可扩展 通过分阶段优化实现浮点可满足性

YUANZHUO ZHANG，弗吉尼亚理工大学，美国ZHOULAI FU，纽约州立大学，韩国；弗吉尼亚理工大学，美国；以及石溪大学，美国BINOY RAVINDRAN，弗吉尼亚理工大学，美国

arXiv:2601.04492v1 [cs.PL] 8 Jan 2026

本文介绍了*StageSAT*，这是一种求解浮点可满足性的新方法，将 SMT 求解与数值优化相结合。StageSAT 将浮点公式重新表述为一个包含三个阶段的优化问题序列，每个阶段都具有更高的精度。它首先采用一种快速的投影辅助下降目标函数，有效引导搜索趋向可行区域；随后通过 ULP^2 优化达到位级准确率，并最终进行 *n*-ULP 格点细化，以确保正确性。按照这种构造，最终阶段使用的表示函数当且仅当候选解满足所有约束时才评估为零。因此，只要优化过程将最终阶段的目标函数降至零，所得赋值就是有效解，从而内置了可靠性保证（不会产生虚假的 SAT 结果）。为进一步改进搜索，StageSAT 通过正交投影技术为线性约束引入部分单调下降性质，避免优化器在平坦或具有误导性的目标函数景观上停滞。更重要的是，该求解器不需要复杂的位级推理，也不需要对浮点运算进行专门抽象；它将复杂运算视为黑盒，并利用运行时评估在输入空间中进行导航。

我们实现了 StageSAT，并在大量基准测试上对其进行评估，包括 SMT-COMP'25 浮点测试套件以及先前研究中的难题。在实验中，StageSAT 的可扩展性和准确性均优于最先进的基于优化的方法。在相同时间预算下，它解决的公式严格多于任何竞争求解器——事实上，StageSAT 找到了基准测试中大多数可满足实例，并且从未为不可满足公式生成伪模型。在我们的基准测试中，它对可满足实例的召回率达到 99.4%，错误判定为可满足的比例为 0%，可靠性超过了我们测试过的先前基于优化的求解器。与传统的位精确 SMT 求解器和早期数值求解器相比，StageSAT 还实现了显著加速（通常快 5–10× 倍）。这些结果表明，我们的分阶段优化策略能够显著提升浮点可满足性求解的性能与正确性。为便于复现，我们在投稿系统中将匿名化制品（实现、基准测试和评估结果）作为补充材料提供。

1 引言

基于优化的浮点求解：传统的浮点（FP）算术 SMT 求解器在可扩展性方面往往表现不佳，尤其是处理包含非线性或超越运算的公式时。一种替代方法是将浮点可满足性问题转化为数值优化问题。在这种范式下，浮点公式（一组约束）会被转换为数值目标函数 R(x)，它表示与满足条件的距离。该目标函数非负，当且仅当候选赋值 x 是真正的解（即满足所有约束）时，其值才为零。因此，判定可满足性就转化为一项优化任务：最小化 R(x)，并检查最小值是否达到零。XSat 等先驱求解器[12],goSAT[3],和 Grater[6] 通过将浮点约束编码为此类目标函数，再运用全局优化技术寻找零值，展现了这一策略的潜力。这类基于优化的求解器的主要优势在于，它们可以将复杂算术视为黑盒函数——无需进行位爆破，也无需显式枚举 IEEE-754 语义的复杂细节。相反，求解器会根据 R(x) 提供的数值反馈进行搜索：随着赋值 x 越来越接近满足公式，R(x) 会逐渐减小；当 R(x) 达到 0 时，即找到解。只要能够计算任意数学函数（例如正弦或余弦等超越函数），就可以对其进行推理；与传统 SMT 方法相比，这是一个显著优势。

作者联系方式：YUANZHUO ZHANG，弗吉尼亚理工大学，美国；Zhoulai Fu，纽约州立大学韩国校区和弗吉尼亚理工大学，美国，以及石溪大学，美国；BINOY RAVINDRAN，弗吉尼亚理工大学，美国。

挑战：早期基于优化的浮点求解器揭示了两个关键挑战。首先，由这种转换得到的目标函数可能不规则且不光滑，从而影响运行时间性能，并可能误导数值搜索方法（基于梯度的方法或随机优化器），使其陷入局部最小值或平坦平台。实践中，即使存在解，求解器也可能无法找到，因为搜索可能会“卡”在目标函数没有明显下降的区域。例如，据观察，XSat 会因最小化不足（陷入远离全局最小值的位置）而将可满足实例错误标记为不可满足；Grater 则避免了错误的不可满足判断，但经常返回许多超时结果，其中一些后来被证实是可满足的。

其次，浮点运算的有限精度以及粗略的步长调整会损害求解器正确性，并迫使求解器效率与正确性之间做出权衡。由于计算 R(x) 时使用有限精度运算并会引入舍入误差，基于优化的求解器（如 Grater）会采用较小的容差阈值来判断公式是否满足[5]，当 $R(x) < tol$ 时便宣告成功，尽管从数学上说，只有 $R(x) = 0$ 才对应一个真实模型。然而，容差过于宽松可能会产生伪解，例如错误地接受 $x \approx 0$ 满足诸如 $x > 0 \wedge x \leq 1e\text{-}160$ 这样的约束。反过来，XSat 完全不采用容差，只有当 $R(x) = 0$ 精确满足时才接受候选解；但如果其步长策略无法将候选值从 $x = 0$ 调整到一个极小的非零值，例如 $x = 1e\text{-}200$，它仍可能错过有效解。

这些挑战意味着，早期基于优化的解决方案在实践中往往难以扩展且不够准确。由于无法找到锯齿状目标函数的全局最小值，它们可能将可满足公式判定为不可满足（或未知）；反过来，也可能因浮点误差和位精度不足，将不可满足公式判定为可满足。值得注意的是，XSat 通过以末位单位（*ULP[15]*）衡量约束违反，在一定程度上解决了精度问题。末位单位是两个可表示浮点数之间的最小差值，因此为其目标函数带来了位级严谨性。然而，纯粹基于 ULP 的目标函数景观具有极强的不连续性，难以搜索，也因此加剧了可扩展性问题。

我们的方法：StageSAT。我们提出了 StageSAT，这是一种新型浮点 SMT 求解器，通过分阶段优化解决准确率和可扩展性方面的问题。其核心思想是有意将目标函数和搜索过程设计为多个阶段，兼顾有效全局搜索所需的平滑性与保证正确性所需的精确性。StageSAT 旨在既能稳健地处理大型基准测试，又能给出精确结果——只有找到正确模型时才报告 SAT；而只有在最后阶段经过穷举搜索后仍表明模型很可能不存在时，才报告 UNSAT-GUESS（即未经证明的启发式 UNSAT 结果）。

StageSAT 的设计融合了多项关键技术，以实现这种平衡：

– **多阶段目标设计：**StageSAT 并非采用单体目标函数，而是分三个阶段逐步细化目标函数。首先使用平滑平方误差项，快速引导搜索进入有希望的区域。随后转为更精确的 1-ULP差异度量（其中 ULP，即末位单位，表示浮点值中的最小步长），以进一步逼近位级正确性。最后，在最后一个阶段应用*n-ULP*裕度细化*[13]*，逐步将允许的误差裕度收紧至零。这种分阶段方法在早期阶段保留了搜索的可行性，同时最终实现位精确的可满足性——只有当最终目标函数在 IEEE-754 语义下真正达到零时，StageSAT 才会报告“SAT”。

– **正交投影：**为缓解线性等式或平坦曲面区域中的误导性梯度，StageSAT 采用正交投影技术[16]。本质上，在优化 $a \cdot x + b = c$ 这类约束时，若它们在目标函数景观中形成平坦的“谷地”，StageSAT 会将搜索步长正交投影到约束曲面上，以便更准确地估算达到可满足状态的真实距离。这能更好地对齐输入空间中的移动

与目标函数下降之间的关系，从而切实避免在平台区域停滞或错误地认为已收敛。

– **目标函数平滑：** StageSAT 在每个阶段都会对目标函数进行轻量级平滑处理（例如，使用误差项的平方形式或微小扰动），以减少不稳定行为。通过“磨平”尖锐拐角并尽早消除零梯度陷阱，这种平滑处理让目标函数景观更易于搜索，从而使数值优化器更可靠地趋向全局最小值。

理论上，StageSAT 对可满足性而言是可靠的：其最终目标函数阶段是位精确的，因此只有在候选解通过严格的位级检查（$R(x) = 0$ 完全精确，即该赋值是有效模型）时，才会报告 SAT。对于优化器收敛到非零最小值的情况，StageSAT 会判定该公式不可满足（但不会提供形式化证明）。尽管这一 UNSAT 结果属于启发式的“有把握的猜测”（因为 StageSAT 与其他数值求解器一样是不完备的），但在我们的所有评估中，它在经验上都准确无误。如果 StageSAT 未能在给定时限内找到解，则会返回未知/超时。

结果： 在实践中，StageSAT 兼具准确性与可扩展性。我们在 SMT-COMP’25[22] 无量词浮点 QF_浮点基准测试集[28],上评估了 StageSAT，其中包括“中型”和“大型”基准测试套件，以及 XSat、Grater 和 JFS 使用的基准测试集[20]。StageSAT 与四种最先进的位精确 SMT 求解器（Z3[9],MathSAT[7], Bitwuzla[24], CVC5[2]）以及四种基于优化的求解器（XSat、goSAT、Grater、JFS）进行了比较[3, 6, 12, 20]。结果表明，StageSAT 具有显著优势：

– 在“中型”基准测试（35 个公式）上，StageSAT 解出了全部 35 个实例（可满足和不可满足），运行时间中位数仅为 0.18 秒。– 在更具挑战性的“大型”基准测试上，StageSAT 解出了 26 个可满足实例中的 24 个，并对 21 个不可满足实例返回了不可满足猜测（与已知真实结果一致）。总体而言，StageSAT 的覆盖率最高；由于超时率低，它解出的公式严格多于任何竞争求解器。– 与最佳 SMT 求解器（如 Bitwuzla 和 CVC5）相比，StageSAT 在可满足情况下的运行时间中位数最多快 *10×*，并避免了许多超时，解出了这些求解器无法解出的若干公式。– 与先前的基于优化的求解器相比，StageSAT 的可靠性和性能均显著提升。在 347 个可满足实例中，它仅漏掉两个，SAT 召回率达到*99.4%*；对于不可满足情况，其错误判定为可满足的比例为 *0%*（从未报告伪模型）。而 goSAT 和 Grater 等工具经常报告未知，或出现错误分类。在我们的实验中，StageSAT 的平均速度也比 XSat、goSAT 和 Grater 快约一个数量级。

为了解每项设计决策的影响，我们对 Stage- SAT 的各个组件（目标分阶段策略、正交投影和平滑）进行了消融研究。移除任何一个组件都会显著降低性能或准确率，这证实了每个组件对求解器的有效性都至关重要。尽管 StageSAT 仍然是不完备的（与其前身一样，它无法生成形式化 UNSAT 证明），但我们的结果表明，数值优化可以成为传统 SMT 求解的有力补充，尤其是在处理规模和运行时间至关重要的困难浮点公式时。

贡献。总而言之，本文的贡献如下：

– **ULP 分阶段目标设计：** 一种多阶段目标策略，先采用平滑的平方距离度量，再过渡到 1 ULP 精度度量，最后进行 n-ULP细化。这种设计在早期阶段实现稳健的全局搜索，并且在最后阶段实现位精确 SAT 检测，从而解决 SMT 求解中的可扩展性和浮点精度问题。

– **通过正交投影改进下降过程：** 一种在优化过程中对线性约束应用正交投影的技术。这能改善朝满足赋值方向的单调下降，并帮助求解器避免停滞在具有误导性的目标函数景观上。

– **增强稳定性的平滑处理：**采用轻量级目标函数平滑处理（例如，对误差项平方），以改善不连续点或零梯度陷阱区域的收敛性。平滑处理可使优化过程更加稳定，从而提高可靠性并加快速度。

– **广泛评估：**在 SMT-COMP'25 QF_FP 基准测试上进行了全面的实验评估，结果表明，StageSAT 在求解的基准测试数量和总体运行时间方面均优于以往的不完备求解器。StageSAT 还能很好地补充完备求解器，以短得多的时间给出一致的结果。我们还进行了消融研究，证实了各项所提组件的必要性。

2 概述与示例

研究范围与目标。我们的目标是 QF_FP（无量词浮点）理论。给定一个由 IEEE-754 原子构成的合取范式公式 C，我们旨在实际应用中准确判断其可满足性：(1) 若 C 可满足，则返回 **可满足**，并给出一个在 IEEE-754 下有效的见证赋值 $\vec{x}^{\star}$；(2) 若优化以严格正的最小值终止，则返回 **不可满足猜测**——这并非证明，而是由评估结果支持的可信判定结果；(3) 若时间预算耗尽且仍未作出判定，则返回 **超时**。

