

# 博士学位论文

晶体结构测定中正空间法的自动化和性能调优

作者姓名: 刘珂

指导教师: 董成 研究员

(中国科学院物理研究所)

学位类别: 理学博士

学科专业: 凝聚态物理

培养单位: 中国科学院物理研究所

2018年12月



**Automation and Performance Optimisation of the**  
**Direct Space Method for Crystal Structure Determination**

A dissertation submitted to  
University of Chinese Academy of Sciences  
in partial fulfillment of the requirement  
for the degree of  
Doctor of Philosophy  
in Condensed Matter Physics  
by  
Liu Yu

**Supervisor: Prof. Dong Cheng**

**Institute of Physics, Chinese Academy of Sciences**

**December 2018**



## **中国科学院大学**

### **研究生学位论文原创性声明**

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

作者签名：

日期：

## **中国科学院大学**

### **学位论文授权使用声明**

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

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名：

导师签名：

日期：

日期：



## 摘要

晶体结构测定的手段可以大致分为倒空间法和正空间法：前者的核心在于提取各衍射峰所对应结构因子的振幅，并设法从振幅反推相位，然后通过 Fourier 变换从各衍射峰的结构因子得到晶胞内的电子密度分布；后者以晶胞内原子的坐标组合为自变量，通过全局最优化算法寻找使计算得到的衍射谱和实际测得的衍射谱最接近的原子坐标。和倒空间法相比，正空间法对衍射谱的分辨率要求更低、对衍射峰重叠的容忍度更高，因此更加适合处理从粉末衍射得到的数据。在正空间法的求解过程中，对结构中成键关系的先验知识可以发挥极大的作用，因此正空间法特别适合求解分子晶体、框架晶体等等类型的结构，但其在求解成键关系总体未知的结构时仍然会遇到许多困难。为了降低最优化算法中搜索空间的维数，对于后一类型的结构常常可以使用等效点系组合（EPC）法：通过将晶胞中的原子分配到可用的 Wyckoff 位置，原先以晶胞中所有  $n$  个原子的坐标为自变量的  $3n$  维最优化问题可以转化为多个互相独立的最优化问题，每个问题以一个 EPC 中独立原子的可变坐标为自变量，而且这些最优化问题的维数在多数情况下远小于  $3n$ 。

在用正空间法求解晶体结构时，常常会遇到计算的衍射谱和实际谱很接近，但在化学上不合理的晶体模型；其中最常见的问题之一是原子重叠，即一些原子对的间距明显小于化学上允许的下限。为了尽量自动化地处理原子重叠问题，须要在全局最优化的过程中高效地实时检测和排除存在原子重叠的模型；对原子重叠的检测属于计算几何中的碰撞检测问题，而我们首先需要的就是一种实时的晶体学碰撞检测算法。本人基于碰撞检测中常用的轴对齐包围盒（AABB）模型，并结合晶胞特殊的几何构造，提出了一种通用的晶体学碰撞检测算法框架。在此基础之上，针对成键关系总体未知的结构，本人对计算几何中常用的 sweep and prune (SAP) 算法进行了修改，使之能以  $O(n \log n)$  的时间复杂度检测晶体模型中的原子重叠；此外，考虑到这类结构的求解往往使用 EPC 法，本人也提出一种利用等效点系对称性显著减少碰撞检测计算量的方法。基于以上的碰撞检测算法，本人设计了一种评估晶胞中原子重叠状况的函数，该函数可用于在最优化的过程中实时排除存在原子重叠的模型；此外，上述的碰撞检测算法不仅有助于更高效地避免原子重叠，而且对于晶胞中配位多面体、原子键价等等的计算也具有重要的意义。

为了让上文所述的排除原子重叠的机制在求解未知结构的过程中发挥实际的作用，本人开发了 *decryst* 这套使用正空间法从粉末衍射数据求解晶体结构的软件。*decryst* 在功能上和以前的 *EPCryst* 类似：两者都使用 EPC 法处理已指标化的数据，并把求解流程分为生成 EPC 列表、对各 EPC 进行统计分析、对每个 EPC 进行全局最优化和导出解模型等 4 个主要步骤；因为 EPC 法的缘故，两者都特别适合求解成键关系总体未知的结构。和 *EPCryst* 相比，*decryst* 最重要的优势在于后者可以在最优化的过程中利用上文中的机制自动、高效地排除原子重叠。此外，受到 Unix 中 `make` 程序的启发，在 *decryst* 的实现中，本人首次将增量计算的思想以一种具有通用性的方式应用于最优化的过程中，使其性能得到了明显的提升；*decryst* 也使用一种增量算法生成 EPC，这不仅极大地降低了其内存需求，而且为在生成 EPC 时对其进行实时筛选做了准备。为了适应目前科学计算的发展趋势，本人在设计 *decryst* 时也加入了对并行和分布式计算的支持，使之可以通过同时利用多个处理器实现对晶体结构测定的进一步加速。考虑到同一结构的各 EPC 互相独立，而且 EPC 数在多数有意义的情形下都相当大，*decryst* 中统计分析和全局最优化任务的并行化将为求解成键关系总体未知的结构带来前所未有的机遇。

*decryst* 是运行在类 Unix 平台上的自由、开源软件，可以从 <https://gitea.com/CasperVector/decryst> 获得；其设计追求简洁、灵活，而本人也希望其中的技巧可以在更多的晶体学软件中得到应用。在现有自动化工具的配合下，用 *decryst* 能简单地实现相当复杂的求解流程：利用重原子法的求解，统计分析和全局最优化后基于 Bragg *R* 因子的 EPC 筛选，求解前筛选必发生原子重叠的 EPC、最优化中实时排除存在原子重叠的晶体模型、求解后筛选仍发生原子重叠的 EPC，等等。本文的最后部分讨论了 *decryst* 的设计和实现，并以美国矿物学家晶体结构数据库（AMCSD）中若干个不同晶系和复杂度的结构为例，循序渐进地演示了 *decryst* 的基本用法和常用技巧。

**关键词：**晶体结构测定，正空间法，碰撞检测，增量计算，平行和分布式计算。

## Abstract

The methods for crystal structure determination can be roughly classified into two categories, the reciprocal space methods and the direct space method (DSM). The reciprocal space methods work by extracting the amplitudes of structure factors for individual reflections and then phasing these reflections, after which the Fourier transform is used to obtain the electron density distribution in the unit cell. In contrast, the DSM uses global optimisation (GO) algorithms to minimise the divergence between the computed and measured diffraction patterns, and uses the coordinate combination of atoms in the unit cell as the variables to be optimised. In comparison with the reciprocal space methods, the DSM is more suitable for handling low-resolution diffraction data, and is more tolerant of overlapping reflection peaks in diffraction patterns. When using the DSM, *a priori* knowledge of the structure to be determined is very helpful, so the DSM is quite suitable for structures like molecular crystals and framework crystals; on the other hand, it is much more difficult to determine structures using the DSM when the bonding relations are largely unknown. To reduce the degree of freedom of the GO problem, the equivalent position combination (EPC) method can often be used for the latter kind of structures: by assigning the atoms to the available Wyckoff positions, we can transform the  $3n$ -dimensional GO problem with regard to all  $n$  atoms in the unit cell into multiple mutually independent problems, each of which only using the non-fixed coordinates of the independent atoms from one EPC, and usually having a degree of freedom much smaller than  $3n$ .

