arXiv:2611.28599v1 [q-bio.BM] 21 Nov 2026

FastCREST：GPU 加速 CREST

能量和梯度一致的构象搜索

Supantha Dey,[†, ‡, ¶] Nilusha L Kariyawasam,[†, ¶] 和 Ratul Chowdhury[*, †]

†爱荷华州立大学化学与生物工程系，美国爱荷华州艾姆斯，邮编 *50011*

‡爱荷华州立大学跨院系遗传学与基因组学项目，美国爱荷华州艾姆斯，邮编 *50011*

¶这些作者对本研究的贡献相同。

电子邮箱：ratul@iastate.edu

摘要

使用 GFN2-xTB 进行 CREST 构象搜索时，大型柔性分子需要数天时间：使用 CREST 3.0.2 对一个含 274 个原子的镓(III)-肽复合物进行一次搜索，在 32 个 CPU 线程上耗时 321.5 小时。FastCREST 保留 CREST 的算法，只改变搜索利用硬件的方式。其元动力学 (MTD) 和分子动力学 (MD) 轨迹采用经过调优的 OpenMP 设置，并使用一个图形处理器；该处理器负责求解自洽电荷 (SCC) 特征值问题并计算三体色散项。每次迭代约有 15,000 次几何优化作为 Slurm 数组作业在其他 CPU 节点上运行，而检查点使搜索能够在可抢占 GPU 上继续进行。在 A100 节点上，OpenMP 设置和循环改动使 MTD 步骤快了 2.4 倍，图形处理器又使其快了 2.3 倍；数组作业则使一批优化快了 39 倍。对于另外六种含 24 至 339 个原子的分子，FastCREST 的单步速度比传统 CREST 快 2.2 至 7.1 倍。一次完整的 FastCREST 搜索（同时取消了 CREST 3 的 200 次优化循环上限）

耗时 54.3 小时，其能量和梯度的计算速度是 CREST 3.0.2 搜索（使用不同硬件）的 8.1 倍，且每次计算所需的 CPU 小时更少。两个可选启用的设置（暖启动雅可比特征值求解器和 GPU驻留SCC 迭代）、LLVM 的 OpenMP 库，以及每条轨迹使用一个 CPU 插槽，使 MTD 步骤比传统 CREST 快 8.3 倍。能量与 CREST 3.0.2 的差异在 1.01×10^{-10} E_h 以内，梯度的差异在 8.2×10^{-11} E_h 玻尔$^{-1}$以内；受控分子动力学运行也通过了预先设定的等价性检验。大多数重复进行的 FastCREST 和传统 CREST 搜索最终得到相似的紧凑构象，能量差异未达到可检测水平。

1 引言

金属结合肽是回收关键矿产的候选材料，因为其序列可以通过工程化设计引入金属配位基团，从而提高对目标金属的亲和力以及相对于竞争性金属离子的选择性。[1–3] 我们设计镓结合肽，以从工业废物流中选择性回收镓。肽可以采取许多不同的构象，其主链和侧链的排列会影响哪些配位基团能够到达金属、结合位点的几何结构以及金属–肽复合物的相对稳定性。从一个起始结构进行几何优化只能找到一个局部极小值，无法捕捉这种多样性。因此，我们对每个复合物的构象进行采样，比较它们与镓的配位方式，并使用密度泛函理论对具有代表性的低能构象进行精修。[4]

采用 GFN2-xTB 紧束缚方法的 CREST 是生成构象系综的标准工具。[5–7] 其 iMTD-GC 工作流程（迭代元动力学与遗传交叉）[5,8] 运行由相对于早期结构的均方根偏差（RMSD）所偏置的元动力学（MTD）[9] 轨迹，优化数千个轨迹快照，并重复这一过程，直到不再出现更低能构象。对于数百个原子，在一个 CPU 核心上进行一次能量与梯度（E/G）评估需要几秒钟，但一次搜索需要数百万次评估。对于一个含 274 个原子的镓(III)-肽复合物，在隐式水中，使用 CREST 3.0.2 版本进行一次搜索，在 32 个 CPU 线程上耗时 321.5 小时（13.4 天），

这一成本限制了可筛选的肽候选结构数量。增加线程几乎没有帮助：即使采用第 2.2 节中的线程设置，使用 64 个线程相比 32 个线程，每个 MTD 时间步的速度也仅提高了 1.06 至 1.09 倍。

CREST 搜索会结合少量长轨迹和数千个相互独立的几何优化。长轨迹的速度取决于一次能量/梯度评估所需的时间，而 CREST 会同时利用单个节点的多个核心运行这两类任务。半经验方法和紧束缚方法的 GPU 实现[10–14]可以加速大型体系上的单次计算，但无法兼顾这种由短暂、顺序执行的轨迹步骤与大量小型独立作业组成的混合工作负载。

FastCREST 改变了搜索利用硬件的方式，同时保持能量和梯度不变（图 1）。轨迹使用 OpenMP 设置运行，使线程能够分担工作，并使用一块 GPU 执行特征值求解器和三体色散项。几何优化则作为 Slurm 数组作业在其他 CPU 节点上运行；检查点使搜索能够在共享集群的可抢占 GPU 上运行。除取消几何优化循环次数的固定上限（第 2.1 节）外，FastCREST 的结果与 CREST 的结果仅在浮点运算顺序上有所不同，差异小于自洽电荷（SCC）过程的收敛阈值。

2 方法

2.1 构象搜索与测试体系

在 CREST 的 iMTD-GC 工作流程（图 1a）中，每次迭代会同时运行六条 MTD 轨迹，并优化其中约 15,000 个快照：先采用粗略阈值，再采用常规阈值，随后使用 CREST 的系综工具 CREGEN 对结果排序。只要找到新的最低能量构象，迭代就会继续。接着，在 400 和 500 开尔文下进行分子动力学阶段，从最低能构象中采样；如果优化后的快照中包含更低能构象，就会启动新一轮 MTD 迭代。随后通过遗传交叉（GC）和最终优化结束搜索，并将能量处于最低能构象以上 5 千卡/摩尔范围内的构象保留在系综中（该范围由 `--quick` 设定）。

测试体系是一个含 15 个残基的肽（MLTREQIEELSKKLK），其中 Ga^{3+} 与五个

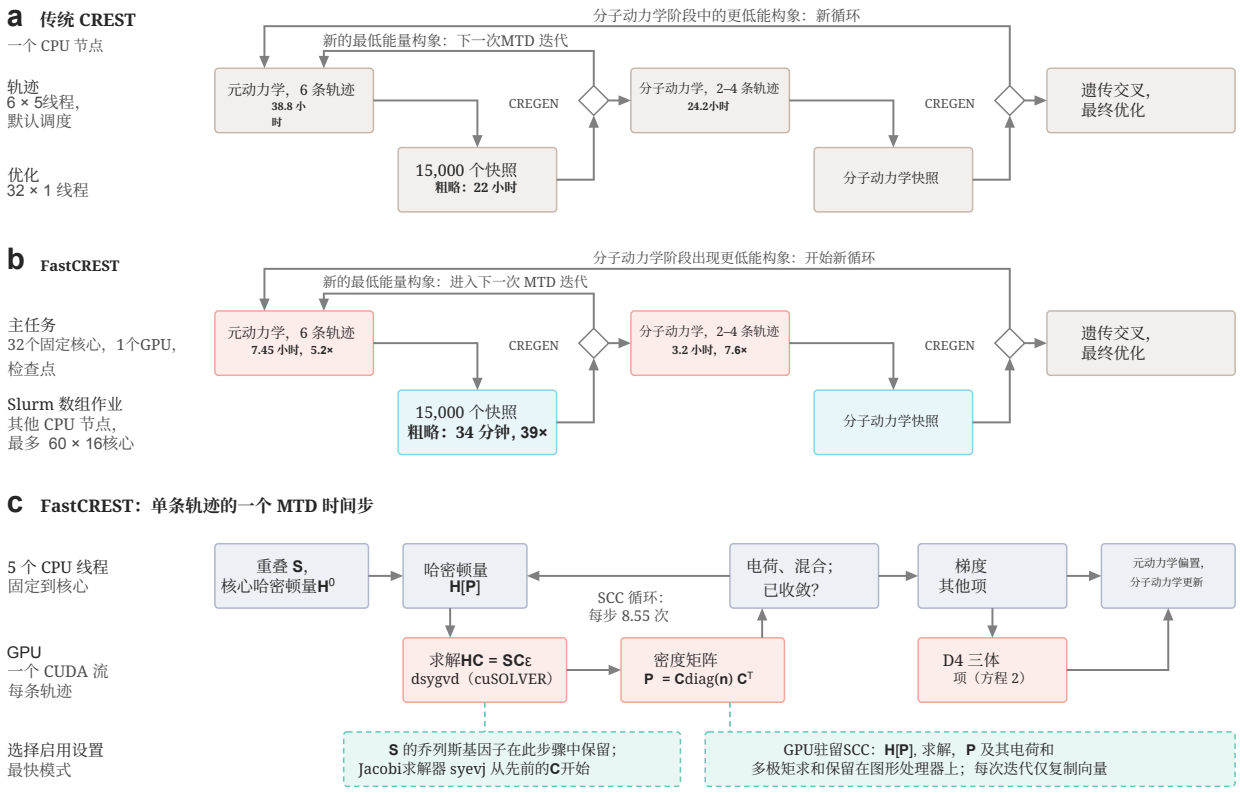


图 1：传统 CREST 和 FastCREST 如何运行相同的 iMTD-GC 搜索。(a) 传统 CREST 在一个 CPU 节点的线程上运行每个阶段；图中给出了 CREST 3.0.2 搜索中每次 MTD 迭代、每个分子动力学阶段以及每批粗略优化的平均时间。(b) FastCREST 在使用一个图形处理器的主任务（红色）中运行轨迹并创建检查点，同时在其他 CPU 节点上以 Slurm 数组作业运行优化（青色）；图中给出了表 1 中 FastCREST 搜索的相同时间（仅使用 GPU 特征值求解器的搜索中的粗略批次），以及相较于 (a) 的加速倍数。(c) 一条 MTD 轨迹的一个时间步：在其五个 CPU 线程（蓝色）和 CUDA 流（红色）上的工作；SCC 循环平均每步运行 8.55 次。绿色虚线：选择启用设置（第 2.4 节和第 2.5 节）。

个氧原子结合：Glu5 的主链羰基氧和一个羧酸根氧原子、Gln6 的侧链酰胺氧，以及 Glu8 和 Glu9 各自的一个羧酸根氧原子（共 274 个原子，N 端和 C 端为中性，电荷为 $+4$）。测试体系的所有 iMTD-GC 搜索均使用 GFN2-xTB、[7] ALPB 隐式水[15] 和 `--quick`，并以同一螺旋结构作为起始构型，该结构已用 GFN2-xTB 预先优化。为进行比较，我们采用了一个使用 CREST 3.0.2 版本[16] 完成的完整搜索，使用 32 个线程和默认 OpenMP 设置；该搜索早先为另一项研究运行于一种未记录的节点类型上，在总计 321.5 h 中，48.2% 的时间用于元动力学，36.7% 用于优化，15.0% 用于分子动力学。我们称之为 CREST 3.0.2 搜索。传统 CREST 指未使用我们 GPU代码的 CREST：CREST

3.0.2 版本、未应用我们的补丁构建的 CREST，或仅在 CPU 上运行的我们的构建版本（传统 CREST 重复搜索）。表 S1 列出了所有搜索的设置。

FastCREST 基于 CREST 提交 cfdc301f、[17] 803.0.2 版本发布后的提交构建，使用 tblite 0.5.0[18] 以及下述补丁。只有其中一个提交会改变结果：CREST 几何优化器（ANCOPT）现在会在达到循环次数上限时立即停止未收敛的优化，而发布版本会先完成当前的步骤块（在循环次数上限为 200 时，最多可达到 239 次）。在循环次数上限为 200 时，与发布版本相比，每 100 个分子动力学快照中未收敛的快照多出 2 到 3 个，超过了我们设定为可接受的每 100 个中 1 个的标准（表 2）。我们保留了该提交，因为在两个版本中均收敛的结构都同样通过或未通过 CREST 的拓扑检查，并且相对于发布版本的受控分子动力学测试通过了（第 3.3 节）。FastCREST 还取消了对优化循环次数的上限。ANCOPT 根据分子的大小设置循环次数上限：对于粗略、常规和严格收敛阈值，分别为每个原子 1、3 或 5 个循环，且至少为 200；在版本 3 之前，xtb 程序的优化器在运行 CREST 优化时也是如此设置的。在每个 CREST 3 版本中，包括当前代码在内，某次搜索的所有优化的循环次数上限都固定为 200（第 S2 节）。FastCREST 保留了依赖尺寸的循环上限（对于测试体系分别为 274、822 和 1,370 个循环），因此，CREST 在 200 个循环后停止并标记为未收敛的优化仍然可以收敛。我们的构建版本在 CPU 上运行时，其单线程能量和梯度与未应用我们的补丁构建的 CREST 按位相同。所有重复搜索，无论是 FastCREST 还是传统 CREST，都使用了依赖尺寸的循环上限和第 2.2 节的线程设置，其中大多数使用了第 2.6 节的分布式优化；这些设置不会改变计算结果。

2.2 线程设置和循环改动

CREST 会同时运行一次 MTD 迭代中的六条轨迹，每条轨迹在嵌套的 OpenMP 并行区域中使用五个线程，[19] 而每项优化使用一个线程。tblite 的并行循环从环境中读取调度方式；在默认设置下，GNU OpenMP 库每次只分配一个循环迭代。因此，五个线程执行能量/梯度评估的速度并不比一个线程快（2.59 秒，而非 2.57 秒）；但采用静态调度后仅需 1.34 秒，速度提高了 1.9 倍（图 2a）。FastCREST 的

线程设置脚本采用静态调度，将每条轨迹的线程绑定到相邻核心，并扩大 C 库的线程栈缓存，使进入嵌套并行区域的速度提高了 3.2 倍（304 微秒，而非 961 微秒）。使用相同的二进制程序时，在两个 A100 节点以及仅配备中央处理器的 AMD EPYC 7502 节点上，MTD 步骤的运行速度比 CREST 3.0.2 搜索的设置快 2.2 至 2.3 倍；在后者上，分子动力学步骤快了 2.0 至 2.5 倍；优化结果没有变化。我们还将 tblite 及其依赖项中 43 处 Fortran 内置函数 `spread`的调用替换为显式循环；这些调用会在内层循环中分配临时数组。以相同的操作数顺序编译且不使用融合乘加时，计算结果的舍入方式与此前完全一致。

