arXiv:2301.11919v2 [cs.LG] 4 May 2023

利用计算机代数系统将背景知识融入符号回归

预印本

查尔斯·福克斯[2], 尼尔·特兰[1], 妮基·纳西翁[1],萨米哈·沙林[1],和泰勒·R.·约瑟夫森[1,2]

[1]马里兰大学巴尔的摩县分校化学、生物化学与环境工程系，马里兰州巴尔的摩市 Hilltop Circle 1000 号，邮编 21250[2]马里兰大学巴尔的摩县分校计算机科学与电气工程系，马里兰州巴尔的摩市 Hilltop Circle 1000 号，邮编 21250

摘要

符号回归（SR）能够生成可解释且简洁的表达式，以拟合给定数据集；与黑箱方法相比，这有助于人们更好地理解数据的结构。加入背景知识（以符号化数学约束的形式）后，生成的表达式既能符合理论，又能与数据一致。我们具体研究了如何将约束加入传统的、基于遗传算法（GA）的符号回归方法 PySR，以及基于马尔可夫链蒙特卡洛（MCMC）的贝叶斯符号回归架构 Bayesian Machine Scientist，并将这些方法用于从实验性历史数据集中重新发现吸附方程。我们发现，硬约束会阻止 GA 和 MCMC 符号回归进行搜索；而软约束则能提升搜索效果和模型的合理性，但计算成本会增加约一个数量级。如果约束与数据集或预期模型的相关性不佳，就可能妨碍表达式搜索。我们还发现，在贝叶斯符号回归中将这些约束用作贝叶斯先验，优于在 GA 中通过修改适应度函数来加入约束。

1 引言

1.1 用于科学发现的符号回归

符号回归（SR）会针对给定数据集生成在复杂度和准确度方面经过优化的数学表达式。自约翰·科扎率先提出通过自然选择进行编程这一范式以来，符号回归在科学发现中的诸多应用应运而生 [1]。与机器学习技术的其他应用不同，科学研究要求具备可解释性和可验证性，而生成人类可理解的数学模型可以使二者更易实现（而不是拟合一个包含数千个参数的模型） [2–4]。此外，即使数据集非常小，符号回归也可能有效，例如由难以开展或成本高昂、无法轻易重复的实验所产生的数据（∼10 项）。符号回归生成的数学表达式可以轻松外推至数据集中未经测试或以其他方式无法到达的领域（例如极端压力或温度条件）。

几十年来，符号回归已从众多应用领域的数据中发现了有趣的模型，包括推断陶氏化学公司的过程模型 [5], 降雨—径流建模 [6], 以及重新发现双摆运动方程 [7]。符号回归已应用于科学研究的各个尺度，包括原子尺度（原子间势 [8]）、宏观尺度（计算流体力学 [9]）和宇宙学尺度（暗物质超密度[10]）。一些技术支持在数十亿个候选表达式中进行搜索，例如在材料性质的非线性描述符空间中进行搜索 [11]。虽然科学领域中大多数符号回归应用都侧重于识别数据中的经验模式，但这类“仅数据”方法并未考虑背景理论可能带来的洞见。事实上，一些符号回归研究强调其能够“无需任何关于物理学、运动学或几何学的先验知识”进行发现 [7]。尽管如此，我们认为先验知识不必被舍弃；在本研究中，我们探讨如何

将理论融入符号回归，从而展示背景知识语境下的机器学习。

1.2 将背景知识融入符号回归

在特定领域有效使用符号回归的一个尤为重要的步骤，是加入先验知识。这一步有望将通用符号回归算法用于寻找具有物理意义的新模型。例如，AI-DARWIN 以预定义函数的形式利用化学反应机理的先验知识，遗传算法可以在搜索方程空间时使用这些函数，从而确保生成的每个模型都具有机理意义 [12]。这种方法具体是将先验知识编码为可用的函数，而不是限制所生成的函数。在另一个近期的例子中，Engle 和 Sahinidis 使用一种确定性的符号回归算法，对可能的方程空间施加约束；这些方程并非必须由具有意义的函数组件库构成，而只需满足理论中的导数约束即可。这提高了热力学状态方程所生成表达式的质量 [13]。将背景知识融入符号回归的另一种方法是贝叶斯机器科学家（BMS） [14]。BMS 以符号表达式上的贝叶斯先验形式，严格地融入背景知识；如果表达式中数学运算符的分布与一组重要方程语料库中的运算符分布相一致，那么这些表达式在 先验上 更有可能出现。然而，他们构建贝叶斯先验的方法并未融入特定科学领域的含义。

在搜索完成之后检查方程的一致性也是可行的。此前我们展示过，可以将生成的表达式与丰富的背景知识（以针对所研究环境的公理形式表达）进行比较，方法是将生成的表达式作为猜想提交给自动定理证明器（ATP） [15]。然而，当前最先进的 ATP 速度过慢，无法在生成符号表达式的同时纳入这一逻辑检查，因此也难以利用这些背景知识来引导方程搜索。此外，将科学理论转换为计算机可解释的形式并非易事。

我们通过将符号回归系统（包括遗传算法和贝叶斯方法）与计算机代数系统（CAS）相结合，解决了这些具体缺陷；CAS 会在方程搜索过程中检查约束。这与逻辑引导遗传算法（LGGA）类似。LGGA 使用与数据集相对应的“辅助真值”（AT），以便对数据集中的项目进行加权，并用更多信息对数据集加以扩充 [16]。LGGA 采用迭代方法：先使用某个数据集训练任意遗传算法，再用 AT 扩充该数据集，然后使用信息更丰富的数据再次训练该算法。我们的工作与 LGGA 之间的一个重要区别在于，数据集不会以任何方式被修改，额外信息的添加是在 GA 执行期间完成的。另一种相关方法是形状约束符号回归 [17, 18], , 该方法利用整数算术的高效应用，将对函数形状的约束（例如偏导数和单调性）纳入符号回归。我们的方法考虑了更广泛的约束（即任何可以由 CAS 定义和检查的约束），并同时考虑遗传编程方法和贝叶斯符号回归方法。

1.3 吸附

吸附是分子与表面结合的现象，可实现包括碳捕集、湿度控制、去除水中的有害污染物以及制氢在内的化学过程 [19] [20] [21] [22]。吸附模型能够预测和设计工程化吸附过程，多年来人们已经提出了许多模型（表 1 展示了其中选定的方程） [23–26]。这些模型将平衡状态下的吸附量表示为压力或浓度的函数，通常以经验方程或基于理论推导的方程表示。例如，Freundlich 等温线 [27], 是为拟合观测数据而设计的经验函数，Langmuir [28]和 BET [29]等温线源自物理模型，而 Sips [30] 等温线则以 Langmuir 为基础，并加入了经验项以提高拟合灵活性。我们不禁思考：“机器学习系统能够生成什么类型的模型？背景知识又能在搜索准确且有意义表达式的过程中发挥什么作用？”

表 1：一些著名的等温线以符号回归可能发现的形式表示，以及它们的复杂度。

等温线	文献表达式	符号回归形式	符号回归复杂度
Langmuir [28]	$\frac{q_{max}K_{eq}p}{1+K_{eq}p}$	$\frac{c_1 p}{c_2+p}$	7
双位点 Langmuir [28]	$\frac{q^a_{max}K^a_{eq}p}{1+K^a_{eq}p} + \frac{q^b_{max}K^b_{eq}p}{1+K^b_{eq}p}$	$\frac{c_1 p}{c_2+p} + \frac{c_3 p}{c_4+p}$	15
BET [29]	$\frac{v_m * c * (p/p_0)}{(1-p/p_0)*(1+(c-1)p/p_0)}$	$\frac{c_1 p}{p^2+c2p+c3}$	13
Freundlich [27]	$c_1 p^{\frac{1}{n}}$	$c_1 p^{c_2}$	5
Sips [30]	$\frac{c_1 p^{\frac{1}{n}}}{1+c_1 p^{\frac{1}{n}}}$	$\frac{p^{c_2}}{c_1+p^{c_2}}$	9

1.4 热力学约束

当模型满足适用于建模这些现象的函数形式所需的热力学约束时，我们认为模型更加 有意义 。也就是说，一个能够拟合数据、但无法正确趋近于零负载量的随机方程，与一个受热力学约束、遵循热力学规律的方程相比，在训练数据之外的可信度更低。我们确定了三个与单组分吸附相关的约束 [15]:

$$\lim_{p\to 0} f(p) = 0 \tag{1}$$

$$\lim_{p\to 0} f'(p) < \infty \tag{2}$$

$$\forall p > 0 \qquad f'(p) \geq 0 \tag{3}$$

约束 1 确保在压力为零的极限情况下，所有分子都必须脱附，且负载量不能为负。约束 2 要求在压力为零的极限情况下，等温线的斜率必须是一个有限的正常数。Talu 和 Myers 表明，当压力趋近于零时，吸附等温线的斜率等于吸附第二维里系数 B_{1S}，该系数表征一个分子与表面之间的相互作用，并且必须是有限的正数 [31] [32]:

$$\lim_{p\to 0} \frac{df}{dp} = \frac{B_{1S}}{RT} = c \tag{4}$$

约束 3 要求对于所有 (∀) 正压力值，负载量不会随着压力增加而降低（等温线是单调非递减的）。需要注意的是，这一约束不适用于混合物吸附（其中竞争作用会发挥影响），也不适用于 BET 吸附；后者在饱和压力处表现为不连续，而不是单调增加。

1.5 PySR：使用遗传算法进行符号回归

PySR（用于符号回归的 Python）是一个 Python 库，它使用遗传算法进行符号回归 [33]。PySR 是一个 Python 封装库，它调用同一作者编写的 Julia 库 SymbolicRegression.jl（SR.jl），以提升数值计算性能。由于本研究所需的算法修改具有特殊性，因此使用了基础 Julia 库；不过，所有新增功能也应能由 Python 封装库继承。

基本前提是，一个或多个模型群体通过随机变异，逐渐趋向更优的解。在每一代中，会根据适应度、年龄或其他某些标准移除群体中的部分成员（PySR 会替换最年长的成员）。通过让群体中更优的成员进行变异和繁殖，可以促进有益解的产生。