In structure determination using the DSM, often encountered are chemically unreasonable crystallographic models with the computed diffraction patterns very similar to the measured patterns, and one of the top problems is atom bumping: in many of these crystallographic models, the distance between some atom pairs are unreasonably short. To automatically handle atom bumping, we need to efficiently detect and eliminate crystallographic models with atom bumping in real time during the GO procedure; since the detection of atom bumping is known as collision detection in computational geometry, what we need first is a real-time algorithm for crystallographic collision detection. Based on the axis-aligned bounding box (AABB) model, and noticing the

unique geometric structure of the unit cell, the author proposed a generic framework for crystallographic collision detection. Based on this framework, regarding structures with largely unknown bonding relations, the author modified the sweep and prune (SAP) algorithm, enabling it to detect atom bumping in the unit cell in  $O(n \log n)$  time bound. In addition, considering that the EPC method is often used in determination of these structures, the author proposed a method that employs the equivalent position symmetry to significantly reduce the computational complexity of collision detection. Based on these algorithms, the author designed an evaluation function for atom bumping in crystallographic models, which can be used to eliminate unreasonable models in real time during the GO procedure; additionally, apart from the prevention of atom bumping, the collision detection algorithms mentioned above can also be very helpful in computation of coordination polyhedra, valence, *etc.* in crystallographic models.

To facilitate practical use of above-mentioned mechanisms in determination of unknown structures, the author wrote *decryst*, a software suite for structure determination from powder diffraction that uses the DSM. *decryst* is functionally similar to *EPCryst*: both use indexed data, employ the EPC method and abstract structure determination as 4 stages, enumerating EPCs, statistical analyses of EPCs, GO of EPCs and exporting the results; and due to the EPC method, both are particularly suitable when the bonding relations are largely unknown. The most important advantage of *decryst* over *EPCryst* is that the former can automatically and efficiently eliminate atom bumping. In addition, inspired by the `make` utility from Unix, the author employed incremental computation in a generic way, for the first time, in the GO procedure, resulting in a dramatic speedup of the procedure; *decryst* also employs an incremental algorithm for the generation of EPCs, allowing for real-time filtering during the generation procedure in the future. In order to accommodate for contemporary developments in scientific computing, the author designed *decryst* with parallel and distributed computing in mind, allowing for further enhancement of the performance by simultaneous use of multiple processors. Noticing that EPCs are mutually independent, and that the number of EPCs is large in most useful cases, it is expected that the parallelisation of statistical analyses and GO of EPCs will offer unprecedented opportunities for determination of structures with largely unknown bonding relations.

*decryst* is free and open source software that runs on Unix-like platforms, and can

be obtained from <https://gitea.com/CasperVector/decryst>; it is designed to be simple and flexible, in the hope that the underlying techniques could be adopted in more crystallographic applications. In combination with well-established automation utilities, it is easy to implement fairly complicated structure determination procedures: the heavy-atom method, filtering of EPCs based on the Bragg  $R$  factor after statistical analyses and GO; filtering of EPCs that would definitely produce crystallographic models with atom bumping before actually solving the structure, real-time elimination of models with atom bumping in GO, filtering of EPCs that have atom bumping in the solutions, *etc.* The final parts of this dissertation discussed the design and implementation of *decryst* and then demonstrated its basic usage and some useful techniques, using example structures from various crystal families and of varying complexities, that are taken from the American Mineralogist Crystal Structure Database (AMCSD).

**Keywords:** Crystal Structure Determination, Direct Space Method, Collision Detection, Incremental Computation, Parallel and Distributed Computing.



## 目 录

第 1 章 引言	1
1.1 晶体和 X 射线的相互作用	1
1.2 倒空间法求解晶体结构	2
1.3 正空间法和等效点系组合法	4
1.4 原子重叠问题和 <i>decryst</i>	5
第 2 章 晶胞中原子重叠的实时检测和排除	9
2.1 晶体学粗碰撞检测	9
2.1.1 晶胞的几何构造	9
2.1.2 晶胞中的 sweep and prune	11
2.1.3 利用等效点系对称性简化粗碰撞检测	14
2.2 晶体学细碰撞检测	16
2.2.1 键长计算和最邻近向量问题	16
2.2.2 一种原子重叠评估函数	18
2.2.3 原子重叠排除机制的测评	19
2.3 讨论和小结	22
2.3.1 关于细碰撞检测正确性的讨论	22
2.3.2 其它讨论	23
2.3.3 本章小结	24
第 3 章 <i>decryst</i> : 高效的粉晶结构测定软件	27
3.1 <i>decryst</i> 中的增量计算	27
3.1.1 增量计算和惰性计算	27
3.1.2 目标函数的增量计算	29
3.1.3 等效点系组合的增量生成	31
3.2 <i>decryst</i> 设计简介	33
3.2.1 <i>decryst</i> 中的并行化	33
3.2.2 <i>decryst</i> 的软件架构	36
3.3 讨论和小结	39
3.3.1 关于 <i>decryst</i> 性能的讨论	39

---

3.3.2 关于浮点数求和的讨论 . . . . .	40
3.3.3 关于等效点系组合生成算法的讨论 . . . . .	41
3.3.4 其它讨论 . . . . .	42
3.3.5 本章小结 . . . . .	43
<b>第 4 章 <i>decryst</i> 的使用方法和常用技巧</b>	<b>45</b>
4.1 <i>decryst</i> 的基本用法 . . . . .	45
4.1.1 <i>decryst</i> 中主控程序的输入格式 . . . . .	45
4.1.2 基本求解流程：以 $\text{Al}_2\text{O}_3$ 和 $\text{As}_2\text{S}_3$ 为例 . . . . .	47
4.1.3 实际求解：以“0000220”和“0000589”结构为例 . . . . .	50
4.1.4 EPC 筛选：以“0000428”和“0000219”结构为例 . . . . .	53
4.2 <i>decryst</i> 的高级用法和常用技巧 . . . . .	56
4.2.1 重原子法：以 $\text{PbSO}_4$ 为例 . . . . .	57
4.2.2 实际 EPC 的搜索：以“0000158”结构为例 . . . . .	60
4.2.3 EPC 的复杂筛选：以“0009563”结构为例 . . . . .	62
4.3 讨论和小结 . . . . .	66
4.3.1 关于动态剪枝的讨论 . . . . .	66
4.3.2 其它讨论 . . . . .	68
4.3.3 本章小结 . . . . .	68
<b>第 5 章 结论和展望</b>	<b>71</b>
<b>参考文献</b>	<b>73</b>
<b>致谢</b>	<b>79</b>
<b>作者简历和研究成果目录</b>	<b>81</b>