另外两项运行时设置只改变线程库以及线程的运行位置。GNU OpenMP 库会在每个并行区域启动嵌套线程组中的线程；替换为 LLVM 的库（libomp 20.1.8）后，无需重新编译，线程便能在各并行区域之间保持运行。单点计算结果与 GNU 库下的结果一致，差异处于多次多线程运行之间的差异范围内（$1.7 \times 10^{-13}\ E_\mathrm{h}$）。当作业的核心在节点的两个处理器插槽之间分配不均时（我们所有的逐步测量均是如此），一条轨迹所用的五个核心可能分布在两个插槽上；这条轨迹的运行速度于是比其他轨迹慢了最多 18%（图 2c），每次元动力学迭代都必须等待它。申请 34 个核心——这些核心总是包含六组各有五个核心且每组都位于同一个插槽上的核心——并将每条轨迹绑定到一组核心（30 个线程），即可消除这条较慢的轨迹。

2.3 GPUEigensolver 与三体色散

每次 SCC 迭代都会求解广义特征值问题

$$\mathbf{H}[\mathbf{P}]\,\mathbf{C} = \mathbf{S}\,\mathbf{C}\,\boldsymbol{\varepsilon}, \qquad \mathbf{P} = \mathbf{C}\,\mathrm{diag}(\mathbf{n})\,\mathbf{C}^{\mathsf{T}}, \tag{1}$$

其中，哈密顿量 $\mathbf{H}$取决于密度矩阵 $\mathbf{P}$，$\mathbf{S}$ 为重叠矩阵，$\mathbf{C}$ 包含轨道系数，$\boldsymbol{\varepsilon}$ 为轨道能量，$\mathbf{n}$ 为费米展宽占据数。对于测试体系，矩阵为 671×671，每个时间步平均需要 8.55 次 SCC 迭代。在元动力学

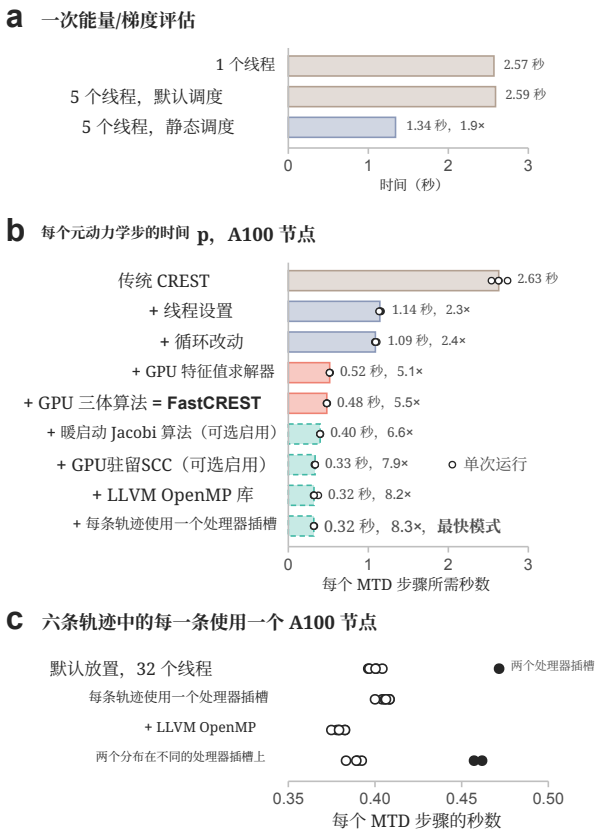


图 2：每个 MTD 步骤的加速来自何处。(a) 在配备 AMD EPYC 7502 的节点上，测试体系进行一次能量/梯度评估所需的时间：分别使用一个线程，以及在默认调度和静态 OpenMP 调度下使用五个线程，并给出相对于默认调度的加速倍数。(b) 在两台 A100 节点（主机为 AMD EPYC 7543）上，逐项加入各项更改后每个 MTD 步骤所需的时间，以及相对于传统 CREST 的加速倍数：四次运行的中位数，每个节点运行两次（柱形）；以及各次单独运行的结果（圆点；一次 LLVM OpenMP 运行因节点上的其他任务而变慢）。除最后一行外，所有行均使用作业的 34 个核心中的 32 个；最后一行使用 30 个核心（每条轨迹分配一个处理器插槽）。虚线表示选择启用的设置。(c) 使用雅可比特征值求解器时六条轨迹各自的每个 MTD 步骤所需时间（两次运行的平均值）。作业在两个处理器插槽上分别使用 16 个和 24 个核心：采用默认线程放置，每条轨迹使用两个插槽（实心）；每条轨迹使用一个插槽，且使用 LLVM 的 OpenMP 库；以及每条轨迹使用两个插槽的对照设置。

和分子动力学中，FastCREST 使用 cuSOLVER[20] 的双精度 `dsygvd` 例程求解方程 1，并使用 cuBLAS[21] 构造**P** ，每条轨迹使用一个 CUDA 流（图 1c）。若 GPU 上的求解未能收敛，则使用传统 CREST 的 LAPACK 例程在中央处理器上重新求解。一个确定性的双精度 GPU 内核用于计算三体（ Axilrod–Teller–Muto ）项

D4色散校正的，[22–24]

$$E^{(3)} = \sum_{A<B<C} C_9^{ABC}\, f_{\mathrm{damp}}\, \frac{3\cos\theta_A \cos\theta_B \cos\theta_C + 1}{(r_{AB}\, r_{BC}\, r_{CA})^3}, \qquad C_9^{ABC} = 5\left(C_6^{AB} C_6^{BC} C_6^{CA}\right)^{1/2}, \tag{2}$$

原子间距离为 r，每个原子三角形的内角为 θ， C_6 零电荷下的系数，以及 GFN2-xTB 阻尼函数 f_{damp}。[7] 该内核在每次能量/梯度评估时运行一次，最多处理 340 万个原子三元组，所得梯度与中央处理器上的结果相差不超过 $4.4 \times 10^{-18}\ E_{\mathrm{h}}$ 玻尔$^{-1}$。优化仍在中央处理器上进行：32 个优化任务共享一个 GPU 时，运行速率仅为使用中央处理器核心时的 0.87 至 0.89 倍；此外，中央处理器上的结果可以逐位核验。

2.4 可选的热启动雅可比特征值求解器

在一次能量/梯度评估中，**S** 保持不变，而 **H** 在相邻的 SCC 迭代之间变化很小。因此，可选择启用一项设置：每次评估时仅对 **S** 进行一次因子分解（Cholesky分解），并使用 cuSOLVER 的雅可比求解器（`syevj`）求解由此得到的标准特征值问题；求解器以此前一次 SCC 迭代的特征向量为初始值（图 1c）。在第 3.1 节的逐步测量中，平均每次求解需要 2.8 次雅可比扫描。对于单点测试中的 306 个结构（第 2.8 节），能量与标准 GPU 求解器所得结果的差异不超过 $3.4 \times 10^{-13}\ E_{\mathrm{h}}$，梯度的差异不超过$3.4 \times 10^{-14}\ E_{\mathrm{h}}$ 玻尔$^{-1}$；在逐步测量中，133,158 次求解均未回退到标准求解器。此设置默认关闭，因为它仅在 A100 GPU 上经过测试，且其收益取决于分子大小（表 3）。

2.5 可选的驻留于 GPU 的 SCC 迭代

在上述 GPU 代码路径的每次 SCC 迭代中，中央处理器构建**H**，图形处理器求解方程 1，随后**C** 和**P** 返回中央处理器；中央处理器根据**P** 计算 Mulliken电荷、原子偶极矩和四极矩以及能带能量；对于该测试体系，每次迭代中中央处理器与图形处理器之间约有 18 MB 的数据传输。第二个选择启用设置会将迭代保留在图形处理器上（图 1c）。

核心哈密顿量、重叠积分以及偶极和四极积分在每次能量/梯度评估时仅复制到图形处理器一次。在每次迭代中，中央处理器随后只发送势能位移；图形处理器构建$\mathbf{H}$ 并求解方程 1；中央处理器设置费米占据数；图形处理器形成 $\mathbf{P}$，并返回每个基函数对应的 11 个求和结果，中央处理器据此组装电荷、矩和能量。电荷混合及所有相互作用势均保留在中央处理器上，而$\mathbf{C}$ 和$\mathbf{P}$ 仅在最后一次迭代后为计算梯度复制回来。一次能量/梯度评估移动的数据量因此从 154 MB 降至 48 MB。图形处理器内核使用 tblite 的公式，并在每个矩阵元内保持相同的运算顺序；只有对基函数的求和采用了不同的顺序。在单点测试的 306 个结构上（第 2.8 节），能量与标准 GPU 代码路径的结果一致，误差在 $3.4\times10^{-13}\ E_\mathrm{h}$ 以内；梯度误差在 $5\times10^{-15}\ E_\mathrm{h}\,\mathrm{bohr}^{-1}$ 以内（对于两个选择启用设置，与不使用 GPU驻留SCC 的 Jacobi求解器相比，误差均在 $2.8\times10^{-13}\ E_\mathrm{h}$ 以内），其一致性与图形处理器和中央处理器代码路径彼此之间的一致性相当。该设置兼容两种本征值求解器；开壳层计算则保留标准 GPU 代码路径。

2.6 分布式优化和检查点

主任务在一个图形处理器节点上运行轨迹，并将每批 128 个或更多优化任务分成每块 64 个结构，发送给一个 Slurm 作业数组[25]；数组作业则在其他中央处理器节点上运行相同的二进制程序（图 1b）。合并之前，主任务会使用自己的中央处理器模型重新优化数组作业中的最多 32 个结构，这些结构必须逐位一致；还会重新优化使用其他中央处理器模型的数组作业中的最多 32 个结构，这些结构的能量差异必须在 $10^{-7}\ E_\mathrm{h}$ 以内，坐标差异必须在 10^{-4} Å 以内；任何不匹配都会导致重新运行整批任务或受影响的任务块。每条轨迹每 40 步写入一个精确检查点，并记录每个优化后的结构，因此，在元动力学迭代期间停止的搜索会从上一个检查点继续；中断的分子动力学阶段则从头开始。启用检查点后，每条轨迹都会从一个全新的初始猜测开始其第一次 SCC 过程；这是检查点导致的唯一行为变化。我们没有测试恢复后的搜索在统计上是否与不间断运行的搜索等效。

2.7 性能指标

搜索会自适应地调整长度，因此我们比较 E/G 吞吐量 $R = N/t_{\mathrm{wall}}$ 以及每次评估的分配成本，

$$K_{\mathrm{CPU}} = \frac{1}{N}\sum_j c_j t_j, \qquad K_{\mathrm{GPU}} = \frac{1}{N}\sum_j g_j t_j, \tag{3}$$

其中， N 统计主任务和所有数组作业的能量/梯度评估次数， t_{wall} 是主任务的运行时间，而作业 j 在其运行时间t_j内分配了 c_j 个硬件线程和 g_j 个 GPU。每个 MTD 步骤的时间取自一批六条 MTD 轨迹，每条轨迹包含 400 个步骤；每种配置均在单个节点上运行，先按图 2b 中的顺序执行，再按相反顺序执行。

2.8 计算测试

我们比较了 FastCREST、发布版本以及未应用我们补丁构建的 CREST 在 306 个结构上的单点计算，并比较了 GPU 代码路径与中央处理器路径。由于多线程求和不具可重复性，轨迹无法逐位比较：[26] 即使是同一二进制程序的两次运行，在中位数为 6.5 ps 后，全原子均方根偏差也超过了 0.5 Å。因此，我们进行了受控分子动力学模拟：从四个起始结构出发，在 400 和 500 开尔文下各运行 20 ps，并使用相同的初始速度，然后比较成键拓扑发生变化的帧所占比例。如果对 24 或 44 对轨迹计算得到的均值差配对 90% 置信区间完全落在 ± 0.15 之内，则认为两种配置等效（双单侧检验[27]）。整个搜索过程也不具可重复性，因此我们比较了重复搜索组在每次搜索都会经过的一个节点上的结果：首次 MTD 迭代后最低的粗略能量，并计算 Hodges–Lehmann 位移及其精确置信区间。[28,29] 除重复搜索比较的规则外，所有标准均在运行前确定；该规则是在看到部分个体结果后、但在计算任何组统计量之前确定的。详情见 S2 和 S3 节。

3 结果与讨论

3.1 每个元动力学步的时间

图 2b 展示了在两个 A100 节点（AMD EPYC 7543 主机）上逐项加入改动后的效果。传统 CREST 在这些节点上每个 MTD 步骤耗时 2.63 秒（CREST 3.0.2 搜索中为 2.8 秒）。线程设置和循环改动将耗时降至 1.09 秒，提速 2.4 倍；GPU 特征值求解器和三体项则进一步降至 0.48 秒，再提速 2.3 倍，因此 FastCREST 在相同节点上的每步速度提高了 5.5 倍。雅可比特征值求解器将耗时降至 0.40 秒（比传统 CREST 快 6.6 倍）。GPU驻留 SCC 迭代单独使用时耗时降至 0.42 秒（提速 6.3 倍），与 Jacobi 求解器结合时降至 0.33 秒（提速 7.9 倍）；采用 LLVM 的 OpenMP 库后耗时为 0.32 秒（提速 8.2 倍）。此外，每条轨迹使用一个处理器插槽时，最快模式（同时启用两项可选设置、LLVM 的 OpenMP 库，并为每条轨迹使用一个插槽）的单步速度比传统 CREST 快 8.3 倍，比 FastCREST 快 1.5 倍。使用 GPU驻留SCC 时，核心跨越两个插槽的轨迹仅比其他轨迹慢约 5%，因此每条轨迹使用一个插槽只使平均步速提高了 1.01 倍。仅使用 Jacobi 求解器时，该轨迹慢了 18%；而每条轨迹使用一个插槽，使决定 MTD 迭代速度的最慢轨迹快了 1.15 倍（图 2c ）。

每一步仍主要是 CPU 运算（在早期版本的性能分析中占 63.5%），在 FastCREST 中，六条轨迹使图形处理器保持 83% 至 87% 的忙碌状态，而在最快模式下则为 79% 至 87%。因此，每次搜索增加第二个图形处理器仅获得 1.2 倍的性能提升；两个搜索共享一个图形处理器和 32 个核心时，综合输出最多提高 1.12 倍，但每个搜索的运行速度慢了 1.6 至 1.9 倍。