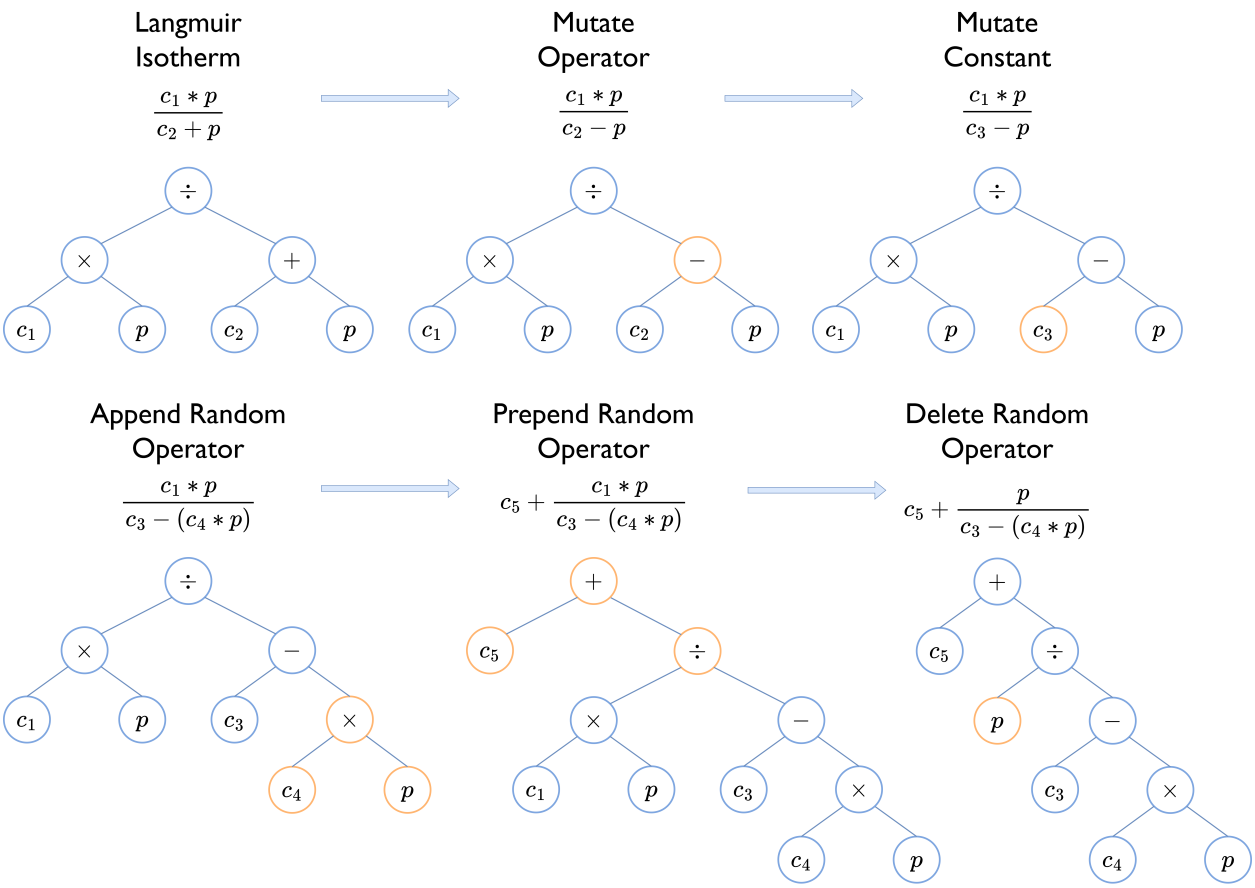


图 1：PySR 中的所有突变（随机树生成和简化除外）依次进行的过程（从左到右、从上到下阅读）。相较于每个先前表达式树的变化以橙色显示。

这些变化包括对单个常数或运算符进行变异、简化表达式，或在两个表达式之间进行交叉（图 1 和图 2）。PySR 采用了类似岛屿方法的多群体机制 [34]。其目的在于通过分别演化彼此独特的群体来实现专业化，同时偶尔允许部分成员在群体之间迁移，以共享这种专业化能力。具体而言，PySR 实现了所谓的名人堂（HOF），它是由各群体中最佳成员构成的帕累托前沿。经过若干代后，每个群体会提交其评分最高的前 10 个成员，然后通过帕累托前沿对它们进行比较和筛选。保留在 HOF 中的表达式将用于各群体未来的变异。

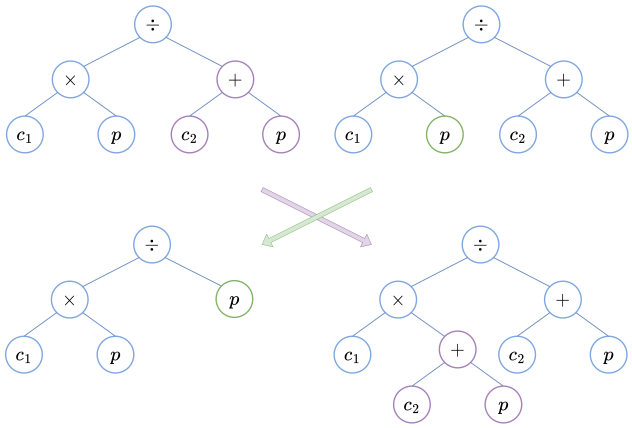


图 2：两个表达式树之间交叉突变的示例。

1.6 贝叶斯符号回归

Guimera 等人提出的贝叶斯机器科学家（BMS） [14] 从贝叶斯视角处理符号回归。贝叶斯符号回归（BSR）将搜索准确、简洁且包含背景知识的模型，表述为：从相对于某个先验并基于数据集拟合的符号模型边缘后验分布中进行采样。马尔可夫链蒙特卡洛（MC）用于生成新的表达式树（图 3），并根据其似然度决定接受或拒绝这些表达式树。作者定义了三种 MC 移动：节点替换、根节点添加或移除，以及基本树替换。这些移动共同实现了表达式树的构建，同时保持详细平衡，从而确保对后验分布进行适当采样。

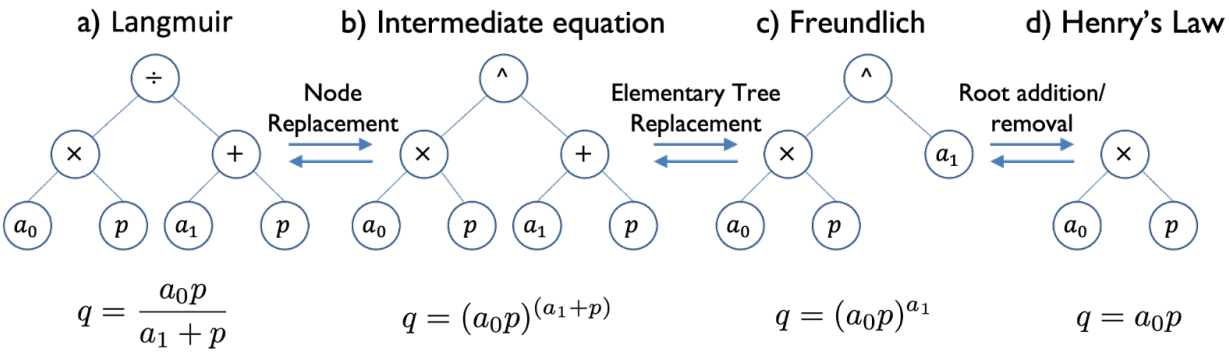


图 3：展示应用于吸附方程时 BMS 算法可用的移动。与 PySR 中可用的突变不同，这些变换满足详细平衡 [14]。

具体而言，给定某些数据时某个模型的概率定义为：

$$p(f_i|D) = \frac{1}{Z}\int_{\Theta_i} d\theta_i p(D|f_i,\theta_i)p(\theta_i|f_i)p(f_i) = \frac{\exp[-\mathcal{L}(f_i)]}{Z} \tag{5}$$

其中， Z 是数据集 $p(D)$ 的概率， Θ_i 是参数 θ_i 的可能取值空间， $\mathcal{L}$ 是模型的描述长度。

BSR 的一个核心思想是引入先验，以强调那些先验上比其他表达式更可能出现的表达式，而不考虑数据的影响。Guimera 等人基于从维基百科（“以人物命名的科学方程列表”）收集的 4080 个数学表达式构建了先验，并通过统计语料库中每个独特运算符（n_o）的出现次数，拟合参数 α 和 β，从而为函数 $p(f_i)$（及其“能量”EP）赋予先验概率，如下所示：

$$\mathrm{EP} = -\log(p(f_i)) = \sum_{o\in O}\left[\alpha_o n_o(f_i) + \beta_o n_o^2(f_i)\right] \tag{6}$$

尽管该方法在不使用数据运行时会产生类似于语料库的表达式分布，但也可以将 $p(f_i)$ 设为常数值，从而消除搜索过程中基于所含运算符产生的偏差。针对我们的问题，我们专门为吸附热力学构建了一个先验（详见方法部分）。

2 方法

2.1 检查热力学约束

我们使用 Python 库 SymPy（一种开源计算机代数系统） [35]，为第1.4节中描述的热力学约束开发了三个约束检查函数。每个函数根据相应约束是否满足返回 true 或 false（如果超出时间限制，则返回 false）。对于 PySR 和 BSR，我们发现，硬约束（拒绝任何违反任一约束的表达式）会严重阻碍搜索过程，截断可能通向更优表达式的中间表达式，而这些更优表达式也可能满足约束。因此，我们将这些约束设为“软”约束：对违反约束的表达式进行惩罚，而不是直接拒绝它们。 [17]和 [18] 也发现，软约束比硬约束更有效。算法1详细说明了这一方法（该方法已在 PySR 中实现）。

约束1和约束2可以使用 SymPy 的 limit 和 derivative 功能进行检查，但约束3更具挑战性。尽管 SymPy 可以检查表达式在给定范围内是否严格递增，但其单调性检查会在表达式的曲率发生任何变化（出现临界点）时返回 false，因此无法将 x^3 等函数视为单调非递减。为允许斜率为零，我们实现了一个自定义的单调非递减检查函数（见算法2）。它不只检查某个范围内的斜率，而是检查所有临界点之间的范围，以及原始范围的起点和终点到相邻临界点之间的范围。
我们推测，符号回归所探索的"方程空间"包含准确但不符合热力学一致性的表达式，而通过纳入背景知识可以剔除这些表达式，引导搜索过程找到更多符合理论的表达式。

2.2 PySR 修改

在 PySR 中，种群中的每个成员都有一个需要最小化的评分，该评分结合了损失和复杂度（由表达式树中的节点总数定义）。当违反热力学约束时，我们会将损失函数乘以一个惩罚项，从而提高评分并降低该表达式的适应度。由于乘法满足交换律，这使得我们可以按任意顺序检查任意数量的约束，并对违反多个约束的表达式施加更大的惩罚。

$$\text{Loss: } L = \ell_2^R * \prod_{i=1,2,3} c_i^{\delta_i} \text{ where } \delta_i = \left\{ \begin{array}{ll} 1 & \text{if constraint } i \text{ passed} \\ 0 & \text{if constraint } i \text{ failed} \end{array} \right\} \tag{7}$$

$$\text{Member Score: } S = L + n_{nodes} * c_l \tag{8}$$

上述方程详细说明了 PySR 中损失和评分的计算方式。 ℓ_2^R是 L2 范数， c_i 是约束 i 的惩罚项， δ_i 表示约束 i 是否通过， c_l 是针对表达式长度/复杂度的惩罚项。
PySR 还可以使用 Julia 或 Python 中定义的任意运算符，包括用户自定义的运算符。本研究仅使用运算符 $+$、$-$、$*$ 和 $\div$ 来控制搜索空间的大小。以规范形式书写的表达式可以使用其他运算符，例如指数运算，但这些运算符仅由生成表达式的简化过程引入。

2.3 BMS 修改

Guimera 等人 [14] 在贝叶斯机器科学家代码中使用的先验，通过考察维基百科中具名方程里的数学运算频率，将"背景知识"纳入方程搜索。作者发现，这有助于搜索一般性的科学方程；而我们的目标是根据研究领域来调整这些背景知识。我们将上述热力学约束视为我们的"先验知识"，并构造以下表达式:

$$\text{EP} = \sum_{o \in O} \left[c_{ops} n_o(f_i) \right] + \sum_{i=1,2} c_i * \delta_i \text{ where } \delta_i = \left\{ \begin{array}{ll} 1 & \text{if constraint } i \text{ passed} \\ 0 & \text{if constraint } i \text{ failed} \end{array} \right\} \tag{9}$$

其中， c_{ops} 是运算符的约束惩罚项（类似于 PySR 中的简约性参数），而 n_o 是表达式 f_i中每个运算符的计数。该表达式直接替代了方程（6），从而改变先验分布。请注意，我们使用 PySR 检查了全部三项约束，而使用 BSR 检查了前两项约束（省略了单调非递减检查）。