三阶段概览。图1 概述了StageSAT——一种分阶段求解器，在可搜索性与位精度之间取得平衡，从平滑引导逐步过渡到位精确表示和离散吸附。此处仅简要介绍数学内容；精确的目标函数定义、弱距离契约、S1 的局部单调下降保证以及 S3 的离散细化策略，请参阅 §3 。

– **S1——投影辅助平方目标函数（快速下降）**。我们首先通过快速但较粗略的优化，朝可行方向推进。我们将约束分为$\mathcal{L}$（线性等式）和$\mathcal{N}$（非线性/不等式），并构造如下形式的目标函数 $R_{\mathcal{L}} + R_{\mathcal{N}}$。其中 $R_{\mathcal{N}}$ 是平方残差/违约量之和（与传统的基于优化的编码方式相同），而 $R_{\mathcal{L}}$ 则利用正交投影到线性流形上，精确计算距离。该投影解决了示例2.1中所示的非单调行为，从而产生部分单调下降效果，并带来稳健的早期优化轨迹。

– **S2——ULP² 目标函数（位级对齐）**。随后，我们改为对每个约束使用平方 *ULP* 距离，使目标函数与 IEEE-754 语义保持一致。平方处理会塑造惩罚项（放大较大的差距、打破平局），从而实现稳定的由粗到细下降，并揭示基于数值大小的 S1 可能遗漏的位级问题（例如次正规数下溢）。

– **S3 — n-在浮点格上进行 ULP 细化**。 最后，我们在浮点数格上围绕 S2 的当前最优解进行有界的离散搜索，将连续搜索无法跨越的近似匹配变为精确零值。

结果与保证。

StageSAT 会给出以下判定结果之一：{**可满足**、**不可满足猜测**、**超时**}。若结果为**可满足**，则会附带一个经过验证的模型（符合 IEEE-754 位精确表示）。若结果为**不可满足猜测**，则表示非证明性判定结果（最小值为正）；实验表明，这一结果与完备求解器高度一致，同时可扩展至更大的实例（见 §5）。若结果为**超时**，则表示在给定预算内既未找到模型，也未达到稳定的正最小值；实践中，在 §5所述预算下，这种情况并不常见。

2.1 示例

直观理解。 我们将任意约束视为合取式 C ≡ L ∧ N，其中 L 汇集了关于浮点变量的线性等式（在几何中视为 $\mathbb{R}$），N 则包含其余

S1（投影辅助的平方距离） → **S2（ULP²）** → **S3（浮点格上的*n*-ULP细化）**。
结果：**可满足**（经验证的模型）/ **不可满足猜测**（最小值为正；非证明）/ **超时**。

图 1. StageSAT的三阶段求解流程。S1 通过对线性等式进行投影实现快速下降；S2 通过平方 ULP 距离保证位级正确性；S3 在浮点格上进行离散细化。形式化目标函数和保证见第 §3节。

非线性和/或不等式原子。线性等式常见于控制和信号处理模型中。

我们的设计原则很直观：随着目标函数值下降，赋值应当更接近模型集 $\mathrm{M} = \{\vec{x} \mid \vec{x} \models \mathrm{C}\}$。如果我们能够计算到模型集的欧几里得距离 $d(\vec{x}, \mathrm{M})$，那么将该函数最小化至零便可直接解决可满足性问题，并且会自动满足我们的设计原则。在实践中，现有的基于优化的求解器会用代理量 $R(\vec{x})$ 替代这一理想但难以计算的距离，该代理量满足两个基本性质：(i) $R(\vec{x}) \geq 0$ 且 (ii) $R(\vec{x}) = 0 \Rightarrow \vec{x} \models \mathrm{C}$。此类代理量有助于搜索，但未必与几何距离一致；正如我们的玩具示例所示，它们可能违反预期的“越近 $\Longrightarrow$ 越小”行为，并误导优化过程。

我们的关键观察是，对于约束的线性等式部分 $\mathrm{L} = \vec{x} \mid A\vec{x} == b$，无需求解方程，就能计算到解流形的精确平方距离：将 $\vec{x}$ 正交投影到 L 上（通过摩尔–彭若斯伪逆 [27]），再测量其与该投影之间的平方距离。这样便得到一个“到 L 的距离”闭式项，可直接在线性部分上贯彻单调下降原则。

我们的阶段 1 将这一几何项与剩余原子 N（非线性和/或不等式）的简单表示函数结合起来：

$$S_1(\vec{x}) = d(\vec{x}, \mathcal{L})^2 + R_N(\vec{x}).$$

其中，当且仅当 N 中的所有原子在 $\vec{x}$处都成立时，$R_N \geq 0$和 $R_N(\vec{x}) = 0$。因此，阶段 1 保持可靠性约定（非负；仅在真实模型处为零），同时保证目标函数的任何严格下降**必然会缩短到** L 的距离。由于投影和评估都以浮点方式执行，阶段 1 用于引导搜索，而不是用于证明 SAT。

后续阶段提供了位级决策能力。阶段 2 切换为与 IEEE 对齐的平方 ULP 表示函数，当且仅当所有文字作为浮点数均得到满足时，该函数才返回零。阶段 3 则在阶段 2 的当前最优解周围，于浮点格上执行有界的 (n) ULP 精化；当连续步骤无法跨越最后一个 ULP 间隙时，将近似命中的候选解吸附到精确模型。因此，整体策略是：在可行之处衡量真实距离（线性部分），并在必要之处使用忠实于 *IEEE* 的目标函数（基于 ULP 的阶段），从而确保目标函数的下降可靠地对应于向模型靠近。

这里将展示朴素选择如何违背单调下降原则，以及投影如何解决这一问题；随后解释为何每个阶段对于高效地做出忠实于 IEEE 的决策都必不可少。

从 f_{naive} 到 S_1：为何需要投影。玩具约束。 $x == 1 \ \wedge \ y == x$。唯一的模型是 (1,1)；线性流形是单点 (1,1)。

朴素目标函数及其失效。一种自然的初步尝试是

$$f_{\text{naive}}(x, y) \ = \ (x - 1)^2 \ + \ (y - x)^2.$$

它衡量两个方程的残差，但与我们的单调下降意图不 一致：从 (2，2) 移动到 (2，1) 时，按欧几里得距离计算确实更接近 (1，1)，但 f_{naive} 却增加了。直观地说，$(y - x)^2$ 会惩罚沿流形切向方向的移动，因此即使点正在接近模型，目标函数也可能上升。

点 $(x,\ y)$	到 $(1,\ 1)$ 的距离	$\mathbf{f}_{\text{naive}}(\mathbf{x},\ \mathbf{y})$
(2, 2)	1.414	1.000
(2, 1)	**1.000**	**2.000**
(1, 1)	0.000	0.000
(0, 1)	**1.000**	**2.000**
(0, 0)	1.414	1.000

投影：让代理量本身成为距离。在 L 上，我们衡量的是到可行集的距离，而不是到其定义方程的残差。具体来说，对于这个玩具示例，任意 $(x,\ y)$ 到线性流形的正交投影都是 $(1,\ 1)$。阶段 1 的目标函数变为

$$\boxed{S_1(x,y)\ =\ (x-1)^2\ +\ (y-1)^2.}$$

由于 S_1（按构造）是到线性解集的平方距离，因此 S_1 中的任何严格下降都必然意味着赋值向该解集更近了一步。这消除了 f_{naive} 中的切向伪影，并恢复了可预测的下降过程。

为什么 *S1* 不声明 *SAT*。S1 是一个搜索阶段。投影和距离是在浮点数中计算的，因此一个极小的近零值可能是舍入伪影。即使 S_1 在数值上很小，我们也不会在 S1 阶段认证 SAT。相反，S1 为位精确阶段提供一个强有力的初始值：
– *S2* 使用成对 ULP^2，当目标函数达到零时，以精确方式判定 SAT。– *S3* 在 S2 的当前最优解周围执行微小的 *n*-ULP 格点细化，将近似匹配调整为精确相等（SAT），否则生成不可满足猜测或超时。

阶段 *2*（*S2*）：平方 *ULP* 目标——位精确的表示函数。在 S1 使我们接近可行后，阶段 2 转而采用与 IEEE-754 一致的位级视角。直观来说，ULP$(a,\ b)$ 计算可表示的浮点值（格点上的“步进”）之间相隔多少步；仅当 a and b 完全相等时，结果才为 0。

– 等式文字 $f(\vec{x})==g(\vec{x})$：距离为 ULP$(f(\vec{x})\ g(\vec{x}))$。– 不等式文字 $f(\vec{x})\ \le g(\vec{x})$：不等式成立时取 0；否则，取使其成立所需的最小 ULP 步进数（并按标准方式通过 ±1修正严格性）。– 子句（析取）：将其中各文字的平方 ULP 距离相乘，使任何一个成立的文字都能令乘积归零；S2 的完整目标函数对各子句的乘积求和。

对于这个玩具示例（每个等式各自构成一个子句），具体的 阶段 2 目标函数为

$$\boxed{S_2(x,y)\ =\ \mathrm{ULP}(x,1)^2\ +\ \mathrm{ULP}(y,x)^2.}$$

根据构造，$S_2(\vec{x})\ \ge 0$ 和 $S_2(\vec{x})\ =0$ 当且仅当 所有约束在 IEEE-754 中都精确地 成立时；也就是说，S_2 是 C 的表示函数 。因此，当其最小值达到 0 时，*S2* 可以声明为可满足 。

近似命中，$\to$ 但不精确。 假设 S1 得到 (x,y)，其中 x 比 1 高 1 个 ULP，y 比 x高 2 个末位单位。那么 $S_2\ =1^2+2^2\ =5\ >0$。S2 会继续调整 (x,y)，直到两个差距都归零；对于这个玩具示例，它通常会达到 (1,1)，使 $S_2(1,1)\ =0$ 并报告可满足。

阶段 *3*（*S3*）：*n-ULP* 细化——通过离散搜索确定最后一位。 即使采用了感知 ULP 的优化，连续方法也可能停滞在距离等式几个 *ULP* 之处——尤其是在次正规数附近或等式链很紧密时——因为连续步进无法跨越最后的离散间隙进行“吸附”。阶段 *3*通过在 S2 当前最优解 (x_2^*,y_2^*) 附近的浮点数格上进行微小的离散搜索 来解决这一问题。

令 nULP$(k,\ z)$ 将浮点数 z移动 k个末位单位（正数：nextUp；负数：nextDown）。我们在步进点上计算相同的 ULP 惩罚，并针对极小的整数偏移量 m（对于 x）和 n（对于 y）进行最小化：

$$\boxed{S_3(m,n)\ =\ \mathrm{ULP}\big(\mathrm{nULP}(m,x_2^*),\ 1\big)^2\ +\ \mathrm{ULP}\big(\mathrm{nULP}(n,y_2^*),\ \mathrm{nULP}(m,x_2^*)\big)^2.}$$

如果小范围邻域内的最优值达到 0，S3便宣告可满足，并给出相应的步进赋值；如果最小值仍严格大于 *0*，则返回不可满足猜测（这并非证明，但根据经验准确度很高）；如果预算先耗尽，则返回超时。

S3 为何不可或缺（直观解释）。在一个包含 20 个变量、以一个极小值 $x_1{=}c$为锚点的链中，$x_2{=}x_1$、$x_3{=}x_2$、……、$x_{20}{=}x_{19}$，S1 会将变量拉向仿射流形，S2 则使其与位级相等对齐；但一两个坐标仍可能相差几个 ULP——在这样的尺度下，连续更新会扰动相邻变量或越过目标。S3 的有界格点搜索（例如在 {−1、0、1} 中取步长）会联合微调末位，并将整个向量吸附到精确的格点上；只要模型存在，就能将目标函数降至 0。

要点。将 f_{naive} 替换为基于投影的距离 S_1，即可恢复 L 上预期的“越近$\Longrightarrow$ 越小”几何关系，并得到稳健的初始值。*S2* 提供一个忠实于 *IEEE* 的ULP²表示函数，当其值达到零时即可据此确认 SAT。*S3* 则通过添加一个小幅度的 *n*-ULP格点细化，弥合平滑优化停滞时剩下的末位差距。整个流程 $S_1 \to S_2 \to S_3$能够带来可预测的进展，在可能时精确判定可满足性，否则则能给出有依据的不可满足猜测或超时。

3 技术方法与理论

本节将精确说明 StageSAT 的工作原理及其有效性的原因。我们对三个阶段进行形式化描述，并确立三个结论：**(i) 阶段 1** 在**线性**部分保证**部分单调下降**；**(ii)阶段 2**和**阶段 3** 是约束集的**表示函数**；(iii) 整体程序是**可靠的**（不会错误判定为可满足），但必然**不完备**。这些结果解释了 StageSAT 在实践中的**准确率**和**可扩展性**。