我们在同类型的一个 A100 节点上，对另外六种由 24 到 339 个原子组成的闭壳层分子重复进行了图 2b 的测量（表 3）。对于每种分子，线程设置都使单步速度提升了 2.1 至 3.0 倍。随着分子增大，图形处理器特征值求解器的收益也随之增加：从咖啡因的 1.05 倍提升至 16 单元直链淀粉链的 2.8 倍，因此 FastCREST 单步速度比传统 CREST 快 2.2 至 7.1 倍。对于中等大小的分子，Jacobi求解器又带来了 1.2 至 1.3 倍的提升，但对于最小和最大的分子，提升仅为 1.02 倍和 1.09 倍。LLVM 的 OpenMP 库对最小的分子咖啡因帮助最大（提升 3.1 倍）。在最快模式下，单步

比传统 CREST 快 6.9 到 11.7 倍。

3.2 完整搜索

表1列出了完整搜索。FastCREST 搜索两次找到了更低能构象，因此运行了三个循环而不是两个；加上额外的循环，并且不受 200 个优化循环上限的限制，它进行的 E/G评估次数是 CREST 3.0.2 搜索的 1.37 倍。它耗时 54.3 小时，因此吞吐量是 CREST 3.0.2 搜索的 8.1 倍。每百万次评估分配了 1,231 个 CPU 小时和 7.5 个 GPU 小时，而后者至少需要 1,939 个 CPU 小时，因此只要 GPU 小时的成本低于约 94 个硬件线程小时（47 个核心小时），它就更便宜。优化所占运行时间从 36.7% 降至 13.6%，因此轨迹现在占运行时间的 86%。在没有图形处理器的情况下，采用线程设置的搜索（其中两次还采用了分布式优化）达到了 CREST 3.0.2 搜索 3.3 至 4.7 倍的吞吐量；采用暖启动 Jacobi 算法特征值求解器后，三次 FastCREST 搜索分别达到了 9.5 倍、10.0 倍和 9.5 倍。最快模式下的两次搜索分别达到了 12.4 倍和 12.7 倍，每百万次评估分配了 1,195 和 1,170 个 CPU 小时，以及 4.9 和 4.8 个 GPU 小时。

3.3 相同计算

表 2 总结了测试结果。306 个结构的单点计算与发布版本的能量差在 $1.01 \times 10^{-10}\ E_\mathrm{h}$ 以内，梯度差在 $8.2 \times 10^{-11}\ E_\mathrm{h}$ 玻尔$^{-1}$ 以内，比我们的判据低两个数量级；FastCREST 的图形处理器和中央处理器代码路径（包括两个选择启用设置）之间的差异最大为 $4.6 \times 10^{-13}\ E_\mathrm{h}$。全部 13 项受控分子动力学比较均通过了等价性检验，其中 6 项采用了一个或两个选择启用设置。

两种代码中的重复搜索结果都分散较广。在已完成的 15 次传统 CREST 重复实验中，有一次得到了所有重复搜索中能量最低的最终构象，12 次的结果比它高 3.7 至 25.2 千卡/摩尔，另有两次高出 110 千卡/摩尔以上，这是 CREST 本身的回退机制所致（第 3.4 节）。首次 MTD 迭代后，未发现 FastCREST 与传统 CREST 之间存在差异：16 次 Fast CREST 搜索的最低能量通常……

表1：测试体系的完整搜索

搜索	硬件	MTD/MD 墙钟时间（小时）	N (10^6)	次数快	K_{CPU}	K_{GPU}
CREST 3.0.2[a]	未记录	4/2	321.5	5.31	1.00×	1,939	0
CPU, 一个节点[b]	Zen 4 , 64个核心	4/2	100.3	5.50	3.32×	2,334	0
CPU, 分布式[c]	Zen 4	4/3	98.7	6.17	3.79×	2,851	0
CPU, GPU代码关闭[d]	Zen 5	2/1	33.8	2.62	4.71×	2,255	0
仅 GPU 特征值求解器[e]	A100/EPYC 7502	4/2	52.0	5.38	6.28×	1,574	9.7
FastCREST[f]	A100/EPYC 7543	5/3	54.3	7.26	8.11×	1,231	7.5
FastCREST + 雅可比 , 搜索 1[g]	A100/EPYC 7543	4/3	36.9	5.77	9.48×	1,194	6.4
FastCREST + 雅可比 , 搜索 2[g]	A100/EPYC 7543	3/2	28.4	4.69	9.99×	1,228	6.1
FastCREST + 雅可比 , 搜索 3[g]	A100/EPYC 7543	4/2	36.8	5.74	9.46×	1,315	6.4
最快模式 , 搜索 1[h]	A100/EPYC 7543	5/3	35.2	7.17	12.36×	1,195	4.9
最快模式 , 搜索 2[h]	A100/EPYC 7543	6/4	42.7	8.97	12.73×	1,170	4.8

每行代表一次搜索，输入和设置相同；但 CREST 3.0.2 搜索将优化限制为 200 个循环（第 2.1 节）：MTD 迭代次数和 MD 阶段数、主任务的耗时、E/G评估次数 N、按每小时 E/G评估次数计算的速度是 CREST 3.0.2 搜索的多少倍（各次搜索的迭代次数不同，因此其实际运行时间不可比较），以及每百万次评估所分配的 CPU 小时和 GPU 小时（$K_{\mathrm{CPU}}, K_{\mathrm{GPU}}$；公式 3）。硬件：主任务使用的图形处理器和主机中央处理器（AMD处理器）。除前两次搜索外，其余搜索都在其他节点上以 Slurm 数组作业的形式运行优化。[a]发布版本，采用默认 OpenMP 设置；CPU 小时按 32 个线程 × 墙钟时间计算，并非记录的资源分配量。[b]线程设置和循环改动，不使用图形处理器。[c]线程设置、分布式优化与检查点，不包含循环改动。[d]FastCREST 的二进制程序关闭图形处理器代码后运行，并计入传统 CREST 重复实验；仅运行一个循环，因为经过 MTD 迭代后，CREGEN 回退至螺旋输入（第 3.4 节）。[e]早期版本：没有循环改动，三体项在中央处理器上运行。[f]第 2.2、2.3 和 2.6 节中的默认改动。[g]使用暖启动 Jacobi 特征值求解器（第 2.4 节）。[h]同时启用两项选择启用设置（第 2.4 节和第 2.5 节），使用 LLVM 的 OpenMP 库，并为每条轨迹分配一个处理器插槽（30 个线程）。EPYC 7502 是 Zen 2处理器，EPYC 7543 是 Zen 3处理器。

在数值上比 17 次常规 CREST 搜索的结果高 4.1 千卡/摩尔（Hodges–Lehmann 位移；95% 置信区间为 -4.9至 $+11.8$ 千卡/摩尔）。不能排除 FastCREST 存在最高 11.8 千卡/摩尔的劣势；若要证明在 ± 2 千卡/摩尔范围内具有等效性，则每组需要进行一百多次搜索。最终能量与此结果一致（中位数：FastCREST 比最低能最终构象高 $+11.6$，传统 CREST 高 $+16.4$ 千卡/摩尔；事后偏移为 -2.7千卡/摩尔，95% 置信区间为 -11.3至 $+6.5$）。表 S5 列出了这些搜索。因此，FastCREST 计算相同能量和梯度并采样等效轨迹的证据依赖于代码级测试。

表 2：计算测试

test	比较对象	标准	结果
搜索设置	八种构型	相同参数	相同（柔性度量 0.391，六条 MTD 轨迹 持续 250 皮秒，272 个 SHAKE 约束[30]）
单点计算	306 个结构，九个构型	最大 $\lvert\Delta E\rvert \le 10^{-8}\ E_\mathrm{h}$，最大梯度差异 $\le 10^{-6}\ E_\mathrm{h}$ 玻尔$^{-1}$	$1.01 \times 10^{-10}\ E_\mathrm{h}$ 和 $8.2 \times 10^{-11}\ E_\mathrm{h}$ 玻尔$^{-1}$ 来自发布版本
优化器停止于循环次数上限（CREST 变化在 3.0.2 之后）	200 个分子动力学快照，我们的构建版本与发布版本对比，两者均为 200 个循环	最多再多出 1 个每 100 个中有未收敛项	每 100 个中再多 2 到 3 个：上限已超出；保留更改（第 2.1 节）
受控分子动力学	24 次比较中的 13 次，或 44 对轨迹	90% 置信区间均值差的在 ± 0.15 内	在分子动力学中，全部 13 个等效快照以及优化后优化
重复实验搜索	最低能量，在首次 MTD 迭代后，16 FastCREST 和 17 传统 CREST 搜索	差异：95% 区间不包含 0；等效性：90% 区间内 ± 2 千卡/摩尔	无法确定：偏移 $+4.1$ 千卡/摩尔，95% 区间 -4.9 至 $+11.8$
发布的代码	干净克隆，重新构建	字节级相同的输出；测试脚本通过	字节级完全相同；17 项测试中有 17 项通过

表 3：不同大小分子的每个 MTD 步骤耗时

			传统 CREST	比传统 CREST 快		
分子	原子	电荷	（每步秒数）	FastCREST	+ 雅可比	最快模式
咖啡因	24	0	0.165	2.2×	2.2×	11.7×
紫杉醇	113	0	0.581	3.1×	4.1×	7.0×
Ga(III)–肽	132	+3	0.782	3.5×	4.6×	6.9×
Ga(III)–肽	174	−5	1.575	5.2×	6.8×	9.5×
Ga(III)–肽	236	−8	2.358	5.2×	6.4×	7.8×
测试体系[a]	274	+4	2.630	5.5×	6.6×	8.3×
直链淀粉，16 个单元	339	0	3.948	7.1×	7.7×	9.0×

传统 CREST 每个 MTD 步骤的时间，以及 FastCREST、采用雅可比特征值求解器的 FastCREST 和最快模式分别快了多少倍。在配备 AMD EPYC 7543 主机的 A100 节点上，每个分子运行一个作业，每种设置取两次运行的平均值，测量方式如图 2b 所示。所有分子均为闭壳层；GFN2-xTB，ALPB 水，`--quick`。[a]图 2b 中四次运行的中位数。

3.4 最终结构与能量

CREST 的拓扑检查只保留具有输入结构中化学键的结构，因此每个最终构象体都保留了五个 Ga–O键，而搜索结果的差异主要在肽的其余部分。重复搜索中能量最低的最终构象来自一次传统 CREST 搜索。它比输入结构低 137.3 千卡/摩尔，且结构紧凑（回转半径为 6.33 Å，而输入结构为 7.56 Å）：螺旋已经展开，Glu8 的第二个羧酸根氧原子距镓 2.33 Å；拓扑检查不会将这种接触计为化学键。这些能量以及对紧凑折叠的偏好均为 GFN2-xTB/ALPB 的结果，我们未作进一步验证。

在已完成的 26 次重复搜索中（10 次 FastCREST、15 次传统 CREST，以及一次启用 Jacobi 选项的 FastCREST 搜索），除两次传统 CREST 搜索外，其余搜索都以紧凑构象收尾（重原子回转半径不超过 7.0 Å，而输入结构为 7.56 Å）；我们将这些搜索称为紧凑搜索。在那两次搜索中，一次 MTD 迭代的常规优化改变了能量窗口内每个结构的成键拓扑，CREGEN 因而回退到螺旋输入（此处使用的 CREST 代码与 3.0.2 版本中的相同）；两次搜索运行的都是关闭 GPU代码的 FastCREST 二进制程序。FastCREST 的最低能构象（十次搜索中最佳者）比能量最低的最终构象高 1.9 千卡/摩尔，并具有相同的 Glu8 接触（2.33 Å）。将二者叠合到结合位点[31] （图3a）后可以看到，从 Arg4 到 Lys12 的部分相差不超过 2.4 Å，而两端则指向其他方向（Met1 到 Thr3，以及 Lys13 到 Lys15，相差 2.5 至 5.2 Å ）。其他搜索均未以这一构象收尾：其他搜索中最接近的最终构象与其相差 1.2 Å（重原子均方根偏差），能量则高出 7.9 千卡/摩尔；其他紧凑搜索的最低能构象与其相差 0.8 至 3.7 Å（主链均方根偏差）（图3b）。FastCREST 和传统 CREST 搜索的结果与该构象的差异主要出现在两端（图3c）：每个残基的四分位距均有重叠；在 Met1、Leu2、Leu10、Leu14 和 Lys15 处，两者的中位数相差超过 0.5 Å，其中 FastCREST 的数值大 0.6 至 1.1 Å。

系综规模差异很大（FastCREST 为 1 至 173 个构象，传统 CREST 为 1 至 82 个），因为每次搜索都只保留其最后一个循环中能量处于自身最低构象以上 5 千卡/摩尔范围内的构象。在这一范围内，FastCREST 和传统 CREST 系综中的构象分布

