性问题操作的停滞设计了简单的前瞻机制。然后，我们详细讨论本工具的具体实现，包括预处理阶段、重启策略、线性方程的快速运算、变量可行域的计算以及参数设置。

5.1　启发式移动和前瞻机制

如前文所述，非线性算数理论的一个挑战就是并不总是存在单变量的关键移动来满足一个特定的约束。之前的做法更多依赖于沿着固定直线的参数方程替代[35]，更一般的做法需要用到柱形代数分解或多项式优化等理论。特别地，在 SMT-LIB 测试样例中，一类来自于生物网络[39] 的名为 Sturm-MBO 的样例覆盖了大量的非常复杂的多项式，并且规定所有变量只能取正值。当多项式包含很多变量时，目前的启发式搜索方法很难找到满足约束的赋值。

本小节提出一种新的应对这种问题的方法，并且保证每次仍然只移动一个变量。为方便说明，我们称一个文字**停滞**当该文字目前处于未满足状态，并且不存在任何的单变量关键移动使其满足。给定一个目前处于停滞状态的文字 l，我们首先在 l 中选择一个系数（根据其他变量的赋值决定）不为 0 的变量 x，然后启发式地选择一系列候选移动作为 x 接下来的赋值。对于每一个候选值来说，我们计算文字 l 在赋值后是否仍然处于停滞状态。我们优先选择那些没有处于停滞状态的候选赋值。

假定当前赋值 x_0，变量 x 的可行域为 I，启发式的移动选择包括以下几种：

1. 可行域 I 的每个区间中，靠近边界值的有理数和整数。有理数被设定为与边界值相差 10^{-4}。比如，对于可行域 $[11.2, 15.1]$，选择的移动有 $\{11.2, 12, 15, 15.1\}$。

2. 比 x_0 大于或小于的临近整数。比如，对于当前赋值 $x_0 = 13.5$，选择的移动有 $\{13, 14\}$。

3. 从区间 $[\frac{x_0}{2}, x_0)$ 中均匀地选取三个值，从区间 $(x_0, 2x_0]$ 中均匀选取三个值。

第一类操作反映了变量 x 约束的信息。第二类操作借鉴了随机游走的思想，并且优先选择简单的整数值。第三种操作是最常见的，允许在更小或者更大的分数上进行搜索。算法5总结了以上的基本思想。我们用集合 S 收集候选赋值的移动，然后循环验证集合 S 中的每一个元素。如果文字 l 在赋值之后没有陷入停滞状态，那么返回这个赋值。否则，返回集合 S 中的随机元素。

算法 5 Heuristic choice of candidate values and look-ahead for critical moves

输入: Literal l that is stuck

输出: Candidate variable v and new value x_1

1: x ←variable in polynomial l with non-zero coefficient;
2: S ←heuristic move selection for variable x;
3: **for** value x_1 in S **do**
4: **if** l has critical move after assigning x to x_1 **then return** $x, x1$;
5: **end if**
6: **end for**
7: $x_1 leftarrow$ random chosen value in S; **return** x, x_1

5.2 实现细节

LS_NRA 主要在 Z3 定理证明器上实现，并且使用 Z3 原生的多项式和代数数库。我们的工具借鉴了 Z3 中 MCSAT 算法的实现，共享了文字、子句的数据结构，但是内部算法逻辑和代码实现完全独立。下面我们将主要介绍 LS_NRA 的几个模块。

预处理阶段：预处理阶段主要负责化简主要的子句形式，以及简单的变量替换，为后续的主要搜索过程提供方便。

- **子句化简：**将同时出现的子句 $p \leq 0$ 和 $p \geq 0$ 合并为 $p = 0$。
- **变量替换：**在形如 $c\dot{x} + q = 0$ 的等式约束中，其中 c 是常数，q 是次数最多为 1 的多项式并且最多包含 2 个变量，替换掉变量 x。这里我们限制 q 的形式以降低变量替换的复杂度。

重启策略：我们设计了一个双层的重启策略，参数分别为 T_1 和 T_2，值为 100。其中一次**小重启**在 T_1 次迭代没有改进后执行，随机选择未满足子句中的一个变量修改赋值。在 T_2 次小重启之后，一次**大重启**会重置所有变量的赋值。

线性方程的快速运算: 本文的实根隔离是基于 Z3 现有的多项式操作库完成的，其原理是牛顿迭代。但是当变量在多项式中以线性项出现时，我们可以通过斜率快速计算可行域，而非使用更通用的实根隔离函数。

参数设置：本工作的可调参数设置如表5.1所示。

符号	参数说明	预设值
sp	PAWS 加权策略的概率	0.006
T_1	小重启执行所需的无提升迭代次数	100
T_2	大重启执行所需的小重启次数	100
ϵ_v	等式松弛所需的复杂度阈值	10^{-4}
ϵ_c	等式松弛的程度	10^{-4}

表 5.1 算法的可调参数设置。

Table 5.1 Tunable parameters of the algorithm.