3 结果

3.1 数据集

为了考察在搜索过程中向符号回归（SR）添加约束所产生的影响，确定了四个实验吸附数据集：氮和甲烷在云母上的吸附 [28], 硅沸石中异丁烷的吸附， [36] 以及氮在 Fe-Al_20_3 催化剂上的吸附 [29]。前两个数据集来自介绍 Langmuir 等温模型的奠基性论文 [28]。该模型假设存在彼此不相互作用的离散装载"位点"，且每个位点要么被占据，要么未被占据。异丁烷数据集可以用具有两种独特位点类型的双位点朗缪尔模型进行良好描述。第四个数据集（称为 BET 数据集）曾被 BET 理论的作者用于

支持其多层吸附模型。这些数据及相应的真实值模型拟合结果如图 4 所示。

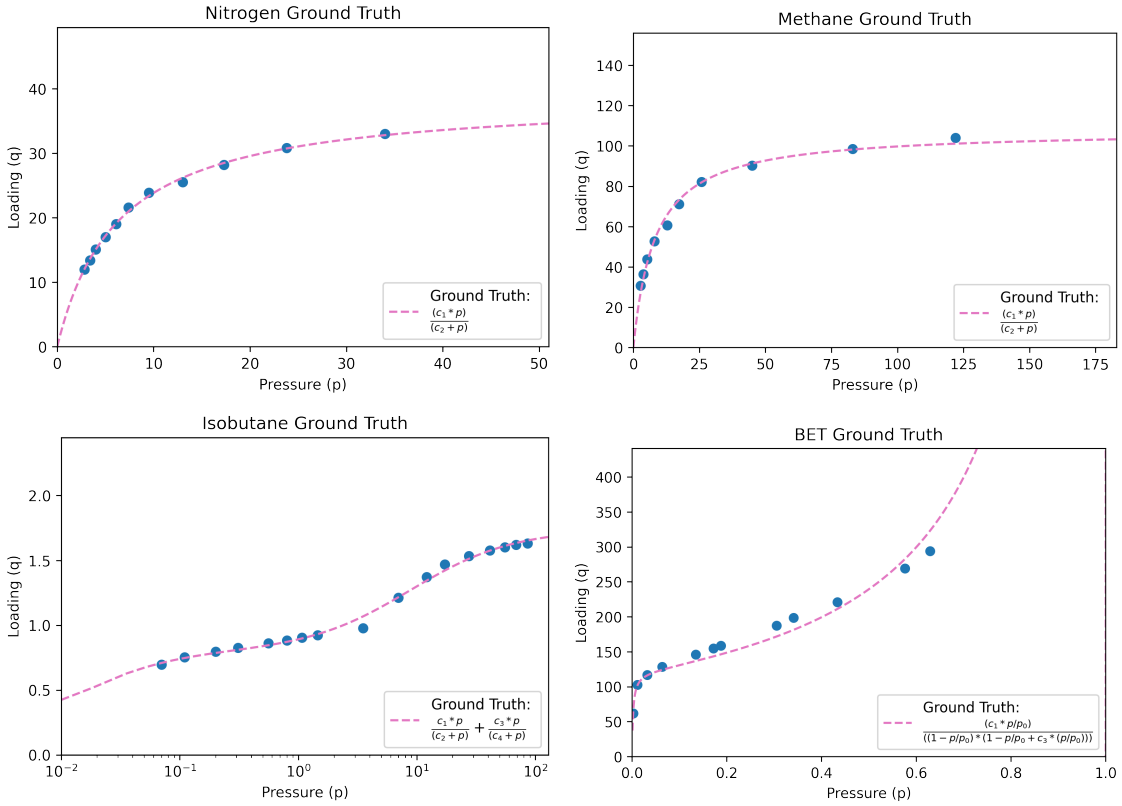


图 4：使用 SciPy 拟合常数后的每个数据集及其对应的真实值模型。异丁烷采用对数刻度的压力显示，以便看清两条彼此分离的曲线。BET 显示的压力增加至 1，以便看清渐近线。

3.2 Langmuir 数据集

本工作的主要结果以两种图表形式展示。左列包含帕累托前沿，显示基于复杂度和准确度的最佳表达式。其中，横轴表示不断增加的复杂度（定义为表达式树中的节点总数），纵轴表示损失，并采用对数刻度，以便更清晰地呈现帕累托前沿的趋势。每种复杂度下的最佳表达式取自 8 次运行中的每一次（灰色曲线），整体帕累托前沿以橙色显示。每个数据集的“真实值”表达式也以符号回归通常可能得到的形式显示，同时给出使用拟合常数计算的损失。每个图的右列显示数据集，以及针对该测试从整体帕累托前沿中选取的表达式。图中仅展示部分表达式，以保持图表的可读性；此外，长度超过真实值表达式的表达式通常存在过拟合，并且会与真实值表达式过于接近而难以区分。真实值采用点线绘制，使得具有相近准确度的表达式仍然可见。将生成的表达式绘制在数据上，有助于说明它们是否遵循热力学约束，以及它们与真实值表达式的相似程度。

图5展示了在 Langmuir 氮数据集上启用和关闭约束时，两种符号回归算法的结果。第一行和第二行分别展示了关闭约束时 BSR 和 PySR 的结果，清楚表明 BSR 找到了真实值，而 PySR 没有。在 BSR 帕累托前沿图（图5a）中，复杂度为 7 的拐角处对应的表达式与真实值无法区分；无论是从数学形式上比较，还是在等温线图（图5b）中观察其在数据上的绘制结果，都是如此。与 PySR 相比，BSR 图（图5a）中 8 次运行的最佳帕累托前沿之间方差要大得多（灰色线条所示），但这可能意味着该算法需要更长时间才能收敛。对应的等温线图（右列）显示，随着表达式变得更加复杂，它们对数据的拟合也更好，这与帕累托前沿的总体趋势一致。这些图还显示，有些表达式虽然能较好地拟合数据，却违反了理论约束，例如 PySR 的图（图5d）。事实上，只有 2.2% 的

PySR 生成的表达式（未强制执行约束）满足第一个约束，只有 33% 满足第二个约束（表2）。在未强制执行约束的情况下，BSR 找到的表达式比 PySR 更一致，其中 37% 的表达式满足第一个约束，67% 满足第二个约束。

启用热力学约束后，其影响在帕累托前沿中清晰可见（最下面两行）。两种符号回归方法都找到了真实值，并达到了相同或相近的准确度（当搜索过程中常数未得到充分优化时，相同表达式的准确度会有所降低）。能够由Langmuir等温线良好表示的数据集很好地体现了这些约束的影响，因为该等温线通常不仅准确度很高，而且表达简洁。等温线图与之前一样显示，随着复杂度增加，表达式对数据的拟合效果会变得更好；但展示复杂度超过7的结果是多余的，因为此时已经发现了真实值，并且几乎完美地匹配预拟合真实值。BSR各次运行之间略有更大差异的趋势在此处也在一定程度上延续，而PySR各次运行之间的差异似乎与禁用约束时大致相近。重要的是，PySR中通过第一项约束的表达式数量增加了5倍（尽管仍仅占10%），而通过另外两项约束的比例也略有提高（分别为8%和13%）。在BSR中，这一变化更加明显：现在通过第一项约束的表达式数量增加了一倍（达到72%），并且相当一部分表达式通过了第三项约束（从0.46%提高到19%），尽管第三项约束并未纳入贝叶斯先验。

虽然甲烷数据集的结果大体相似，但仍存在一些重要差异。与氮数据集类似，在未启用约束的情况下，BSR 能够找到真实值，而 PySR 则不能。这一点从帕累托前沿（图 6a 和图 6c）中可以明显看出。在本例中，PySR 找到的一个复杂度为 9 的表达式，其准确度高于真实值；然而，由于分子中多了一个常数，该表达式违反了热力学约束（图 6d）。施加约束后，该表达式相对于真实值的损失受到惩罚，但惩罚程度不足以抵消准确度的提升（图 6h）。与氮数据集一样，即使未启用约束，BSR 在寻找符合约束的表达式方面也表现更好：它找到的表达式中有 33% 通过了第一项约束，51% 通过了第二项约束（而 PySR 对应的比例分别为 4.1% 和 48%）。

3.3 异丁烷数据集

不同于甲烷和氮数据集（图 5 和图 6），后两者最适合用 Langmuir 等温线建模；异丁烷数据集（图 7）最适合用双位点 Langmuir 等温线建模，其复杂度是前者的两倍。尽管复杂度显著增加，双位点 Langmuir 等温线的准确度并没有显著高于许多比它更短的表达式。图 7c 和图 7g 最能说明这一点，其中分别展示了 PySR 在关闭和启用约束时的帕累托前沿。在两幅图中，复杂度仅为其一半的表达式几乎达到了相同的准确度，从复杂度 7 开始形成平台。这一点在相应的等温线图中也体现得十分明显：复杂度为 7 时找到的表达式与真实值一样，都能很好地拟合数据。需要特别指出的是，这些表达式并不满足热力学约束。

与 PySR 不同，B SR 要到复杂度相同的时候，才会找到准确度接近真实值的表达式 .

对于 BSR，加入约束会使整个帕累托前沿下移（从图 7a 到图 7e），这表明在许多复杂度水平下都找到了更准确的表达式。PySR 没有找到符合约束的准确表达式，而 BSR 找到了。在这种情况下，BSR 找到了真实值表达式，而 PySR 没有。不过，从帕累托前沿或等温线图中都看不出这一点，因为所找到表达式的准确度比拟合真实值以及该复杂度下找到的最佳表达式差约 10 倍。这很可能是因为，尽管找到了真实值，但它最初生成时的形式（简化之前）复杂得多。
在 PySR 中，对违反约束的表达式施加惩罚，反而使方程群更频繁地违反约束二和约束三，两种情况下的比例都下降了约 10%（见表 2）。这令人意外——我们原以为施加惩罚项会减少违反约束的表达式，但结果恰恰相反。对于 BSR，在先验中加入约束也导致满足第二项约束的表达式比例下降（从 46% 降至 36%），而满足第一项和第三项约束的表达式比例略有上升。

3.4 BET数据集

BET数据集很特别，因为当压力接近 1 时，真实值表达式会趋于无穷大（此处的压力是相对蒸气压，p/p^{sat}；当 $p/p^{\mathrm{sat}} \to 1$时，被吸附的蒸气会变成液体）。因此，在这种情况下，第三项约束（即单调非递减）不再对所有压力都成立（见图 8）。尽管如此，我们发现，无论是否启用约束，PySR 为该数据集生成的许多最准确的表达式都满足第三项约束（未启用约束时为 78.65%，启用时为 81.29%），这与真实值理论相悖。此外，与未启用约束相比，启用约束时 PySR 满足前两项约束的频率更低。对此的一种可能解释是，从 PySR 算法的角度来看，该数据集本身更容易用满足单调非递减的表达式拟合

的表达式拟合。总体而言，虽然 PySR 能为 BET数据集找到准确的表达式，但即使启用了约束，它仍无法找到同时满足这些约束的表达式。

相比之下，BSR 并未生成许多单调非递减的表达式，而加入约束对搜索产生了显著影响。具体而言，无论启用还是禁用第二个约束，该约束通过的比例都约为 92%；启用第一个约束后，通过该约束的比例则从 16% 大幅提高到 85%。这使得大量模型满足 BET 所要求的约束，但其中没有一个重新发现真实值。相反，在两种情况下，两种算法都找到了许多准确度接近真实值（或优于真实值）的表达式，但所绘制的等温线没有一条看起来相似。分压力为 1 处的渐近线没有被任何准确度相当的表达式复现，数据集中央真实值的轻微曲线也同样缺失。综合来看，这些结果似乎表明，这些约束虽然在热力学上是正确的，却没有提供足够的信息（甚至提供了相互矛盾的信息）来重新发现 BET 真实值表达式。