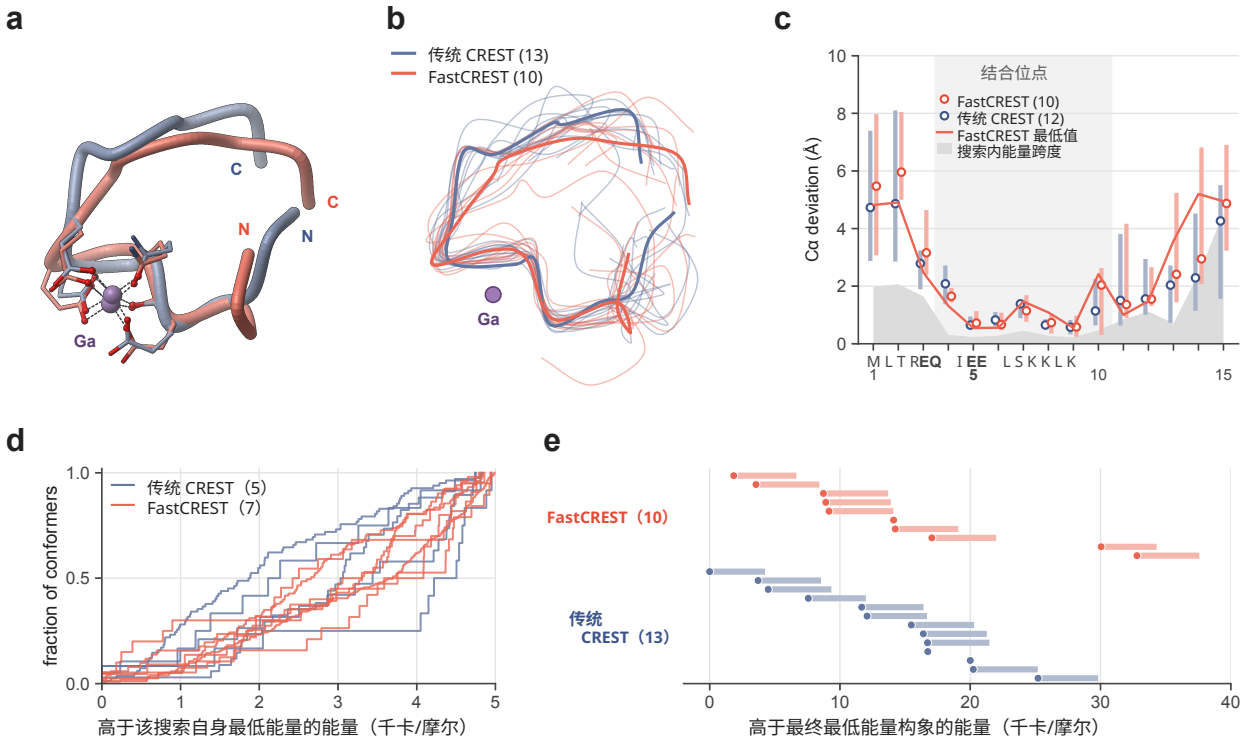


图 3：重复 FastCREST 搜索（红色）和传统 CREST 搜索（蓝色）的最终构象；括号中的数字表示搜索次数。（a–c）叠加在结合位点上的结构（镓、Glu5 的 Cδ、Gln6、Glu8 和 Glu9，以及残基 4–10 的主链）。（a）重复搜索（传统 CREST）中的最低能最终构象，以及最低能 FastCREST 构象（十次搜索中最佳者，高出 1.9 千卡/摩尔）；管状结构描绘 Cα 原子，虚线连接镓与 2.6 Å 范围内的氧原子。（b）每次紧凑搜索中最低能构象的 Cα 轨迹，其中（a）中的两个构象以粗体显示。（c）对于每次其他紧凑搜索中的最低能构象，显示各残基的 Cα 相对于最低能最终构象的偏差：中位数（圆圈）和四分位距（条形），（a）中的 FastCREST 构象（线），以及（a）中两次搜索的任一最终构象相对于其自身搜索中最低能构象的最大偏差（阴影）。粗体残基与镓结合。（d）对于包含至少 10 个构象的系综，显示构象相对于其所属搜索中最低能构象的能量差的累积分数。（e）每行对应一次紧凑搜索：其最低能构象（点）及其最终构象的能量范围（条形），均以最低能最终构象为参照。未在（b–e）中显示：两次最终停留在接近螺旋输入位置的传统 CREST 搜索（+116 和 +125 千卡/摩尔），以及启用雅可比选项的 FastCREST 搜索（+33.1 千卡/摩尔）。

相似：FastCREST 系综中，构象的中位能量比该次搜索自身的最低能量高 2.5 至 3.8 千卡/摩尔；传统 CREST 系综中则高 1.9 至 4.4 千卡/摩尔（图 3d）。相对于最低能最终构象，FastCREST 搜索的能量分布范围与传统 CREST 搜索一样宽（最低构象分别高出 1.9 至 32.8 千卡/摩尔和 0 至 25.2 千卡/摩尔）；10 次 FastCREST 搜索中有 2 次、另外 12 次规模较小的传统 CREST 搜索中也有 2 次，最终结果与该构象的能量差在 5 千卡/摩尔以内（图 3e）。

3.5 其他应用和基准测试

我们还在其他体系上比较了 FastCREST 与传统 CREST；每个体系各进行了一次 CREST 3.0.2 版本搜索（S7 至 S9 节以及表 S7 中 NCI-MTD 的最后一行）。对于第二个 Ga(III)–肽复合物（264 个原子），FastCREST 在一块 A100 GPU 上通过数组作业耗时 58.4 h，而传统 CREST 在 CPU 节点的 32 个线程上耗时 265.8 h，因此 FastCREST 快了 4.6 倍，尽管它运行了三个循环，而非两个。两个最终系综都与螺旋输入相近；每次搜索的最后一个循环都从该输入重新开始。它们的最低能构象彼此相差不超过 0.7 Å（Cα 均方根偏差）和 1.1 千卡/摩尔，且供体残基相同（S8 节，图 S3(d-e)）。FastCREST 在 13 至 34 h 内完成了全部四个 Zn–肽体系（三个 12 残基肽，p1 至 p3，以及低 pH 条件下的 p1；S7 节）。传统 CREST 在 97 h 内完成了一个体系，即低 pH 条件下的 p1（FastCREST 快了 7.3 倍）；运行 53 h 后，我们停止了对中性 p1 的搜索，而对 p2 和 p3 的搜索在 6 天后仍未完成（FastCREST 至少快 10 倍和 4.2 倍）。在两种代码都得到结构的体系中，肽都以大致相同的供体残基缠绕在锌周围。

对测试体系（Ga(III)–肽）进行 NCI-MTD 搜索，使用 FastCREST 耗时 34.2 小时，使用 CREST 3.0.2 耗时 93.7 小时（FastCREST 快 2.7 倍）；借助 NCI-MTD 的壁势，FastCREST 可在主任务中执行优化。两次运行获得的最低能构象具有相同的金属配位残基和供体原子。

我们还使用 FastCREST 和 CREST 3.0.2 对一个高自旋 Fe(II)–肽复合物进行了测试（第 S9 节）。两种搜索得到的最低能构象存在差异，其 Fe(II) 配位环境也不同：CREST 更倾向于一种由四个氧供体配位的结构，而 FastCREST 则识别出一种由六个氧供体配位、更加紧凑的结构。FastCREST 得到的最低能结构在 GFN2-xTB 能量上比传统 CREST 最终得到的最低能结构低 85.5 kcal mol^{-1}（图 S3 d、e；表 S9）。此类差异可能反映了构象采样的随机性[32] 以及开壳层过渡金属复合物更高的电子结构复杂性；对于这类体系，已有研究报道了近简并电子态和自洽场收敛困难，这些问题会出现在扩展紧束缚计算中。[33]

在 CREST 的其他运行类型中，质子化、去质子化和互变异构体搜索使用 FastCREST 和传统 CREST 得到了相同的最低能质子异构体和互变异构体；NCI-MTD 和构象熵运行在两种代码中都会因运行次数不同而产生结果散布，而在我们重复进行这些运行的情况下，其结果在这一散布范围内保持一致（表 S7）。

3.6 局限性

重复搜索、表1中的搜索以及大多数计算测试均来自一个包含 `--quick` 的体系；其他金属–肽体系在每个代码版本中仅搜索一次（第 3.2 节）。每个 MTD 步骤的性能增益取决于分子的大小（表 3），也取决于中央处理器和图形处理器（表 S6）。表1中的大多数配置在不同硬件上完成了一次完整搜索，而重复组则在不同条件下运行。与 CREST 3.0.2 版本或与未应用我们的补丁构建的 CREST 进行的比较（表1、表 S7 以及第 S7 至 S9 节）还包含 FastCREST 所取消的 200 次循环上限；取消该上限使更多优化得以收敛，并增加了能量 / 梯度评估；我们未测量其对吞吐量或每次评估 CPU 小时数的影响。在启动的 40 次 FastCREST 重复搜索中，有 16 次完成了首次 MTD 迭代：其中 17 次在恢复运行时因一个 GPU 发生内存错误而停止；由于已经获得了中期结果，我们没有重新启动这些搜索；另外 7 次被提前取消或仍在排队。在这 16 次 FastCREST 搜索中，有 13 次在首次迭代期间从检查点恢复，而 17 次传统 CREST 搜索中仅有 7 次如此；其中 16 次中有 12 次在 L40S 或 V100 GPU 上完成了首次迭代，而代码级测试未涵盖这些 GPU；此外，有 6 次在部分运行期间由中央处理器计算三体项。在后续循环的首次 MTD 迭代中恢复运行时，有 4 次搜索失败，且每次都发生在该搜索完成首次 MTD 迭代之后（已在 1.2.0版本中修复）。对于使用每种代码对一个高自旋 Fe(II)–肽复合物进行的单次搜索（264 个原子；第 S9 节），在这些设置下，大多数 MTD 和 MD 轨迹均提前停止（传统 CREST 中为 50 次中的 48 次，FastCREST 中为全部 20 次），这与对相关肽的 Cu 和镍配合物进行的 FastCREST 单次搜索相同（未显示）；两种搜索都未按预期进行采样，并且最终得到的结构不同。尽管这一行为被列为 FastCREST 的局限性，但在 CREST 中完全相同。对于该

Fe(II)配合物的 10 个结构，FastCREST 的能量和梯度与 CREST 3.0.2 的结果一致，误差在$9 \times 10^{-11}\ E_\mathrm{h}$ 和 $1.2 \times 10^{-10}\ E_\mathrm{h}$ 玻尔$^{-1}$以内。当 CREST 找到更低能构象时，它会从优化后的输入结构而不是该构象开始下一组 MTD 轨迹；当这一操作启动新循环时，它还会丢弃截至目前找到的系综（3.0.2 版本也是如此）。在已完成的 26 次重复搜索中，至少有 24 次出现了这种情况，且两种代码中均如此，因此最终系综的能量可能高于搜索过程中发现的最低能构象。FastCREST 1.3.0 可修复这两个问题（可选择启用；第 S6 节）；第 S7 节给出了一个锌结合肽的示例。

4 结论

在 GPU集群上，FastCREST 执行测试体系的 CREST iMTD-GC 搜索时，E/G 吞吐量达到 CREST 3.0.2 搜索的 8.1 倍（最快模式下为 12.4 倍和 12.7 倍），且能量和梯度保持不变。对于 274 原子的测试体系，每个 MTD 步骤的加速收益约有一半来自任何 CREST 用户都能采用的线程设置，另一半来自使用一块图形处理器求解特征值和计算三体色散项；随着分子规模增大，图形处理器带来的收益占比也会提高（表 3）。将优化作为数组作业在其他节点上运行，使其占总运行时间的比例从 37% 降至 14%，因此进一步提速必须着眼于 MTD 步骤本身。使用雅可比求解器时，将 SCC 迭代保留在图形处理器上可使单步速度再提高 1.2 倍；仍在中央处理器上计算的梯度是下一个可优化环节。仅缓存重叠矩阵分解、在轨迹之间共享空闲线程、在优化过程中批量进行特征值求解、采用其他编译选项以及使用 Intel MKL，带来的收益都很小或没有收益；而密度矩阵纯化则会改变算法。

FastCREST 的重要性在于，在不改变传统 CREST 所采用的底层 GFN2-xTB 理论水平的情况下，能够对大型且柔性的分子体系开展更广泛的构象探索。能量和梯度之间高度一致的数值结果，加上对结构相似且能量具有竞争力的构象体的识别，支持将其作为构象筛选的加速替代方案。通过大幅降低计算成本，FastCREST 能够研究更大的体系，探索多个起始

几何结构，并更快地筛选候选分子。尽管由于构象采样具有随机性，单次搜索可能会识别出不同的能量极小值，但计算效率的提升能够扩大对构象空间的探索，并有助于为后续更高理论水平的计算选择具有代表性的低能结构。

数据与软件可用性

FastCREST 可从 `https://github.com/chowdhurylab-studio/fastCREST`（版本 1.3.0）获取。它由应用于 CREST cfdc301f、[17] tblite、[18] dftd4,[34] simple-dftd3,[35] multicharge[36] 和 mctc-lib 的补丁组成，[37] 每个补丁均遵循其所修改代码的许可证；同时还包含一个用于获取固定上游提交的构建脚本、线程设置脚本、Slurm 作业模板和测试脚本。关闭 FastCREST 的选项后，单线程能量和梯度与未应用我们补丁构建的 CREST 按位相同。从 GitHub 进行全新构建后，得到的单点输出与本文所使用二进制程序的输出逐字节相同，并通过了测试脚本（表 2）。输入结构、最终系综和原始输出可向作者索取。

致谢

我们感谢 Mahbuba Ferdaous 提供相关肽的开壳层金属配合物结构，并感谢爱荷华州立大学研究信息技术部门提供 Nova 集群的使用权限。本研究部分由美国能源部科学办公室生物与环境研究办公室生物系统科学部 2025 年实验室关键材料与矿产（CMM）试点计划资助，受资助者为 R.C.。艾姆斯国家实验室由爱荷华州立大学根据合同编号 DEAC02-07CH11358 代表美国能源部运营。FastCREST 模型构建工作由关键材料创新中心资助。该中心是由美国能源部关键矿产与能源创新办公室（CMEI）、先进材料与制造技术办公室（AMMTO）资助的能源创新中心，合同编号为 DEAC52–07NA27344（LLNL-JRNL-857236）。N.L.K 获得艾姆斯国家实验室 LDRD 资助，负责设计用于绑定关键矿产金属的肽，

该资助授予 R.C.。作者声明不存在任何竞争性经济利益。

可获取支持信息

所有搜索的设置、软件和硬件；计算测试的详细信息（单点计算、受控分子动力学、优化器循环次数上限）；每次重复搜索；在其他硬件上每个元动力学步骤所需的时间；其他 CREST 运行类型；CREST 的新循环步骤和 FastCREST 的选择加入式修复；锌结合测试肽的搜索；对第二种肽的 Ga(III) 和 Fe(II) 配合物进行的搜索（PDF）。

参考文献

(1) Braun, R.; Bachmann, S.; Schönberger, N.; Matys, S.; Lederer, F.; Pollmann, K. 肽作为生物吸附剂——资源回收的有前景工具。《微生物学研究》**2018年**, *169*, 649–658.

(2) Schönberger, N.; Zeitler, C.; Braun,R.; Lederer, F. L.; Matys, S.; Pollmann, K. 镓结合噬菌体克隆的定向进化与工程改造——一项初步研究。《仿生学》**2019年**, *4*, 35.

(3) Schönberger, N.; Taylor, C.; Schrader,M.; Drobot, B.; Matys, S.; Lederer, F. L.; Pollmann, K. 镓结合肽作为可持续处理工业废物流的工具。《危险材料杂志》**2021年**, *414*, 125366.