第 6 章　实验设计及结果分析

本章节主要介绍基于前述算法设计的 LS_NRA 求解工具的实验结果，并基于此对其性能进行了分析。本章节主要可以分为三个部分：首先我们介绍实验安排和实验条件，包括测试样例、比较方法、运行环境等；然后我们给出 LS_NRA 与其他主流求解器的求解个数与时间对比；最后我们对前述算法进行了消融实验，确保本文介绍的算法的有效性。

6.1　实验设置

实验样例：本工作选取 SMT-LIB 中 QF_NRA 理论作为测试样例。测试样例大多包含来自工业问题的实例，如非线性混成自动机的分析 (hycomp)，程序终止验证的秩函数生成 (LassoRanker) 等。Kissing 样例则包含了球体吻合问题，即给定一个维度，最多可以有多少个一样的球可以和给定的球相切，并且保证两两之间不相交。通过 SMT 求解器的计算，每一个测试样例会被返回可满足 (sat)，不可满足 (unsat) 或者未知 (unknown)。在实际测试中，很多标注为未知的问题往往被其他的求解器 (Z3、cvc5) 证明为不可满足。由于局部搜索往往用于计算可满足问题，因此在实际测试中，我们首先从 SMT-LIB 中选择标注为可满足或未知的样例，然后剔除掉被其他求解器证明为不可满足的部分。SMT-LIB 包含了各种不同难度的样例，但是并没有显著标注难度等级。比如，metitarski 里面的一些样例 (来自于 MetiTarski 工具) 一般只包含很少的约束，对局部搜索和完备算法均不构成挑战。最终，我们收集了 6216 个测试样例。

实验环境：所有实验都在一台配备有 Intel Xeon Platinum 8153（2.00GHz）和 2048G RAM 的服务器上进行，系统为 Centos 7.7.1908。每一个样例设置的时间限制为 20 分钟（和 SMT 比赛相同），内存限制为 30GB。

6.2　LS_NRA 求解 NRA 样例的能力

我们在表6.1中展示了我们的工具 LS_NRA 与其他主流 SMT 求解器的性能对比。我们的工具一个主要优势是针对 Sturm-MBO 样例，一种只包含一个非常复杂度数很高的多项式的约束。目前主流的完备算法在该问题上没有求解成功。我们的工具在其他样例上表现也很突出，基本可以和主流求解器的效果持平。

除此之外，我们的算法和 Z3、cvc5 有很大的互补性。根据表6.1，一共有来自不同类别共 148 个样例仅仅可以使用局部搜索算法求解，而非 Z3、cvc5、Yices 等完备算法。更具体来说，有 291 个样例可以被局部搜索求解而非 Z3，有 378 个样例可以被局部搜索求解而非 cvc5。

我们在图6.1中描绘了 LS_NRA 与 Z3、cvc5 求解时间对比的散点图。图中每一个点的横坐标代表了求解该样例 LS_NRA 所需要的时间，纵坐标表示了其他

类别	个数	Z3	cvc5	Yices	LS_NRA （本文）	单独求解
20161105-Sturm-MBO	120	0	0	0	84	84
20161105-Sturm-MGC	2	2	0	0	0	0
20170501-Heizmann	60	2	1	0	6	5
20180501-Economics-Mulligan	93	93	89	91	87	0
2019-ezsmt	61	56	50	52	18	0
20209011-Pine	237	234	199	235	224	0
20211101-Geogebra	112	110	91	99	100	0
20220314-Uncu	74	69	62	70	70	0
LassoRanker	351	167	305	122	284	15
UltimateAtomizer	48	35	35	39	26	2
hycomp	492	307	225	227	270	17
kissing	42	33	17	10	33	2
meti-tarski	4391	4391	4343	4369	4356	0
zankl	133	70	58	58	99	26
总和	6216	5569	5475	5372	5657	151

表 6.1 LS_NRA 和其他 SMT 求解器的求解能力对比。

Table 6.1 Comparison of LS_NRA and other SMT solvers.

求解器的求解时间，虚线则说明二者在该问题上消耗的求解时间一致。我们注意到在 SMT-LIB 测试样例中存在大量相对简单的问题。为了考虑这些问题的因素，我们统计了 Z3、cvc5 和 LS_NRA 在 1 秒内解决的问题个数，共计 4765 个，这样仅仅剩下 1451 个被认为是难以求解的样例。从这点来看，LS_NRA 在求解总数、单独求解和互补性上均有优势。

我们还参考了不同求解器的求解时间对比，如图6.2所示。可以看出，几乎在任何的限制时间内，局部搜索算法的求解数量都是最多的，这说明了局部搜索的相对轻捷。随着求解时间的增加，局部搜索算法的求解数量增长速度远远快于其他求解器，这说明在充足的时间内，更多的样例可以通过多次迭代找到邻近的可行解。