3.1 无量词_浮点数设置中的记法

语言。我们考虑无量词浮点数（IEEE-754）公式。项由浮点变量和常量通过标准 IEEE-754 算术构成；原子的形式为 $e_1\ \mathrm{op}\ e_2$，其中 $\mathrm{op} \in \{==、\ \le、\ <、\ \ge、\ >\}$。允许混合精度（FP32/FP64）；每个原子都按照其声明的格式求值。

合取范式与模型。设 C 为一个合取范式：由子句构成的合取 ϕ（每个子句都是文字的析取）。若在 $\vec{x}$ 上按照 IEEE-754 求值时所有子句均为真，我们记作 $\vec{x} \models \mathrm{C}$。模型集为 $\mathrm{M}_C := \{\vec{x}: \vec{x} \models \mathrm{C}\}$。为简便起见，我们用 $\sum_{\phi\in C}(\cdot)$（或简写为 $\sum_{\phi}(\cdot)$）表示对各子句求和。

点到集合的欧几里得距离。 对于 $S \subseteq \mathbb{R}^d$，其（欧几里得）距离为

$$d(\vec{x}, S) \;:=\; \inf_{\vec{y}\in S} \|\vec{x} - \vec{y}\|_2.$$

当 S 是一个仿射集（例如线性流形）时，下确界在 $\vec{x}$ 到 S的正交投影处取得。

我们将正式的目标函数定义保留在各阶段内部，以避免冗余并突出其不同作用。

3.2 阶段 1（S1）：投影辅助平方目标函数与部分单调下降 nt

令 $\mathrm{L} = \{\vec{x}: A\vec{x} = b\}$ 表示从 C 中提取的线性等式，并令 N 表示其余原子（非线性等式和所有不等式）。

文字和子句的平方距离（$\vee$中的乘法）。对于文字 ℓ：– 等式 $h(\vec{x}) == 0$：$dist_{\square}(\ell; \vec{x}) := h(\vec{x})^2$；– 不等式 $f(\vec{x}) \le g(\vec{x})$：$dist_{\square}(\ell; \vec{x}) := max\{0, f(\vec{x}) - g(\vec{x})\}^2$。

对于一个子句 ϕ（文字的析取），我们通过乘法对 $\vee$ 进行编码：

$$\mathrm{Dist}_{\square}(\phi;\vec{x}) \;:=\; \prod_{\ell\in\phi} \mathrm{dist}_{\square}(\ell;\vec{x}),$$

因此，如果 ϕ 中的任意文字得到满足，则有一个因子为 0，且子句距离为 0（如 XSat 中所示）。令 $\mathsf{C}_{\mathcal{N}}$ 为仅由 N 中的原子构成的子句集合。*S1*目标函数为

$$S_1(\vec{x}) \;=\; \underbrace{\|\vec{x}-\Pi_{\mathcal{L}}(\vec{x})\|_2^2}_{\text{exact squared distance to } \mathcal{L}} \;+\; \sum_{\phi\in C_{\mathcal{N}}} \mathrm{Dist}_{\square}(\phi;\vec{x}).$$

投影（定义与闭式表达）。

$$\Pi_{\mathcal{L}}(\vec{x}) \;=\; \vec{x} - A^{\top}\big(AA^{\top}\big)^{\dagger}(A\vec{x}-\vec{b}), \qquad \|\vec{x}-\Pi_{\mathcal{L}}(\vec{x})\|_2^2 \;=\; \Big\|A^{\top}\big(AA^{\top}\big)^{\dagger}(A\vec{x}-\vec{b})\Big\|_2^2,$$

其中 $(\cdot)^{\dagger}$ 摩尔–彭若斯伪逆[27]。

单调　下降性质（*MDP*）。对于目标函数 R，我们称 R 对于C 满足单调下降，当且仅当

$$\boxed{R(\vec{x}') \;<\; R(\vec{x}) \implies d(\vec{x}',\,\mathcal{M}_C) \;<\; d(\vec{x},\,\mathcal{M}_C).}$$

引理***1***（投影事实）。 $\Pi_{\mathcal{L}}(\,\vec{x})$ 是 L 中距离 $\vec{x}$ 最近的唯一点，并且 $\|\vec{x}-\Pi_{\mathcal{L}}(\vec{x})\|_2 = d(\,\vec{x},$ L)。上述闭式成立。

引理***2***（当$\mathcal{N}$ 成立时 ***S1*** 的部分 ***MDP***）。若 $\vec{x}\models\mathcal{N}$ 且 $\vec{x}'\models\mathcal{N}$，则

$$S_1(\vec{x}') \;<\; S_1(\vec{x}) \iff \|\vec{x}'-\Pi_{\mathcal{L}}(\vec{x}')\|_2^2 \;<\; \|\vec{x}-\Pi_{\mathcal{L}}(\vec{x})\|_2^2,$$

因此 $d(\,\vec{x}',\ \mathrm{M}_C) \;<\; d(\,\vec{x},\ \mathrm{M}_C)$，因为在 N 条件下，（局部）模型集与 L 重合。解释。当非线性/不等式部分已经成立时，S1 的下降恰好反映了向模型集移动的过程。S1 提供了一个很好的起点；它不用于判定可满足。

3.3 阶段 2（S2）：平方 ULP 目标函数是表示函数

现在，我们使目标函数与 IEEE-754 语义保持一致，同时通过在每个析取内部相乘来遵循合取范式。

文字和子句的 *ULP*距离（ $\vee$ 中的乘法）。对于文字 ℓ：
– 等式 $f(\vec{x}) == g(\,\vec{x})$： $d_{\mathrm{ulp}}(\ell;\,\vec{x})$ 是 $f(\vec{x})$ 与 $g(\,\vec{x})$ 的 IEEE 值之间相邻浮点步数（通过 IEEE nextUp/nextDown）；若二者位相等，则为 0；– 不等式 $f(\,\vec{x}) \le g(\vec{x})$：若成立，则为 $d_{\mathrm{ulp}}(\ell;\,\vec{x}) = 0$；否则为使其成立所需的最少浮点步数（仅用作距离）。

对于子句 ϕ，定义

$$\mathrm{Dist}_{\mathrm{ulp}}(\phi;\vec{x}) \;:=\; \prod_{\ell\in\phi} d_{\mathrm{ulp}}(\ell;\vec{x})^2,$$

因此，满足的文字会使乘积为 0。

表示函数和 *S2* 的定义。标量 R 是适用于C 的表示函数，当且仅当

$$R(\vec{x}) \;\ge\; 0 \quad\text{and}\quad R(\vec{x}) = 0 \iff \vec{x}\models C.$$

我们设

$$S_2(\vec{x}) \;:=\; \sum_{\phi\in C} \mathrm{Dist}_{\mathrm{ulp}}(\phi;\vec{x}).$$

引理 ***3***（***S2*** 是表示函数）。 S_2 是 C 的表示函数。理由。每个子句距离均为非负数，且当且仅当该子句得到满足时为 0；这些距离之和为 0，当且仅当所有子句都得到满足，即 $\vec{x} \models \mathrm{C}$。

3.4 阶段 3（S3）：n-ULP细化表示函数（离散步进）

令 $\vec{x}_2^\star$ 为*S2* 返回的当前最佳赋值。对于整数向量 $\vec{n}$（通过 IEEEnextUp/nextDown 逐分量应用的有符号 ULP 步进），定义步进点处的子句距离，并对所有子句求和：

$$\mathrm{Dist}_{\mathrm{ulp}}\big(\phi;\ \mathrm{nULP}(\vec{n}, \vec{x}_2^\star)\big) \ := \ \prod_{\ell \in \phi} d_{\mathrm{ulp}}\big(\ell;\ \mathrm{nULP}(\vec{n}, \vec{x}_2^\star)\big)^2,$$

$$S_3(\vec{n}) \ := \ \sum_{\phi \in C} \mathrm{Dist}_{\mathrm{ulp}}\big(\phi;\ \mathrm{nULP}(\vec{n}, \vec{x}_2^\star)\big).$$

引理 ***4***（***S3*** 在步进点处具有表示能力）。对于所有 $\vec{n}$, $S_3(\vec{n}) \ge 0$ 和 $S_3(\vec{n}) = 0 \Leftarrow \mathrm{nULP}(\vec{n},\ \vec{x}_2^\star) \models \mathrm{C}$。解释。S3 在 S2 结果周围搜索浮点数格点，并可将近似满足的解精确调整为精确模型，同时保持合取范式语义（∨取乘积，∧取和）。

3.5 总体程序与保证

输入。
– 一个关于 QF_FP 原子的合取范式约束 C。– 一个黑盒 MP_Inverse(M)，用于返回摩尔–彭若斯伪逆 $M^\dagger$。^ ^ – 一个黑盒全局最小化器 Global_Min(F, init，budget) $\to (\vec{x},\ F(\vec{x}),$ status)。– 阶段预算 budget$_1$,budget$_2$,budget$_3$。

算法。

1. **根据 C 构建目标函数**。将所有线性等式收集为一个系统 $A\vec{x} == b$，并令其余原子构成 N。

   定义 $S_1(\vec{x})$ 为以下两项之和：(i) 通过 $\Pi_{\mathcal{L}}$ 计算的点 $\vec{x}$ 到 L 的平方距离；(ii) C_N 的平方违反项的子句级乘积（参见第 3 节）。定义 $S_2(\vec{x})$ 为对各子句中每个文字的 ULP 距离平方进行乘积后再求和（使用符合 IEEE 标准的表示函数；参见第 3 节）。给定来自 S_2 的当前最优解 $\vec{x}_2^\star$，定义 $S_3(\vec{n};\ \vec{x}_2^\star)$ 为在浮点数格点上计算的 S_2 目标函数值；该格点由各坐标步进 n_j 个末位单位，从而得到 $\mathrm{nULP}(\vec{n}, \vec{x}_2^\star)$（参见第 3 节）。

2. **最小化 S1（快速下降）**。$(\vec{x}_1^\star,\ v_1,\ _) \leftarrow$ 全局_最小值(S_1, MultiStartBox(C)，预算$_1$)。注意：此处不判定 SAT。

3. **最小化 S2（位精确优化）**。$(\vec{x}_2^\star,\ v_2,\ _) \leftarrow$ 全局_最小值(S_2, $\vec{x}_1^\star$，预算$_2$)。如果$v_2 = 0$，则返回**可满足**，其值为 $\vec{x}_2^\star$。

4. **最小化 S3（在浮点数格上进行 n-ULP 细化）**。为每个维度选择整数步长界 $\vec{N}$。在 $\vec{n} \in [-\vec{N}, \vec{N}] \cap \mathbb{Z}^d$ 上执行有界离散搜索，以最小化 $S_3(\vec{n};\ \vec{x}_2^\star)$。如果最佳值 $v_3 = 0$，则返回**可满足**，其 nULP 值为($\vec{n}^\star$, $\vec{x}_2^\star$)。

5. 否则，返回**不可满足猜测** 及分数 min(v_2, v_3)；若时间预算耗尽，则返回**超时** 。

引理3.1（结果三分性）。对于任意 *IEEE-754* 原子上的合取范式C 以及任意有限时间预算，上述程序恰好返回{可满足、不可满足猜测、超时}之一。

证明。对控制流进行分类讨论。步骤 (3)–(4) 在验证后达到目标函数值 0 时返回 **可满足**；否则，在以严格大于 0 的最佳值终止时，步骤 (5) 返回**不可满足猜测**；如果在满足上述任一条件之前时间预算耗尽，步骤 (4) 返回 **超时**。这些情况互斥且完备。□

Theo ˆ ˆ余 3.2（SAT可靠性）。如果程序返回可满足 并给出赋值 $\vec{x}$，那么 $\vec{x} \models C$.