数据集	启用的约束	BSR C1	BSR C2	BSR C3	PySR C1	PySR C2	PySR C3
氮	假	37%	67%	0.46%	2.2%	33%	46%
氮	True	72%	73%	19%	10%	41%	59%
甲烷	假	33%	51%	0.51%	4.1%	48%	61%
甲烷	True	59%	59%	2.5%	5.7%	54%	62%
异丁烷	假	24%	46%	0.45%	3.4%	65%	68%
异丁烷	True	36%	36%	1.3%	5.5%	56%	58%
BET	否	16%	92%	0.09%	6.2%	35%	79%
BET	True	85%	92%	2.4%	4.7%	30%	81%

表 2：生成的表达式中通过三个约束的各项百分比。结果涵盖两种符号回归方法、所有数据集，以及启用和禁用约束的情况。

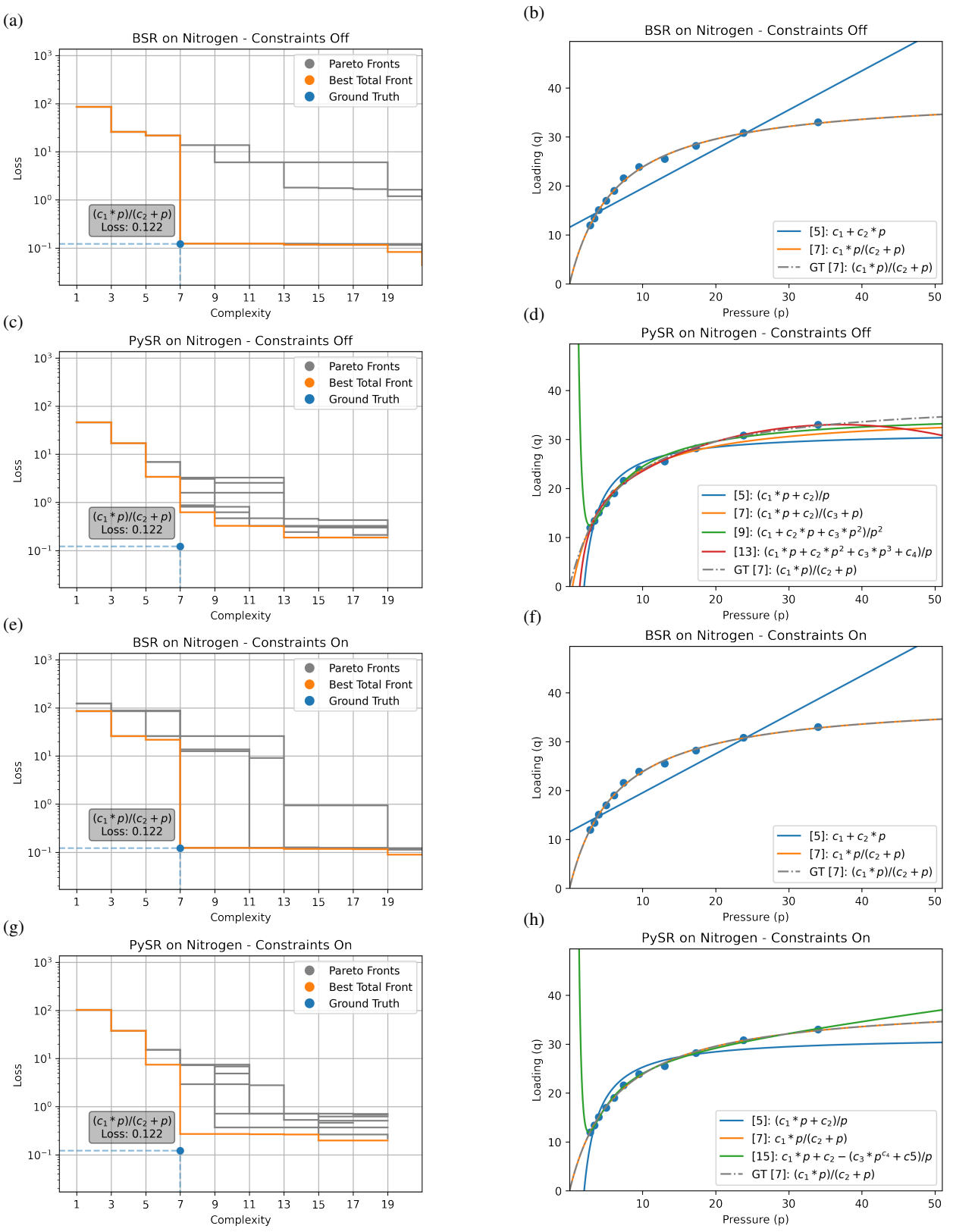


图5：BSR 和 PySR 在氮数据集上的表现。左列展示了 8 次运行中的综合帕累托前沿，右列展示了在这些帕累托前沿的定义性拐角处发现的有趣等温线。顶部四个子图禁用了约束，底部四个子图启用了约束。各行交替展示 BSR 和 PySR。

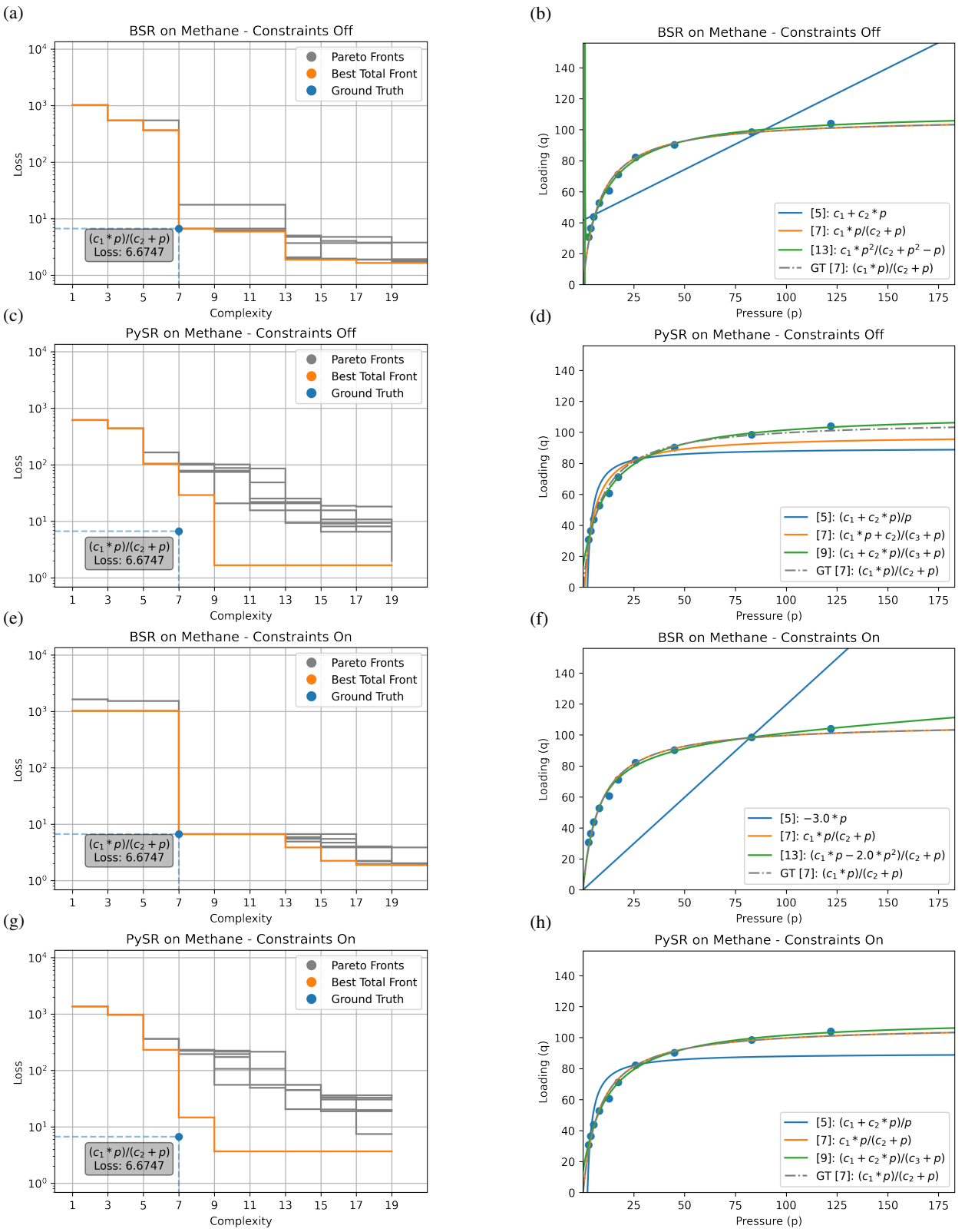


图 6：BSR 和 PySR 在甲烷数据集上的表现。左列展示了 8 次运行中的合并帕累托前沿，右列展示了在这些帕累托前沿的定义性拐角处发现的有趣等温线。上方四个子图禁用了约束，下方四个子图启用了约束。各行交替展示 BSR 和 PySR。

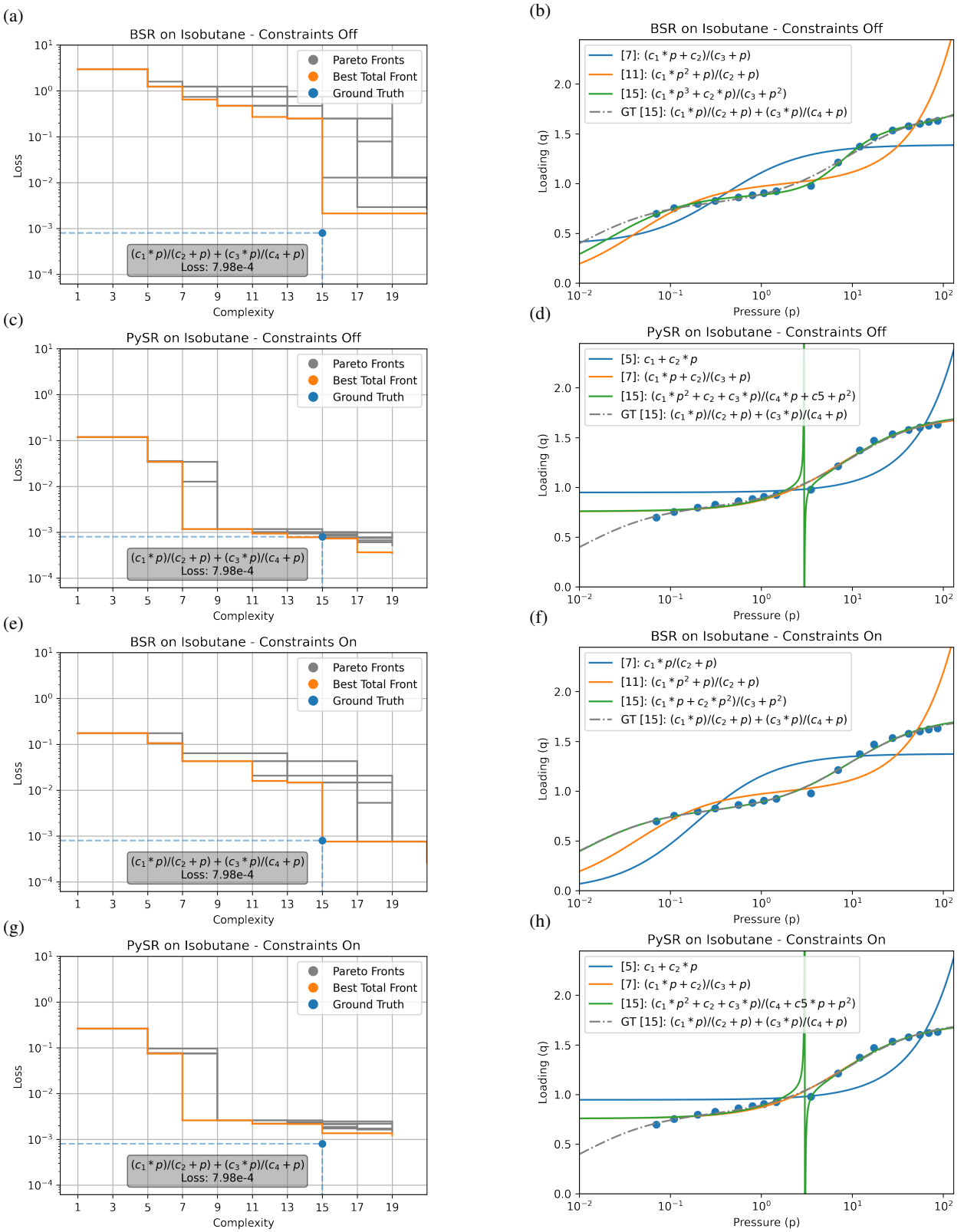


图 7：BSR 和 PySR 在异丁烷数据集上的表现。左列显示 8 次运行的组合帕累托前沿，右列显示在这些帕累托前沿的定义角点处找到的有趣等温线。顶部四个子图禁用约束，底部四个子图启用约束。各行交替显示 BSR 和 PySR。