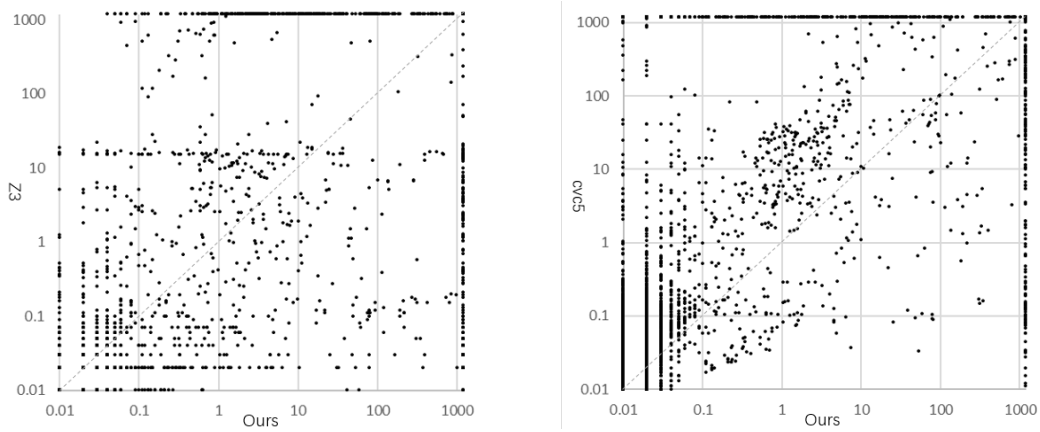

图 6.1 LS_NRA 与 Z3、cvc5 求解时间对比。

Figure 6.1 Comparison of LS_NRA and Z3, cvc5 solving time.

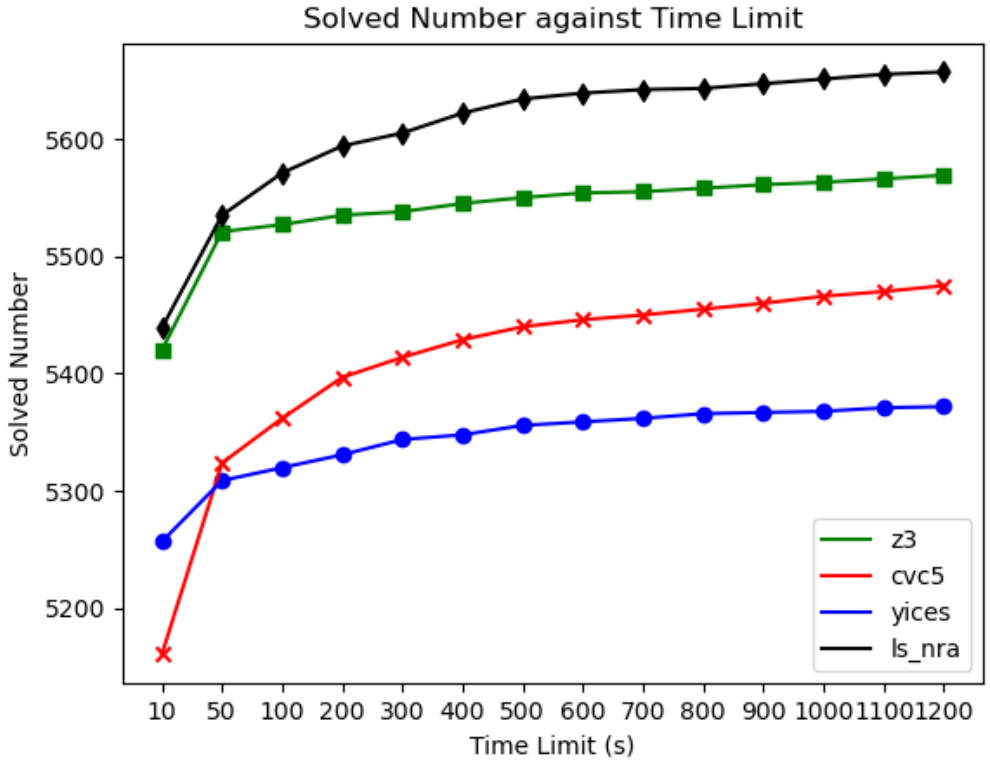

图 6.2 LS_NRA 与 Z3、cvc5 在不同时间求解个数对比。

Figure 6.2 Comparison of LS_NRA and Z3, cvc5 solving number in different time.

6.3 和其他局部搜索工作的对比

我们还讨论了 LS_NRA 与以往局部搜索工作的对比。在[34] 考虑的 979 个多线性样例中，LS_NRA 可以求解 826 个，略小于前序工作的 891 个。这种微弱的差异主要来自于[34] 更高效的实现，考虑到其工作进需要考虑有理数赋值，而非代数数，因此在数据结构设计和参数优化上更有针对性。在[35] 考虑的 2736 个样例中，我们的方法可以求解 2589 个，高于前序工作的 2246 个。事实上，我们不仅比局部搜索算法求解的个数要多，也要比前面工作用于对比的其他求解器个数还要多。我们注意到[35] 使用了不同的计算软件（比如 Maple），并且在不同机器上测试，因此数据可能略有误差。

6.4 消融实验 1：变量分数增量式计算的影响

本小节主要展示章节3中提出的变量分数增量式计算对迭代加速的影响。我们比较如下集中实现：基于边界的增量式计算 (Incremental)，不采用增量式计算的传统方法 (Naive)，以及传统方法但是限制每次考虑的子句数最大为 45(Limit-45)。相关的结果如表格6.2所示。

我们注意到不同实现方法的求解总数差异并不大，并且主要集中在 LassoRanker 类别上。LassoRanker 样例一般需要比较长时间求解，因此不同实现方法对迭代速度的影响十分重要。通过对比求解时间可以看出对于一个特定的问题而言，Naive 和 Limit-45 的方法基本需要消耗 2-10 倍于 Incremental 方法的时间，具体的倍数在不同问题上有所不同。