## 插 图

1.1 PbSO <sub>4</sub> 的几种晶体模型 . . . . .	6
2.1 二维晶胞建模为二维环面: $S^1 \times S^1 = T^2$ . . . . .	10
2.2 轴对齐投影和分数坐标下 AABB 的等价性 . . . . .	10
2.3 原子在 <i>a</i> 轴上的轴对齐投影 . . . . .	11
2.4 一种 AABB 比晶胞大的情形 . . . . .	11
2.5 非周期边界 SAP 算法示例 . . . . .	12
2.6 周期边界下 SAP 算法示例 . . . . .	13
2.7 SAP 实现中一些技术细节的示例 . . . . .	14
2.8 等效点系对称性示例 . . . . .	15
2.9 实际使用的对势和 Lennard-Jones 势的对比 . . . . .	15
2.10 键长计算和 CVP 的等价性 . . . . .	16
2.11 一种不好的晶胞选择 . . . . .	17
2.12 Voronoi 图和 Delaunay 三角化之间的对偶关系 . . . . .	17
2.13 PbSO <sub>4</sub> 随机解模型的 ( <i>D</i> , <i>B</i> ) 分布 . . . . .	21
2.14 一个指标几乎合理的错误解 . . . . .	22
2.15 不同 $\mu$ 值所对应解的实验正确率 . . . . .	22
2.16 格点的 Voronoi 图随晶胞参数的变化 . . . . .	23
3.1 make 如何构建一个示例程序 . . . . .	27
3.2 EPC 的生成被建模为深度优先树搜索 . . . . .	32
3.3 EPC 树搜索中根据晶胞化学式进行的剪 . . . . .	32
3.4 单元素 EPC 的生成 . . . . .	33
3.5 总 EPC 的生成 . . . . .	33
3.6 复杂依赖关系示例 . . . . .	34
3.7 MapReduce 式依赖关系 . . . . .	34
3.8 EPC 的并行处理 . . . . .	34
3.9 模拟退火的并行化 . . . . .	35
3.10 <i>decryst</i> 的软件架构 . . . . .	37

4.1 “0000589” 的典型解 . . . . .	52
4.2 “0000428” 的几个解 . . . . .	54
4.3 “0000219” 典型解在 $a$ 方向上的投影 . . . . .	56
4.4 CPU 单线程浮点性能在 1995–2011 年间的增长 . . . . .	57
4.5 “0009563” 的一些解 . . . . .	67

## 表 格

2.1 碰撞检测算法的有效性测评 . . . . .	20
2.2 碰撞检测算法的性能测评 . . . . .	21
3.1 <i>decryst</i> 中增量计算的性能测评 . . . . .	30



## 第1章 引言

### 1.1 晶体和X射线的相互作用

在 Rietveld (1969) 关于全谱拟合的工作之后, 晶体结构解析常常被分为结构测定和结构精修两个主要步骤: 前者对空间群、晶胞参数、晶胞化学式和原子坐标等等基本参数进行大致的确定, 而后者不仅对这些参数进行精细的调整, 而且也对由仪器灵敏度、衍射峰形等等更多因素决定的各种参数进行拟合, 以使所有这些参数和实际测得的整个衍射谱之间达到最大程度的吻合。本文主要讨论通过 X 射线粉末衍射测定晶体结构的问题, 而本节将对晶体和 X 射线之间相互作用的理论进行简要的总结。

一般而言, 在对晶体样品进行 X 射线衍射实验时, 光源、检测器到样品的距离都远大于样品的尺寸, 而 X 射线的波长和晶胞的尺寸在相近的数量级上, 因此样品对 X 射线的散射满足 Fraunhofer 远场条件 (钟锡华 2003)。晶体和 X 射线之间的相互作用主要是其中电子云对 X 射线的散射, 而在电子云对单波长电磁波的远场散射中, 散射波的复振幅  $A$  和散射体的电子密度分布  $\rho$  之间满足以下的 Fourier 关系:

$$A(\mathbf{q}) \propto f(\mathbf{q}) = \int \rho(\mathbf{r}) e^{2\pi i \mathbf{q} \cdot \mathbf{r}} d\tau_r, \quad (1.1)$$

其中  $\mathbf{q}$  为散射向量,  $\tau_r$  为正空间的体积元,  $f$  即散射因子。由此结合 Fourier 变换的可加性和位移-相移关系, 并忽略 X 射线的次级散射 (Pecharsky、Zavalij 2009, p. 4), 可知原点在  $(0, 0, 0)$  的单个晶胞的散射因子正是其结构因子

$$F(\mathbf{q}) = \sum_j f_j(\mathbf{q}) e^{2\pi i \mathbf{q} \cdot \mathbf{r}_j}, \quad (1.2)$$

其中  $j$  为单个原子<sup>1</sup>的编号,  $f_j$  和  $\mathbf{r}_j$  分别为其原子散射因子和坐标。

再次在忽略次级散射后利用 Fourier 变换的可加性和位移-相移关系, 可得整块晶体的散射因子为

$$\sum_j F(\mathbf{q}) e^{2\pi i \mathbf{q} \cdot \mathbf{r}_j} = F(\mathbf{q}) \sum_j e^{2\pi i (h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^*) \cdot (u_j \mathbf{a} + v_j \mathbf{b} + w_j \mathbf{c})} = F(\mathbf{q}) \sum_j e^{2\pi i (hu_j + kv_j + lw_j)}, \quad (1.3)$$

其中  $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ 、 $\{\mathbf{a}^*, \mathbf{b}^*, \mathbf{c}^*\}$  分别为正空间和倒空间晶格的基向量组<sup>2</sup>,  $j$  为单个晶

<sup>1</sup> 在讨论晶体结构时, 常常将其中的离子也称为“原子”, 而本文中也采用这一约定。

<sup>2</sup> 本文统一使用不含  $2\pi$  因子的倒基向量定义 (Prince 2004, p. 1), 以下不再特别说明。

胞的编号,  $\mathbf{r}_j$  为该晶胞的原点坐标。假设有一个晶胞的原点在  $(0, 0, 0)$ , 那么为了使来自所有晶胞的散射波发生相长干涉,  $hu_j + kv_j + lw_j$  应均为整数; 又由晶体<sup>3</sup>的平移周期性可知  $u_j$ 、 $v_j$ 、 $w_j$  均为整数, 因此  $h$ 、 $k$ 、 $l$  也应为整数。

由此可知, 产生衍射峰的散射向量必须是倒晶格向量, 这正是我们熟知的 Laue 条件; 在满足 Laue 条件时, 由 (1.3) 式进一步可知总散射因子正比于结构因子。在具体的衍射实验中, 还有其它的许多因素影响实际测得的衍射谱 (梁敬魁 2011): 例如衍射装置的几何结构主要由 Lorentz–极化因子反映, 粉末衍射中倒易向量模相等的衍射峰互相重叠由多重度因子反映, 此外还有热振动、样品吸收等等多种影响因素。但无论如何, 由本节的推导可见, 结构因子和晶胞中原子排布之间的紧密联系决定了对结构因子的处理是晶体结构测定的核心。

## 1.2 倒空间法求解晶体结构

因为 Fourier 变换具有可逆性, 如果获得了各主要衍射峰所对应结构因子的值, 就可以通过逆变换获得晶胞内的电子密度分布, 从而得到晶胞内各原子的位置; 然而在进行衍射实验时, 对 X 射线的测量会丢失相位信息, 因为仪器测量到的只是衍射峰的强度, 而其正比于结构因子振幅的平方。因此如果要得到完整的结构因子, 就须要设法重建结构因子的相位, 而重建结构相位的数值依据就是结构振幅。这正是结构测定中倒空间法的基本思路, 其名字源于对倒空间中结构振幅的提取; 以下对目前在用粉末衍射数据进行结构测定 (David、Shankland、McCusker 等 2002) 的算法中被最广泛使用的几种倒空间法进行简要的总结, 其中各方法的常用软件列表可以参考下文中相应的文献。