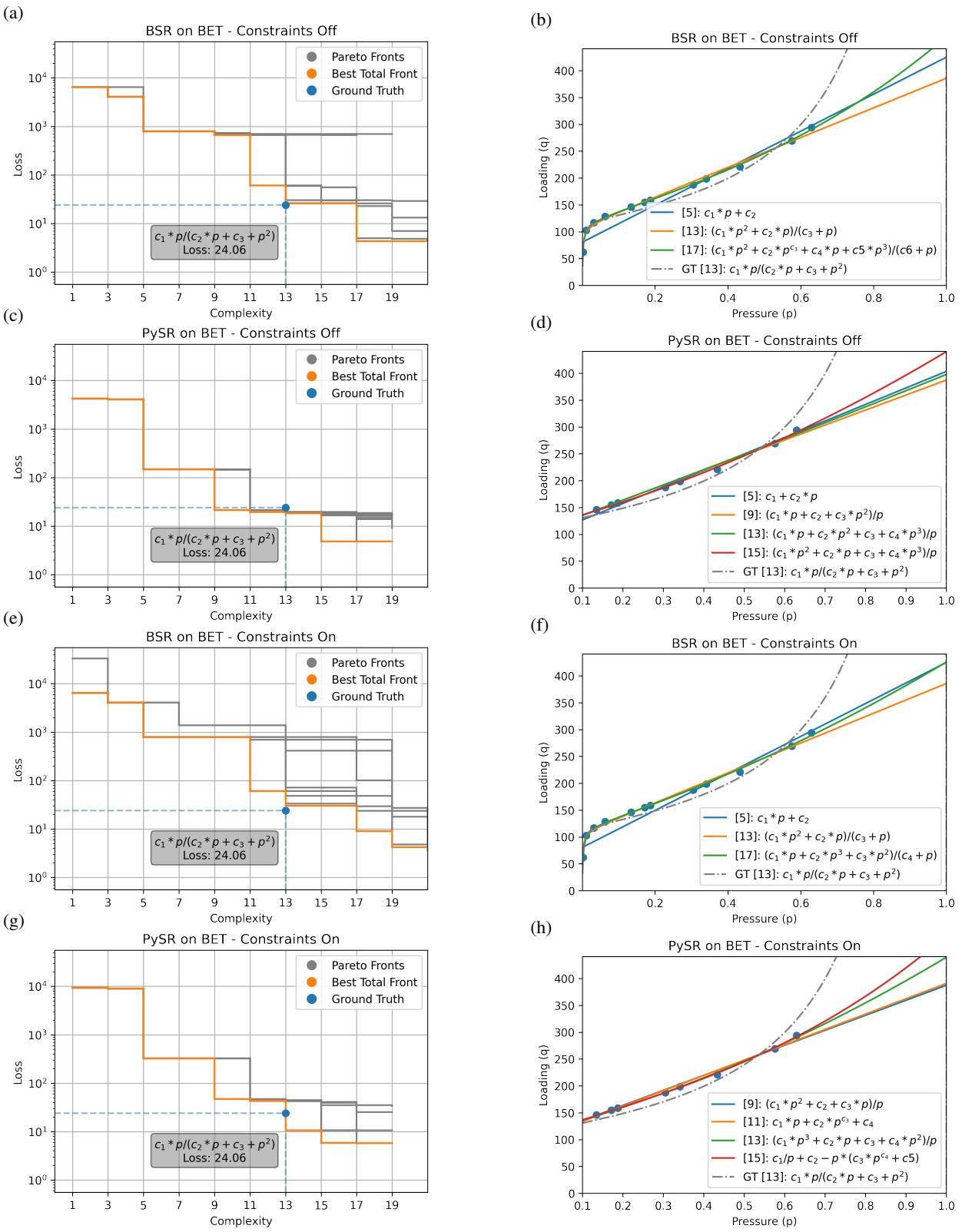


图 8：BET数据集上的 BSR 和 PySR。左列展示了 8 次运行得到的合并帕累托前沿，右列展示了这些帕累托前沿定义性角点处发现的有趣等温线。上方四个子图中禁用了约束，下方四个子图中启用了约束。各行交替展示 BSR 和 PySR。

4 讨论

4.1 有效性

本研究强调，有时通过方程空间进行随机搜索，能够找到优于真实值表达式的方程——在我们的案例中，这些方程不仅达到了与真实值表达式相当的准确度，而且复杂度更低（表达式长度更短）。这一点在异丁烷的案例中尤为明显（图 7）；无论是否启用约束，PySR 都会找到表达式 $\frac{c_1 p + c_2}{c_3 + p}$。该表达式能够很好地拟合数据，但在趋近于 0 时会偏离真实值。然而，许多此类表达式虽然与数据一致，却不符合热力学规律，因为它们违反了我们所检验的约束。我们已经证明，在搜索过程中纳入这些约束，可以引导表达式群体或分布趋向于符合热力学规律的表达式，并有助于识别真实值表达式。有时，这会得到更加一致的方程；但有时，它并不会改善搜索结果。

4.2 约束的例外情况

虽然本文提出的三个约束确实源自更广泛的热力学理论，但本文中的等温线模型并非都满足所有约束。具体而言，BET 等温线不满足第三个约束，因为当 p/p_0 接近 1 时，它会达到渐近线。虽然这确实违背了单调性约束，但考虑到 p_0 所代表的含义（吸附质变为液体且相互作用发生根本变化时的压力），这种情况似乎也合情合理。这引出了一个问题：应纳入哪些约束，理由是什么？决定检验表达式是否满足第三个约束，就必然会排除 BET模型，因为尽管理论上有依据，它仅适用于范围 $(0, p_0)$ 内。此外，所用数据并未超出该范围，因此通过第三个约束检验的表达式，在相关方面可能不会有太大差异。我们建议仔细考量希望检验哪些约束，以及实际应在什么范围内检验。引入不正确的约束可能会使搜索受到我们的偏见影响，并阻止算法发现假设之外的现象。

4.3 计算复杂度/运行时间

如图 10 所示，考虑约束会使运行时间增加一个数量级；即使我们已将计算机代数系统仔细集成到符号回归算法中以减少开销，并利用内存避免对先前访问过的表达式进行重复检查，情况仍然如此。平均而言，以数值方式检查模型比以符号方式处理模型快得多（尤其是对于较大的表达式）——检查每一个新表达式的代价都相当高。如果要将约束作为搜索不可或缺的一部分加以考虑，这种做法就不可避免。若需要更经济的解决方案，可以在不考虑约束的情况下进行搜索，然后再对约束进行检查。事实上，这种方法甚至能够采用更复杂的背景知识处理方法，例如使用自动定理证明器，将结果与包含多个复杂前提的背景理论进行比较 [15]。

4.4 关于复杂度、简化和规范形式的挑战 m

在本研究中，简化对于确定生成的表达式是否与真实值匹配，以及为生成的表达式赋予适当的复杂度而言是必要的。我们扩展了 SymPy 的“simplify”函数，将生成的大量有理数表达式缩短为“规范形式”（详见补充信息）。尽管 BSR 等方法会在运行时进行简化，但 PySR 不会这样做，因为这会增加每个表达式所需的计算量。为 PySR 生成的表达式构造“规范形式”有时会增加复杂度，有时则会降低复杂度。有些表达式会以复杂的表达式树形式生成，其复杂度远高于自身的规范形式（图 9）。

简化是这项工作的一个关键挑战，因为复杂度在符号回归中发挥着重要作用。毕竟，在搜索过程中，需要根据准确度和复杂度比较模型，以便作出决策。单个模型可能会因为表达方式不同而得到不同的评分／似然，从而不仅影响其排名，也会影响搜索中的后续步骤。理想情况下，每个模型都应始终以最简形式表示，但在某些情况下，这在计算上不可处理 [37]。因此，根据行为（符号约束）比较函数或许更合适，因为极限行为不受表达式众多表示方式的影响。

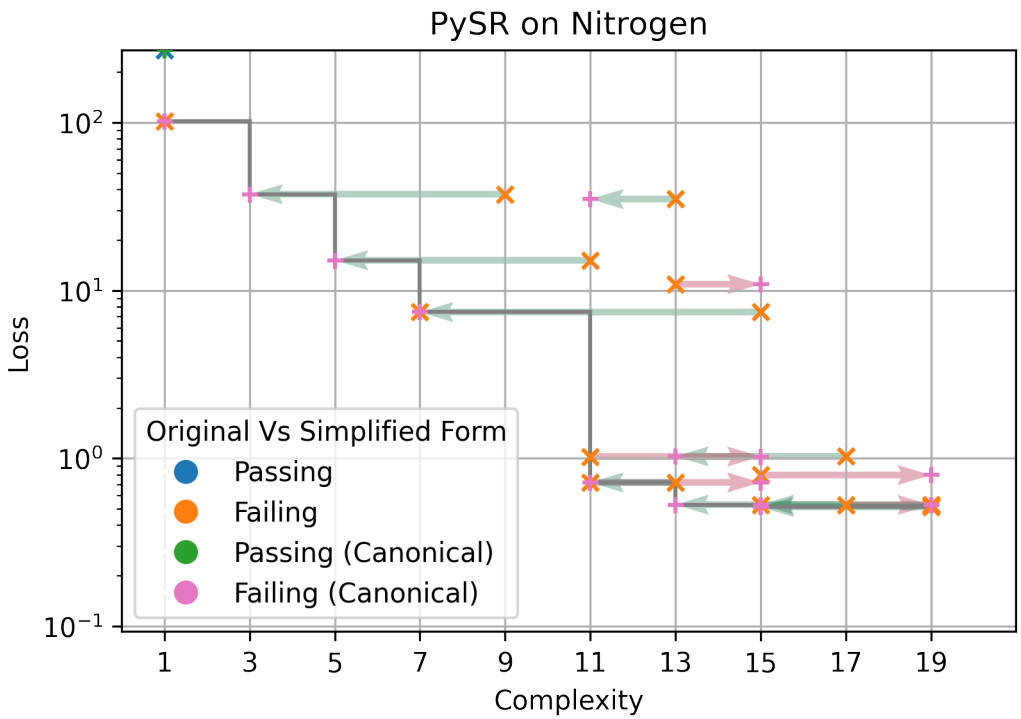


图9：规范形式检查函数对帕累托前沿的影响，展示了 PySR 在氮数据集上的结果。只有通过所有已检查约束的点才会在此图中标记为“通过”。

4.5 通过归纳偏置减少欠规格化

Google 的机器学习研究人员最近强调了欠规格化在机器学习流程中的作用 [38]。他们指出，应对欠规格化的一种方法是采用可靠的归纳偏置，从效果相近的模型中进行选择；如果选择得当，这些约束对准确度的负面影响应该很小。在这项工作中，我们发现，有些表达式在准确度和复杂度方面大致相当，却具有不同的函数形式，因此在数据范围之外表现出不同的行为——这与 [38] 中讨论的特征相似。我们发现，加入热力学约束有助于改进对优良表达式的搜索，但这不一定会像归纳偏置那样限制假设空间；我们无法使用硬约束进行有效搜索，因此假设空间仍包含与约束不一致的表达式。相反，我们可以在搜索完成后缩小假设空间：利用背景知识剔除准确但不一致的表达式，从而改善欠规格化问题。尽管如此，对于行为相当复杂的数据集，仍存在多个彼此不同、符合热力学约束且准确度和复杂度相近的表达式。由此处考虑的有限数量运算符所定义的方程空间，即使对于一维数据集，也实在太过广阔！