图6.3显示了是否使用增量式计算在不同时间限制下的求解个数。对于 1200 秒的限制来说，是否使用增量式计算对求解个数的影响并不明显，但是在更少的求解时间时（300 秒以下），更多的计算资源消耗在了实根隔离和可行域计算上，因此在迭代中缓存这些昂贵的信息尤为重要。在求解时间为 10 秒的条件下，二者的差距有 150 个以上，这说明了增量式计算对于提升求解速度的重要性。

类别	个数	Incremental	Naive	Limit-45
20161105-Sturm-MBO	120	84	84	84
20161105-Sturm-MGC	2	0	0	0
20170501-Heizmann	60	6	5	5
20180501-Economics-Mulligan	93	87	88	88
2019-ezsmt	61	18	19	15
20209011-Pine	237	224	221	221
20211101-Geogebra	112	100	99	99
20220314-Uncu	74	70	70	70
LassoRanker	351	284	262	267
UltimateAtomizer	48	26	26	26
hycomp	492	270	257	257
kissing	42	33	32	33
meti-tarski	4391	4356	4345	4345
zankl	133	99	98	98
总和	6216	5657	5606	5608

表 6.2 增量式计算对算法的影响。

Table 6.2 Impact of incremental calculation on algorithm.

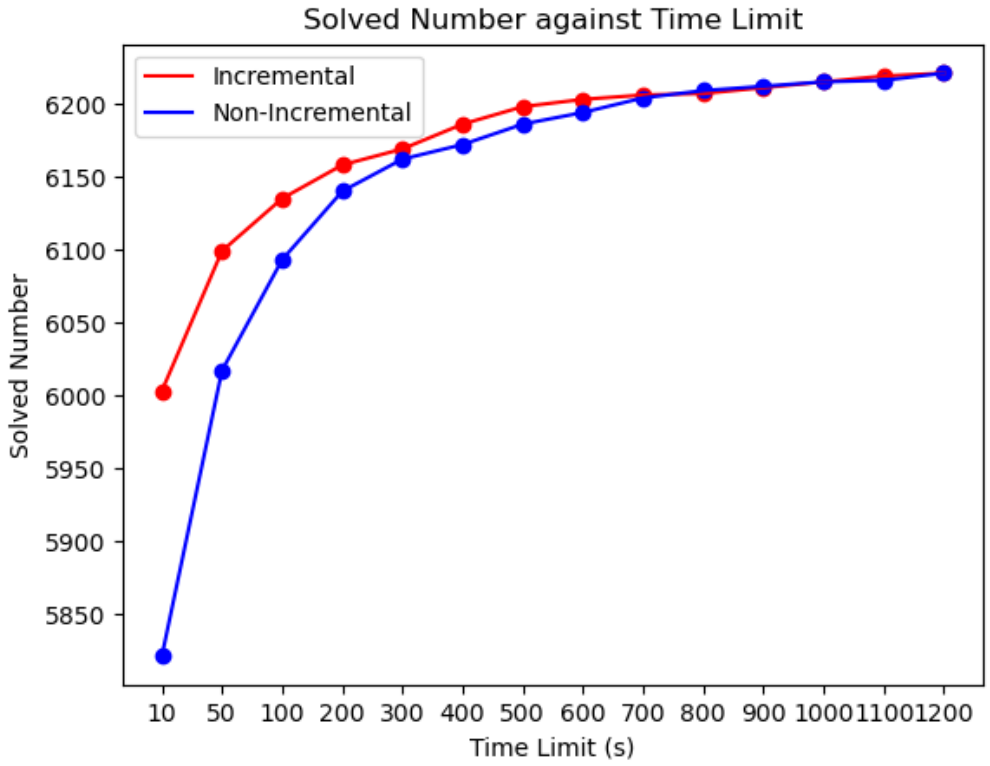

图 6.3 LS_NRA 使用增量式计算在不同时间下求解的个数。

Figure 6.3 Number of solutions found by LS_NRA with incremental calculation in different time.

6.5 消融实验 2：等式约束松弛的影响

本小节主要展示章节4中介绍的等式约束松弛对整体算法的影响。我们给出以下三种版本：使用等式松弛机制 (Relaxation)，不适用等式松弛但优先选取低复杂度的赋值 (Threshold)，不使用等式松弛并且不考虑赋值复杂度 (NoOrder)。相关结果在表6.3中展示。

试验结果表明，当把赋值复杂度考虑在内时可以保证对于大多数类别提升求解能力。

类别	个数	Relaxation	Threshold	NoOrder
20161105-Sturm-MBO	120	84	85	84
20161105-Sturm-MGC	2	0	0	0
20170501-Heizmann	60	6	9	3
20180501-Economics-Mulligan	93	87	89	86
2019-ezsmt	61	18	18	18
20209011-Pine	237	224	220	220
20211101-Geogebra	112	100	100	92
20220314-Uncu	74	70	70	70
LassoRanker	351	284	283	278
UltimateAtomizer	48	26	24	19
hycomp	492	270	204	158
kissing	42	33	31	27
meti-tarski	4391	4356	4348	4355
zankl	133	99	99	99
总和	6216	5657	5580	5669

表 6.3 等式约束松弛对算法的影响。

Table 6.3 Impact of relaxation on algorithm.