ˆ 证明。只有在 S2 或 S3 中才会得到**可满足** 结果。在 S2 中，$S_2(\vec{x}) = 0$ 以及引理 3（S2 是表示函数）可推出 ˆ$\vec{x} \models$ C。在 S3 中，$S_3(\vec{n}^\star) = 0$ 以及引理 4（S3 在步进点处是表示函数）可推出 $\mathrm{nULP}(\vec{n}^\star, \vec{x}_2^\star) \models$ˆ C，这正是返回的 $\vec{x}$。在这两种情况下，返回前我们还会依据 IEEE-754 进行验证。因此，任何**可满足** 输出都是真正的 C 的模型。□

推论 3.3（无虚假 SAT）。程序绝不会对非模型报告可满足。

证明。由定理3.2立即得出。 □

命题 3.4（不完备性）。存在可满足的C 和有限预算，使程序返回不可满足猜测或超时。

证明概要。S1/S2 中的数值最小化可能收敛到非全局最小值；S3 只搜索有界集合内的整数步向量。在有限时间内，即使存在零点，也无法保证找到它。□

讨论。 尽管该设计并不完备，但在实践中表现出良好的效果：(1) S1 的投影在线性等式上带来可预测的进展，从而稳定了大型实例上的搜索；(2) S2 和 S3 是符合 IEEE 标准的表示函数，可确保无虚假 SAT，并明确呈现末位间隙；(3) S3 在浮点数格上的离散细化无需穷举，就能将近似匹配修正为精确模型。这些特性解释了我们在评估中观察到的准确率和可扩展性。

4 实现细节

$\vec{}$对 S_1的投影。 我们根据在 C 中检测到的浮点线性等式构造 A 和 b，并使用 $AA^\top$ 和与秩无关的摩尔–彭若斯伪逆计算投影。当 $AA^\top$ 在数值上奇异或不存在任何线性等式时，我们会跳过投影项，并保留其余原子的平方惩罚；目标函数仍然定义良好。

子句聚合。 在每个析取中，我们将逐文字距离相乘，并对各子句求和。这种“析取内求积、∨ 子句间求和”的方案在 S_1 （使用平方距离）和 S_2/S_3 （使用平方 ULP 距离）中始终如一，与第 3 节中的表示函数契约相符。

全局最小化。 Global_Min 通过多起点策略与无导数局部求解器实例化；当目标函数为零时提前终止，并将各起点并行分配到多个核心上。阶段预算 budget_1,budget_2,budget_3 可配置，其中 S_1 通常获得最大的预算份额，以构造良好的吸引盆，而 S_3 使用较小且有界的邻域。

ULP 距离与混合精度。 我们在 IEEE 754 中为每个文字实现 ULP 距离，并使用标准 ±1 偏移量处理严格不等式，同时根据变量为 FP32/FP64 变体进行分派。混合精度原子通过与第 3 节一致的逐操作数规则得到支持。

常数时间 *nULP* 步进。 原语 $\mathrm{nULP}(\vec{n}, \vec{x})$ 对每个坐标的耗时为 $O(1)$：我们将每个浮点值重新解释为浮点数格上的单调有符号整数索引，将n_j 作为一次单独的 整数运算相加，然后再进行位转换还原为浮点数；必要时将其钳制为最近的有限值。次正规数和带符号零在构造上得到保留。

数值卫生。 我们将 NaN/$\pm\infty$ 排除在搜索空间之外，预先计算每个变量的排序（FP32/FP64），以便高效分派内核；当 A 在重启过程中保持不变时，缓存投影项所需的矩阵分解。

只有 S_2 和 S_3 才能返回 **可满足**； S_1 仅用于快速初始化。这与第 3 节中的理论相呼应： S_1 在线性部分上提供部分单调下降； S_2 和 S_3 表示

函数（当且仅当 $\vec{x} \models C$ 时为零）；整体流程是可靠的（不会错误地判定为**可满足**），必然是不完备的，并且在经验上准确且可扩展。

附录A介绍了所实现的算法。我们的补充制品（已匿名处理）包含 StageSat 工具、使用指南、基准测试和实验脚本，使审阅者能够复现本文报告的表格和图表。

5 实验

5.1 执行摘要

StageSAT 在广泛使用的浮点可满足性基准测试上展现出稳健的性能和可靠性：

– 广泛覆盖：在最困难的 FP 基准测试套件中，StageSAT 为 49 个公式中的 45 个生成了结果（最佳求解器为 39 个），并且得益于显著更低的超时率，其处理的基准测试总数超过了任何竞争求解器。– *SAT* 检测近乎完美的准确率：在我们的基准测试套件中，StageSAT 为除两个之外的所有可满足公式找到了解（99.4% 召回率），并且从未对任何不可满足公式报告可满足结果（0% 错误判定为可满足）。这些结果与完整求解器的 UNSAT 判定完全一致，在任何情况下都没有生成错误模型。– 显著的时间改进：StageSAT 比完整求解器快得多地取得了这些结果，整体求解加速比平均达到 5–10× 倍。它遇到的超时也少得多，在我们的基准测试中几乎能够在时限内解决所有实例。– 不可或缺的流程：移除 StageSAT 三个优化阶段中的任意一个，都会显著减少其能够处理的实例数量（或导致遗漏解），这证实了其设计中的每个组件对于有效性都至关重要。

5.2 实验设置

我们在五个基准测试套件上评估 StageSAT，这些套件涵盖了范围广泛的浮点 SMT 问题：MathSAT-Small（130 个文件，≤10KB）、MathSAT-Middle（35 个文件，11–20 KB）、MathSAT-Large（49 个文件，>20KB）、Grater（118 个文件；过滤掉 5 个非 RNE 用例）和 JFS（111 个文件）。Large 集合用作主要压力测试；其他集合则覆盖了不同规模和来源。我们比较两类求解器：不完备求解器——XSat、goSAT、Grater 和 JFS（基于优化/模糊测试；无法证明不可满足）——以及完备求解器——Z3、CVC5、MathSAT 和 Bitwuzla（位精确决策过程）。所有实验均在配备 Intel Core i9-13900HK（14 核）、32 GB RAM、Ubuntu 24.04.3 LTS 的机器上运行，每个实例的墙钟时间上限为 20 分钟（压力测试除外，该测试运行 48 小时，结果见表3）。不完备求解器（包括 StageSAT ）的每个基准测试运行 5× 。

我们的评估采用以下指标。*SAT* 覆盖率（又称 *SAT* 召回率）是指求解器返回可满足并给出经验证的模型的真实 SAT 实例所占的比例。所有真实 SAT/UNSAT 标签均来自完备求解器：只要至少有一个完备求解器在 48 小时限制内完成，我们就采用其结果（而且在所有此类情况下，已完成的求解器结果一致）。对于一个特别困难的基准测试，所有完备求解器都达到了 48 小时超时限制，因此我们以更长的超时限制重新运行速度最快的完备求解器 Bitwuzla，并将其最终结果作为真实值。

$$\text{SAT-Recall} \;=\; \frac{\#\{\text{reported SAT} \;\wedge\; \text{ground-truth=SAT}\}}{\#\{\text{ground-truth=SAT}\}}\,.$$

超时率 是指 20分钟内未得出判定结果的所有基准测试所占的比例：

$$\text{Timeout-rate} = \frac{\#\{\text{timeout}\}}{\#\{\text{all benchmarks}\}}.$$

平均值（算术平均值）时间 是每个测试套件的算术平均值，其中超时按上限时间计入（以反映在预算限制下的稳健性）。中位时间 与平均值一并报告，以反映典型表现并减轻重尾效应。

我们不使用**SAT 精确率**（报告为可满足的判定结果中实际确实可满足的比例）进行比较。在我们的实验中，所有求解器的这一比例实际上都为 100%——仅在*Grater* 接受低于容差的目标值时出现过少数例外——因此这一指标无法区分求解器。此外，只在简单案例上报告可满足可以虚增精确率，而将所有问题都标记为可满足则可以虚增 SAT 覆盖率。因此，我们着重考察**SAT 覆盖率、超时率和时间统计**。对于*StageSAT*，不可满足猜测是一种不同于证明的结果，我们绝不会将其计作已证明不可满足或“已解决”。

5.3 定量结果

我们按四个研究问题（RQ1-4）组织实验结果。

RQ1：***StageSAT*** 与不完备求解器的比较（***MathSAT-Large***）与以往基于优化和启发式的不完备求解器相比，*StageSAT* 在大型浮点基准测试集上的表现如何？我们在 49 个 MathSAT-Large 公式上比较了 StageSAT、XSat、goSAT、Grater 和 JFS。结果表明，在这些不完备求解器中，StageSAT 的覆盖率和准确率最高。表1报告了各基准测试的详细结果。在 49 个实例中，StageSAT 能够对 45 个返回结果（可满足或不可满足猜测）；而排名第二的不完备求解器 XSat 在相同时间限制内解决的实例明显更少。就MathSAT-Large测试集的 SAT 覆盖率而言，StageSAT在 26 个可满足基准测试中为 24 个找到了满足赋值，在解决 SAT 实例的数量上超过了所有以往的数值竞争对手。事实上，XSat 解决的实例数量最接近 StageSAT，但两者有一个关键区别——可靠性。XSat 在未能找到解时，常将困难的可满足问题错误分类为“不可满足”；而在我们的测试中，StageSAT 从未错误地报告不可满足。具体而言，XSat 对实际存在解的基准测试给出了 12 个错误的不可满足答复（StageSAT 对这 12 个问题均找到了有效模型）。所有 StageSAT 模型都经过位精确检查器（Z3）的独立验证，确认正确性[10]。这表明，StageSAT 的优化策略不仅提高了解决问题的数量，也提升了对具有挑战性的公式所得结果的精确性。

其他基线方法在 MathSAT-Large 上的表现较差。作为最先进的基于优化的求解器，Grater 在 StageSAT 能够解决的大多数大型基准测试上都超时，因此总体解决的问题数量较少。此外，Grater 使用固定容差可能会导致虚假 SAT 报告——有一次，它报告“SAT”，但实际上该公式不可满足（Z3 和 CVC5 均证明了其不可满足）。StageSAT 通过要求 SAT 情况下的目标函数严格为零来避免这一问题，因此绝不会对不可满足的实例报告 SAT。启发式模糊测试器 JFS 和全局优化器 goSAT 在这组测试中解决的问题最少，它们难以处理复杂约束（许多运行都以超时或结果未知告终）。总体而言，StageSAT 在覆盖率和可靠性之间取得了最佳平衡：它解决的大型基准测试显著多于 JFS 或 goSAT，也多于 Grater，并且覆盖率与 XSat 相当，同时不 会出现 XSat 的误分类错误[12]。在实际应用中，StageSAT 能够解决这些大型实例挑战中的绝大多数（通常每次试运行都能解决），而其他不完备求解器则要么表现不稳定，要么在许多实例上彻底失败。这些结果证实，StageSAT 新颖的分阶段优化方法显著提升了求解器在困难浮点问题上的有效性（更高的求解率）和可靠性（不产生错误答案）。

RQ2：***StageSAT*** 与完备求解器的比较（***MathSAT-Large***）在相同的大型基准测试上，*StageSAT* 与现代完备 *SMT* 求解器相比表现如何？在本分析中，我们评估了 StageSAT 与 Bitwuzla、CVC5、Z3 和 MathSAT5 在 49 个 MathSAT-Large 公式上的表现，如表 2 所示。一个关键