5 结论

在这项工作中，我们将计算机代数系统与两种符号回归算法结合，以检查生成的表达式是否与背景知识一致。我们发现，加入适当的数学约束可以提高搜索效率，也可能彻底中断搜索，具体取决于数据集和实现细节。尽管计算成本增加了一个数量级，但将符号回归与计算机代数系统紧密集成，是在搜索过程中检查每个生成表达式是否满足约束的一种实用方法。

我们已经表明，考虑约束有助于从实验数据中重新发现真实等温线模型，包括 Langmuir 和双位 Langmuir 等温线（尽管双位 Langmuir 等温线未在帕累托前沿上被识别出来，但它存在于生成的模型中）。相比之下，BET 等温线未被重新发现；系统反而生成了更准确且更简洁的模型，因此遗漏了最具意义的模型（BET）。我们发现，与通过修改传统遗传算法中的适应度函数相比，贝叶斯符号回归是一个更有效且更直观的平台，适用于

在贝叶斯先验中纳入符号约束；使用 BSR 后，生成的表达式群体对这些约束的适应性更强。最后，尽管背景知识可以筛除准确但不一致的解，但在我们的语境中，符号回归流程仍未得到充分限定，能够生成多个性能相近且对约束遵循程度相似的不同解。

6 致谢

感谢 Marta Sales-Pardo 和 Roger Guimerà 就贝叶斯机器科学家展开讨论，并感谢 Miles Cranmer 对 PySR 提供帮助。本材料得到美国国家科学基金会第 #2138938 号拨款以及马里兰大学巴尔的摩县分校的启动资金支持。

7 补充信息

修改后的 PySR 版本及用于运行该版本的代码均已在 GitHub 上提供。PySR 代码于 2020 年 6 月 6 日从原始代码仓库分叉而来，现可在 https://github.com/CharFox1/SymbolicRegression.jl 获取。用于运行 PySR、解析其输出并绘制结果的代码可在 https://github.com/ATOMSLab/pySR_吸附 获取。本文所使用的修改版 BMS 代码可在 https://github.com/ATOMSLab/BayesianSymbolicRegression 获取。

补充信息包括：1）对本文所考虑的吸附模型的进一步描述；2）进一步讨论我们在 PySR 和 BMS 代码中实施的更改，以实现热力学约束检查，其中包括新算法的伪代码；3）描述我们收集和分析所生成表达式的流程；4）进一步讨论在自动化流程中识别“有趣”表达式的细微问题，以及用于简化和模式匹配的算法；5）常数拟合的详细信息；6）比较算法在不同数据集上运行时间的实验；以及 7）在 UMBC 超级计算机上进行的测试环境的详细信息。

参考文献

[1] John R. Koza。《遗传编程》，1992 年。[2] Felipe Oviedo、Juan Lavista Ferres、Tonio Buonassisi 和 Keith T Butler。面向材料科学与化学的可解释和可说明机器学习。《Accounts of Materials Research》，3(6)：597–607，2022 年。[3] Xiaoting Zhong、Brian Gallagher、Shusen Liu、Bhavya Kailkhura、Anna Hiszpanski 和 T Han。材料科学中的可解释机器学习。《npj Computational Materials》，8(1)：1–19，2022 年。[4] Jacques A Esterhuizen、Bryan R Goldsmith 和 Suljo Linic。用于异相催化知识生成的可解释机器学习。《Nature Catalysis》，5(3)：175–184，2022 年。[5] Arthur Kordon、Flor Castillo、Guido Smits 和 Mark Kotanchek。遗传编程在工业中的应用问题。载于《Genetic Programming Theory and Practice III》，第 241–258 页。Springer，2006 年。[6] Dragan A. Savic、Godfrey A. Walters 和 James W. Davidson。遗传编程方法在降雨–径流建模中的应用。《Water Resources Management》，13(3)：219–231，1999 年 6 月。[7] Michael Schmidt 和 Hod Lipson。从实验数据中提炼自由形式的自然定律。《Science》，324(5923)：81–85，2009 年 4 月。出版方：美国科学促进会。[8] Alberto Hernandez、Adarsh Balasubramanian、Fenglin Yuan、Simon Mason 和 Tim Mueller。通过符号回归快速、准确且可迁移的多体原子间势，2019 年 8 月。arXiv:1904.01095[cond-mat，physics:physics]。[9]Mehrad Ansari、Heta A. Gandhi、David G. Foster 和 Andrew D. White。用于学习输运方程的迭代符号回归。《AIChE Journal》，68(6)：e17695，2022 年。_电子印本：https://onlinelibrary.wiley.com/doi/pdf/10.1002/aic.17695。[10] Miles Cranmer、Alvaro Sanchez Gonzalez、Peter Battaglia、Rui Xu、Kyle Cranmer、David Spergel 和 Shirley Ho。利用归纳偏置从深度学习中发现符号模型。载于《Advances in Neural Information Processing Systems》，第 33 卷，第 17429–17442 页。Curran Associates, Inc.，2020 年。[11] Runhai Ouyang、Stefano Curtarolo、Emre Ahmetcik、Matthias Scheffler 和 Luca M. Ghiringhelli。SISSO：一种压缩感知方法，用于从海量候选描述符中识别最佳低维描述符。《Physical Review Materials》，2(8)：083802，2018 年 8 月。出版方：美国物理学会。[12] Arijit Chakraborty、Abhishek Sivaram 和 Venkat Venkatasubramanian。AI-DARWIN：一种基于第一性原理、利用机器学习的模型发现引擎。《Computers & Chemical Engineering》，154：107470，2021 年 11 月。[13] Marissa R. Engle 和 Nikolaos V. Sahinidis。利用导数信息进行确定性符号回归：状态方程的一般方法与应用。《AIChE Journal》，68(6)：e17457，2022 年。_电子印本：https://onlinelibrary.wiley.com/doi/pdf/10.1002/aic.17457。[14] Roger Guimerà、Ignasi Reichardt、Antoni Aguilar-Mogas、Francesco A. Massucci、Manuel Miranda、Jordi Pallarès 和 Marta Sales-Pardo。贝叶斯科学家：协助解决具有挑战性的科学问题。《Science Advances》，2020 年 1 月。出版方：美国科学促进会。[15] Cristina Cornelio、Sanjeeb Dash、Vernon Austel、Tyler Josephson、Joao Goncalves、Kenneth Clarkson、Nimrod Megiddo、Bachir El Khadir 和 Lior Horesh。AI Descartes：结合数据与理论，实现可推导的科学发现，2021 年 10 月。arXiv:2109.01634 [cs]。[16] Dhananjay Ashok、Joseph Scott、Sebastian J. Wetzel、Maysum Panju 和 Vijay Ganesh。逻辑引导的遗传算法。《Proceedings of the AAAI Conference on Artificial Intelligence》，35(18)：15753–15754，2021 年 5 月。编号：18。[17] Gabriel Kronberger、Fabricio Olivetti de França、Bogdan Burlacu、Christian Haider 和 Michael Kommenda。形状约束符号回归：利用先验知识改进外推。《Evolutionary Computation》，30(1)：75–98，2022 年 3 月。arXiv:2103.15624 [cs，stat]。[18] C. Haider、F.O. De Franca、B. Burlacu 和 G. Kronberger。用于符号回归的形状约束多目标遗传编程。《Applied Soft Computing》，132：109855，2023 年 1 月。[19] R. Ben-Mansour、M. A. Habib、O. E. Bamidele、M. Basha、N. A. A. Qasem、A. Peedikakkal、T. Laoui 和 M. Ali。物理吸附捕集碳：材料、实验研究以及数值建模与模拟综述。《Applied Energy》，161：225–255，2016 年 1 月。[20] James A. Ritter 和 Armin D. Ebner。用于化学和石化工业制氢的先进吸附与膜分离过程。《Separation Science and Technology》，42(6)：1123–1193，2007 年 4 月。出版方：Taylor & Francis_电子印本：https://doi.org/10.1080/01496390701242194。