第 7 章　总结与展望

本章节主要对前面提出的创新点和工具实现进行系统的总结，归纳本工作主要的贡献，并讨论后续工作的可行性。

7.1　工作总结

SMT 问题一直是形式化验证与软件工程领域的核心问题，SMT 求解器在很多方面有着广泛的应用，很多验证算法最终都需要 SMT 求解器给出解答。非线性实数理论因为其约束的复杂性和无理数赋值的困难性，一直是 SMT 问题中的难点。以往的工作既有根据多项式胞腔理论设计的系统方法，也有从 SAT 问题借鉴而来的启发式方法，但是这些方法在实际应用中都有一定的局限性，在特定问题上表现不佳。本工作提出了一种针对非线性实数所有样例的局部搜索算法工具 LS_NRA。具体来说，其包含以下几部分组成：基于边界的胞腔跳跃分数缓存机制、等式约束松弛机制、非单变量操作的前瞻机制。实验表明，LS_NRA 在大多数情况下都能够在较短时间内找到解，并且相比于主流 SMT 求解器 (Z3、cvc5) 在高次约束上优势明显。

以往工作提出了用于算术理论的关键移动操作，针对非线性问题拓展成了胞腔跳跃操作。但是以往的工作只是针对单一子句而言，并没有基于变量级别的可行域来分析，造成了操作的冗余和迭代地低效。本工作首先提出了边界数据结构来模拟胞腔之间的间隔，基于此提出一种变量分数的增量式计算方法，而非传统的迭代计算，并且给出了边界信息的迭代条件和迭代信息，从而使得局部搜索单词计算量的下降和迭代性能的提升。

本工作的另一个创新点是针对等式约束的特殊处理。非线性实数理论是唯一一个涉及到无理数赋值的算术理论，这也是以往的局部搜索工作避开高次非线性约束的一个理由。本工作首先引入了赋值复杂度的概念，给出了赋值对迭代次数影响的分析，进而提出等式松弛机制，以暂时扩大可行域来增加赋值的可能性，从而提高了算法的求解能力。本工作还给出了两种恢复方法，即基于约束结构的传播机制和受限局部搜索，最终完成 SMT 问题精确解的搜索。

此外，本工作还探讨了实现细节。比如，本工作定义了无单变量操作的文字状态，并给出一种两种变量前后移动的前瞻机制，可以确保在给定的候选赋值中增加变量跳出停滞状态的可行性。此外，本工作还引入了两种重启机制，单一变量或所有变量的重新赋值方法，并规定在一定步数迭代没有效果时使用。最后，本工作详细阐述了 LS_NRA 的不同实现组件，包括预处理部分、线性方程的快速计算和参数的选择。

基于上述思想，本工作测试了 LS_NRA 在同一测试环境下与其他局部搜索算法和主流 SMT 求解器的求解能力。实验表明，LS_NRA 整体上要优于其他主

流方法，并且在高次约束上表现最优。经过消融实验，我们证明了以上方法的有效性，并且给出了每个组件的重要性。

7.2 未来展望

7.2.1 局部搜索算法的拓展

本工作作为局部搜索算法在算术理论的拓展，对后面的工作有很多启发作用。首先，目前的缓存机制仍然是基于边界数据结构的单变量可行域执行的，但是在实际的解空间 R^n 中，同一胞腔内每个变量的可行域都是固定的，因此更好的解决方法是能够定位到当前遍历的胞腔序号，这样可以确保重复遍历带来的可行域重复计算问题。其次，目前对于等式约束的松弛算法很基础，复杂度阈值是一个预设的常数，但是在实际搜索过程中应该区别不同难度的等式约束，即为不同难度的等式约束设置不同等级的阈值条件。再者，前序工作的多变量移动机制涉及到参数方程的方向向量选定问题，目前的工作都是基于预设的几组方向向量，而非动态调整。一个启发式的创新是使用强化学习方法实时检测不同方向对搜索状态的作用，然后实时调整多变量的搜索方向。

7.2.2 与完备算法的互补

如章节6所述，局部搜索算法往往在高次约束上表现甚好，而完备算法则在强推理性地样例上表现突出。工作[40,41] 分别提出了用于 SAT 和 SMT 问题的混合求解方法，目前已经成为 SAT-COMP 和 SMT-COMP 的主流求解方法。然而，目前的算法更多的还是和 CDCL/CDCL(L) 进行耦合，而非 MCSAT 算法。此外，本文提出的 LS_NRA 基于胞腔的操作机制可以和 MCSAT 算法中的 CAD 解释相关联，比如在局部搜索遍历一个不满足胞腔时自动学习一个相关的引理。我们将目前和潜在的技术路线归纳在图7.1中。