(4) Grimme, S.; Bohle, F.; Hansen, A.; Pracht, P.; Spicher, S.; Stahn, M. 高效计算非刚性分子的结构系综和自由能的量子化学计算。《物理化学杂志*A*辑》**2021年**, *125*, 4039–4054.

(5) Pracht, P.; Bohle, F.; Grimme, S. 使用快速量子化学方法自动探索低能化学空间。《物理化学与化学物理》**2020年**，*22*，7169–7192。

(6) Pracht, P.; Grimme, S.; Bannwarth, C.; Bohle, F.; Ehlert, S.; Feldmann, G.; Gorges, J.; Müller, M.; Neudecker, T.; Plett, C.; Spicher,S.; Steinbach, P.; Wesołowski, P. A.; Zeller, F. CREST——一种用于探索低能分子化学空间的程序。《化学物理学杂志》**2024年**，*160*，114110。

(7) Bannwarth, C.; Ehlert, S.; Grimme, S. GFN2-xTB——一种具有多极静电学和密度依赖色散贡献的准确且参数化范围广泛的自洽紧束缚量子化学方法。*J.Chem. Theory Comput.* **2019年**，*15*，1652–1671。

(8) Grimme, S. 基于紧束缚量子化学计算、利用元动力学模拟探索化合物、构象体和反应空间。*J.Chem. The-ory Comput.* **2019年**，*15*，2847–2862。

(9) Laio, A.; Parrinello, M. 逃离自由能极小值。*Proc. Natl. Acad. Sci. U.S.A.* **2002**，*99*，12562–12566。

(10) Wu, X.; Koslowski, A.; Thiel, W. 在混合多核 CPU–GPU 计算平台上加速的半经验量子化学计算。*J. Chem.Theory Comput.***2012**，*8*，2272–2281。

(11) Maia, J. D. C.; Urquiza Carvalho, G. A.; Mangueira, C. P.; Santana, S. R.; Cabral, L. A. F.; Rocha, G. B. 用于加速 MOPAC 半经验量子化学计算的 GPU 线性代数库和 GPGPU 编程。*J. Chem. TheoryComput.* **2012年**,*8*, 3072–3081。

(12) Zhou, G.; Nebgen, B.; Lubbers, N.; Malone, W.; Niklasson, A. M. N.; Tretiak, S. 使用 PyTorch 加速的图形处理器半经验 Born–Oppenheimer 分子动力学。*J. Chem. TheoryComput.***2020年**,*16*, 4951–4962。

(13) Vuong, V.-Q.; Cevallos, C.; Hourahine, B.; Aradi, B.; Jakowski, J.; Irle, S.; Camacho, C. 利用图形处理器加速密度泛函紧束缚方法。《化学物理学杂志》**2023年**，*158*，084802。

(14) Friede, M.; Hölzer, C.; Ehlert, S.; Grimme, S. dxtb——一种高效且完全可微的扩展紧束缚框架。《化学物理学杂志》**2024年**，*161*，062501。

(15) Ehlert, S.; Stahn, M.; Spicher, S.; Grimme, S. 适用于快速半经验方法的稳健高效隐式溶剂化模型。*J.Chem.TheoryComput.* **2021年**, *17*, 4250–4261.

(16) CREST：构象体–旋转异构体系综采样工具，版本 3.0.2。2024年； `https://github.com/crest-lab/crest/releases/tag/v3.0.2`，于2024年8月25日发布；conda-forge 构建。

(17) CREST：构象体–旋转异构体系综采样工具，提交 cfdc301f。2026年；`https://github.com/crest-lab/crest/tree/cfdc301f759686b0fd66ced63b5ddbd6c693fa4f`，2026年6月16日的提交。

(18) Ehlert, S. tblite：轻量级紧束缚框架，版本 0.5.0。2025年； `https://github.com/tblite/tblite/releases/tag/v0.5.0`，提交 6f6cd7d。

(19) OpenMP应用程序接口，第5.2版。OpenMP架构评审委员会，2021年；`https://www.openmp.org/specifications/`，访问日期：2026年10月26日。

(20) cuSOLVER库，《CUDA工具包 12.4.1 文档》。英伟达公司，2024年；`https://docs.nvidia.com/cuda/archive/12.4.1/cusolver/index.html`，访问日期：2026年10月26日。

(21) cuBLAS库，《CUDA工具包 12.4.1 文档》。英伟达公司，2024年；`https://docs.nvidia.com/cuda/archive/12.4.1/cublas/index.html`，访问日期：2026年10月28日。

(22) Axilrod, B. M.; Teller, E. 三个原子之间的 van der Waals 型相互作用。*J.Chem.Phys.* **1943**，*11*，299–300。

(23) Muto, Y. 非极性分子之间的作用力。《日本数学物理学会志》**2019年**，*17*，629–631，日文。

(24) Caldeweyher, E.; Ehlert, S.; Hansen, A.; Neugebauer, H.; Spicher, S.; Bannwarth, C.; Grimme, S. 一种普遍适用的原子电荷依赖的伦敦色散校正。《化学物理学杂志》**2019年**，*150*，154122。

(25) Yoo, A. B.; Jette, M. A.; Grondona, M. 收录于并行处理的作业调度策略；Feitelson, D., Rudolph, L., Schwiegelshohn, U. 编；《计算机科学讲义》；施普林格：柏林、海德堡，2003；第2862卷；第44–60页。

(26) Demmel,J.; Nguyen,H.D. 并行可重现求和。*IEEETrans.Comput.***2015**, *64*, 2060–2070.

(27) Schuirmann,D. J. 评估平均生物利用度等效性时，双单侧检验程序与功效法的比较。《药代动力学与生物药剂学杂志》**1987**,*15*, 657–680.

(28) Hodges, J. L.; Lehmann, E. L. 基于秩检验的位置估计。*Ann. Math.Stat.***1963**, *34*, 598–611.

(29) Bauer, D. F. 使用秩统计量构造置信集。美国统计学会杂志**1972**，*67*，687–690。

(30) Ryckaert, J.-P.; Ciccotti, G.; Berendsen, H. J. C. 对带约束体系的笛卡尔运动方程进行数值积分：正构烷烃的分子动力学。计算物理学杂志 **1977**，*23*，327–341。

(31) Kabsch, W. 求解使两组向量相对应的最佳旋转。晶体学报 *A* **1976**，*32*，922–923。

(32) King, N. J.; LeBlanc, I. D.; Brown, A. CREST iMTD算法的一种变体，用于柔性分子非共价簇。《计算化学杂志》**2024年**，*45*，2431–2445。

(33) Moradi, S.; Tomann, R.; Head-Gordon, M.; Stein, C. J. 含过渡金属体系扩展紧束缚方法的拓展。《计算化学杂志》**2026年**，*47*，e70346。

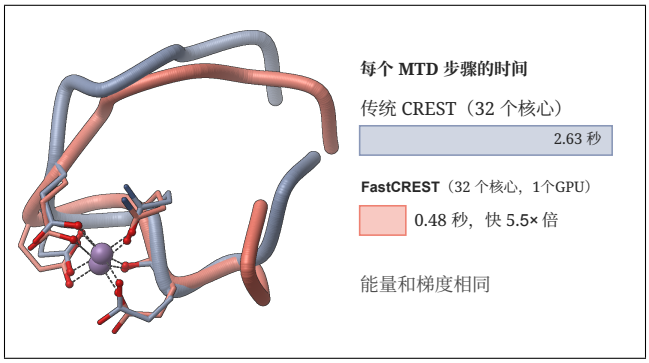
(34) Caldeweyher, E.; Ehlert, S.; Grimme, S. dftd4：普适的基于原子电荷的伦敦色散校正，版本 3.7.0。2024年； https://github.com/dftd4/dftd4/releases/tag/v3.7.0。

(35) Ehlert, S. simple-dftd3：DFT-D3方法的简单重新实现，版本 1.2.1。2024年；https://github.com/dftd3/simple-dftd3/releases/tag/v1.2.1。

(36) Ehlert, S. multicharge：用于原子部分电荷的电负性均衡模型，版本 0.3.0。2024年；https://github.com/grimme-lab/multicharge/releases/tag/v0.3.0。

(37) Ehlert, S. mctc-lib：模块化计算工具链库，版本 0.6.1。2026年； https://github.com/grimme-lab/mctc-lib/releases/tag/v0.6.1。

目录图


每个 MTD 步骤的时间
传统 CREST（32 个核心）
2.63 秒
FastCREST（32 个核心，1个GPU）
0.48 秒，快 5.5× 倍
能量和梯度相同

支持信息：

FastCREST：GPU 加速 CREST

采用一致的能量和梯度进行构象搜索

Supantha Dey,[†, ‡, ¶] Nilusha L Kariyawasam,[†, ¶] 和 Ratul Chowdhury[*, †]

†爱荷华州立大学化学与生物工程系，美国爱荷华州艾姆斯市 *50011*

‡爱荷华州立大学跨院系遗传学与基因组学项目，美国爱荷华州艾姆斯市 *50011*，美国

¶这些作者对本研究的贡献相同。

电子邮件：ratul@iastate.edu

S1 搜索设置、软件和硬件

测试体系（274 个原子，电荷为 $+4$，闭壳层）的所有 iMTD-GC 搜索均从同一结构开始，并使用 GFN2-xTB、ALPB 水和 `--quick`。FastCREST 和传统 CREST 搜索均以 `crest --input search.toml --newversion --gfn2 --chrg 4 --uhf 0 --alpb water --quick --ewin 5 --noreftopo -T 32` 启动；输入文件设置了运行类型（`imtd-gc`）、计算级别，以及 FastCREST 的 `tblite_solver = "cuda"`。CREST 3.0.2 搜索使用了与该发布版本等效的命令行。在其命令中，`--quick` 位于 `--ewin 10` 之后，并将能量窗口重置为 5 千卡/摩尔，因此

Ta表 S1：测试体系 iMTD-GC 搜索的有效搜索设置

item	设置
算法	iMTD-GC， `--quick`
MTD	每次迭代 6 条轨迹，每条 250 皮秒，温度为 300 开尔文；采用带推动作用的均方根偏差偏置强度分别为 0.548 和 0.274，宽度分别为 α = 1.2、0.6 和 0.3 玻尔$^{-2}$；偏置每 1 皮秒存储一次结构
积分	蛙跳算法，时间步为 5 fs，氢质量为 2 u，Berendsen 恒温器（500 fs），每 100 fs 保存一次快照
键约束	对具有 GFN0 Wiberg 键级 > 0.5 且键长为.....的化学键使用 SHAKE 在 1.2($r_i + r_j$) 范围内；金属–配体键从不受约束（272 个约束）
分子动力学阶段	最低能的一个或两个构象体，在 400 和 500 开尔文下各进行 125 皮秒
优化	元动力学快照采用粗略设置，然后采用常规设置；分子动力学快照采用常规设置；遗传先粗略交叉，再常规交叉；最终系综紧密
收敛（能量，梯度）	粗略 5×10^{-4} E_h, 1×10^{-2} E_h 玻尔$^{-1}$；常规 5×10^{-6}, 1×10^{-3}；紧密 1×10^{-6}, 8×10^{-4}
优化器	ANCOPT；循环次数上限分别为粗略、常规和严格阈值下每个原子 1、3 或 5 个循环，至少 200 次（CREST 3.0.2 搜索：200 次，发布版本最多可超过 39 次）； `hlow` 0.005；20 次微迭代
CREGEN	能量窗口为 5 千卡/摩尔（粗略级别下为其两倍）；均方根偏差为 0.125 Å；能量为 0.05 千卡/摩尔；转动常数为 1%；拓扑检查 on
遗传交叉	最多 2,500 个结构
SCC	能量 1×10^{-6} E_h，密度 2×10^{-5}

所有搜索的有效设置均相同（表 S1），但优化器循环上限（第 S2 节）除外。

CREST 3.0.2 版本是 conda-forge 构建版本，其中捆绑了 tblite 0.3.0 的一个分支。我们的构建版本（FastCREST，以及未应用我们补丁的 CREST）使用了 CREST 提交 cfdc301f、tblite 0.5.0、gfortran 11.5（`-O3`，不含 `-march`，因此不使用融合乘加指令）、OpenBLAS 0.3.27 和 CUDA 12.4.1。在 `src/` 下的 200 个文件中，有 151 个在两个 CREST 代码树中相同，其中包括分子动力学和元动力学代码、CREGEN，以及除包含循环次数上限的文件之外的所有优化器文件。

所有作业均通过 Slurm 在爱荷华州立大学的 Nova 集群上运行。GPU 节点配备 A100（40 GB 和 80 GB）、H200、V100、L40S 或 RTX PRO 6000 GPU，搭载 AMD EPYC 7413、7502、7513、7543 或 9645、AMD Zen 5 或 Intel Xeon 主机。仅 CPU 搜索在 EPYC 7502、Zen 4（EPYC 9654）和 Zen 5（EPYC 9645）节点上运行；作为 Slurm 数组作业运行的优化任务则分配到调度器指定的 CPU 节点。

S2 计算测试

构型。测试比较了九种配置：CREST 3.0.2 版本；使用命令行输入和输入文件的传统 CREST；仅使用 CPU 求解器的 FastCREST，以及启用暖启动 Jacobi 算法特征值求解器的 FastCREST；还有在图形处理器上运行的 FastCREST，包括单独运行、搭配雅可比特征值求解器、搭配 GPU驻留SCC，以及同时启用这两项选择启用设置的情况。每项图形处理器计算都必须证明使用了图形处理器（特征值求解和三体项在图形处理器上运行，且没有 CPU 回退机制）；所有计算均满足这一要求。

搜索设置。 八种构型重放了搜索过程，直到首次元动力学。所有构型都做出了与 CREST 3.0.2 搜索相同的决策：相同的预优化结构（其第一次和最后一次优化步骤的能量分别为 -415.2625315087 和 $-415.2625672259\ E_\mathrm{h}$，精确到 $1\times10^{-10}\ E_\mathrm{h}$），柔性度量为 0.391，进行六条 250 皮秒的元动力学轨迹，偏置设置相同，没有失败的试验性元动力学，274 个 Wiberg 键级高于 0.5 的化学键，以及 272 个 SHAKE 约束。