表 1. 比较 : MathSAT-Large 基准测试：StageSAT 与不完备求解器。超时设为 20 分钟。

基准测试	大小（字节）	变量数	XSat		goSAT		Grater		JFS		StageSAT	
			判定结果	时间（秒）	判定结果	时间（秒）	判定结果	时间（秒）	判定结果	时间（秒）	判定结果	时间（秒）
sqrt.c.10	21701	26	不可满足	7.81	未知	0.72	超时	>1200	超时	>1200	sat	102.48
test_v5_r15_vr5_c1_s8246	21791	5	不可满足	1.62	未知	0.02	超时	>1200	超时	>1200	不可满足猜测	10.76
test_v5_r15_vr1_c1_s26845	21811	5	不可满足	1.24	未知	0.02	超时	>1200	超时	>1200	不可满足猜测	7.00
test_v5_r15_vr10_c1_s25268	21818	5	不可满足	1.54	未知	0.02	超时	>1200	超时	>1200	不可满足猜测	8.87
test_v5_r15_vr5_c1_s26657	22070	5	不可满足	1.48	未知	0.02	超时	>1200	超时	>1200	不可满足猜测	6.70
test_v5_r15_vr1_c1_s32559	22072	5	不可满足	1.58	未知	0.02	超时	>1200	超时	>1200	不可满足猜测	5.78
test_v5_r15_vr1_c1_s8236	22072	5	不可满足	1.22	未知	0.02	超时	>1200	超时	>1200	不可满足猜测	5.88
test_v5_r15_vr5_c1_s23844	22072	5	不可满足	1.66	未知	0.02	超时	>1200	超时	>1200	不可满足猜测	9.21
test_v5_r15_vr10_c1_s14516	22252	5	不可满足	1.40	未知	0.02	超时	>1200	超时	>1200	不可满足猜测	10.56
qurt.c.5	23169	30	不可满足	9.44	未知	0.83	sat	1177.90	超时	>1200	不可满足猜测	156.14
test_v7_r12_vr5_c1_s29826	23736	7	sat	0.18	sat	0.02	sat	0.07	sat	3.25	sat	0.06
test_v7_r12_vr10_c1_s15994	23828	7	sat	0.14	sat	0.02	sat	0.21	sat	16.94	sat	0.06
test_v7_r12_vr10_c1_s30410	24070	7	sat	1.40	sat	0.02	sat	0.89	超时	>1200	sat	0.09
test_v7_r12_vr5_c1_s14336	24250	7	sat	0.14	sat	0.03	sat	0.07	sat	0.38	sat	0.05
test_v7_r12_vr5_c1_s8938	24251	7	sat	0.14	sat	0.02	sat	0.07	sat	0.14	sat	0.05
test_v7_r12_vr1_c1_s10576	24274	7	不可满足	3.04	未知	0.03	超时	>1200	超时	>1200	不可满足猜测	14.56
test_v7_r12_vr1_c1_s22787	24345	7	不可满足	2.24	未知	0.04	超时	>1200	超时	>1200	不可满足猜测	11.65
test_v7_r12_vr10_c1_s18160	24437	7	不可满足	2.40	未知	0.04	超时	>1200	超时	>1200	不可满足猜测	21.18
test_v7_r12_vr1_c1_s703	24441	7	不可满足	2.76	未知	0.03	超时	>1200	超时	>1200	不可满足猜测	16.54
sin2.c.15	25235	52	sat	25.81	未知	1.78	sat	13.88	超时	>1200	sat	3.92
gaussian.c.25	29883	79	sat	0.72	未知	2.52	sat	9.12	超时	>1200	sat	3.38
sqrt.c.15	32192	36	不可满足	13.10	未知	1.06	超时	>1200	超时	>1200	sat	101.76
test_v7_r17_vr5_c1_s2807	32711	7	不可满足	2.20	未知	0.04	超时	>1200	超时	>1200	不可满足猜测	15.21
test_v7_r17_vr1_c1_s30331	32876	7	不可满足	2.58	未知	0.05	超时	>1200	超时	>1200	不可满足猜测	14.96
test_v7_r17_vr5_c1_s25451	32964	7	不可满足	2.30	未知	0.04	超时	>1200	超时	>1200	不可满足猜测	11.58
sin2.c.20	33016	67	不可满足	41.81	未知	1.74	sat	33.89	超时	>1200	sat	169.92
test_v7_r17_vr10_c1_s8773	33151	7	sat	0.74	sat	0.03	sat	0.35	超时	>1200	sat	0.06
test_v7_r17_vr5_c1_s4772	33222	7	不可满足	2.30	未知	0.05	超时	>1200	超时	>1200	不可满足猜测	35.20
test_v7_r17_vr1_c1_s23882	33226	7	sat	0.14	sat	0.03	sat	0.29	超时	>1200	sat	0.05
test_v7_r17_vr1_c1_s24331	33226	7	不可满足	3.12	未知	0.04	超时	>1200	超时	>1200	不可满足猜测	11.63
test_v7_r17_vr10_c1_s3680	33335	7	不可满足	3.06	未知	0.05	超时	>1200	超时	>1200	不可满足猜测	13.26
test_v7_r17_vr10_c1_s18654	33410	7	sat	0.16	sat	0.03	sat	5.73	超时	>1200	sat	0.05
sin.c.25	40536	81	不可满足	76.20	未知	2.31	sat	98.30	超时	>1200	sat	294.71
sin2.c.25	40747	82	不可满足	61.49	未知	2.84	sat	93.33	超时	>1200	sat	8.62
sqrt.c.20	46804	63	不可满足	45.91	未知	1.73	超时	>1200	超时	>1200	sat	498.60
sqrt.c.25	46804	63	不可满足	46.33	未知	1.82	超时	>1200	超时	>1200	sat	581.95
qurt.c.10	47946	60	不可满足	22.89	未知	1.28	超时	>1200	超时	>1200	不可满足猜测	393.02
qurt.c.15	73125	90	不可满足	53.40	未知	2.95	超时	>1200	超时	>1200	不可满足猜测	728.38
gaussian.c.75	89686	229	sat	20.05	未知	13.31	sat	542.51	超时	>1200	sat	36.45
qurt.c.20	93126	114	不可满足	68.26	未知	3.96	超时	>1200	超时	>1200	超时	>1200
qurt.c.25	93126	114	不可满足	76.92	未知	4.31	超时	>1200	超时	>1200	超时	>1200
sin2.c.75	119790	231	不可满足	242.70	未知	11.70	超时	>1200	超时	>1200	sat	363.37
sin.c.75	119794	231	不可满足	214.49	未知	12.32	超时	>1200	超时	>1200	sat	660.70
gaussian.c.125	150792	379	sat	13.20	未知	40.24	sat	89.02	超时	>1200	sat	90.62
sin.c.125	200503	381	不可满足	1000.64	未知	34.38	错误	–	超时	>1200	sat	379.39
sin2.c.125	200503	381	不可满足	984.14	未知	33.26	错误	–	超时	>1200	sat	313.85
gaussian.c.175	210711	529	不可满足	1101.04	未知	80.37	超时	>1200	超时	>1200	sat	273.68
sin2.c.175	280962	531	超时	>1200	未知	76.20	错误	–	超时	>1200	超时	>1200
sin.c.175	280984	531	超时	>1200	未知	72.63	错误	–	超时	>1200	超时	>1200
SAT 覆盖率			**46.2%**		**30.8%**		**57.7%**		**15.4%**		**92.3%**	
超时率			**4.1%**		**0.0%**		**51.0%**		**91.8%**		**8.2%**	
平均时间（秒）			**134.02**		**8.27**		**819.24**		**1102.46**		**208.00**	
中位时间（秒）			**3.04**		**0.05**		**>1200**		**>1200**		**14.96**	

问题是，StageSAT 的启发式方法能否达到与这些完备（位精确）求解器相近的覆盖率；原则上，只要时间充足，这些求解器便能求解或反驳任何实例。我们发现，StageSAT 在大型基准测试上的覆盖率可与最优秀的完备求解器相媲美。在 20分钟时限内，StageSAT 为 49 个实例中的 45 个给出了结果，略多于表现最好的完备求解器（Bitwuzla 在时限内求解了 39 个）。CVC5 求解了 36 个，而 Z3 和 MathSAT5 在超时前处理的案例更少。换言之，StageSAT 的不完备策略使其能够处理这些困难实例中约 92%，基本与

在这一大规模数据集上实现了最先进的覆盖率。这一点值得关注，因为完备求解器有时会在极复杂的公式上耗尽时间，而 StageSAT 在大多数情况下能更快地找到模型，或得出不可满足性猜测。

表 2. 对比 MathSAT-Large 基准测试：StageSAT 与完备求解器的对比。超时设为 20 分钟 s.

基准测试	大小（字节）	#变量	cvc5		Bitwuzla		Z3		MathSAT		StageSAT	
			判定结果	时间（秒）	判定结果	时间（秒）	判定结果	时间（秒）	判定结果	时间（秒）	判定结果	时间（秒）
sqrt.c.10	21701	26	sat	1.01	sat	6.15	sat	492.62	sat	28.08	sat	102.48
test_v5_r15_vr5_c1_s8246	21791	5	不可满足	269.56	不可满足	95.73	超时	>1200	不可满足	294.65	不可满足猜测	10.76
test_v5_r15_vr1_c1_s26845	21811	5	不可满足	75.80	不可满足	47.68	不可满足	737.95	不可满足	131.37	不可满足猜测	7.00
test_v5_r15_vr10_c1_s25268	21818	5	不可满足	216.89	不可满足	156.56	超时	>1200	超时	>1200	不可满足猜测	8.87
test_v5_r15_vr5_c1_s26657	22070	5	不可满足	160.27	不可满足	63.23	超时	>1200	不可满足	177.60	不可满足猜测	6.70
test_v5_r15_vr1_c1_s32559	22072	5	不可满足	43.48	不可满足	33.09	不可满足	139.16	不可满足	24.20	不可满足猜测	5.78
test_v5_r15_vr1_c1_s8236	22072	5	不可满足	49.45	不可满足	25.43	不可满足	369.09	不可满足	12.66	不可满足猜测	5.88
test_v5_r15_vr5_c1_s23844	22072	5	不可满足	266.86	不可满足	95.28	超时	>1200	不可满足	231.11	不可满足猜测	9.21
test_v5_r15_vr10_c1_s14516	22252	5	不可满足	530.11	不可满足	202.41	超时	>1200	超时	>1200	不可满足猜测	10.56
qurt.c.5	23169	30	不可满足	6.59	不可满足	0.22	不可满足	13.44	不可满足	6.53	不可满足猜测	156.14
test_v7_r12_vr5_c1_s29826	23736	7	sat	129.16	sat	163.13	sat	425.77	sat	65.19	sat	0.06
test_v7_r12_vr10_c1_s15994	23828	7	sat	176.39	sat	98.18	sat	509.78	sat	130.89	sat	0.06
test_v7_r12_vr10_c1_s30410	24070	7	超时	>1200	超时	>1200	超时	>1200	超时	>1200	sat	0.09
test_v7_r12_vr5_c1_s14336	24250	7	sat	22.82	sat	115.91	sat	284.32	sat	84.11	sat	0.05
test_v7_r12_vr5_c1_s8938	24251	7	sat	19.00	sat	37.72	sat	105.88	sat	30.66	sat	0.05
test_v7_r12_vr1_c1_s10576	24274	7	不可满足	322.33	不可满足	170.45	超时	>1200	不可满足	206.12	不可满足猜测	14.56
test_v7_r12_vr1_c1_s22787	24345	7	不可满足	597.04	不可满足	217.96	超时	>1200	不可满足	581.35	不可满足猜测	11.65
test_v7_r12_vr10_c1_s18160	24437	7	超时	>1200	超时	>1200	超时	>1200	超时	>1200	不可满足猜测	21.18
test_v7_r12_vr1_c1_s703	24441	7	不可满足	639.55	不可满足	234.56	超时	>1200	超时	>1200	不可满足猜测	16.54
sin2.c.15	25235	52	sat	29.95	sat	159.21	超时	>1200	超时	>1200	sat	3.92
gaussian.c.25	29883	79	sat	3.54	sat	2.47	sat	642.11	sat	11.65	sat	3.38
sqrt.c.15	32192	36	sat	8.95	sat	1.29	超时	>1200	sat	27.84	sat	101.76
test_v7_r17_vr5_c1_s2807	32711	7	超时	>1200	超时	>1200	超时	>1200	超时	>1200	不可满足猜测	15.21
test_v7_r17_vr1_c1_s30331	32876	7	不可满足	205.82	不可满足	227.17	超时	>1200	不可满足	1140.11	不可满足猜测	14.96
test_v7_r17_vr5_c1_s25451	32964	7	超时	>1200	不可满足	888.67	超时	>1200	超时	>1200	不可满足猜测	11.58
sin2.c.20	33016	67	sat	560.39	sat	693.96	超时	>1200	超时	>1200	sat	169.92
test_v7_r17_vr10_c1_s8773	33151	7	sat	794.03	sat	530.94	超时	>1200	超时	>1200	sat	0.06
test_v7_r17_vr5_c1_s4772	33222	7	超时	>1200	超时	>1200	超时	>1200	超时	>1200	不可满足猜测	35.20
test_v7_r17_vr1_c1_s23882	33226	7	sat	273.29	sat	340.07	sat	1162.59	sat	524.15	sat	0.05
test_v7_r17_vr1_c1_s24331	33226	7	超时	>1200	不可满足	787.68	超时	>1200	超时	>1200	不可满足猜测	11.63
test_v7_r17_vr10_c1_s3680	33335	7	超时	>1200	不可满足	1188.71	超时	>1200	超时	>1200	不可满足猜测	13.26
test_v7_r17_vr10_c1_s18654	33410	7	sat	1126.63	sat	588.61	超时	>1200	超时	>1200	sat	0.05
sin.c.25	40536	81	sat	802.21	sat	874.55	超时	>1200	超时	>1200	sat	294.71
sin2.c.25	40747	82	sat	713.26	sat	902.61	超时	>1200	超时	>1200	sat	8.62
sqrt.c.20	46804	63	sat	0.96	sat	5.96	sat	721.16	sat	18.39	sat	498.60
sqrt.c.25	46804	63	sat	0.96	sat	5.95	sat	717.69	sat	18.42	sat	581.95
qurt.c.10	47946	60	不可满足	6.59	不可满足	0.21	不可满足	23.86	不可满足	6.62	不可满足猜测	393.02
qurt.c.15	73125	90	不可满足	0.27	不可满足	0.02	不可满足	1.93	不可满足	6.61	不可满足猜测	728.38
gaussian.c.75	89686	229	sat	13.46	sat	6.57	sat	429.71	sat	56.89	sat	36.45
qurt.c.20	93126	114	不可满足	0.30	不可满足	0.01	超时	>1200	不可满足	7.25	超时	>1200
qurt.c.25	93126	114	不可满足	0.32	不可满足	0.01	超时	>1200	不可满足	7.21	超时	>1200
sin2.c.75	119790	231	超时	>1200	超时	>1200	错误	–	超时	>1200	sat	363.37
sin.c.75	119794	231	超时	>1200	超时	>1200	错误	–	超时	>1200	sat	660.70
gaussian.c.125	150792	379	sat	20.54	sat	36.94	超时	>1200	sat	338.55	sat	90.62
sin.c.125	200503	381	超时	>1200	超时	>1200	错误	–	超时	>1200	sat	379.39
sin2.c.125	200503	381	超时	>1200	超时	>1200	错误	–	超时	>1200	sat	313.85
gaussian.c.175	210711	529	sat	460.36	sat	745.12	超时	>1200	超时	>1200	sat	273.68
sin2.c.175	280962	531	超时	>1200	超时	>1200	错误	–	超时	>1200	超时	>1200
sin.c.175	280984	531	超时	>1200	超时	>1200	错误	–	超时	>1200	超时	>1200
SAT/UNSAT 覆盖率			**73.5%**		**79.6%**		**32.7%**		**53.1%**		**91.8%**	
超时率			**26.5%**		**20.4%**		**55.1%**		**46.9%**		**8.2%**	
平均时间（秒）			**492.82**		**443.89**		**911.09**		**648.33**		**208.00**	
中位时间（秒）			**269.56**		**170.45**		**>1200**		**581.35**		**14.96**	