**Patterson 法** (Rossmann、Arnold 2001): 对结构振幅的平方 (正比于衍射峰的强度) 进行 Fourier 变换, 可以得到 Patterson 函数, 后者可以理解为电子密度分布的自卷积或“自关联”函数

$$P(\mathbf{R}) = \int \rho(\mathbf{r})\rho(\mathbf{r} + \mathbf{R})d\tau_r. \quad (1.4)$$

含  $n$  个原子的晶胞会在 Patterson 函数中产生  $n^2$  个峰, 因此即使不考虑峰的重叠问题, 一般而言总的 Patterson 峰数仍然会很大。除了原点处的强峰之外, 具有高强度的 Patterson 峰主要由含重原子的原子对产生, 所以由这些重原子峰可以确定重原子的位置, 而由重原子的位置结合 Patterson 函数中更多的信息便可以

<sup>3</sup> 国际晶体学联合会 (IUCr 1992) 目前将晶体定义为 “any solid having an essentially discrete diffraction diagram”, 即包括具有平移周期性的常规晶体和不具有平移周期性的准晶 (Shechtman 等 1984); 然而, 本文只讨论常规晶体的结构测定, 而本文所称的“晶体”也一律指常规晶体。

进一步推测整个晶胞中各原子的位置。

**直接法** (Giacovazzo 2001): 根据电子密度分布函数的正定性和原子性, 可以得到不同衍射峰所对应 (为抵消高角度峰的强度衰减) 归一化结构因子  $E_{\mathbf{q}}$  以及单位结构因子  $U_{\mathbf{q}}$  之间的一系列数值关系, 其中最重要的是组根据给定衍射峰的结构振幅对一些相关衍射峰结构相位的取值进行限制的不等式关系, 以及一组概率性的等式关系: 例如对  $P\bar{I}$  空间群有

$$U_{\mathbf{q}}^2 \leq (1 + U_{2\mathbf{q}})/2, \quad (1.5)$$

而以下的正切公式 (其中  $G_{\mathbf{k}} \propto |E_{\mathbf{k}}||E_{\mathbf{k}}||E_{\mathbf{q}-\mathbf{k}}|$ ) 给出最概然的结构相位

$$\tan \arg E_{\mathbf{q}} = \frac{\sum_{\mathbf{k}} G_{\mathbf{k}} \sin(\arg E_{\mathbf{k}} + \arg E_{\mathbf{q}-\mathbf{k}})}{\sum_{\mathbf{k}} G_{\mathbf{k}} \cos(\arg E_{\mathbf{k}} + \arg E_{\mathbf{q}-\mathbf{k}})}. \quad (1.6)$$

在实际求解时, 在计算  $E_{\mathbf{q}}$ 、 $U_{\mathbf{q}}$  后, 选出较强衍射峰所组成的  $\{\mathbf{q}, \mathbf{k}, \mathbf{q} - \mathbf{k}\}$  三元组, 根据上述的不等式和等式关系可以对其中一小部分结构相位的符号甚至具体数值进行直接估计; 在此之后, 通过迭代性地应用等式关系和不等式关系, 就可以对各主要衍射峰的结构相位进行重建。

**电荷反转法** (Palatinus 2013): 在满足 Friedel 定律 (Pecharsky、Zavalij 2009, pp. 218–220) 的前提下, 如果对各结构因子赋予随机相位, 由此得到的电子密度分布往往不具备正定性和原子性。为了满足正定性条件, 可以强行将负电子密度部分的符号反转; 此外为了引入随机微扰, 考虑到实际电子密度分布的集中性 (可以认为是由原子性导致的), 实际计算时会将电子密度小于某一预设正值的部分进行反转。在这样修改电子密度分布之后, 用原结构振幅结合反演得到的结构相位, 得到的结构因子又可以通过同样的流程产生新的结构相位, 如此便可进行迭代; 根据实际测试, 对于许多衍射谱, 通过上述迭代算法可以获得满意的结构相位。

一般而言, 为了能使用倒空间法正确求解晶体结构, 衍射谱中可确定结构振幅的衍射峰数须要远多于晶胞中待定的原子数, 因此衍射数据通常需要较高的分辨率, 这在上述几种算法的参考文献中多少有所提及。在对粉末衍射数据使用倒空间法时, 还须要设法分离互相重叠的衍射峰, 这又使得粉末数据需要比单晶数据更高的分辨率。在很多情况下, 高分辨的衍射数据很难获得, 特别是对纳米材料, 即使应用同步辐射也很难获得高质量的单晶衍射数据; 在没有高分辨衍射数据的情况下, 常常不得不使用正空间法求解晶体结构, 而正空间法就是本文将要讨论的主题。

### 1.3 正空间法和等效点系组合法

求解晶体结构的另一种主要思路是直接从原子坐标建立试探性的晶体模型，然后计算晶体模型所对应的衍射谱，将计算得到的衍射谱和实际测得的衍射谱比对，并根据结果设法对晶体模型进行优化，直到获得满意的模型。这正是正空间法 (Černý、Favre-Nicolin 2007) 的思路，后者是除了上文中讨论的倒空间法之外，用粉末衍射数据进行结构测定的主要算法；显然，正空间法并不依赖对结构相位的估计，而且甚至不需要单个衍射峰的结构振幅，因此其特别适合处理低分辨率的衍射数据，以及衍射峰重叠严重的粉末数据。

在计算机尚未被广泛应用于科学计算的年代，晶体结构测定仍严重依赖手动计算和尝试，而对于晶体模型的优化也极大程度地依赖具有极强技巧性（乃至艺术性）的人工处理：例如在包括陆学善、梁敬魁 (1965)，Reddy 等 (1965) 在内的许多先驱的工作中，晶体学家在大致确定各原子的等效点系组合（参考下文和第 3.3.3 小节）之后，通过立体几何的原理确定具体的原子坐标。在现代的正空间法中，尝试性晶体模型的建立和优化仍然不可或缺，只是往往被自动化的计算机程序以极高的频率反复进行；因为这样的原因，正空间法也往往被称为尝试法 (Wolfson 1997) 和模型法 (Černý、Favre-Nicolin 2007)。

从数学上容易看出，正空间法将结构测定抽象为寻找使一目标函数最小化的原子坐标组合

$$\mathbf{x} = (x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n)^T \quad (1.7)$$

的全局最优化问题，其中  $n$  是晶胞内的原子数，而目标函数中最重要的部分是表征计算谱和实际谱之间差异的 **Bragg R 因子**。为了利用计算机自动化地寻找合理的  $\mathbf{x}$ ，现代的正空间法使用各种最优化算法，例如模拟退火 (David、Shankland、Streek 等 2006)、并行回火 (Černý、Favre-Nicolin 2007)、遗传算法 (Shankland 等 1997) 等等来求解上述的问题，因此正空间法的另一个别称是全局最优化法 (David、Shankland、McCusker 等 2002, p. 252)。