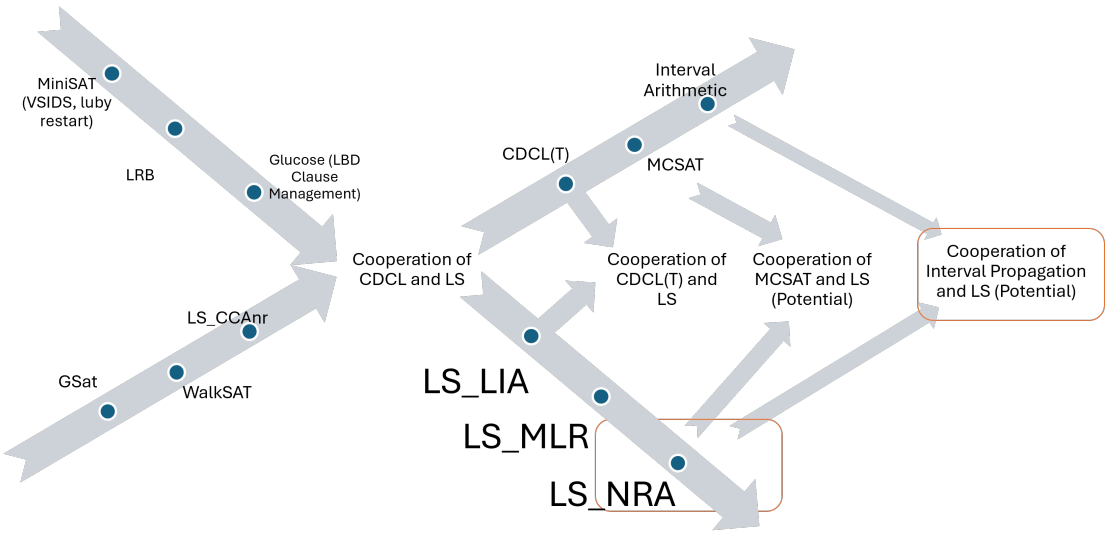

图 7.1 局部搜索算法和完备算法的主要技术路线。

Figure 7.1 Main Technical Routes of Local Search Algorithms and Complete Algorithms.

参考文献

[1] Cadar C, Dunbar D, Engler D. Klee: unassisted and automatic generation of high-coverage tests for complex systems programs [C]//OSDI'08: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation. USA: USENIX Association, 2008: 209–224.

[2] Godefroid P, Klarlund N, Sen K. Dart: directed automated random testing [C/OL]//PLDI '05: Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation. New York, NY, USA: Association for Computing Machinery, 2005: 213–223. https://doi.org/10.1145/1065010.1065036.

[3] Yao P, Shi Q, Huang H, et al. Program analysis via efficient symbolic abstraction [J/OL]. Proc. ACM Program. Lang., 2021, 5(OOPSLA). https://doi.org/10.1145/3485495.

[4] Beyer D, Dangl M, Wendler P. A unifying view on smt-based software verification [J/OL]. J. Autom. Reason., 2018, 60(3): 299–335. https://doi.org/10.1007/s10817-017-9432-6.

[5] Wang J, Wang C. Learning to synthesize relational invariants [C/OL]//ASE '22: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. New York, NY, USA: Association for Computing Machinery, 2023. https://doi.org/10.1145/3551349.3556942.

[6] Tappler M, Aichernig B K, Lorber F. Timed automata learning via smt solving [C/OL]//NASA Formal Methods: 14th International Symposium, NFM 2022, Pasadena, CA, USA, May 24–27, 2022, Proceedings. Berlin, Heidelberg: Springer-Verlag, 2022: 489–507. https://doi.org/10.1007/978-3-031-06773-0_26.

[7] Xu R, An J, Zhan B. Active learning of one-clock timed automata using constraint solving [C/OL]//Automated Technology for Verification and Analysis: 20th International Symposium, ATVA 2022, Virtual Event, October 25–28, 2022, Proceedings. Berlin, Heidelberg: Springer-Verlag, 2022: 249–265. https://doi.org/10.1007/978-3-031-19992-9_16.

[8] Amir G, Wu H, Barrett C, et al. An smt-based approach for verifying binarized neural networks [C/OL]//Tools and Algorithms for the Construction and Analysis of Systems: 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings, Part II. Berlin, Heidelberg: Springer-Verlag, 2021: 203 – 222. https://doi.org/10.1007/978-3-030-72013-1_11.

[9] Katz G, Barrett C, Dill D L, et al. Reluplex: An efficient smt solver for verifying deep neural networks [C]//Majumdar R, Kunčak V. Computer Aided Verification. Cham: Springer International Publishing, 2017: 97-117.

[10] Paulsen B, Wang C. Linsyn: Synthesizing tight linear bounds for arbitrary neural network activation functions [C/OL]//Tools and Algorithms for the Construction and Analysis of Systems: 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2 – 7, 2022, Proceedings, Part I. Berlin, Heidelberg: Springer-Verlag, 2022: 357 – 376. https://doi.org/10.1007/978-3-030-99524-9_19.

[11] Paulsen B, Wang C. Example guided synthesis of linear approximations for neural network verification [C/OL]//Computer Aided Verification: 34th International Conference, CAV 2022,

Haifa, Israel, August 7–10, 2022, Proceedings, Part I. Berlin, Heidelberg: Springer-Verlag, 2022: 149–170. https://doi.org/10.1007/978-3-031-13185-1_8.

[12] Bae K, Gao S. Modular smt-based analysis of nonlinear hybrid systems [C/OL]//2017 Formal Methods in Computer Aided Design (FMCAD). 2017: 180-187. DOI: 10.23919/FMCAD.2017.8102258.

[13] Shoukry Y, Chong M, Wakaiki M, et al. Smt-based observer design for cyber-physical systems under sensor attacks [J/OL]. ACM Trans. Cyber-Phys. Syst., 2018, 2(1). https://doi.org/10.1145/3078621.

[14] Cimatti A. Application of smt solvers to hybrid system verification [C]//2012 Formal Methods in Computer-Aided Design (FMCAD). 2012: 4-4.