表3. 在 Bitwuzla 超时的 MathSAT-Large 基准测试上的比较：StageSAT 与 Bitwuzla。超时时限为 48 小时。

基准测试	大小（字节）	#变量	StageSAT		Bitwuzla	
			判定结果	时间（秒）	判定结果	时间（秒）
test_v7_r12_vr10_c1_s30410	24070	7	sat	0.09	sat	3042.79
test_v7_r12_vr10_c1_s18160	24437	7	不可满足猜测	21.18	不可满足	7347.42
test_v7_r17_vr5_c1_s2807	32711	7	不可满足猜测	15.21	不可满足	1466.26
test_v7_r17_vr5_c1_s4772	33222	7	不可满足猜测	35.20	不可满足	93950.26
sin2.c.75	119790	231	sat	363.37	sat	4180.57
sin.c.75	119794	231	sat	660.70	sat	9361.38
sin.c.125	200503	381	sat	379.39	超时	>48小时
sin2.c.125	200503	381	sat	313.85	sat	105514.52
sin2.c.175	280962	531	超时	>48小时	sat	151645.17
sin.c.175	280984	531	超时	>48小时	sat	68898.42

就性能而言，StageSAT 在可满足实例上也展现出优势：它通常比完备求解器更快地找到解。许多可满足的 MathSAT-Large 基准测试中，Bitwuzla 或 CVC5 只能在接近超时时才求解成功，而 StageSAT 只需极短时间即可求解成功（通常快 5–10× 倍；在我们测试中，对于最大的可满足案例，速度则超过 10× 倍）。我们还使用 StageSAT 和 Bitwuzla 进行了压力测试，并将超时时间延长至 48 小时，以评估它们在 Bitwuzla 超时的最大规模基准测试上的表现，结果汇总于表3。例如，在一个特别困难的可满足公式（sin.c.125）上，StageSAT 很快就求解成功，而 Bitwuzla 即使搜索了 *48* 小时也未能找到任何模型。这凸显了 StageSAT 高效地在搜索空间中寻找模型的能力。另一方面，完备求解器在证明不可满足性方面更具优势。按照设计，StageSAT 无法提供形式化 UNSAT 证明——当其优化过程得出无法找到更优（更低）目标函数值的结论时，它会输出不可满足猜测，但这只是一种启发式指示。我们检查了所有案例：任何被完备求解器证明为不可满足的基准测试，都被 StageSAT 归类为不可满足猜测。关键的是，我们没有发现任何差异——StageSAT 没有将任何被完备求解器证明为不可满足的实例标记为可满足；而每当 StageSAT 给出不可满足猜测时，根据完备求解器的判断，该实例确实无解。这种实证一致性表明，在 MathSAT-Large 上，StageSAT 的不可满足性猜测是可靠的：在所有测试案例中，它们都能作为真正不可满足性的正确启发式替代指标。不过，由于这些并非形式化证明，我们不会将其计作已证明不可满足；相反，它们表明 StageSAT 能以合理的置信度识别无解实例。在 StageSAT 超时（没有结果）的四个最困难案例中，完备求解器经过延长运行后，最终判定其中两个可满足，另外两个不可满足。这凸显了一项局限性：对于规模最大的或最棘手的可满足公式，StageSAT 也可能难以应对（正如它在两个需要超过 20分钟的超大规模可满足基准测试上的表现）。反过来，对于某些棘手的不可满足案例，完备求解器凭借强力推理成功求解，而 StageSAT 的启发式方法则未能在时限内完成。

尽管存在这些局限性，StageSAT 相较于完备求解器仍表现出色。在标准时间限制内，它的求解数量与这些求解器持平或更高，并且在能够求解的实例上更快地给出了结果。这表明，StageSAT 可以作为传统 SMT 求解器的实用补充。在一种使用场景中，可以将 StageSAT 与完备求解器并行运行：对于困难的可满足问题，如果存在模型，StageSAT 能够快速找到它（可能省去数小时的搜索）；而对于少数需要不可满足性证明的实例，则可交由完备求解器处理。StageSAT 在大型基准测试中展现出的竞争力，证明了其方法的优势：

通过结合数值优化与 SMT，在浮点约束上同时实现可扩展性和经验证的正确性。

RQ3：***StageSAT*** 设计的消融研究 *StageSAT* 的三个阶段和关键设计组件对于其性能而言是否都必不可少？ StageSAT 的求解过程由三个依次执行的阶段（S1、S2、S3）组成，各阶段采用不同的目标函数形式，此外还包含 S1 中的投影项以及后续阶段中的按子句的ULP聚合等额外技术。为了解各部分的贡献，我们进行了消融研究：运行移除或更改一个组件后的 StageSAT 变体，并测量其对 MathSAT-Large 结果的影响。表4 展示了各个变体的结果数量。完整的 StageSAT（按设计采用 S1–S2–S3）解决了 49 个大型基准测试中的 45 个（报告为可满足的有 24 个，不可满足猜测有 21 个），每个实例的平均求解时间为 208 秒。我们以此作为比较基线。

表 4. MathSAT-Large 基准测试的消融汇总。超时时间设为 20 分钟。

变体	可满足 (#)	不可满足 (#)	超时 (#)	平均时间 (s)
完整 S1–S2–S3	24	21	4	208.00
无 S1（直接从 S2 开始）	13	22	14	409.07
无 S3（仅 S1–S2）	22	23	4	262.61
S1 不含投影项	21	24	4	291.82
S1 使用绝对残差	18	27	4	196.05
S2/S3 不使用按子句 ULP 乘积	22	21	6	264.06

移除阶段 1 并直接从阶段 2 开始，对性能产生了显著影响。这个“无 S1”变体仅求解了 35/49 个实例（从 45 个下降），并出现了 14 次超时（完整 StageSAT 中为 4 次）。尤其是，找到的可满足实例数量大幅下降（从 24 个降至仅 13 个）。已求解案例的平均运行时间也几乎翻倍（409 秒，而非 208 秒）。这表明阶段 1 对可扩展性至关重要——其投影辅助的平方残差目标函数能迅速引导求解器进入可行区域。没有 S1 的快速粗略引导，求解器会陷入困境，超时也频繁得多。直观而言，S1 的轻量级目标函数（基于残差幅值）比 S2 和 S3 中更精确的基于 ULP 的目标函数更易计算，因此能够广泛探索搜索空间，并迅速找到有希望的区域。结论：阶段 1 对于在大型问题上实现较高覆盖率不可或缺。

从概念上说，当纯平方幅值目标函数陷入停滞时，阶段 2 至关重要，例如在次正规数区域和下溢情形中。MathSAT-Large 数据集中不包含此类压力测试用例，因此我们在这项消融研究中没有纳入“无 S2”配置。相比之下，JFS 和 Grater 基准测试包含许多结构相似的公式；对于这些测试集，阶段 2 对于引导 StageSAT 找到正确模型至关重要。

接下来，移除阶段 3 的影响更为微妙，但同样重要。“无 S3”变体解决的实例总数相同（45/49），但 找到的 SAT 解更少（22 对 24），相应地将更多案例判定为不可满足猜测（23 对 21）。事实上，有两个基准测试中，完整的 StageSAT 成功将其判定为 SAT，但仅含 S1–S2 的求解器未能解决；它收敛到了一个非零最小值，并将它们错误地报告为不可满足。这表明，对于边界情况，阶段 3 对正确性至关重要——它会在浮点数格上执行离散高精度搜索；当阶段 2 的连续优化陷入停滞时，这种搜索能够“弥合差距”。如果没有阶段 3，StageSAT 的 SAT 覆盖率会下降，并会将一些可满足问题误判为不可满足猜测。结论：阶段 3 是确保 StageSAT 的 SAT 结果达到 100% 可靠性的必要条件，因为它能够捕捉仅靠数值平滑无法处理的棘手情况。

我们还测试了阶段内技术的不同变体。从阶段 1 中移除正交投影项（同时仍执行 S1–S2–S3），使总求解数保持在 45，但 StageSAT 的

SAT 查找率略有下降（21 个 SAT 解，而非 24 个），平均求解时间则增加到 ~292 秒。这表明，投影项——它施加了部分单调下降性质——确实有助于求解器找到更多模型（完整版多找到 3 个），而且速度更快。缺少该项很可能导致优化器有时陷入平坦或具有误导性的曲面，使一些原本可能可满足的情况变成不可满足猜测，并总体拖慢求解进度。将 S1 的平方残差目标函数替换为更简单的绝对残差目标函数，效果则喜忧参半：运行时间略有改善（平均 196 秒），但 SAT 解数量大幅减少（仅找到 18 个 SAT 解，另有 27 个不可满足猜测）。这种权衡表明，尽管绝对残差可能在某些情况下加快收敛，但它对精确解的引导较弱，导致 StageSAT 漏掉许多平方残差版本能够求解的可满足情况。最后，在阶段 2–3 中禁用按子句 ULP 乘积（一种强调同时满足所有约束的聚合方案），使 StageSAT 的总求解数降至 43/49，并导致 6 次超时（完整 StageSAT 为 4 次）。SAT 覆盖率也略有下降（22 个 SAT 解，而完整 StageSAT 为 24 个）。这表明，按子句乘积虽然只是组合各子句错误的一种简单启发式，但有助于 StageSAT 优先选择能够满足每个约束的赋值，从而避免一些超时，并多找到几个模型。

总之，消融研究证实，每个阶段和每项设计选择都对 StageSAT 的成功作出了重要贡献。阶段 1 的投影辅助粗搜索对于扩展规模和提升速度至关重要；阶段 2 基于 ULP 的连续优化对于在棘手的浮点区域中进行搜索至关重要；阶段 3 的离散细化对于确保最终结果正确至关重要；而额外的改进（投影项、平方指标、ULP 聚合）则进一步提升了性能和结果质量。移除其中任何一项，都会削弱 StageSAT 求解困难实例的能力，或导致其无法找到解，这证明完整的 S1–S2–S3 设计是实现所报告的高覆盖率和高准确率所必需的。

RQ4：所有基准测试套件的整体性能 考虑所有基准类别（小型、中型、大型以及不同来源）时，*StageSAT* 的整体性能如何？与其他求解器相比，其综合表现如何？ 为回答这个问题，我们汇总了表 5 中全部五组基准测试的结果：MathSAT-Large 、 MathSAT-Middle 、 MathSAT-Small 、 JFS 和 Grater。StageSAT 在大型套件中呈现的趋势也延续到了其他套件，表现始终强劲。首先，在超时限制内，StageSAT 成功完成了 MathSAT-Small、MathSAT-Middle、JFS 和 Grater 数据集中的每一个实例。在这四个套件中（共计 280 多个公式），StageSAT 没有发生超时，并求解了所有可满足实例，覆盖率达到 100%。MathSAT-Large 套件仍是 StageSAT 唯一发生超时的数据来源（49 个实例中有 4 个，如前所述）。这意味着，在所有测试的基准测试中，StageSAT 求解了 45 个（大型） + （全部 35 个中型） + （全部 130 个小型） +（全部 111 个 JFS） + （全部 118 个 Grater） = 439 问题，只漏掉了 4 个难度最高的大型实例。相比之下，其他不完备求解器往往有许多问题未能求解（尤其是在较大的实例上）；即使是完备求解器，在面对公式复杂的部分中型或小型基准测试时，也偶尔会超时。StageSAT 能够处理较简单或中等难度套件中的每一个实例，这凸显了它在这些问题上的稳健性和效率。