[21] M. H. Stenzel。通过活性炭吸附去除有机物。*Chemical Engineering Progress; (United States)*，89:4，1993年4月。[22] Douglas Ruthven。吸附与吸附过程原理 | Wiley，1984年6月。[23] G Limousin、J-P Gaudet、L Charlet、Stephanie Szenknect、V Barthes 和 M Krimissa。吸附等温线：关于物理基础、建模与测量的综述。*Applied geochemistry*，22(2):249–275，2007年。[24] Keng Yuen Foo 和 Bassim H Hameed。对吸附等温线系统建模的深入认识。*Chemicalengineeringjournal*，156(1):2–10，2010年。[25] Nimibofa Ayawei、Augustus Newton Ebelegi 和 Donbebe Wankasi。吸附等温线的建模与解释。*Journalofchemistry*，2017，2017年。[26] Jianlong Wang 和 Xuan Guo。吸附等温线模型：分类、物理意义、应用与求解方法。*Chemosphere*，258:127279，2020年11月。[27] Herbert Freundlich。溶液中的吸附。教授资格论文，通过该论文……邀请参加即将举行的试讲“毛细管化学与生理学”的 *Herbert Freundlich* 博士。W. Engelmann，1906年。[28] Irving Langmuir。玻璃、云母和平面铂表面的气体吸附。*Journal oftheAmericanChemical Society*，40(9):1361–1403，1918年9月。出版商：美国化学学会。[29] Stephen Brunauer、P. H. Emmett 和 Edward Teller。多分子层中的气体吸附。*Journalofthe American Chemical Society*，60(2):309–319，1938年2月。出版商：美国化学学会。[30] Robert Sips。关于催化剂表面的结构。*The journal of chemical physics*，16(5):490–495，1948年。[31] Orhan Talu 和 Alan L. Myers。气体吸附的严格热力学处理。*AIChE Journal*，34(11):1887–1893，1988年。_电子版：https://onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690341114。[32] J Toth。应用错误气固吸附等温线方程的若干后果。*Journal ofColloid and InterfaceScience*，185(1):228–235，1997年1月。[33] Miles Cranmer、Dhananjay Ashok 和 Johann Brehmer。MilesCranmer/PySR：v0.6.0，2021年6月。[34] Z. Konfrst。并行遗传算法：进展、计算趋势、应用与发展前景。载于第*18*届国际并行与分布式处理研讨会，*2004*年。会议论文集。，第162页起，2004年4月。ˇ[35] Aaron Meurer、Christopher P. Smith、Mateusz Paprocki、Ondˇrej Certík、Sergey B. Kirpichev、Matthew Rocklin、AMiT Kumar、Sergiu Ivanov、Jason K. Moore、Sartaj Singh、Thilina Rathnayake、Sean Vig、Brian E. Granger、Richard P. Muller、Francesco Bonazzi、Harsh Gupta、Shivam Vats、Fredrik Johansson、Fabian Pedregosa、Matthew J. Curry、Andy R. Terrel、Štˇepán Rouˇcka、Ashutosh Saboo、Isuru Fernando、Sumith Kulal、Robert Cimrman 和 Anthony Scopatz。Sympy：Python 中的符号计算。*PeerJComputer Science*，3:e103，2017年1月。[36] T. J. H. Vlugt、W. Zhu、F. Kapteijn、J. A. Moulijn、B. Smit 和 R. Krishna。沸石 Silicalite-1 中直链和支链烷烃的吸附。*Journal of theAmerican Chemical Society*，120(22):5599–5600，1998年6月。出版商：美国化学学会。[37] Dan Richardson 和 John Fitch。初等函数与常数的恒等性问题。载于国际符号与代数计算研讨会论文集，ISSAC ’94，第285–290页，纽约，美国，1994年8月。计算机协会。[38] Alexander D’Amour、Katherine Heller、Dan Moldovan、Ben Adlam、Babak Alipanahi、Alex Beutel、Christina Chen、Jonathan Deaton、Jacob Eisenstein、Matthew D. Hoffman、Farhad Hormozdiari、Neil Houlsby、Shaobo Hou、Ghassen Jerfel、Alan Karthikesalingam、Mario Lucic、Yian Ma、Cory McLean、Diana Mincu、Akinori Mitani、Andrea Montanari、Zachary Nado、Vivek Natarajan、Christopher Nielson、Thomas F. Osborne、Rajiv Raman、Kim Ramasamy、Rory Sayres、Jessica Schrouff、Martin Seneviratne、Shannon Sequeira、Harini Suresh、Victor Veitch、Max Vladymyrov、Xuezhi Wang、Kellie Webster、Steve Yadlowsky、Taedong Yun、Xiaohua Zhai 和 D. Sculley。欠规格化给现代机器学习的可信度带来挑战，2020年11月。arXiv:2011。03395 [cs，stat]。[39] T. J. H. Vlugt、R. Krishna 和 B. Smit。硅沸石中直链和支链烷烃及其混合物的吸附等温线分子模拟。*TheJournalof PhysicalChemistry B*，103(7):1102–1118，1999年2月。出版商：美国化学学会。[40] Jeff Reback、jbrockmendel、Wes McKinney、Joris Van den Bossche、Matthew Roeschke、Tom Augspurger、Simon Hawkins、Phillip Cloud、gfyoung、Sinhrks、Patrick Hoefler、Adam Klein、Terji Petersen、Jeff Tratner、Chang She、William Ayd、Shahar Naveh、J. H. M. Darbyshire、Richard Shadrach、Marc Garcia、Jeremy Schendel、Andy Hayden、Daniel Saxton、Marco Edward Gorelli、Fangchen Li、Torsten Wörtwein、Matthew Zeitlin、Vytautas Jancauskas、Ali McMaster 和 Thomas Li。pandas-dev/pandas：Pandas 1.4.3，2022年6月。

[41] Patrick Kofod Mogensen、John Myles White、Asbjørn Nilsen Riseth、Tim Holy、Miles Lubin、Christof Stocker、Andreas Noack、Antoine Levitt、Christoph Ortner、Blake Johnson、Dahua Lin、Kristoffer Carlsson、Yichao Yu、Christopher Rackauckas、Josua Grawitter、Alex Williams、Ben Kuhn、Benoît Legat、Jeffrey Regier、cossio、Ron Rock、Thomas R. Covert、Benoit Pasquier、Takafumi Arakaki、Alexey Stukalov、Andrew Clausen、Arno Strouwen 和 Benjamin Deonovic。JuliaNLSolvers/Optim.jl：v1.7.0，2022年5月。

[42] Pauli Virtanen、Ralf Gommers、Travis E. Oliphant、Matt Haberland、Tyler Reddy、David Cournapeau、Evgeni Burovski、Pearu Peterson、Warren Weckesser、Jonathan Bright、Stéfan J. van der Walt、Matthew Brett、Joshua Wilson、K. Jarrod Millman、Nikolay Mayorov、Andrew R. J. Nelson、Eric Jones、Robert Kern、Eric Larson、C. J. Carey、 ̇Ilhan Polat、Yu Feng、Eric W. Moore、Jake VanderPlas、Denis Laxalde、Josef Perktold、Robert Cimrman、Ian Henriksen、E. A. Quintero、Charles R. Harris、Anne M. Archibald、Antônio H. Ribeiro、Fabian Pedregosa 和 Paul van Mulbregt。《SciPy 1.0：Python 科学计算的基础算法》。自然方法，17(3)：261–272，2020 年 3 月。编号：3。出版商：自然出版集团。

8 补充信息

8.1 Langmuir

Lang Muir 等温线模型最初于 1918 年提出，至今仍广泛使用 [28]。

$$q = \frac{q_m K_L p}{1 + K_L p} \quad \text{or} \quad \theta = \frac{K_L p}{1 + K_L p} \tag{10}$$

原始 Langmuir 等温线将吸附在表面的体积（q_e）、吸附强度（K_L）、气体压力 p 和最大吸附容量（q_m）[26] [28]联系起来。该等温线通常可使用分数吸附 θ_A以更简洁的形式表示，表示被吸附分子占据的表面分数。在实践中，q_m并非已知先验，因此我们的目标是重新发现 q_e的表达式。然而，符号回归从未找到这一表达式，因为可以将其写成一个长度更短的等价表达式（7 对 11），即

$$q = \frac{q_m K_L P}{1 + K_L P} \rightarrow q = \frac{c1 * p}{(c2 + p)} \tag{11}$$

Langmuir 模型的基本前提是，将吸附剂（表面原子或分子正在其上发生吸附）表示为一个简单表面，其上有一定数量的空闲位点和已占据位点，吸附质可以附着于这些位点。该模型不考虑粗糙或多孔表面，也不考虑可能占据多个“位点”的分子，或吸附后彼此发生相互作用的分子，尤其不考虑分子的任何层叠。Wang 等人将 Langmuir 模型归类为化学吸附模型，因为该模型只关注单层吸附，在此过程中吸附剂与吸附质之间会形成化学键 [26]。

8.2 双位点

Langmuir 模型 可以扩展用于描述更复杂的材料。具体而言，如果一个表面具有两种类型的 f
吸附位点（具有 不同的性质），则可以添加另一项，从而得到“双位点 Langmuir”模型：

$$q = \frac{q_a K_A p}{(1 + K_A p)} + \frac{q_b K_B p}{(1 + K_B p)} \rightarrow q = \frac{c1 * p}{(c2 + p)} + \frac{c3 * p}{(c4 + p)} \tag{12}$$

在这种情况下，由于存在两个不同的 Langmuir 项，因此可吸附的最大体积（q_a 和 q_b）以及“朗缪尔常数”（K_A 和 K_B）也各有两个不同的项。该模型被视为异丁烷数据集的真实值，因为吸附剂材料 MFI沸石对异丁烷具有两个不同的吸附位点 [39]。

8.3 BET

BET模型将 Langmuir 模型扩展为考虑多层吸附 [29]。除第一层外，范德华力还会吸引吸附质靠近表面，并使其与已吸附的分子相互吸引。因此，Wang 等人将 BET 等温线归类为物理模型，而非化学模型 [26]。主要的 BET 模型假设可能存在无限多层；也可以推导出最大层数有限的其他形式 [29]。当最大层数为一层时，它简化为 Langmuir 模型（仅对一层进行建模）；当有 n 层时，模型则会变得复杂得多（而且不太可能通过符号回归找到）。幸运的是，$n = \infty$ 给出的形式简洁，可以简化为：

$$q = \frac{v_m * c * (p/p_0)}{(1 - p/p_0) * (1 + (c - 1)p/p_0)} \tag{13}$$

其中，p_0 为饱和蒸气压，v_m 和 c 为描述系统中单层吸附和相互作用能的常数。数据可以作为 p 或 p/p_0提供给符号回归；采用第二种方式相当于进行先验 特征选择，即假定我们知道压力应归一化至 p_0。由于我们并未作此假定，p_0 便成为由符号回归拟合的第三个常数。符号回归通常会将其拟合为错误的值，因为它不受约束，无法确保 p_0 与任何物理量相匹配，并且会优先拟合给定数据。

8.4 热力学约束函数

我们使用 Python 库 SymPy [35]开发了三个约束检查函数（分别对应第1.4节前文介绍的三个热力学约束）。每个函数都会根据相应约束是否满足，返回布尔值 TRUE 或 FALSE。尽管这些函数有助于检查运行后生成的表达式，但若在运行过程中仅因某个表达式未通过一项或多项约束就将其舍弃，可能会严重限制搜索潜力，因为这会截断通往更优表达式的中间步骤，而这些表达式也可能满足约束。因此，每个函数都配有相应的权重，使其能够作为“软约束”发挥作用。具体而言，若表达式满足约束，其评分不会受到影响；若不满足，则评分会被乘以一个系数而降低。这一方法（在 PySR 中的实现）详见算法1。算法2介绍了我们如何使用 SymPy 检验函数是否单调非递减。

算法1改进遗传算法

输入： $tree,\ dataset,\ options$
输出： $score,\ loss$
  函数 SCOREFUNC($tree, dataset, options$)
    $loss \leftarrow$ 计算损失($tree, dataset, options$) ▷ 传统误差计算
    $penalties \leftarrow options.penalties$
    如果 $penalties$不为空 **则** ▷ 如果可能，跳过缓慢计算
      $expr,\ var \leftarrow$ 解析树($tree$)
      对于 $p \in penalties$ 且 $tf \in thermoFunctions$ **执行**
        如果 $tf(expr, var) =$ 假 **则**
          $loss \leftarrow loss * p$
        结束如果
      结束循环
    结束如果
    $score \leftarrow$ 评分函数($loss, tree, options$) ▷ 计入复杂度
    返回 $score\ , loss$
  结束函数

算法2单调非递减检查

输入： $expr,\ var,\ start,\ stop$
输出： $passing$ （如果表达式是单调非递减的）
  函数 MONTONICNONDECREASING($expr,\ var,\ start,\ stop$)
    如果 $expr$ 为常数**则**
      返回True
    结束如果
    $turningPoints \leftarrow\ expr$的拐点（零点） ▷ 使用 SymPy 计算
    如果 $turningPoints$为空 **则**
      如果 $expr(stop)\ - expr(start)\ \geq 0$ **则** ▷ 测量斜率
        返回 真
      else
        返回 假
      结束如果
    结束 if
    $turningPoints \leftarrow [start, turningPoints, stop]$ ▷ 包含边界
    对于每一对连续的点 $p,\ q \in turningPoints$ **执行**
      如果 $expr(q) - expr(p)\ \leq 0$ **则** ▷ 测量零点之间的斜率
        返回 False
      结束 if
    结束 for
    返回 真
  结束 function

8.5 热力学约束的实现

为了确保 PySR 与 BMS 之间保持一致，我们使用了 Julia 库 PyCall。这样一来，便可以在 SymbolicRegression.jl 库（PySR 的后端）的 Julia 代码中使用与 BMS 中检查热力学约束相同的 Python 代码。除了需要在不同平台之间进行公平比较之外，PyCall 对本研究而言也在某种程度上是必不可少的。截至本文撰写时，还没有功能接近 SymPy 那样全面的 Julia 库（而且 SymPy 也存在一些严重的局限和难题）。我们尝试过的一种方法是使用 Julia 库 MATLAB.jl，从 Julia 内部调用 MATLAB 代码，方式与 PyCall 调用 Python 代码类似。虽然最终实现了这一点（MATLAB 确实拥有功能强大的符号数学工具箱），但由于两种语言存储数据的方式存在复杂差异，配置过程非常困难。在我们的测试中，MATLAB 并没有比 Python 快多少。