单点计算。 对 306 个结构计算了能量和梯度：两个输入几何结构、CREST 3.0.2 搜索的 135 个最终构象和 50 个分子动力学快照、其首次优化的两个步骤，以及 FastCREST 搜索的 50 个分子动力学阶段输入和 67 个构象。判据为 $|\Delta E|\le10^{-8}\ E_\mathrm{h}$，最大梯度分量差异为 $\le10^{-6}\ E_\mathrm{h}$ 玻尔$^{-1}$；每个结构在每次比较中都满足该判据（表 S2）。我们未使用图形处理器的构建版本得到了按位相同的结果；这些版本与发布版本之间最高达 $10^{-10}\ E_\mathrm{h}$ 的差异源于构建版本不同（tblite 版本、编译器和编译选项）。脱辅基肽以及一种相关肽的 Co、Cu 和镍配合物（241 或 242 个原子；这些配合物为开壳层，含有一个或两个未成对电子）的结果也一致：对于每种结构的十个以上构型，发布版本、在中央处理器上运行的 FastCREST 和在图形处理器上运行的 FastCREST 之间的差异分别在 $1.1\times10^{-10}\ E_\mathrm{h}$ 和 $7.4\times10^{-11}\ E_\mathrm{h}$ 玻尔$^{-1}$以内。

受控分子动力学。 轨迹无法逐位比较，因此我们比较了对搜索而言真正重要的指标：分子动力学快照改变其成键拓扑的频率，该频率决定 CREGEN 保留多少个结构。对于从四个结构开始的每种配置，我们分别在 400 和 500 K 下进行了 20 ps 的分子动力学

表 S2：306 个结构的单点计算

比较	逐位相同	最大 $\lvert\Delta E\rvert$ (E_h)	最大 $\lvert\Delta g\rvert$ (E_h 玻尔$^{-1}$)
传统 CREST 与发布版本对比	0	1.01×10^{-10}	8.16×10^{-11}
输入文件与命令行对比（传统 CREST）	306	0	0
FastCREST，CPU 求解器与传统 CREST	306	0	0
FastCREST +雅可比，CPU 求解器与 FastCREST，CPU 求解器	306	0	0
FastCREST，图形处理器与发布版本对比	0	1.01×10^{-10}	8.16×10^{-11}
FastCREST +雅可比，图形处理器与发布版本对比	0	1.00×10^{-10}	8.16×10^{-11}
FastCREST + GPU驻留SCC与发布版本对比	0	1.00×10^{-10}	8.16×10^{-11}
两个选择启用设置与发布版本对比	0	1.00×10^{-10}	8.16×10^{-11}
FastCREST，图形处理器与中央处理器求解器	0	3.98×10^{-13}	4.0×10^{-15}
FastCREST +雅可比，图形处理器与中央处理器求解器	0	3.41×10^{-13}	3.3×10^{-14}
FastCREST + GPU驻留SCC 与中央处理器求解器对比	0	4.55×10^{-13}	4.0×10^{-15}
两个选择启用设置与中央处理器求解器对比	0	4.55×10^{-13}	6.0×10^{-15}

计算，每种配置均使用相同的初始速度（取自 CREST 重启文件）：两个结构各使用三组初始速度，另外两个较晚循环中的结构各使用八组初始速度；每次比较包含 44 对轨迹（仅运行前三组时为 24 对）。每隔十个快照使用 200 次循环上限进行优化。对于每一对轨迹，我们计算了优化前后拓扑不同于输入拓扑的快照比例之差 d。如果均值的 90% 置信区间 d 落在 ± 0.15 以内，则比较通过（双单侧检验；边界在运行前预先固定）。全部 13 次比较均通过（表 S3 ）。在每次与发布版本的比较中，配对轨迹均在中位数 4.7 至 4.8 ps 后发生分离（全原子均方根偏差超过 0.5 Å）；当同一构建运行两次时，则在 6.5 ps 后发生分离，因此没有配置过早出现发散。

优化器循环上限。CREST 提交 9908b42（介于 3.0.2 版本和 cfdc301f 之间）会在未收敛的 ANCOPT 优化达到循环次数上限时立即停止；发布版本则会让它完成最后一个弛豫区块，因此当上限为 200 个循环时，最多会运行到 239 个循环。未收敛优化会被丢弃，因此较早停止只会减少结构数量。我们在常规级别下，对两组各 100 个分子动力学快照进行了优化（表 S4）。较早停止使每 100 个快照中多出 2 到 3 个未收敛项，超过了我们设定的每 100 个最多 1 个的可接受标准；循环次数上限设为 240 的运行

表 S3：受控分子动力学：拓扑变化比例均值差的 90%置信区间

比较	对	分子动力学快照	优化后
传统 CREST 运行两次	24	-0.024 到 $+0.017$	-0.084 到 $+0.041$
传统 CREST 与发布版本	44	-0.015 到 $+0.017$	-0.039 到 $+0.030$
输入文件与命令行	24	-0.005 到 $+0.034$	-0.030 到 $+0.110$
FastCREST ，中央处理器求解器与传统 CREST 的比较	24	-0.043 到 $+0.005$	-0.137 到 -0.007
FastCREST ，图形处理器与中央处理器求解器	24	-0.000 至 $+0.050$	$+0.008$ 至 $+0.134$
FastCREST ，图形处理器与传统 CREST	44	-0.001 至 $+0.030$	$+0.005$ 至 $+0.072$
FastCREST ，图形处理器与发布版本	44	-0.003 至 $+0.033$	-0.014 至 $+0.082$
FastCREST + 雅可比 vs FastCREST , GPU	44	-0.033 到 $+0.010$	-0.076 到 $+0.025$
FastCREST + 雅可比 vs 发布版本	44	-0.020 到 $+0.026$	-0.048 到 $+0.064$
FastCREST + GPU驻留SCC 与 FastCREST , GPU	44	-0.024 至 $+0.016$	-0.065 至 $+0.040$
两个选择启用设置 与 FastCREST +雅可比	44	-0.021 至 $+0.021$	-0.059 至 $+0.040$
FastCREST + GPU驻留SCC 与发布版本	44	-0.005 至 $+0.027$	-0.029 至 $+0.072$
两个选择启用设置与发布版本	44	-0.013 至 $+0.020$	-0.044 至 $+0.042$

表 S4：在 200 次循环限制下收敛的优化，占 100 个优化中的

快照	发布版本	我们的构建版本	差异	上界
CREST 3.0.2 搜索的分子动力学阶段，第二个循环	95	92	3	8
FastCREST 搜索的分子动力学阶段，第二个循环	93	91	2	4

将差异的上界设为每 100 个 4 到 8 个（表 S4）。在两个版本中都收敛的结构，通过或未通过 CREST 的拓扑检查的情况完全相同；其能量差异不超过$1.5 \times 10^{-3}\ E_\mathrm{h}$，这与不同构建版本中的优化结果一致。我们保留了该提交。

本测试在两个版本中均使用了 200 个循环的上限，这与 CREST 在其搜索中的做法一致。当未设置上限时，ANCOPT 根据原子数推导该上限（表 S1），所采用的规则与 xtb 程序的优化器相同；在版本 3 之前，CREST 的优化由该优化器执行。然而，在每个 CREST 3 版本（3.0、3.0.1、3.0.2 以及当前代码）中，将优化设置写入输出的例程还会把上限设为 200，以用于搜索的其余部分。FastCREST 版本的该例程不改变任何设置，因此每次优化都会保留这个依赖规模的上限，而在 200 个循环后仍未收敛的结构仍可能收敛。FastCREST 和传统 CREST 的每次重复搜索均使用了依赖规模的上限。

S3 重复搜索

在看到部分个体结果后、计算任何组统计量之前，比较方案即已确定。比较的量是首次 MTD 迭代的粗略优化之后的最低能量（14,994 个快照；每次搜索均使用相同的 CPU 优化器），该数值取自所有达到该阶段的搜索，无论搜索是否完成，因此不会偏向于需要较少循环、从而更早完成的搜索。比较的组为图形处理器上的 FastCREST 和传统 CREST，即仅在 CPU 上运行的我们构建版本；后者的单点计算结果与未应用我们补丁构建的 CREST 按位相同。统计量为 Hodges–Lehmann 位移（FastCREST 减去传统 CREST），并采用精确的 Mann–Whitney 置信区间。如果 95% 区间不包含零，则判定存在差异；如果 90% 区间位于 ± 2 千卡/摩尔以内（小于 5 千卡/摩尔能量窗口的一半），则表明等效性；否则结果为不确定。

40 次 FastCREST 搜索中有 16 次、18 次传统 CREST 搜索中有 17 次在该点具有相应数值（表 S5）；第 18 次传统搜索通过了该点，但未保留其数值。其余 FastCREST 搜索中的大多数在一块 GPU 上运行数秒后因内存错误停止，且未重新启动。达到该点的 16 次搜索分别在 9 个 L40S、3 个 V100、3 个 A100 和 1 个 H200 GPU 上运行了第一次迭代；其中 13 次在该点之前从检查点恢复，而 17 次传统搜索中只有 7 次如此。偏移为 $+4.09$ 千卡/摩尔（95% 区间为 -4.94 至 $+11.75$，达到的置信水平为 0.951；90% 区间为 -3.58 至 $+10.13$；双侧 $p = 0.35$）：未检测到差异，也未证明等效性。仅考虑在 A100 或 H200 GPU 上运行的 4 次 FastCREST 搜索时，偏移为 $+3.12$（-14.50 至 $+19.80$）；而仅与使用 FastCREST 自有构建在中央处理器上运行的 9 次传统搜索比较时，偏移为 $+2.75$（-6.44 至 $+13.89$）。已完成搜索的最终能量（FastCREST 10 次、传统搜索 15 次）显示偏移为 -2.70 千卡/摩尔（95% 区间为 -11.27 至 $+6.54$）；这一比较是在看到数据后选定的。

表 S5：测试体系的重复搜索。相对于这些搜索中的最低能最终构象的能量，单位为千卡/摩尔（常规 7，$-415.48139\ E_{\mathrm{h}}$）

搜索	GPU	首次 MTD 迭代	最终构象		新循环
常规 1	CPU	+13.1	+16.7	19	1
常规 2	CPU	+16.8	+16.4	8	1
传统 3	CPU	+21.1	+25.2	3	2
传统 4	CPU	+22.3	未完成		
传统 5	CPU	+24.5	+4.5	7	3
传统 6	CPU	+26.4	未完成		
常规 7	CPU	+29.4	+0.0	3	1
传统 8	CPU	+29.5	+7.6	2	2
传统型 9	CPU	+32.1	+20.2	12	1
传统型 10	CPU	+32.7	+16.7	1	1
传统型 11	CPU	+34.0	+15.5	12	2
传统型 12	CPU	+34.7	+11.7	34	2
传统 13	CPU	+35.0	未完成		
传统 14	CPU	+38.0	+124.7	11	0
传统 15	CPU	+41.1	+116.0	10	0
传统 16	CPU	+53.3	+3.7	82	2
传统 17	CPU	+54.7	+20.0	1	3
传统 18	CPU	–	+12.1	9	1
FastCREST 1	H200	+17.2	尚未完成		
FastCREST 2	L40S	+18.2	+3.6	22	2
FastCREST 3	L40S	+22.5	尚未完成		
FastCREST 4	L40S	+23.2	+1.9	51	1
FastCREST 5	A100	+26.6	+8.7	3	2
FastCREST 6	L40S	+31.4	+32.8	40	2
FastCREST 7	L40S	+34.0	尚未完成		
FastCREST 8	L40S	+34.7	+14.1	1	2
FastCREST 9	L40S	+36.2	+30.0	5	1
FastCREST 10	V100	+37.1	未完成		
FastCREST 11	V100	+40.9	未完成		
FastCREST 12	A100	+42.1	+9.2	20	3
FastCREST 13	V100	+42.2	未完成		
FastCREST 14	L40S	+47.7	+8.9	173	1
FastCREST 15	A100	+49.5	+14.2	103	2
FastCREST 16	L40S	+50.8	+17.1	19	1
FastCREST + 雅可比	A100	+37.9	+33.1	10	2

首次 MTD 迭代：粗略优化后的最低能量（比较的量）。最终：最终系综的最低能量。新循环：在分子动力学阶段或遗传交叉找到更低能构象后启动的循环。图形处理器：首次迭代所用的图形处理器；传统 CREST 在 CPU 上运行。常规 18 没有首次迭代的数值：其首次迭代从检查点重新运行，且未保留其粗略优化结果。

表 S6：不同硬件上测试体系每个 MTD 步骤所需的时间

代码和硬件	线程	每步耗时（秒）	快于……倍 CREST 3.0.2 搜索
CREST 3.0.2 搜索，节点未记录	32	2.8	1.0×
使用该线程设置的传统 CREST，EPYC 7502	32	1.43–1.51	1.9×
same, 整个 EPYC 7502 节点	64	1.35–1.37	2.1×
不使用图形处理器的 FastCREST，EPYC 7502	64	1.26–1.27	2.2×
不使用图形处理器的 FastCREST，Zen 4（共享节点）	64	0.93	3.0×
不使用图形处理器的 FastCREST，Zen 5（共享节点）	64	0.74	3.8×
FastCREST，V100 和较旧的 Intel Xeon[a]	8	1.86	1.5×
FastCREST，RTX PRO 6000 和 EPYC 9645[a]	32	0.68	4.1×
FastCREST，A100 和 EPYC 7543	32	0.46–0.49	5.7–6.0×
FastCREST，H200 和 Zen 5	32	0.39	7.1×

范围：两次运行或两个节点。各行使用的主机不同，因此这些倍数综合反映了代码和硬件的影响；在相同的 A100 节点上，FastCREST 的速度比传统 CREST 快 5.5 倍（正文，图 2b）。[a]此架构的构建版本中没有 GPU 代码：三体项由中央处理器运行。V100：一条部分轨迹。

S4 其他硬件上的元动力学步骤耗时

表 S6 列出了在其他节点上进行搜索和基准测试时测得的测试体系每个元动力学步骤的耗时。主机中央处理器与图形处理器同样重要，因为每个步骤的大部分运算仍由中央处理器完成。

S5 其他 CREST 运行类型

CREST 的其他运行类型使用相同的轨迹和优化代码，因此 FastCREST 也会运行这些类型。我们将每种类型与使用相同提交构建、但未应用我们补丁的 CREST 进行了比较；后者会在 200 个循环后停止优化（第 S2 节；表 S7）。质子化、去质子化和互变异构体搜索不包含分子动力学，并且得到的结构和能量相同。NCI-MTD 和熵运行具有自适应性，因此两种代码的运行结果都会有所分散；对于重复运行的情况，其结果在这种分散范围内一致。紫杉醇的熵运行每种代码各运行一次。测试体系的 NCI-MTD 搜索分别使用 FastCREST 运行了一次，并使用 CREST 3.0.2 版本运行了一次（最后一行 NCI-MTD）。在 NCI-MTD 的壁势作用下，FastCREST 会将优化运行到