尤其是考察 SAT 覆盖率——也就是求解器能够找到模型的真实可满足基准测试所占的比例——StageSAT 在所有测试套件中都优于其他不完备求解器，始终找到了最多的解。例如，在 Grater 基准测试中（其中许多是棘手的可满足案例），StageSAT 找到了所有可满足实例的解，而 XSat、goSAT 或 JFS 则遗漏了一些，和／或过早放弃。在专为模糊测试方法设计的 JFS 测试套件中，StageSAT 也成功解决了所有可满足案例，实际上达到了或超过了 JFS 的覆盖率，但采用的是系统化方法。同时，StageSAT 的不可满足猜测启发式在较小的基准测试中也没有误判：对于所有完备求解器报告 UNSAT 的问题，StageSAT 要么也报告不可满足猜测，要么在许多情况下能够更快地判定为不可满足猜测，

表 5. MathSAT-Large、MathSAT-Middle、MathSAT-Small、JFS 和 Grater 基准测试汇总。#未知/不可满足猜测 =#未知/不可满足猜测。超时设为 20 分钟。

数据集 求解器	#可满足	#不可满足	#超时	#未知/不可满足猜测	#错误	可满足覆盖率	平均时间（秒）	中位时间（秒）
MathSAT-Large（49 个基准测试，26 个可满足，23 个不可满足）								
CVC5	19	17	13	0	0	73.1%	492.82	269.56
Bitwuzla	19	20	10	0	0	73.1%	443.89	170.45
Z3	10	6	27	0	6	38.5%	911.09	>1200
MathSAT	12	14	23	0	0	46.2%	648.33	581.35
XSAT	12	23	2	0	0	46.2%	134.02	3.04
goSAT	8	0	0	41	0	30.8%	8.27	0.05
Grater	16	0	25	0	4	57.7%	819.24	>1200
JFS	4	0	45	0	0	15.4%	1102.46	>1200
StageSAT	24	0	4	21	0	92.3%	208.00	14.96
MathSAT-Middle（35 个基准测试，29 个可满足，6 个不可满足）								
CVC5	29	6	0	0	0	100.0%	94.65	39.24
Bitwuzla	29	6	0	0	0	100.0%	31.10	22.59
Z3	19	5	6	0	5	65.5%	370.31	323.14
MathSAT	28	6	1	0	0	96.6%	77.19	33.47
XSAT	29	6	0	0	0	100.0%	1.06	0.14
goSAT	23	0	0	12	0	79.3%	0.26	0.02
Grater	26	0	9	0	0	89.7%	337.13	0.18
JFS	9	0	26	0	0	31.0%	892.06	>1200
StageSAT	29	0	0	6	0	100.0%	8.59	0.18
MathSAT-Small（130 个基准测试，63 个可满足，67 个不可满足）								
CVC5	63	55	12	0	0	100.0%	136.44	3.34
Bitwuzla	63	64	3	0	0	100.0%	35.64	1.93
Z3	56	53	13	0	8	88.9%	194.18	24.79
MathSAT	63	51	16	0	0	100.0%	168.37	7.31
XSAT	62	67	0	0	1	98.4%	0.46	0.14
goSAT	58	0	0	72	0	92.1%	0.07	0.02
Grater	63	0	38	0	28	100.0%	471.50	0.81
JFS	53	0	77	0	0	84.1%	717.96	>1200
StageSAT	63	0	0	67	0	100.0%	1.64	0.66
JFS（111 个基准测试，111 个可满足，0 个不可满足）								
CVC5	104	0	6	0	1	93.7%	110.87	4.74
Bitwuzla	108	0	3	0	0	97.3%	103.65	3.24
Z3	85	0	18	0	8	76.6%	327.55	4.27
MathSAT	102	0	9	0	0	91.9%	149.23	11.60
XSAT	101	4	0	0	6	91.0%	13.04	0.14
goSAT	88	0	0	19	4	79.3%	2.21	0.03
Grater	107	0	4	0	0	96.4%	71.86	0.06
JFS	69	0	42	0	0	62.2%	458.33	0.72
StageSAT	111	0	0	0	0	100.0%	13.10	0.05
Grater（118 个基准测试，118 个可满足，0 个不可满足）								
CVC5	118	0	0	0	0	100.0%	37.62	2.13
Bitwuzla	117	0	1	0	0	99.2%	64.30	10.61
Z3	103	0	12	0	3	87.3%	241.12	64.73
MathSAT	111	0	7	0	0	94.1%	131.05	25.29
XSAT	115	2	0	0	1	97.5%	1.62	0.14
goSAT	92	0	0	26	0	78.0%	0.70	0.02
Grater	118	0	0	0	0	100.0%	4.88	0.41
JFS	40	0	78	0	0	33.9%	793.65	>1200
StageSAT	118	0	0	0	0	100.0%	10.34	0.18

比完备求解器超时还要早。没有任何实例中 StageSAT 的答案与已知真实值不一致。这种一致程度让我们有信心认为，StageSAT 的方法不仅能扩展到更大规模，在较小规模下也表现良好。

将 StageSAT 与所有测试集上的完备求解器进行比较后，我们观察到两者的表现互为补充。Bitwuzla 和 CVC5 等完备 SMT 求解器始终保持完美的可靠性（绝不会误判为不可满足），只要有足够时间，最终就能解决每个问题；但在可满足的情况下，它们通常需要耗费显著更多时间。另一方面，StageSAT 的优势在于速度：总体而言，它大幅缩短了可满足基准测试的求解时间；与各类别中的完备求解器相比，其平均运行时间降低了 3×–100×，运行时间中位数则降低了一个到两个数量级

。尤其是在 MathSAT-Large 中较难的部分，以及 JFS/Grater 测试集（其中包含复杂的数学约束）上，StageSAT 的数值搜索比穷举搜索快得多地找到解，而完备求解器则往往遭遇许多超时或长时间运行。在覆盖率方面，StageSAT 也颇具竞争力：在 Large 上，它解决的实例数量不少于表现最好的完备求解器；在其他测试套件中，它解决了每一个实例，而一些完备求解器则难以处理其中少数实例（这是某些公式进行位爆炸时出现指数级膨胀所致）。例如，在 Grater 基准测试中，StageSAT 没有发生超时，而一些完备求解器在少数实例上超时；JFS 测试套件也有类似情况。这表明，对于这类浮点问题，StageSAT 可以作为有效的前端求解器，快速解决容易的可满足问题以及许多较难的问题，从而减轻完备求解器的负担。

6 相关工作

主流浮点 *SMT* 求解器。现代 SMT 求解器，例如 Z3 [9], CVC5 [2],MathSAT5 [7, 17], 和 Bitwuzla [24] 代表着浮点约束位精确推理的最先进水平。这些求解器会将浮点公式规约为较低层次的理论（通常通过位爆炸转换为位向量），并利用 DPLL(T)/CDCL(T) [8, 14, 21, 25]搜索来确保完备性。这种方法已在许多基准测试中证明非常有效——事实上，Bitwuzla 和 CVC5 在近期 SMT-COMP 的浮点除法竞赛中表现出色[22]。然而，位精确方法可能难以应对低层编码所带来的巨大搜索空间，尤其是在处理复杂数值约束时。相比之下，StageSAT 采用了另一种途径：它不进行位级穷举逻辑推理，而是利用数值优化技术 [13, 23, 26] 在搜索空间中探索。这种策略与位爆炸方法有着根本区别，且与之互为补充。StageSAT 无需显式枚举位模式，因此有望扩展到传统求解器难以处理的约束，同时最终仍能生成位精确满足赋值。与 Bitwuzla 最近引入的位级局部搜索启发式方法不同[11], StageSAT 在数值层面运行，利用分阶段优化过程引导求解器找到解。这一新颖视角使 StageSAT 能够从全新角度处理浮点约束，而无需与 Z3、CVC5、MathSAT 和 Bitwuzla 复杂但开销巨大的位精确程序直接竞争。

启发式与基于模糊测试的求解器。 另一条研究路线探索了用于 SMT 求解的启发式方法。值得注意的是，JFS（“Just Fuzz It”求解器） [20] 将覆盖率引导模糊测试 [4, 31]应用于浮点可满足性，根据求解器执行的反馈随机变异输入。这种方法能够快速找到某些令传统求解器束手无策的浮点数问题的解。然而，JFS 既不完备，也不基于优化：它无法证明不可满足性，除了代码覆盖率启发式之外，也缺少有原则的目标函数。StageSAT 则截然不同：我们定义了一个明确的数学目标函数，其最小化对应于满足该公式。JFS 展示了非常规搜索策略的价值，而 StageSAT 则提供了一种更系统、以优化为基础的方法。与 JFS 的随机探索不同，StageSAT 利用其优化框架，在有原则的引导下寻找模型，而非依靠猜测。

基于优化的浮点求解器。我们的工作最直接地受到此前通过数学优化求解浮点约束的尝试的启发。XSat [12] 开创了这一方向：将约束转换为实值优化问题，并依据 IEEE-754 语义进行检验 [1]。这带来了快速的求解器，但可能遗漏实数解与浮点解不一致时的边界情况。GoSAT [3] 将浮点可满足性表述为全局优化问题，能够高效地为困难的 FP 问题寻找模型，但仍不完备，且容易受到不连续区域的影响。最近，Grater [6] 通过精心构造连续目标函数并采用基于梯度的方法，改进了这一范式，取得了令人瞩目的性能，达到或

在多个基准测试中超越 Bitwuzla 和 CVC5。parSAT [19] 通过在多核 CPU 上并行运行多种随机优化方法，进一步拓展了这项工作 [18, 29, 30] 。尽管取得了这些进展，所有先前基于优化的求解器都采用单阶段流程：将整个公式简化为一个单体目标函数，再应用现成优化器。这种单阶段方法使它们容易陷入局部最小值，也难以应对具有分段定义语义或突变不连续点的不规则约束地形。

StageSAT 凭借其分阶段优化 策略脱颖而出，据我们所知，先前的 SMT 求解器尚未探索过这种策略。StageSAT 不再只执行一次优化，而是将求解过程拆分为多个阶段，每个阶段解决一个子问题，逐步逼近可满足状态。这使 StageSAT 能够克服早期工具难以应对的障碍：它首先求解一个放宽后的约束版本（平滑非线性或不连续行为），然后逐步重新引入完整的浮点语义。这种方法避免了单次求解方法容易陷入的局部最小值，并通过将不连续点隔离在特定阶段来加以处理。XSat、goSAT 和 Grater 均未采用这种多阶段方法。这种多阶段设计兼顾了数值优化的平滑性与位级精度，因此，与先前技术相比，该求解器在某些困难的基准测试中更加稳健。

7 结论

我们介绍了 StageSAT，这是一款用于 SMT 逻辑 QF_FP 可满足性求解的求解器，它通过新颖的三阶段架构应对浮点约束带来的挑战。StageSAT 的流程结合了 (i) 快速的投影辅助下降、(ii) ULP^2 形目标函数优化，以及 (iii) n-ULP 格点细化。通过将数值搜索与最终的位级细化相结合，StageSAT 能够高效地遍历浮点搜索空间，同时确保解的正确性。这种设计既准确又切实有效，克服了先前数值求解器的关键局限性；这些求解器往往会出现搜索精度不足或结果不完备的问题。具体而言，StageSAT 的格点细化阶段能够保证高精度，这使其有别于早期数值 SMT 求解器，后者并不总能确保结果正确或完备。总的来说，StageSAT 的多阶段方法代表了浮点 SMT 求解领域一项有原则的进步，它以此前工具未曾采用的方式，结合了连续优化与离散搜索的优势。

我们的实验结果验证了 StageSAT 方法的有效性：在我们的基准测试中，其 SAT 召回率不低于其他所有求解器。它解决的大型、高难度基准测试也多于任何竞争求解器，并在最复杂的问题集上取得了最高覆盖率。同时，在所有受评估的工具中，StageSAT 的超时率最低，表明它在处理困难约束时具有更高的可靠性。例如，在规模最大的基准测试类别中，StageSAT 解决的实例最多，超时也最少；而表现最佳的替代方案仍有多个可满足问题未能解决。这些具体结果表明，StageSAT 在实践中表现稳健且准确，在困难的浮点问题上显著优于以往的数值方法和最先进的位精确求解器。总之，StageSAT 求解器新颖、准确且有效——在我们的评估中，它通过高精度、高效率地求解复杂浮点约束，推动了 QF_FP 可满足性的最先进水平。