若直接以晶胞中所有  $n$  个原子的坐标组合为正空间法中全局最优化的自变量，则最优化问题的维数为  $3n$ ；注意到最优化问题中搜索空间的规模随其维数指数增长，设法尽量降低维数显然很有必要。对于成键关系大部分已知的结构，例如分子晶体 (Andreev 等 1997)、框架晶体 (Falcioni、Deem 1999) 以及部分其它类型的晶体 (Favre-Nicolin、Černý 2002)，可以将原子团的平移、振动、转动等等自由度作为自变量，利用原子团的相对刚性来降低最优化问题的维数，并对

键长、键角进行适当限制以进一步缩小搜索空间。相比之下，在求解成键关系总体未知的结构时，关于成键关系的先验知识就缩小搜索空间而言作用很有限；对于这类结构，等效点系组合法（Deng、Dong 2009）常常是一种有效的手段，而下文就将对其进行简要的总结。

在空间群和晶胞化学式已知<sup>4</sup>的前提下，考虑到绝大多数晶体中晶胞的对称性，可以将晶胞中独立原子的坐标作为自变量，从而极大地降低最优化问题的维数。为了得到独立原子的具体列表，我们须要将晶胞中的原子分配到可用的 Wyckoff 位置上，而每一种分配方案就是一种等效点系组合（equivalent position combination，简称 EPC）。例如对于  $\text{PbSO}_4$  ( $Pnma$ ,  $Z = 4$ )，因为  $Pnma$  空间群有  $4a$ 、 $4b$ 、 $4c$ 、 $8d$  等共 4 种 Wyckoff 位置，为了满足正确的晶胞化学式，其中原子有且仅有 35 种 EPC：

- $\text{Pb}^{2+}: 4a \times 1; \text{S}^{6+}: 4b \times 1; \text{O}^{2-}: 4c \times 4$  (8 维);
- $\text{Pb}^{2+}: 4a \times 1; \text{S}^{6+}: 4b \times 1; \text{O}^{2-}: 4c \times 2 + 8d \times 1$  (7 维);
- .....
- $\text{Pb}^{2+}: 4c \times 1; \text{S}^{6+}: 4c \times 1; \text{O}^{2-}: 4c \times 2 + 8d \times 1$  (11 维，正确 EPC)。
- $\text{Pb}^{2+}: 4c \times 1; \text{S}^{6+}: 4c \times 1; \text{O}^{2-}: 8d \times 2$  (10 维)。

原先 72 维的最优化问题被转化为 35 个介于 6–12 维之间的最优化问题，可见 EPC 法极大地缩减了  $\text{PbSO}_4$  结构求解问题的总规模。

正空间法和 EPC 法的常用软件列表参考第 1.4 小节；此外有必要指出，随着近年来双空间迭代技术，如电荷反转法（参考第 1.2 节），Altomare、Caliandro 等（2008）的直接法–模拟退火混合方案和 Ma 等（2004）的 UACIEM 方法等等的发展，倒空间法和正空间法之间的界限正在逐步地模糊。然而，双空间迭代法目前的发展趋势并非是取代倒空间法或正空间法，而是让这两类算法更加紧密地互相结合，因此针对后两类算法本身的研究仍然是具有重要意义的。

## 1.4 原子重叠问题和 *decryst*

正空间法遇到的一个重要问题是随机产生的晶体模型在化学上不合理，而其中最常见的情况之一是原子重叠（atom bumping），即产生的模型中一些原子对的间距明显小于化学上允许的下限，而且这些模型中一部分甚至具有和正确

---

<sup>4</sup> 本文主要考虑对已指标化粉末衍射数据的处理，因此在讨论结构测定时一般假定晶胞参数和空间群是已知的，或至少能确定空间群在某个比较小的范围之中。此外，因为结构测定所用样品的名义组成一般是已知的，在确定晶胞参数之后常常可以通过密度测量获得晶胞的化学组成，所以本文一般也假定晶胞化学式是已知的。

模型相近的 Bragg  $R$  因子。例如对于  $\text{PbSO}_4$  的正确等效点系组合 (EPC)，如果试图通过最小化  $R$  因子来求解原子坐标，最终得到的结果往往类似于图 1.1：虽然正确的晶体模型中没有原子重叠，但是绝大部分解模型中有严重的原子重叠。对于成键关系大部分已知的结构，通过对键长、键角等等参数的限制，可以极大地减轻这一问题；然而对于成键关系总体未知的结构，这一方法的效果很有限。因此为了尽量自动化地处理原子重叠，我们迫切需要一种具有通用性的机制在全局最优化的过程中高效地实时检测和排除存在原子重叠的模型，而本文第 2 章将要讨论的就是本人 (Liu 2017) 提出的自动化排除原子重叠的机制。

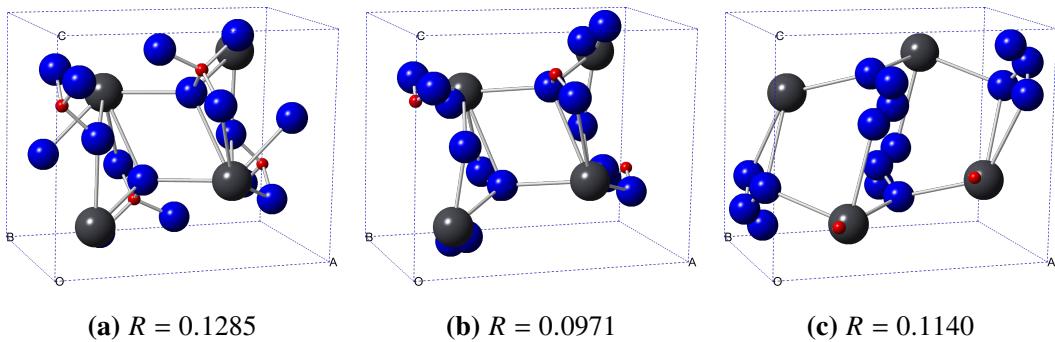


图 1.1  $\text{PbSO}_4$  的几种晶体模型，其中只有 (a) 正确（数据来源参考第 2.2.3 小节）

现有的使用正空间法从粉末衍射数据求解晶体结构的主流软件，如 *FOX* (Favre-Nicolin、Černý 2002)、*DASH* (David、Shankland、Streek 等 2006)、*EXPO* (Altomare、Cuocci 等 2013) 等等多数适合求解成键关系大部分已知的结构，但在求解成键关系总体未知的结构时不甚有效；针对后一类结构的求解，Deng、Dong (2011) 基于 EPC 法开发了 *EPCryst* 这一软件，但其对原子重叠的处理方式是在全局最优化步骤完成之后对解模型进行筛选，因此排除原子重叠的效率相当低。考虑到以上的现状，本人开发了 *decryst* (Liu 2018)，以使上述的排除原子重叠的机制在求解未知结构的过程中发挥实际的作用。