表 S7：FastCREST 和传统 CREST 的其他运行类型

运行类型	分子	结果
质子化，去质子化，互变异构体	尿嘧啶、咖啡因、鸟嘌呤（气相）	每次运行中相同的最低能量质子化异构体和互变异构体（最快模式：最大 $\lvert\Delta E\rvert$ 6.4×10^{-6} E_h）
质子化	Ga(III)–肽（132 个原子，$+3$，ALPB 水）	最低质子化异构体在所有运行中均相同；保留的结构数量因运行而异，从一次运行到另一次运行，在传统 CREST 本身中（53、69 和 60）以及 FastCREST 中（74 和 58）
NCI-MTD	水三聚体（气相）	九次运行均得到相同的最低能量（-15.235025 E_h）
NCI-MTD	Ga(III)–肽（132 原子，ALPB 水，`--quick`）	六次运行中最低的能量在 1.8 千卡/摩尔范围内；传统 CREST 耗时 59.6 和 217.7 分钟，FastCREST 18.1 和 29.0 分钟（在两种情况下分别快 3.3 倍和 7.5 倍运行对，不同节点），最快模式为 16.0，且 14.6 分钟（速度分别快 3.7 倍和 14.9 倍）
NCI-MTD	测试体系（ALPB 水）	从相同结构各进行一次搜索：CREST 3.0.2 版本耗时 93.7 小时（32 个线程，EPYC 7502 节点），FastCREST 用时 34.2 小时（A100 GPU，EPYC 7513 主机；快 2.7 倍，进行了三次 MTD 迭代，而另一方进行了两次）；最低能量相差 5.0 千卡/摩尔（FastCREST 的能量较高）；
熵	n-辛烷（气相）	FastCREST 得到 1 个构象体，发布版本得到 7 个构象熵 8.12 cal mol^{-1} K^{-1}（FastCREST）、8.13（最快模式）、8.20、8.20 和 8.32（传统 CREST）；对于这个 26 原子分子 FastCREST 用时 14.8 分钟，而传统 CREST 用时 7.3 分钟（慢 2.0 倍）
熵	紫杉醇（113 个原子，ALPB 水，`--quick`）	构象熵 9.65（传统 CREST，63.3 小时），9.89（FastCREST，46.5 小时，快 1.4 倍）以及 9.37 cal mol^{-1} 开尔文$^{-1}$（最快模式，21.6 小时，快 2.9 倍更快）；不同节点

主任务而非数组作业，因此其收益小于 iMTD-GC 搜索中的收益。

S6 CREST 的新循环步骤与 FastCREST 的选择加入式修复

当 CREST 的 iMTD-GC 搜索找到一个更低能构象时，其代码中的两个部分（CREST 3.0.2 和 cfdc301f；即命令行运行的搜索例程）并未按照代码设计的意图执行。首先，负责登记新的最低能量构象的例程会在调用方保存其中最新的 `crest_rotamers` file 以供下一次合并之前，删除所有 `crest_rotamers` file，因此当分子动力学阶段或遗传

遗传交叉会启动一个新循环，此前找到的系综会丢失，后续循环也只会合并该循环自身的 MTD 迭代结果。其次，新的最低能量构象会写入文件 `coord`，作为下一批 MTD 轨迹的起点，但该例程之后不会再读取它：每条 MTD 轨迹都从优化后的输入结构开始。不过，CREGEN 的拓扑检查会与 `coord` 进行比较。

在测试体系的 26 次已完成重复搜索中，传统 CREST 和 FastCREST 都有 24 次在分子动力学阶段或遗传交叉后至少启动了一个新循环（总计 41 个新循环）（表 S5）；在 MTD 迭代过程中找到更低能构象后，下一批 MTD 轨迹有 60 次从输入结构重新开始。由于两种代码都有这种行为，因此这不会影响二者之间的比较，但这解释了为什么最终系综的能量可能远高于搜索过程中发现的最低能构象。

FastCREST 1.3.0 可按需修复这两个步骤。设置 `CREST_KEEP_ENSEMBLE=1` 后，最新系综会在删除前复制，并纳入之后的每次合并；设置 `CREST_RESTART_FROM_BEST=1` 后，下一批 MTD 轨迹会从新的最低能量构象开始。两者都未设置时，FastCREST 的计算行为与之前相同：在被迫进入第二个循环的搜索中，构象和搜索日志与前一版本完全相同，单点计算的字节也完全一致。两者都设置后，被迫进行的搜索会从新的最低能量构象重新开始，保留之前的系综，并在每次于 33 个检查点步骤中的任意一步被终止后正常恢复。

S7 锌结合测试肽

在 PyMOL 中构建了三个具有随机序列的 12 残基 α-螺旋（不含 Cys 或 Pro，含一个 His，且至少含一个 Asp 或 Glu）：DEGLAHSMAAFK（p1）、GINGQALHAEYW（p2）和 LKERTTEIHENI（p3）；其末端为中性，Asp 和 Glu 去质子化，Lys 和 Arg 质子化。在距 His NE2 2.05 Å 处放置了一个 Zn^{2+} 离子（电荷 $+1$；174 至 211 个原子）。p1 还以 Asp 和 Glu 质子化的形式构建（低 pH，电荷 $+3$）。每个体系分别使用传统 CREST（3.0.2 版本，32 个 CPU 核心）、FastCREST 1.2.0，以及启用两项选择加入式修复的 FastCREST 1.3.0（S6 节）搜索一次；FastCREST 按正文所述运行，使用一个

A100 或 H200 图形处理器。所有搜索均使用 GFN2-xTB、ALPB 水、 `--quick` 和 `--noreftopo`；FastCREST 使用依赖尺寸的循环上限（表 S1），传统 CREST 使用 200 个循环的上限。Zn 被排除在 CREGEN 的拓扑检查之外（`--notopo` 及其原子编号），因为该检查会将金属–配体接触计为化学键，并删除 Zn 更换配体的所有结构：采用默认检查时，对 p3 进行的 FastCREST 搜索最终比表 S8 中的最低能结构高出 119 千卡/摩尔，而 Zn 仍与 His 配位。

在其他所有搜索到结构的搜索中，肽都环绕着锌（图 S1、表 S8）。螺旋解旋，锌变为六配位（其中一个结构为七配位），其首个和最后一个供体残基之间相隔 5 至 9 个位置，回转半径从 6.6–6.8 Å 降至 5.4–6.0 Å。在中性 pH 下，供体为组氨酸、天冬氨酸和谷氨酸羧酸盐，以及主链羰基；在 p2 中还包括 N 端氨基和天冬酰胺3 侧链；在 p3 中，羧酸盐占据五个或六个位点，而组氨酸则被释放。在低 pH 下，组氨酸与四个或五个主链羰基氧结合。

FastCREST 1.2.0 在 13 至 34 h 内完成了所有体系（采用两项修复后为 42 至 79 h，这两项修复会增加循环次数）。传统 CREST 仅完成了低 pH p1 的搜索，耗时 97 h（FastCREST 快 7.3 倍）；对 p2 和 p3 的搜索在 6 天后仍未完成（FastCREST 至少快 10 倍和 4.2 倍）。在中性 p1 中，输入结构的初始优化在 N 端形成了输入结构中不存在的两个化学键。CREGEN 会将结构与输入结构进行比较，并移除 MTD 轨迹中的全部或几乎全部结构；这些轨迹从优化后的输入结构开始，因此有效结构来自分子动力学阶段；采用重启修复后，后续循环的 MTD 轨迹从一个有效结构开始。停止传统 CREST 对 p1 的搜索时，其已完成两次 MTD 迭代，但尚未获得有效结构，耗时 53 h。已完成的传统搜索通过 S6 节的新循环步骤，最终结果比其找到的最低能结构高出 9.1 千卡/摩尔。采用两项修复后，FastCREST 为 p1、p2 和 p3 找到了所有搜索中最低能的最终结构。每种体系均由每种代码搜索一次；这些体系的搜索未测量不同运行之间的离散程度（S3 节）。

在两种代码都生成结构的情况下，它们与锌的结合所用的供体基本相同（图 S2）。将锌位点叠合后，两种代码得到的最低能结构相距 2.2 至 2.8 Å

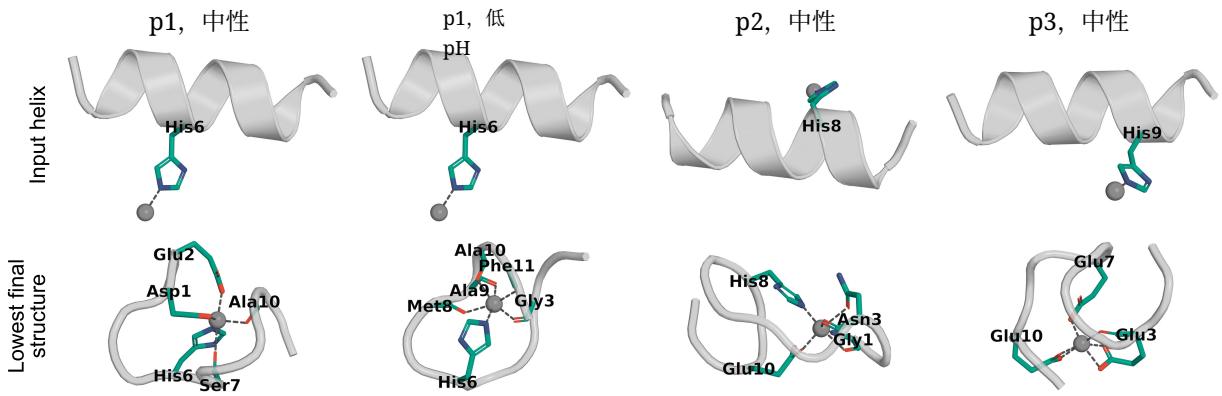


图 S1：每种肽的输入螺旋（上）和最低能最终结构（下；FastCREST 使用两项修正），均采用相同比例。灰色球体：锌；虚线：距离在 2.5 Å 以内的供体；绿色棒状结构：结合基团。使用开源 PyMOL 渲染。

（对锌、结合残基的主链及其供体基团计算的均方根偏差），且共享五个或六个供体；对每种肽进行的两次 FastCREST 搜索，在相同原子上的结果彼此相差 1.7 至 3.3 Å。链端的差异更大（Cα RMSD 为 3.8 至 4.1 Å）。

S8 第二个 Ga(III)–肽复合物

肽 MLTREQIEELSKLLE 与测试体系有两个残基不同（Leu13 和 Glu15，而非 Lys）。其 Ga(III) 复合物含有 264 个原子，电荷为 $+1$，且为闭壳层；末端为中性，四个谷氨酸已去质子化，Arg4 和 Lys12 已质子化（中性 pH）。在预优化的螺旋输入结构中，Ga 与六个氧原子结合（表 S9）。使用 GFN2-xTB、ALPB 水、 `--quick`和 `--noreftopo`，并采用与测试体系相同的设置（最终能量窗口为 5 千卡/摩尔），分别用两种代码从该输入结构进行了单次搜索：传统 CREST（CREST 3.0.2）在 AMD Zen 5 节点上使用 32 个线程运行；FastCREST 1.1.1 则在配备 AMD EPYC 7543 主机的单块 A100 GPU 上运行，其优化任务作为 1,583 个 Slurm 数组任务在其他节点上执行（分配的中央处理器时间为 6,060 小时）。两种代码还采用了不同的优化器循环次数上限（第 S2 节）：CREST 3.0.2 为 200 次循环；FastCREST 在粗略、常规和严格阈值下分别为 264、792 和 1,320 次循环。在两个 MD 阶段中，CREST 3.0.2 的 5,061 次和 5,028 次优化分别有 3,013 次和 3,220 次收敛，而 FastCREST 的所有 MD 阶段优化均收敛。FastCREST 耗时 58.4 小时，传统 CREST 耗时 265.8 小时，

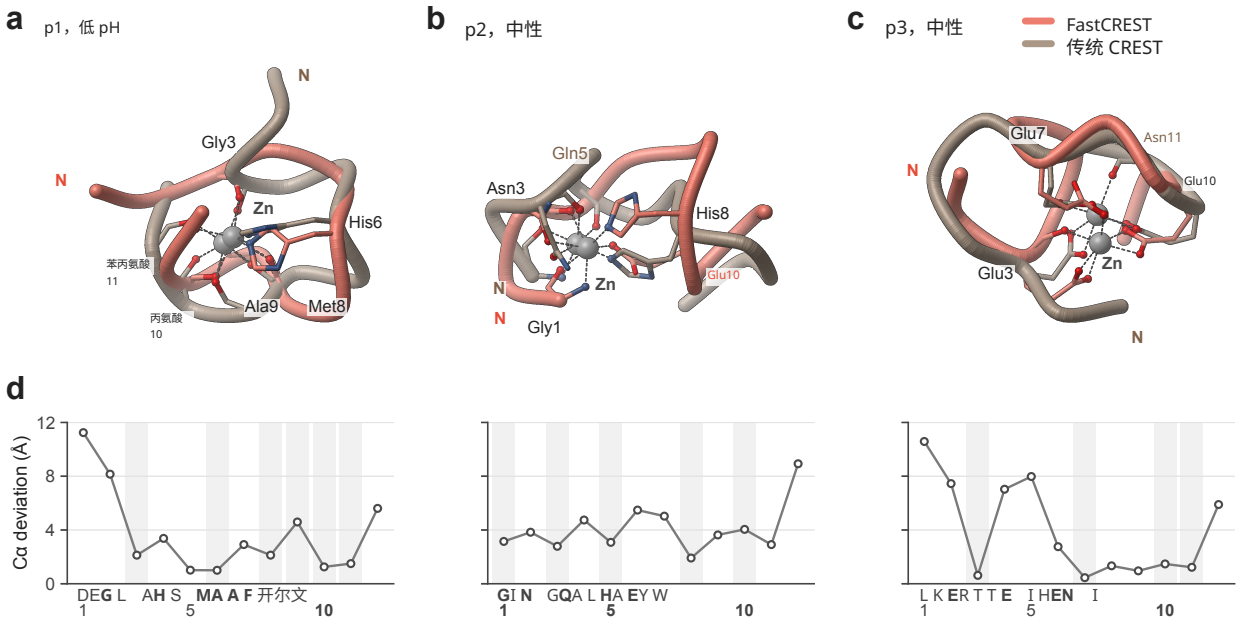


图 S2：FastCREST 采用两项修正得到的最低能结构（红色）与传统 CREST 得到的最低能结构（棕色），叠合于锌位点（锌、锌结合残基的主链及其侧链供体基团）：(a) 低 pH 的 p1、(b) p2 和 (c) p3。虚线连接锌（灰色）与其供体；红色或棕色标出的残基名称表示该残基仅在一种结构中与锌结合。(d) 叠合后每个残基的 Cα 偏差；阴影标出的残基在任一结构中均与锌结合。传统 CREST：对于低 pH 的 p1，显示其搜索找到的最低能结构，但该结构在新循环中被从最终结果中剔除；对于 p2 和 p3，显示在 6天时限之前找到的最低能结构。未显示中性 p1，因为传统 CREST 未能得到结构。