[15] Leike J, Heizmann M. Ranking templates for linear loops [J/OL]. Log. Methods Comput. Sci., 2015, 11(1). https://doi.org/10.2168/LMCS-11(1:16)2015.

[16] Heizmann M, Hoenicke J, Leike J, et al. Linear ranking for linear lasso programs [C/OL]// Hung D V, Ogawa M. Lecture Notes in Computer Science: volume 8172 Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings. Springer, 2013: 365-380. https://doi.org/10.1007/978-3-319-02444-8_26.

[17] Cimatti A, Mover S, Tonetta S. A quantifier-free SMT encoding of non-linear hybrid automata [C/OL]//Cabodi G, Singh S. Formal Methods in Computer-Aided Design, FMCAD 2012, Cambridge, UK, October 22-25, 2012. IEEE, 2012: 187-195. https://ieeexplore.ieee.org/document/6462573/.

[18] Nieuwenhuis R, Oliveras A, Tinelli C. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T) [J/OL]. J. ACM, 2006, 53(6): 937-977. https://doi.org/10.1145/1217856.1217859.

[19] Jovanovic D, de Moura L M. Solving non-linear arithmetic [C/OL]//Gramlich B, Miller D, Sattler U. Lecture Notes in Computer Science: volume 7364 Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings. Springer, 2012: 339-354. https://doi.org/10.1007/978-3-642-31365-3_27.

[20] de Moura L M, Jovanovic D. A model-constructing satisfiability calculus [C/OL]// Giacobazzi R, Berdine J, Mastroeni I. Lecture Notes in Computer Science: volume 7737 Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings. Springer, 2013: 1-12. https://doi.org/10.1007/978-3-642-35873-9_1.

[21] Caviness B F, Johnson J R. Quantifier elimination and cylindrical algebraic decomposition [C]//Texts and Monographs in Symbolic Computation. 2004.

[22] Ábrahám E, Davenport J H, England M, et al. Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings [J/OL]. J. Log. Algebraic Methods Program., 2021, 119: 100633. https://doi.org/10.1016/j.jlamp.2020.100633.

[23] Li H, Xia B, Zhang H, et al. Choosing better variable orderings for cylindrical algebraic decomposition via exploiting chordal structure [J/OL]. J. Symb. Comput., 2023, 116: 324-344. https://doi.org/10.1016/j.jsc.2022.10.009.

[24] Cimatti A, Griggio A, Irfan A, et al. Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions [J/OL]. ACM Trans. Comput. Log., 2018, 19(3): 19:1-19:52. https://doi.org/10.1145/3230639.

[25] Khanh T V, Ogawa M. SMT for polynomial constraints on real numbers [C/OL]//Jeannet B. Electronic Notes in Theoretical Computer Science: volume 289 Third Workshop on Tools for Automatic Program Analysis, TAPAS 2012, Deauville, France, September 14, 2012. Elsevier, 2012: 27-40. https://doi.org/10.1016/j.entcs.2012.11.004.

[26] Tung V X, Khanh T V, Ogawa M. raSAT: an SMT solver for polynomial constraints [J/OL]. Formal Methods Syst. Des., 2017, 51(3): 462-499. https://doi.org/10.1007/s10703-017-0284-9.

[27] Fontaine P, Ogawa M, Sturm T, et al. Subtropical satisfiability [C/OL]//Dixon C, Finger M. Lecture Notes in Computer Science: volume 10483 Frontiers of Combining Systems - 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings. Springer, 2017: 189-206. https://doi.org/10.1007/978-3-319-66167-4_11.

[28] Nalbach J, Ábrahám E. Subtropical satisfiability for SMT solving [C/OL]//Rozier K Y, Chaudhuri S. Lecture Notes in Computer Science: volume 13903 NASA Formal Methods - 15th International Symposium, NFM 2023, Houston, TX, USA, May 16-18, 2023, Proceedings. Springer, 2023: 430-446. https://doi.org/10.1007/978-3-031-33170-1_26.

[29] Cimatti A, Griggio A, Lipparini E, et al. Handling polynomial and transcendental functions in SMT via unconstrained optimisation and topological degree test [C/OL]//Bouajjani A, Holík L, Wu Z. Lecture Notes in Computer Science: volume 13505 Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, October 25-28, 2022, Proceedings. Springer, 2022: 137-153. https://doi.org/10.1007/978-3-031-19992-9_9.

[30] Ni X, Wu Y, Xia B. Solving smt over non-linear real arithmetic via numerical sampling and symbolic verification [C]//SETTA 2023. 2023.

[31] Li H, Xia B, Zhao T. Square-free pure triangular decomposition of zero-dimensional polynomial systems [J]. J. Syst. Sci. Compl., 2023.

[32] Cai S, Li B, Zhang X. Local search for SMT on linear integer arithmetic [C/OL]//Shoham S, Vizel Y. Lecture Notes in Computer Science: volume 13372 Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II. Springer, 2022: 227-248. https://doi.org/10.1007/978-3-031-13188-2_12.

[33] Cai S, Li B, Zhang X. Local search for satisfiability modulo integer arithmetic theories [J/OL]. ACM Trans. Comput. Logic, 2023, 24(4). https://doi.org/10.1145/3597495.

[34] Li B, Cai S. Local search for smt on linear and multi-linear real arithmetic [C/OL]//2023 Formal Methods in Computer-Aided Design (FMCAD). 2023: 1-10. DOI: 10.34727/2023/isbn.978-3-85448-060-0_25.