受 Unix 中 `make` 程序 (Feldman 1979) 的启发，本人在 *decryst* 中将增量计算的技术应用于最优化步骤中，使其性能得到了很大的提升；除此之外，考虑到目前科学计算（乃至通用计算）中大规模并行化的发展趋势，本人在设计 *decryst* 时也加入了对并行和分布式计算的支持，使之可以通过同时利用多个处理器实现对晶体结构测定的进一步加速。因为各 EPC 在计算上互相独立，而多数未知结构的 EPC 数很大，可以认为 EPC 法对搜索空间的缩减作用和 EPC 任务的大规模并行化将为求解成键关系总体未知的结构带来前所未有的机遇。针对这些

内容，本文第3章将展开讨论 *decryst* 的基本设计和实现。

第1.3节提到，在早期的利用正空间法进行结构测定的工作中，晶体学家通过具有很强经验性和技巧性的手段对备选的EPC进行分析，从而筛除掉其中不可能产生合理晶体模型的EPC。在 *EPCryst* 中，用户可以通过对随机晶体模型 Bragg *R* 因子的统计分析实现对 EPC 的简单筛选；在此基础之上，*decryst* 在筛选 EPC 时使用包含原子重叠的目标函数，因此可以在筛选中同时考虑 *R* 因子和原子重叠两种因素。*decryst* 的设计追求简洁、灵活，其在现有自动化工具的配合之下可以实现相当复杂的求解流程，如重原子法和多步骤的精细 EPC 筛选。针对以上问题，本文第4章将比较详细地讨论 *decryst* 的使用方法和常用技巧。

最后有必要指出，除了 EPC 法之外，FOX 中的动态占据率校正（dynamical occupancy correction）也是一种处理特殊位置上原子的有效方法，但其更适用于以原子团为基础的晶体模型。对于成键关系总体未知的结构，该方法就降低最优化问题的复杂度而言效果有限，因此在实用性上不如 EPC 法。

*Zum Nutzen und Gebrauch der Lehrbegierigen Musicalischen Jugend, als auch derer in diesem studio schon habil seyenden besonderem Zeitvertreib [...]*

— J. S. Bach, “*Das Wohltemperierte Klavier*”



## 第2章 晶胞中原子重叠的实时检测和排除

### 2.1 晶体学粗碰撞检测

对物体重叠的检测属于计算几何中的碰撞检测（collision detection）问题（Ericson 2005），所以为了在全局最优化中高效地实时检测和排除存在原子重叠的晶体模型，首先需要的就是一种实时的晶体学碰撞检测算法。对于含  $n$  个原子的晶胞，如果通过直接逐对计算原子间距来检测重叠，那么就须要计算所有  $n(n - 1)/2$  个原子对的间距，因此算法的时间复杂度为  $O(n^2)$  量级；由此可见  $n$  较大时逐对算法在性能上是不利的，而这一问题也是实现实时碰撞检测的主要障碍之一。为了解决这一问题，实际的碰撞检测通常被分为粗碰撞检测（broad phase）和细碰撞检测（narrow phase）两个步骤（Ericson 2005, p. 329）：前者排除那些明显不会发生碰撞的物体对，而后者对剩下的物体对进行精确的测试以判断其是否发生碰撞。本节主要就粗检测展开讨论。

#### 2.1.1 晶胞的几何构造

对于直角坐标系下的低维 Euclid 空间，碰撞检测是较为成熟的研究领域；然而晶胞有其独特的几何构造，这使我们难以在晶胞中原封不动地应用现有的碰撞检测算法。首先的问题是晶胞的  $a$ 、 $b$ 、 $c$  轴在很多情况下并不互相正交，因此在晶胞中常常不便使用直角坐标系；针对这一问题，晶体学家的惯用对策是使用分数坐标。假设晶胞中某位移向量  $\mathbf{x}$  的分数坐标为  $\mathbf{x} = (x, y, z)^T$ ，则其 Cartesian 坐标为

$$\mathbf{r} = x\mathbf{a} + y\mathbf{b} + z\mathbf{c} = (\mathbf{a} \ \mathbf{b} \ \mathbf{c}) \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{Ax}, \quad (2.1)$$

其中  $\mathbf{a}$ 、 $\mathbf{b}$ 、 $\mathbf{c}$  为正空间各基向量的 Cartesian 坐标。可见坐标变换矩阵  $\mathbf{A}$  正是基向量组矩阵，且进一步可知  $\mathbf{x}$  的长度为

$$\mathbf{r}^T \mathbf{r} = \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{x}^T \mathbf{G} \mathbf{x}, \quad (2.2)$$

其中二次型  $\mathbf{G} = \mathbf{A}^T \mathbf{A}$  正是分数坐标下以矩阵形式表示的晶胞度规张量（Dodson、Poston 1991, p. 66），而后者决定了晶胞中的距离构造。在分数坐标下，晶胞中的原子位置可以用 3 个互相独立的坐标  $x$ 、 $y$ 、 $z$  表示；又因为每个坐标分别满足周期边界条件，即它们各是一个一维圆环  $S^1$  上的坐标，所以分数坐标下的晶胞

事实上是被建模为 3 个一维圆环的 Cartesian 积（类似于图 2.1 中的情形），即一个三维环面  $T^3$ （肖盖 2009），而这个三维环面具有上述的特殊距离构造。

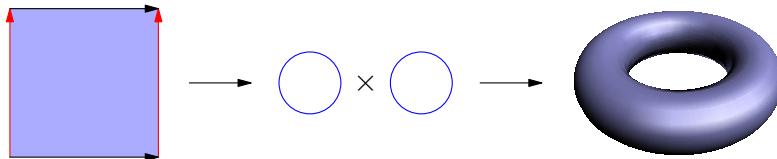


图 2.1 二维晶胞建模为二维环面： $S^1 \times S^1 = T^2$

粗碰撞检测中一种很普遍的做法是围绕每个被检测的物体构造一个包围体 (Ericson 2005, pp. 75–123)，然后利用包围体的特定几何属性来进行优于  $O(n^2)$  的粗检测；一种常用的包围体是轴对齐包围盒 (axis-aligned bounding box, 简称 AABB)，即以物体在各坐标轴上的投影区间为边界的盒子 (图 2.2)。两个 AABB 间发生碰撞当且仅当其在 3 个坐标轴上的投影区间均发生碰撞，因此使用 AABB 在很多时候可以将三维问题化为 3 个一维问题；注意到这里应用的是变量分离的数学思想，而在晶胞内用分数坐标正好可以实现 3 个坐标的分离，在使用 AABB 对晶胞内原子进行粗检测时使用分数坐标显然是最合适的选择<sup>1</sup>。最后，为了能使用 AABB 进行粗检测，还须要解决计算 AABB 的问题：具体而言，给定原子的分数坐标和 Cartesian 半径  $r$ ，如何计算其在分数坐标下的 AABB？