参考文献

[1] 2019。IEEE 浮点算术标准。IEEE Std 754-2019（IEEE 754-2008 的修订版）（2019），1–84。doi:10.1109/IEEESTD.2019.8766229[2] Haniel Barbosa、Clark Barrett、Martin Brain、Gereon Kremer、Hanna Lachnitt、Makai Mann、Abdalrhman Mohamed、Mudathir Mohamed、Aina Niemetz、Andres Nötzli、Alex Ozdemir、Mathias Preiner、Andrew Reynolds、Ying Sheng、Cesare Tinelli 和 Yoni Zohar。2022。cvc5：一种通用且具备工业级能力的 SMT 求解器。载于《系统构造与分析工具和算法：第 28 届国际会议 TACAS 2022，作为欧洲软件理论与实践联合会议 ETAPS 2022 的一部分举行，德国慕尼黑，2022 年 4 月 2–7 日，论文集，第一部分》（德国慕尼黑）。施普林格出版社，柏林、海德堡，415 442 10 1007 978 3 030 99524 9_24 –。doi: . / - - - - M. Ammar Ben [3] 2017 Khadra、Dominik Stoffel 和 Wolfgang Kunz。goSAT：将浮点可满足性作为全局优化问题。载于《2017 年计算机辅助设计形式化方法，FMCAD 2017》，奥地利维也纳，2017 年 10 月 2–6 日，Daryl Stewart 和 Georg Weissenbacher（编）。IEEE，11 14 10 23919 2017 8102235 –。doi: . /FMCAD. . Marcel Böhme、[4] 2019 Van-Thuan Pham 和 Abhik Roychoudhury。基于马尔可夫链的灰盒模糊测试。《IEEE 软件工程汇刊》45, 5 2019 489 506 10 1109 2017 2785841 ( )，–。doi: . /TSE. . Qian Chen、[5] 2025 Chenqi Cui、Fengjuan Gao、Yu Wang、Ke Wang 和 Linzhang Wang。Grater-experiment：用于“使用连续优化求解浮点约束”的制品和代码。https://github.com/grater-exp/grater-experiment。提交版本 e405648，访问于 2025年11月13日。[6] Qian Chen、Chenqi Cui、Fengjuan Gao、Yu Wang、Ke Wang 和 Linzhang Wang。2025。使用连续优化求解浮点约束。《ACM 程序设计语言会议论文集》9, 2025 725 747 10 1145 3729279 PLDI ( )，–。doi: . / [7] 2013 5 Alessandro Cimatti、Alberto Griggio、Bastiaan Schaafsma 和 Roberto Sebastiani。《MathSAT SMT 求解器》。载于 TACAS 论文集（LNCS，第 7795 卷），Nir Piterman 和 Scott Smolka（编）。施普林格。[8] Martin Davis 和 Hilary Putnam。1960。量化理论的计算程序。《J. ACM》7，3（1960 年 7 月），201–215。doi:10.1145/321033.321034[9] Leonardo De Moura 和 Nikolaj Bjørner。2008。Z3：一种高效的 SMT 求解器。载于《软件理论与实践论文集：第 14 届系统构造与分析工具和算法国际会议》（匈牙利布达佩斯）（TACAS'08/ETAPS'08）。施普林格出版社，柏林、海德堡，337–340。[10] Leonardo de Moura、Nikolaj Bjørner 和 Eric Pony。2012。Z3Py 教程。https://ericpony.github.io/z3py-tutorial/guide-examples.htm。访问于 2025年11月13日。[11] Andreas Fröhlich、Armin Biere、Christoph M. Wintersteiger 和 Youssef Hamadi。2015。用于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》，1 月 *25 30, 2015,* 1136 1143 10 1609 29 1 9372 -，美国得克萨斯州 [12] 2016 奥斯汀，Blai Bonet 和 Sven Koenig（编）。AAAI Press，–。doi: . /AAAI.V I . Zhoulai Fu 和 Zhendong Su。XSat：一种快速的浮点可满足性求解器。载于《计算机辅助验证——第 th 届国际会议 CAV 2016》，加拿大安大略省多伦多，2016 年 7 月 17–23 日，论文集，第二部分（《计算机科学讲义》，第 *9780* 187 209 10 1007 978 3 319 41540 6_11 卷），Swarat [13] 2019 Chaudhuri 和 Azadeh Farzan（编）。施普林格，–。doi: . / - - - - Zhoulai Fu 和 Zhendong Su。通过弱距离最小化实现有效浮点分析。载于《第 40 届 ACM SIGPLAN 编程语言设计与实现会议论文集》（美国亚利桑那州凤凰城）（PLDI *2019* 439 452 10 1145 3314221 3314632 ）。计算机协会，纽约，纽约 [14] 2004 州，美国，–。doi: . / . Harald Ganzinger、George Hagen、Robert Nieuwenhuis、Albert Oliveras 和 Cesare Tinelli。DPLL(T)：快速判定过程。载于《计算机辅助验证：第 16 届国际会议 CAV 2004》，美国马萨诸塞州波士顿，2004 年 7 月 13–17 日，论文集（《计算机科学讲义》，第 3114 卷），Rajeev Alur 和 Doron A. Peled（编）。施普林格，175–188。doi:10.1007/978-3-540-27813-9 14_[15] David Goldberg。1991。每位计算机科学家都应该了解的浮点运算知识。《ACM 计算调查》23, 1 1991 5 48 10 1145 103162 103163 （3 月），–。doi: . / . Gene H. Golub 和 Charles F. Van Loan。《矩阵计 [16] 1996 算》，第三版。约翰·霍普金斯大学出版社。[17] Leopold Haller、Alberto Griggio、Martin Brain 和 Daniel Kroening。2012。通过系统化抽象判定浮点逻辑。载于《计算机辅助设计形式化方法》，FMCAD 2012，英国剑桥，2012 年 10 月 22–25 日，Gianpiero Cabodi 和 Satnam Singh（编）。IEEE，131–140。https://ieeexplore.ieee.org/document/6462565/ [18] P. Kaelo 和 M. M. Ali。2006。受控随机搜索算法的若干变体及其在全局优化中的应用。《优化理论与应用杂志》130, 2 2006 253 264 10 1007 10957 006 9101 0 ( )，–。doi: . /s - - - Markus [19] 2025 Krahl、Matthias Güdemann 和 Stefan Wallentowitz。parSAT：浮点可满足性的并行求解。CoRR abs/2509 16237 2025 2509 16237 10 48550 2509 16237 。( )。arXiv: . doi: . /ARXIV. . Daniel [20] 2019 Liew、Cristian Cadar、Alastair F. Donaldson 和 J. Ryan Stinnett。Just fuzz it：使用覆盖率引导模糊测试求解浮点约束。载于《ACM 欧洲软件工程会议与软件工程基础研讨会联合会议论文集》，ESEC/SIGSOFT FSE 2019，爱沙尼亚塔林，2019 年 8 月 26–30 日，Marlon Dumas、Dietmar Pfahl、Sven Apel 和 Alessandra Russo（编）。ACM，521–532。doi:10.1145/3338906.3338921

[21] J.P. Marques-Silva 和 K.A. Sakallah。1999年。GRASP：用于命题可满足性的搜索算法。IEEE Trans. Comput. 48, 5 1999 506 521 10 1109 12 769433 ( ), – 。doi：。/。Martin Jonáš、François Bobot、David Déharbe 和 [22] Dominik Winterer。2025年。SMT-COMP 2025：第20届理论模可满足性国际竞赛。https://smt-comp.github.io/2025/。访问日期：2025年11月12日。[23] W. Miller 和 D.L. Spooner。1976年。浮点测试数据的自动生成。IEEE 软件工程汇刊 SE-2, 3 1976 223 226 10 1109 1976 233818 ( ), – 。doi：。/TSE. 。Aina Niemetz 和 Mathias Preiner。 。 [24] 2023 Bitwuzla。载于《计算机辅助验证——第35届国际会议，CAV 2023，法国巴黎，2023年7月17日至22日，会议论文集，第二部分》（《计算机科学讲义》，第13965卷），Constantin Enea 和 Akash Lal（编）。施普林格，3 17 10 1007 978 3 031 37703 7_1 – 。doi：。/ - - - - Robert Nieuwenhuis、Albert Oliveras 和 [25] 2006 Cesare Tinelli。 。求解 SAT 和理论模 SAT：从抽象的 Davis–Putnam–Logemann–Loveland 过程到 DPLL(T)。J. ACM 53, 6 2006 937 977 10 1145 1217856 1217859 ( ), – 。doi：。/。 [26] 1999 10 1007 Jorge Nocedal 和 Stephen J. Wright。 。数值优化。施普林格。doi：。/B 98874 [27] 1955 R. Penrose。 。矩阵的广义逆。《剑桥哲学学会数学学报》 51, 3 1955 406 413 10 1017 0305004100030401 ( ), – 。doi：。/S Mathias Preiner、 [28] Hans-Jörg Schurr、Clark Barrett、Pascal Fontaine、Aina Niemetz 和 Cesare Tinelli。2025年。SMT-LIB 2025版（非增量基准测试）。https://zenodo.org/records/16740866 SMT-LIB非增量基准测试库。[29] T.P. Runarsson 和 Xin Yao。2005年。约束进化优化中的搜索偏差。《系统、人类与控制论》IEEE汇刊，C辑（应用与评论）35, 2 2005 233 243 10 1109 2004 841906 ( ), – 。doi：。/ [30] 1997 TSMCC. 。David J. Wales 和 Jonathan P. K. Doye。 。通过盆地跳跃进行全局优化及包含最多110个原子的 Lennard-Jones团簇的最低能量结构。《物理化学杂志》A辑 101，28（1997年7月），5111–5116。doi:10.1021/jp970984n [31] Michał Zalewski。 [n. d.]。afl-fuzz 的技术白皮书。http://lcamtuf.coredump.cx/afl/technical_details.txt。访问日期：2025年11月13日。

StageSat 实现伪代码

算法1 使用全局最小化求解目标函数 1：输入：2：C：FP 的无量词浮点约束 3：R_{sq}：快速投影辅助目标函数 4：R_{ulp}：ULP距离目标函数 5：输出：6：如果找到，则输出带有 C模型的 sat，否则输出不可满足猜测 7：8：程序 Global_Minimizer(C, R_{sq}, R_{ulp}) 9：for $i = 1$ to $nStartOver$ do 10：▷ 第1轮：使用 R_{sq}快速收敛 11：$x_1^{(0)} \leftarrow$ 随机_起始_点() 12：$x_{(L)\,1} \leftarrow$ 局部_最小化($R_{sq}, x_{(0)\,1}$) 13：▷ 第2轮：使用精确的 R_{ulp}进行细化 14：$x_{(L)\,2} \leftarrow$ 局部_最小化($R_{ulp}, x_{(L)\,1}$) 15：if IsModel($C, x_{(L)\,2}$) then ▷ 使用 Z3库函数进行模型验证 16：返回 sat，$x_2^{(L)}$ 17：结束 if 18：▷ 第3轮：在浮点邻域中进行细粒度搜索 19：$N^{(0)} \leftarrow$ zeros(dim) ▷ 从零偏移量开始 20：定义 $f(N) = R_{ulp}($)第_个_浮点数($N[j], x_{(L)\,2}[j]$)，其中 $j = 1$ to dim]) 21：$N_{(L)} \leftarrow$ 局部_最小化($f, N_{(0)}$) 22：$x_{(L)\,3} \leftarrow$ [第_个_浮点数($N_{(L)}[j], x_{(L)\,2}[j]$)，其中 $j = 1$ to dim]23：if IsModel($C, x_{(L)\,3}$) then 24：返回 sat，$x_3^{(L)}$ 25：结束 if 26：结束 for 27：返回“不可满足猜测”28：结束程序 29：30：函数 Local_Minimize($R, x_{(0)}$) 31：$x_{(L)} \leftarrow$ scipy.optimize.basinhopping($R, x_{(0)}$) 32：返回 $x^{(L)}$33：结束函数 34：35：函数 Nth_Floating_Point(n, x) 36：▷ 时间复杂度：O(1) 37：返回从 x开始的第 n个可表示浮点数 38：结束函数

算法2 生成目标函数

输入： C (无量词浮点约束)
输出： R_{sq}， R_{ulp}

程序生成_实数集_Sqare(C)
    C_{leq}， $C_{other} \leftarrow$ 解析_C(C)
    如果 $C_{leq} = \emptyset$ **那么**
        $R_{sq}(x) \leftarrow \sum C_{other}$
    else
        A， $b \leftarrow$ 构建_矩阵(C_{leq})
        $G^{+} \leftarrow$Moore–Penrose伪逆($AA^{\top}$)
        $P \leftarrow A^{\top}G^{+}$; $M \leftarrow I - PA$; $c \leftarrow Pb$
        $R_{sq}(x) \leftarrow \sum_{i=1}^{n}(x_i - (Mx + c)_i)^2 + \sum C_{other}$
    结束如果
    返回 R_{sq}
结束程序

程序生成_实数集_ULP(C)
    $C = \{c_i :$ 左操作数$_i =$ 右操作数$_i \mid i = 1, \ldots, m\}$
    $R_{ulp}(x) \leftarrow \sum_{i=1}^{m}$ ULP(lhs$_i$， rhs$_i$)
    返回 R_{ulp}
结束程序