[35] Li H, Xia B, Zhao T. Local search for solving satisfiability of polynomial formulas [C/OL]// Enea C, Lal A. Lecture Notes in Computer Science: volume 13965 Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II. Springer, 2023: 87-109. https://doi.org/10.1007/978-3-031-37703-7_5.

[36] Barrett C, Fontaine P, Tinelli C. The Satisfiability Modulo Theories Library (SMT-LIB) [Z]. 2016.

[37] Collins G E. Quantifier elimination for real closed fields by cylindrical algebraic decomposition–preliminary report [J/OL]. SIGSAM Bull., 1974, 8(3): 80 – 90. https://doi.org/10.1145/1086837.1086852.

[38] McCallum S. An improved projection operation for cylindrical algebraic decomposition [C]//

Caviness B F, Johnson J R. Quantifier Elimination and Cylindrical Algebraic Decomposition. Vienna: Springer Vienna, 1998: 242-268.

[39] Akutsu T, Hayashida M, Tamura T. Algorithms for inference, analysis and control of boolean networks [C/OL]//Horimoto K, Regensburger G, Rosenkranz M, et al. Lecture Notes in Computer Science: volume 5147 Algebraic Biology, Third International Conference, AB 2008, Castle of Hagenberg, Austria, July 31-August 2, 2008, Proceedings. Springer, 2008: 1-15. https://doi.org/10.1007/978-3-540-85101-1_1.

[40] Cai S, Zhang X. Deep cooperation of CDCL and local search for SAT [C/OL]//Li C, Manyà F. Lecture Notes in Computer Science: volume 12831 Theory and Applications of Satisfiability Testing - SAT 2021 - 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings. Springer, 2021: 64-81. https://doi.org/10.1007/978-3-030-80223-3_6.

[41] Zhang X, Li B, Cai S. Deep combination of cdcl(t) and local search for satisfiability modulo non-linear integer arithmetic theory [C/OL]//ICSE '24: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. New York, NY, USA: Association for Computing Machinery, 2024. https://doi.org/10.1145/3597503.3639105.

致 谢

几年的硕士生涯走到了最后，值此论文完成之际，谨向求学路上所有给予我指导、支持与陪伴的师长、亲友致以最诚挚的感谢。

首先，我要对帮助过我的各位老师表达深深的感谢，包括曾经的导师詹博华副研究员、现任导师张立军研究员。他们作为我的导师帮助我初步拟定了课题的选定、基本的研究路线，并且帮助我养成了研究生该有的成熟心态。除此之外，蔡少伟老师作为约束求解领域的专家也对我的研究给出了十分宝贵的指导意见，吴志林老师也曾在整个毕业流程中多次对我施以援手。软件所的其余导师都无形之中或多或少对我给予了专业上的帮助和指引，他们的言传身教将是我未来学术道路上的重要财富。

其次，我也要向国重实验室的各位兄弟姐妹表达感谢。作为中途换过导师的学生，我曾经流转过多个课题组，他们在很多方面都给予了我极大的帮助。詹博华老师的学生帮助我很好地熟悉了学校的环境和基本的研究思路，蔡少伟老师的学生则更多在算法设计和具体的课题指引上担任起了领路人的帮助，张立军老师的学生则在最后一学年给予了我很多就业和未来发展的关键意见。可以说，国重实验室是一个充满了爱和帮助的大家庭，我很荣幸能够成为其中的一员。

再者，我也要感谢很多形式化社区的学者和从业人员。比如，我的工作更多是基于现有的 SMT 求解器开发的，因此 Z3 和 cvc5 等主要软件的开发团队是我的技术前辈，他们是使我看得更远的巨人，也是我永远要学习的楷模。我在英国参加 VMCAI 会议时，曾和来自不同国家、不同研究背景的同龄人热烈讨论过，他们对我的意见和对我工作的欣赏是我莫大的欣慰。我也要感谢国内的形式化社区，比如中国计算机学会的形式化方法专委会，他们为我提供了很多关于形式化方法的学习资源和交流机会。当然，我最应该感谢的其实还是投稿 VMCAI 会议时的无名审稿人，他们的审稿意见让我更好地打磨了自己的作品，让我欣慰地交上了这份答卷。功利地说，如果不是他们最后给出了正面的审稿意见，可能我的论文还未发表，我也未能顺利毕业。

未来，我可能会从事不同的行业和研究方向，但是硕士期间积累的宝贵经历是我人生地巨大财富。

作者简历及攻读学位期间发表的学术论文与研究成果

作者简历：

王忠汉，男，辽宁大连人，1999 年生，中国科学院软件研究所硕士研究生。

2017 年 9 月——2021 年 6 月，在南开大学电子信息与光学工程学院获得学士学位。

2021 年 9 月——2025 年 6 月，在中国科学院软件研究所攻读硕士学位。

已发表（或正式接受）的学术论文（加星号的表示共一作者）：

1. **Zhonghan Wang**, Bohua Zhan, Bohan Li, Shaowei Cai. Efficient Local Search for Nonlinear Real Arithmetic. (VMCAI 2024)

参加的研究项目及获奖情况：

1. 参与了课题组可满足性模理论求解工具的工具开发和测试。
2. 参与了课题组交互式定理证明的讨论。