表 S8：锌结合测试肽的搜索。相对于该体系任意搜索找到的最低能结构的能量（千卡/摩尔）

体系	搜索	时间（小时）	能量	锌供体（配位数）	R_g (Å)
p1	FastCREST	32.6	+2.5	组氨酸6；谷氨酸2（2 个氧）；甲硫氨酸8 的主链氧，丙氨酸9、丙氨酸10（6）	5.42
	FastCREST +两项修复	41.8	0.0	组氨酸6；天冬氨酸1（2 个氧）；谷氨酸2；主链氧丝氨酸7、丙氨酸10（6）	5.43
	传统 CREST[a]	53		无有效结构	
p1，低 pH	FastCREST	13.3	+7.1	组氨酸6；甘氨酸3、亮氨酸4、丙氨酸9的主链氧，丙氨酸10、苯丙氨酸11 (6)	5.55
	FastCREST +两种修正	43.6	+3.3	组氨酸6；甘氨酸3、甲硫氨酸8、丙氨酸9的主链氧，丙氨酸10、苯丙氨酸11 (6)	5.50
	传统 CREST	96.9	+9.1	组氨酸6；甲硫氨酸8、丙氨酸9的主链氧，丙氨酸10、苯丙氨酸11；甲硫氨酸8 S (6)	5.99
	找到的最低能构象[b]		0.0	组氨酸6；甘氨酸3、甲硫氨酸8、丙氨酸9的主链氧，丙氨酸10、苯丙氨酸11（6）	5.46
p2	FastCREST	14.3	+2.8	组氨酸8；谷氨酸10（2 个氧）；天冬酰胺3 OD1；甘氨酸1、谷氨酰胺5的主链氧（6）	5.72
	FastCREST +两项修复	41.5	0.0	组氨酸8；谷氨酸10；甘氨酸1 N 和 O；天冬酰胺3 O 以及 OD1（6）	5.96
	传统 CREST[c]	>144	+1.6	组氨酸8；甘氨酸1 N 和 O；天冬酰胺3 O 和 OD1；谷氨酰胺5 O（6）	5.79
p3	FastCREST	34.4	+6.2	谷氨酸3（2 个 O）；谷氨酸7；谷氨酸10（2 个 O）；天冬酰胺11 O（6）	6.02
	FastCREST +两项修正	79.1	0.0	谷氨酸3（2 个 O 和主链氧）；谷氨酸7；谷氨酸10（2 个 O）（6）	5.62
	传统 CREST[c]	>144	+2.2	Glu3、Glu7、Glu10（各含 2 个氧）；Asn11 氧（7）	5.73

FastCREST：1.2.0版本；应用两项修正后：1.3.0版本。时间：主任务的耗时，为搜索运行的各项作业耗时之和。能量、供体（距锌 2.5 Å 以内；配位数）以及肽重原子的回转半径 R_g：最低能的最终结构；输入的回转半径 R_g：6.68（p1）、6.62（p2）和 6.80 Å（p3）。[a]在两次元动力学迭代后由我们停止。[b]传统搜索找到的最低能结构；新循环将其从最终结果中剔除。[c]运行 6 天后仍未完成：找到的最低能结构。

因此，尽管使用的硬件不同、计算工作量更大，FastCREST 仍快了 4.6 倍：它运行了三个循环（六次 MTD 迭代），而传统 CREST 运行了两个循环（四次）。按每次 MTD 迭代计算，FastCREST 耗时 7.2 小时，传统 CREST 耗时 35.2 小时（快了 4.9 倍）。两种搜索中都没有轨迹提前停止。

两种搜索结束时的结构都接近输入结构（图 S3a–c）。与测试体系一样，每个结构都保留了输入中的 Ga–O键，且两个最终系综中的每个构象体都保留了输入结构的九个螺旋残基，并与输入结构的距离在 1.5 Å 以内（Cα 均方根偏差）。两个最低能最终构象体在镓位点处重合（均方根偏差为 0.08 Å），而在所有 Cα 原子上相差 0.7 Å，差异主要位于 Met1；FastCREST 的构象体能量低 1.1 千卡/摩尔。两个最终系综存在重叠：传统 CREST 的 79 个构象体各自与 FastCREST 的 48 个构象体中的一个相距不超过 1.1 Å（Cα 均方根偏差）。两种搜索都曾找到能量更低的结构，但这些结构随后在新循环中被舍弃（S6 节）。传统 CREST 找到的结构比 FastCREST 的最低能最终构象体低 2.4 千卡/摩尔，且与其仅相差 0.2 Å（Cα 均方根偏差）。FastCREST 在第二个循环中找到的结构比该构象体低 30.7 千卡/摩尔，且具有相同的 Ga–O键，但螺旋已展开成紧凑结构（回转半径为 6.7 Å，输入结构为 7.5 Å）。FastCREST 的最终系综来自其第三个循环，该循环的轨迹再次从输入结构开始。由于每种代码只进行了一个搜索，因此未测量该复合物的运行间离散程度（S3 节）。

S9 一个开壳层 Fe(II) 配合物

对同一肽的高自旋 Fe(II)配合物（264 个原子、电荷为 0、含四个未成对电子，采用 S8 节中的质子化状态和中性末端）进行了搜索：使用 S8 节中的代码和设置，分别用每种代码从一个 Fe 与三个氧原子配位的预优化螺旋输入结构开始运行一次。传统 CREST 在 Intel Skylake 节点的 32 个线程上耗时 242.3 小时，而 FastCREST 在配备 AMD EPYC 7513 主机的单个 A100 GPU 上耗时 7.3 小时（63 个数组任务，222 个中央处理器小时）。两次搜索均正常结束，但两种代码中的大多数轨迹都提前停止：传统 CREST 的 50 条 MTD 和 MD 轨迹中有 48 条提前停止，FastCREST 的 20 条轨迹则全部提前停止；此外，MTD

迭代仅产生了完整迭代所含快照数量的 2% 到 29%。因此，两次搜索都未能按预期进行采样。这两个时间并不具有可比性：传统 CREST 运行了四个循环，而 FastCREST 运行了一个循环，且两者使用的都是截断轨迹。

两次搜索最终得到的结构不同（图 S3d、e；表 S9）。传统 CREST 得到的最低能最终构象与输入结构接近（Cα 均方根偏差 1.3 Å），通过输入结构中的三个氧原子以及 Glu5 的一个羧酸根氧原子与铁配位。FastCREST 得到的结构较为紧凑，通过五个主链羰基氧和 Gln6 的侧链氧原子与铁配位，不含羧酸根；其能量低 85.5 千卡/摩尔。在第三个循环中，传统 CREST 曾找到一个比 FastCREST 的最低能最终构象低 13.7 千卡/摩尔的结构，该结构同样紧凑且不含羧酸根供体，但新一轮循环使其能量升高（S6 节），最终系综的能量比该结构高 99.2 千卡/摩尔。对两次搜索的最终构象进行单点计算后，使用 CREST 3.0.2 和 FastCREST 得到的能量相同（差异在 $9 \times 10^{-11}\ E_\mathrm{h}$ 以内），因此能量差异源于结构不同。这些结构和能量是 GFN2-xTB 计算结果，我们未进一步验证。

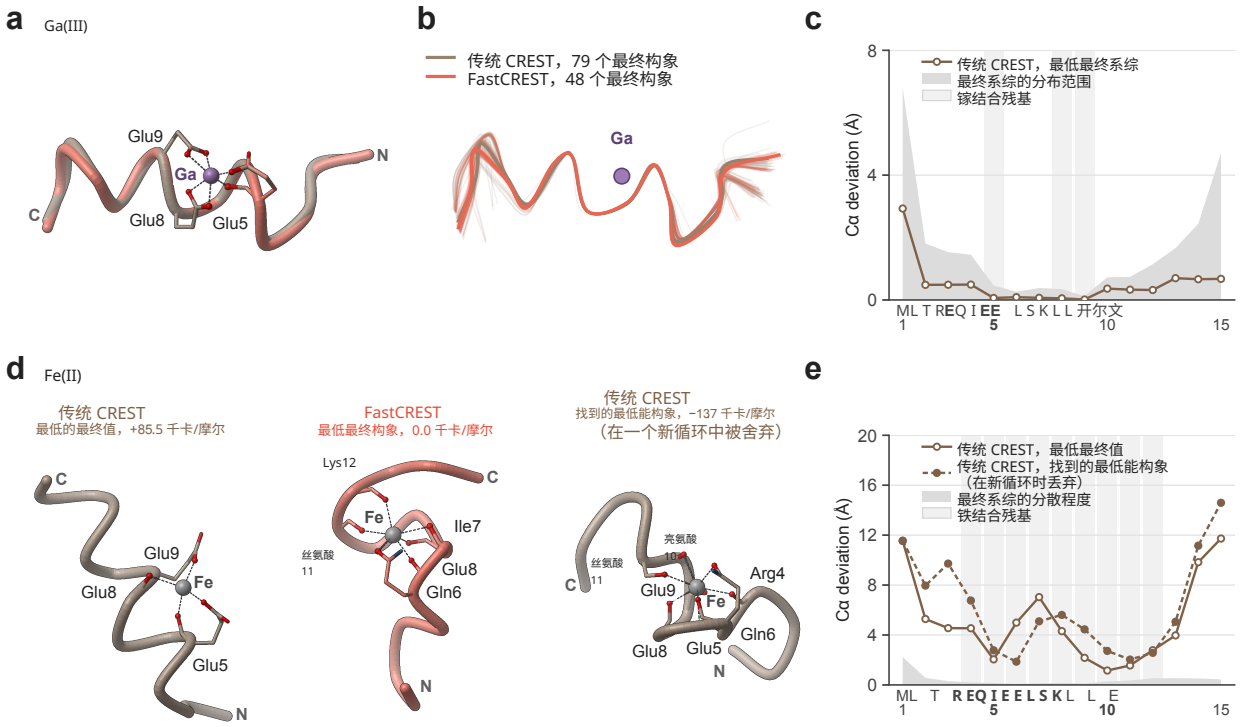


图 S3：MLTREQIEELSKLLE 的 Ga(III)（a–c）和高自旋 Fe(II) 配合物（d、e）的最低能结构；每个代码进行一次搜索；FastCREST 为红色，传统 CREST 为棕色。每个结构均叠合于 FastCREST 最低能最终构象的金属位点：金属、每个所比较结构中以任何方式与其结合的残基主链（Ga：Glu5、Glu8、Glu9；Fe：Arg4 至 Lys12）以及结合侧链原子，其中羧酸根作为其 Cδ。管状线表示 Cα 原子，棒状结构表示结合基团，虚线将金属与 2.6 Å（Ga）或 2.8 Å（Fe）以内的氧原子连接。（a）最低能最终构象（FastCREST 的能量低 1.1 千卡/摩尔）。（b）所有最终构象的 Cα 轨迹，最低能构象以粗体显示。（c）对于传统 CREST 的结构（实线）以及（b）中虚线结构，各残基的 Cα 相对于 FastCREST 最低能最终构象的偏差；同时显示（b）中任一构象相对于其自身搜索所得最低构象的最大偏差（阴影区域）；粗体残基属于该位点。传统 CREST 找到的最低能结构的偏差最大为 0.8 Å，未在图中显示。（d）并排展示，方向和比例相同：传统 CREST 和 FastCREST 的最低能最终构象（低 85.5 千卡/摩尔），以及传统 CREST 找到的最低能结构（比 FastCREST 的最低能最终构象低 13.7 千卡/摩尔，在新循环中被舍弃）。（e）与（c）相同，但针对（d）中的两个传统 CREST 结构。

表 S9：对 MLTREQIEELSKLLE 的 Ga(III) 和高自旋 Fe(II) 配合物进行的搜索。能量相对于每种配合物经 FastCREST 搜索得到的最低能最终构象，单位为千卡/摩尔

配合物	结构	时间（小时）	循环	构象	能量	金属供体（配位数）	R_g (Å)
Ga(III)	输入					Glu5；Glu8、Glu9（2 个氧各）；Glu5 的主链氧（6）	7.49
	FastCREST	58.4	3	48	0.0	与输入相同（6）	7.66
	找到的最低能构象[a]				−30.7	与输入相同 (6)	6.71
	传统 CREST	265.8	2	79	+1.1	与输入相同 (6)	7.36
	找到的最低能构象[a]				−2.4	与输入相同 (6)	7.65
Fe(II)	输入					Glu9；主链氧 Glu5、Glu8 (3)	7.57
	FastCREST[b]	7.3	1	5	0.0	Gln6 OE1；主链氧 Gln6, Ile7, Glu8, 丝氨酸 11，Lys12（6）	6.57
	传统 CREST	242.3	4	4	+85.5	Glu5、Glu9；主链氧 Glu5、Glu8（4）	7.25
	找到的最低能构象[a]				−13.7	Gln6 OE1、丝氨酸11 OG；Arg4 的主链氧，Glu5, Gln6, Glu8, Glu9, 亮氨酸10 (8)	6.45

时间：主任务的耗时。构象：最终系综。能量、供体（距镓 2.6 Å 或铁 2.8 Å 以内；配位数，配位数）以及肽的重原子回转半径 R_g：最低能最终构象，或所命名的结构；未标注数目的谷氨酸结合一个羧酸根氧原子。[a]搜索找到的最低能结构；新循环将其从最终结果中剔除（S6 节）；能量来自搜索日志。[b]其最低能最终构象也是搜索找到的最低能结构。