我们当前版本的 SR.jl 及其所依赖语言的结构有一个缺点：必须以分布式模式运行，而非多线程模式（这意味着需要创建多个进程，因此开销更大）。这是因为 PyCall 规定，同一时间只能有一个 Julia 实例尝试使用 Python。使用多个进程时，每个进程都会创建一个新的 PyCall Julia 包实例，从而避免内存访问冲突。

8.6 PySR 数据收集

尽管 PySR 会生成并处理许多表达式，但要获取所有成员及各个群体随时间变化的完整信息，事实证明并不容易。PySR 的目标是在每次迭代中生成一个帕累托前沿，展示它所发现的、在各个复杂度级别上最准确的表达式。这意味着群体中的绝大多数表达式都不会显示出来（尽管其中一些可能是重复项）。事实上，如果某个表达式在运行期间始终未能进入帕累托前沿，那么按照搜索的定义，它就没那么重要。然而，这使我们无法调查是否有与真实值一致的表达式被生成后又丢弃，因为它们没有进入帕累托前沿（原因可能是参数优化不足或简化所致）。

尽管如此，为了跟踪我们可以轻松访问的表达式（通常每次运行会有数百个帕累托前沿），我们使用 Python 解析完整的输出文本，并将其存储在 Pandas DataFrame [40] 中。从原始文本中收集的值包括表达式本身、评分、损失、复杂度、运行时间、迭代次数和运行编号。虽然这已经是相当大量的数据，但仍有一些有趣的属性尚未计算出来——具体而言，一类与表达式的简化形式有关，另一类与热力学约束有关。为了使进一步的解析易于管理，在生成每个表达式的简化形式后，只保留具有该形式的最准确表达式。这一步通常会使 DataFrame 中的行数减少约 1000 倍。每个表达式的简化形式都会分别使用原始（已优化/拟合的）常数和替换后的常数进行计算。简化形式的新复杂度可能小于原始复杂度，也可能不小于，因此两个属性都会保留。最后，会针对每个表达式检查可能用于引导搜索的热力学约束。

8.7 识别“有趣”的表达式

这项工作的一项重要内容，是识别符号回归算法生成的有趣或有意义表达式。满足约束的表达式比不满足约束的表达式更有趣，但最有意义的表达式是那些可推导出 [15]的表达式。在我们的案例中，这表现为能够将其化简为某个已知表达式的规范形式，例如 Langmuir 或 BET 吸附等温线的规范形式。遗憾的是，确定一个表达式是否等价于另一个表达式是一个非常困难、有时甚至不可判定的问题。事实上，尽管由于我们对所使用的运算符和常数的限制，Richardson 定理并不适用于本项工作，但该定理表明，如果包含某些运算符和超越数 π 以及 e，则可能无法证明一个表达式等价于另一个表达式 [37]。因此，我们通过自动寻找与规范形式的直接匹配，或手动检查满足所有相关约束且误差较低的表达式，来识别精确表达式；但这并不能保证我们找到所有重新发现的真实值表达式实例。

8.8 化简函数

我们起初倾向于将常数保留为符号，并将其视为变量（例如 c1、c2.....），但我们发现，对于即使规模适中的完全符号表达式，SymPy 也无法可靠地进行完全化简。因此，我们随后使用搜索得到的拟合参数进行替换，或者用一系列素数替换符号常数。我们首先使用 SymPy 的默认工具化简表达式，但该工具无法将某些表达式化简为最简形式。例如，如果所得表达式是有理数（仅含整数指数且不存在除零），就可以将其写成分数，并可能进一步化简。具体来说，如果分子和分母之间存在公因子（因为它们的次数都不小于 1），则可能可以消去首项常数，从而得到一个比 SymPy 默认工具所得结果更简单的表达式，提供一个

更简单的表达式。例如，

$$\frac{2x}{3x^2+4x+5} \to \frac{x}{(3/2)x^2+2x+5/2} \quad \text{or} \quad \frac{(2/3)x}{x^2+(4/3)x+5/3} \tag{14}$$

$$\frac{c_1 x}{c_2 x^2+c_3 x+c_4} \to \frac{x}{c_1 x^2+c_2 x+c_3} \quad \text{or} \quad \frac{c_1 x}{x^2+c_2 x+c_3} \tag{15}$$

在这种情况下，一个包含 4 个常数的表达式被简化为一个包含 3 个常数的表达式。

有些表达式会在多个位置使用同一个常数。例如，文献中的 BET 表达式及其简化形式都多次出现常数 p_0 或 c_2：

$$\frac{v_m * c * (p/p_0)}{(1-p/p_0) * (1+(c-1)p/p_0)} \to \frac{c_1 * p}{(p^2+c_2 * p+c_2)} \tag{16}$$

在我们的情形中，我们接受形式与真实值相同、但常数不唯一的解。表达式$c1*p/(p^2+c_2*p+c_3)$的复杂度相同（按我们的指标衡量），同时拟合数据的能力更强；如果我们要求$c_2=c_3$，就永远无法得到真实值。

算法 3化简函数

输入： $expr$, $vars$, $pars$
输出： can（表达式的规范形式）
函数 SIMPLIFY($expr$, $vars$, $pars$)
$expr \leftarrow$表达式
$vars \leftarrow$ 变量
$pars \leftarrow$ 参数
对于 p **中的每个** $pars$**，执行**
$p \leftarrow$ 素数 ▷ 将常数替换为唯一的素数
结束循环
使用 SymPy 进行简化
如果 $expr$ 是有理数**则**
$num, denom \leftarrow expr$ 作为分数 ▷ 使用 SymPy 计算
如果次数(num) $>$ 次数($denom$) **那么**
$factor \leftarrow num$ 的首项
else
$factor \leftarrow denom$ 的首项
结束条件判断
$expr \leftarrow \frac{num/factor}{denom/factor}$ ▷ 移除公因子
结束 if
返回 $expr$
结束函数

8.9 在 PySR 中拟合常数

默认情况下，PySR 使用 Nelder-Mead优化在生成表达式时拟合常数 [33] [41]。Nelder-Mead非常适合为任意生成的表达式优化参数，因为它不需要任何导数或梯度信息。该算法在 $n+1$ 个点处计算函数值，其中 n 是参数的数量。下一点的选择方式是：找出函数值最高的点，然后在其余点（构成一个单纯形）的另一侧寻找。该过程会迭代地将最差的点移至可能更优的位置，最终无论函数如何计算，都会逐渐趋向一个局部最优解。

这种方法的一个缺点是，它并不能保证（甚至也不一定期望）找到全局最小值。通常可以通过从随机位置进行多次初始化并选取最佳结果来解决这一问题。在 PySR 中，每个表达式默认尝试 8 次，每次尝试之间都会随机化参数。这通常已经足够，但偶尔也会出现这样的情况：由于常数拟合不佳，某个表达式本应更加准确，实际却并非如此。

为了拟合真实表达式并在后处理中检查常数，我们还使用 SciPy [42] 进行了 Nelder-Mead优化。我们在后处理中允许更多次迭代，以确保真实值和有趣的函数形式都得到充分优化。

8.10 运行时间

在考察热力学约束对符号回归的引导效果时，一个重要的考量因素是其对计算时间的影响。符号数学运算（在 SymPy 中进行）虽然算不上特别慢，但速度确实较慢，尤其是与 PySR 前端背后高效且经过优化的 Julia 代码相比。下图展示了不同数据集以及更重要的不同约束惩罚项下，迭代时间（即 PySR 打印出一个帕累托前沿与下一个帕累托前沿之间的时间）的差异。结果表明，不使用 SymPy 时运行速度很快，且不受数据集影响；而检查约束会使运行时间增加约一个数量级。

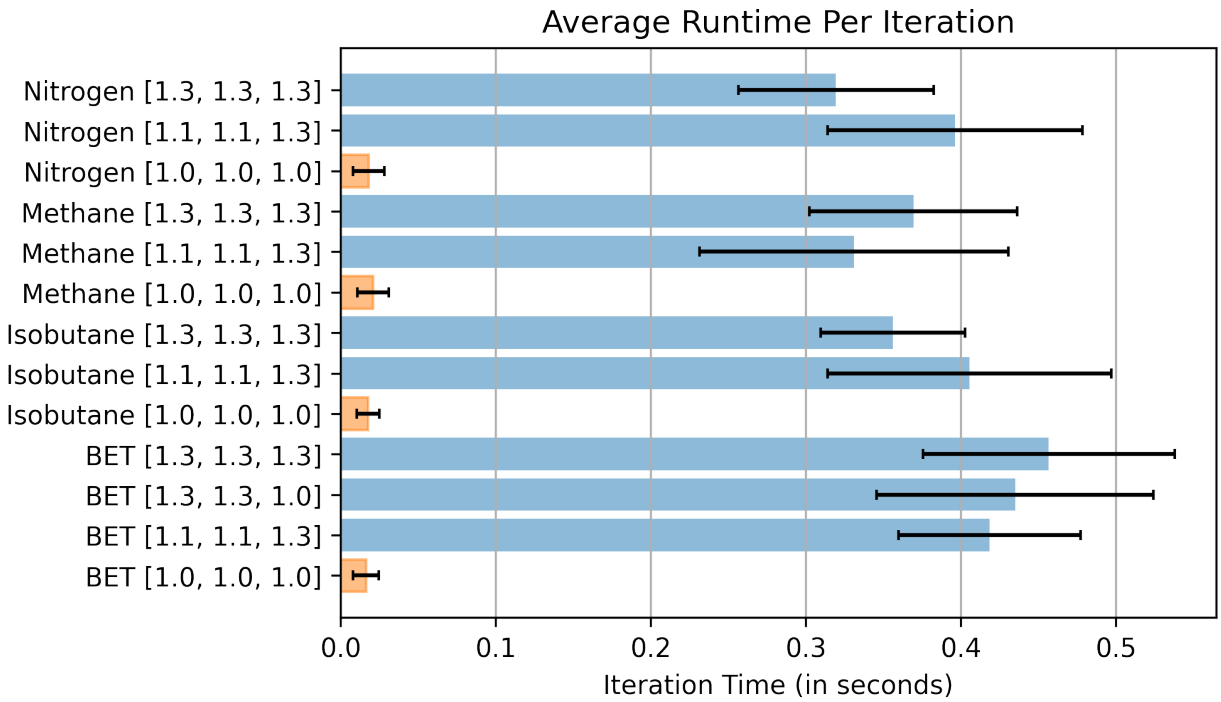


图 10：在所有数据集以及热力学约束惩罚项的各种组合下的平均运行时间。所有惩罚项均设置为 1.0 的运行以橙色突出显示。每个柱状条顶部的误差线表示标准差。

8.11 环境

测试在 UMBC 高性能计算设施（https://hpcf.umbc.edu/）的 batch 和 cpu2021 分区上进行。这使得在添加带有可变惩罚项的热力学约束后，可以进行所需的更长时间运行和更大规模的参数探索。

BMS 完全基于 Python，而 SymbolicRegression.jl（PySR 的后端）则完全基于 Julia。由于需要使用 SymPy，因此使用了 Julia 包 PyCall.jl，以便 Julia 能够运行 Python 代码和库（本研究完成时，Julia 中的符号数学库尚无法评估本研究所考虑的约束）。为了在 Julia 中并行使用 Python / SymPy，PySR 以分布式模式运行，因此比多线程模式需要更多开销。