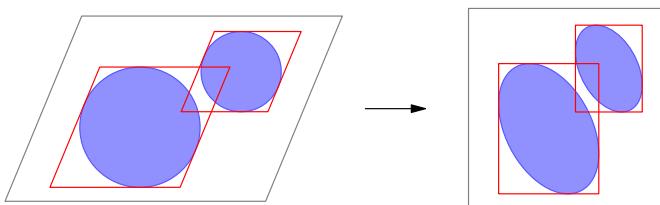


图 2.2 轴对齐投影和分数坐标下 AABB 的等价性

考虑到分数坐标下使用 AABB 正好等价于在以  $a$ 、 $b$ 、 $c$  轴为坐标轴的衍射坐标下使用和坐标轴对齐的斜投影 (图 2.2)，不妨从一个和  $a$  轴及  $a^*$  轴均正交 (因此也和  $bOc$  平面平行) 的方向考察上述原子在  $a$  轴上的轴对齐投影 (图 2.3)，此时只须计算投影区间的半宽  $R$  即可。根据相似三角形的性质可知

$$\frac{R}{a} = \frac{r}{d} \Rightarrow \frac{R}{ar} = \frac{1}{d} = a^* = \frac{\sin \alpha}{av}, \quad (2.3)$$

<sup>1</sup> 此外，因为上述的 AABB 碰撞判据只涉及对投影区间上下界的比较而不涉及距离，所以分数坐标下晶胞的特殊距离构造并不影响 AABB 的使用。

其中  $d$  为相邻两  $bOc$  晶面的间距，而

$$v = V/abc = \sqrt{1 - \cos^2 \alpha - \cos^2 \beta - \cos^2 \gamma + 2 \cos \alpha \cos \beta \cos \gamma} \quad (2.4)$$

为晶胞的无量纲化体积； $a^*$  和  $V$  的计算公式引自文献（Prince 2004, pp. 1–4）。在实际的粗检测中使用的是分数坐标，因此最终用到的半宽应为

$$\frac{R}{a} = \frac{r}{d} = \frac{r \sin \alpha}{av}; \quad (2.5)$$

同理可对  $b$ 、 $c$  轴得到类似的结论。

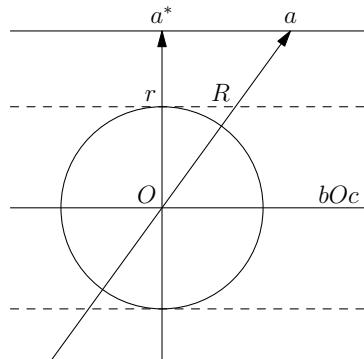


图 2.3 原子在  $a$  轴上的轴对齐投影

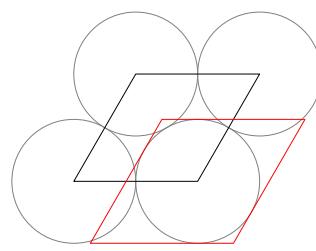


图 2.4 一种 AABB 比晶胞大的情形

晶胞的周期边界也会影响碰撞检测，但考虑到  $T^3$  是三维流形，流形的概念（Dodson、Poston 1991, p. 161）可以给我们一定的启发：可以把看成盒子的晶胞强行分为“边界”和“内部”两部分，其内部完全类似于具有非平凡 Euclid 度规张量的普通空间。因此，对于完全包含于晶胞内部的 AABB，可以按常规流程进行粗碰撞检测，不用担心受周期边界的影响；而对于跨边界的 AABB，我们须要进行专门的额外检测。

### 2.1.2 晶胞中的 sweep and prune

在使用轴对齐包围盒（AABB）的碰撞检测算法中，利用树状数据结构的算法，如 Grudinin、Redon（2010）的算法可以考虑已有的成键信息，因此特别适合处理成键关系大部分已知的结构；对于成键关系总体未知的结构，我们需要的是能从头检测碰撞的算法，而这类算法中一种最简单的是 **sweep and prune**（简称 SAP，也称为 sort and sweep；参考 Ericson 2005, pp. 329–338）。假设我们希望在非周期边界的前提下检测  $n$  个物体（例如图 2.5 中的那些）之间的碰撞；在 SAP 算法中，我们会首先将物体投影到某个方便使用的坐标轴上，然后对投影区间进行碰撞检测：

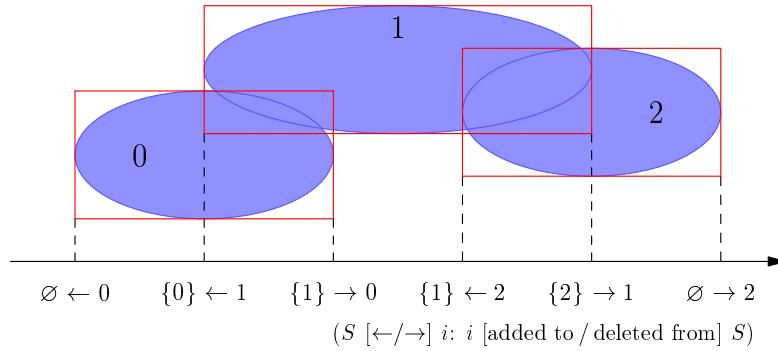


图 2.5 非周期边界 SAP 算法示例

- 将各投影区间的上下界在排序后存入一个列表  $l$  (sort 步骤), 然后建立一个空集  $S$ , 并通过从小到大扫描  $l$  中的各个上下界来跟踪各上下界处投影区间之间的碰撞 (sweep 步骤)。
- 当扫描到一个下界时, 报告相应投影区间和  $S$  中每个区间的碰撞, 然后将该区间插入  $S$  中: 例如在图 2.5 中, 当扫描到区间 #1 的下界时, 算法会先报告区间 #0 和 #1 之间的碰撞, 然后将区间 #1 插入  $S$  中。
- 当扫描到一个上界时, 从  $S$  中删除相应的投影区间: 例如在图 2.5 中, 当扫描到区间 #1 的上界时, 该区间会被从  $S$  中删除。

在 SAP 中,  $S$  上的插入和删除操作常常并入和 SAP 的输出 (报告) 代码, 因此习惯上并不在粗检测的复杂度分析中考虑 (Ericson 2005, p. 333)。虽然如此, 不难注意到这些操作的总复杂度取决于实际发生碰撞的物体对数: 如果每个物体都和其它所有物体发生碰撞, 那么  $S$  上的插入和删除操作无论如何都将至少花费  $O(n^2)$  时间。如果 sort 步骤中使用的是合适的排序算法, 例如归并排序 (Knuth 1998, pp. 158–168), 则该步骤花费的时间在最坏的情形下也是  $O(n \log n)$  量级; 而 sweep 步骤花费的时间显然是  $O(n)$  量级, 所以 SAP 的总时间复杂度可以为  $O(n \log n)$  量级。

如第 2.1.1 小节所述, 晶胞中周期边界对粗碰撞检测算法的影响往往可以通过对跨边界的 AABB 进行专门的额外检测来解决。就 SAP 而言, 因为其正确性依赖于每个投影区间的下界在上界之前被扫描到, 周期边界所带来的问题在于跨边界区间的上半部分在 sweep 步骤中会被算法错误地忽略: 例如在图 2.6 中, 区间 #0 的上半部分和区间 #1 之间的碰撞就被忽略了。然而考虑到在 sweep 步骤的末尾, 集合  $S$  中会包含所有的跨边界区间, 我们可以对常规的 SAP 算法进行如下的修改:

- 执行原 SAP 算法的步骤, 但当扫描到上界时须要注意, 如果相应的下界

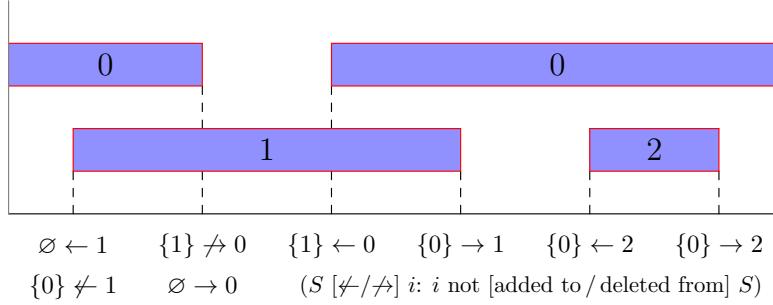


图 2.6 周期边界下 SAP 算法示例

尚未遇到，那么它们所对应的投影区间将不在集合  $S$  中，因此试图从  $S$  中删除相应区间将是无意义的。例如在图 2.6 中，当扫描到区间 #0 的上界时，我们不能从  $S$  中删除该区间。

- 原步骤执行完之后，再次从小到大扫描列表  $l$ ，但不在  $S$  中插入区间：跨边界区间互相之间的碰撞在原 sweep 步骤的末尾已被处理，所以我们只关心不跨边界的区间和跨边界区间之间的碰撞。例如在图 2.6 中，再次扫描到区间 #1 的下界时，该区间不会被插入  $S$  中。
- 当  $S$  成为空集时，算法终止。例如在图 2.6 中，再次扫描到区间 #0 的上界时，算法终止。

上述改动中额外加入的新 sweep 步骤在最坏的情形下将花费  $O(n)$  量级的时间，因此修改后 SAP 算法的时间复杂度仍可以为  $O(n \log n)$ 。在此基础之上有必要指出，为了正确、高效地实现 SAP，一些技术细节须要特别注意：

- 在上述新加的 sweep 步骤中，部分碰撞对有可能会被重复报告，例如图 2.6 中区间 #0 和 #1 之间的碰撞会被报告两次，而这些重复的报告应该在实现时加以正确处理。
- 上述改动通过判断投影区间的下界是否小于上界来检测跨边界区间，但这样的做法在区间长度不小于坐标轴的长度时（例如图 2.4 中的情形）会出错。为了处理这种情形，可以强行限制区间在分数坐标下的半宽最多为  $0.5 - \varepsilon$ ，其中  $\varepsilon > 0$  为一很小的常数。
- 在  $n$  很大的情形下，如果只在一个坐标轴上使用 SAP，碰撞对可能不会被有效筛除：例如在图 2.7(a) 中，只在  $x$  或  $y$  轴上使用 SAP 都不能有效筛除碰撞对；此时在多个坐标轴上使用 SAP<sup>2</sup> 有可能会实现更高的总体性能（但也未必，参考第 2.2.3 小节）。

<sup>2</sup> 顺便指出，在使用多坐标轴 SAP 时正相当于构造了以物体在各坐标轴上的投影区间为边界的盒子，即物体的 AABB，因此 SAP 被分类为使用 AABB 的算法。

- 被检测物体在特定坐标轴上的投影区间可能特别拥挤，例如图 2.7(b) 中的  $x$  轴；此时在这些轴上使用 SAP 效果将很差，因此在 SAP 中可以跳过这些轴。由第 2.1.1 小节可知，在  $R/ar$  值特别大的坐标轴上容易出现拥挤问题，因此在晶胞中使用 SAP 时建议不用这些轴。
- 在物理建模中，往往可以认为一对只在彼此边界上发生碰撞的物体（例如一对相切的球）并不真正发生相互作用；因此如果一些上下界在  $l$  中排名相同时，可以先处理上界，然后处理下界。
- 快速排序算法（Knuth 1998, pp. 113–122）在最坏情形下具有  $O(n^2)$  的复杂度，但其平均复杂度仍为  $O(n \log n)$ ，且常数因子更小（参考第 3.1.1 小节），因此其比归并排序更适用于 SAP 中的 sort 步骤。

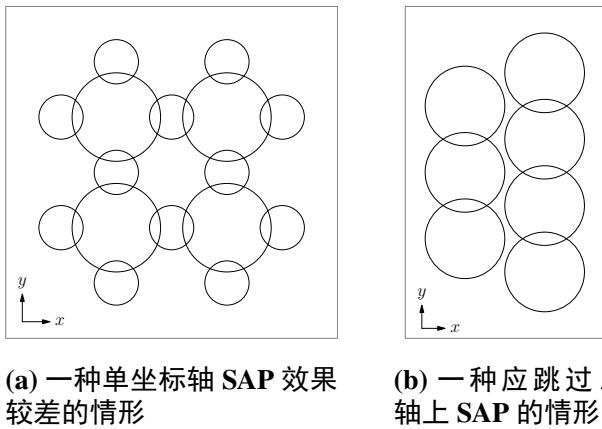


图 2.7 SAP 实现中一些技术细节的示例

最后，考虑到因为 Coulomb 相互作用，两个原子间的正常键长可能不同于其半径之和，我们引入对缩放因子（pairwise zoom factor） $p$  的概念，其定义为上述正常键长和半径之和的比。此外我们定义原子的碰撞检测半径（collision-detection radius）为用来计算其投影区间的半径，其等于原子的正常半径  $r_0$  乘以其原子缩放因子（atomic zoom factor） $q$ 。显然，为了保证粗检测的正确性，对任意的原子对  $\{k_0, k_1\}$  应有

$$p(k_0, k_1)(r_0(k_0) + r_0(k_1)) \leq q(k_0)r_0(k_0) + q(k_1)r_0(k_1)。 \quad (2.6)$$

### 2.1.3 利用等效点系对称性简化粗碰撞检测

因为等效点系组合法（参考第 1.3 节）的使用，我们可以利用等效点系对称性来降低晶胞粗碰撞检测的复杂度，而本节将就此进行讨论。为了方便推导，首

先约定用下标  $i$  标记晶胞中的独立原子，其中  $i = 0, 1, \dots, m - 1$ ；用下标  $ij$  标记独立原子  $i$  的等效原子，其中  $j = 0, 1, \dots, n_i - 1$ ：例如在图 2.8 中，第 1 个独立原子的第 2 个等效原子被标记为 01 ( $i = 0, j = 1$ )。考虑一个两体函数

$$c(i_0j_0, i_1j_1) = c(i_1j_1, i_0j_0) \quad (i_0j_0 \neq i_1j_1), \quad (2.7)$$

该函数满足等效点系对称性：对于任何兼容于上述晶胞的对称操作  $T$ ，有

$$c(i_0j_0, i_1j_1) = c(T(i_0j_0), T(i_1j_1)); \quad (2.8)$$

例如 Euclid 距离函数和第 2.2.2 小节中的对势函数（图 2.9）都满足这一条件。

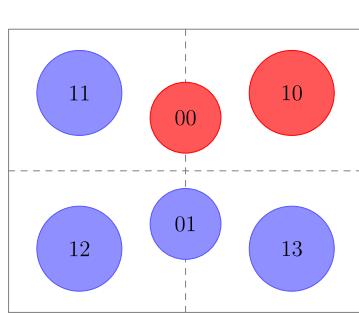


图 2.8 等效点系对称性示例

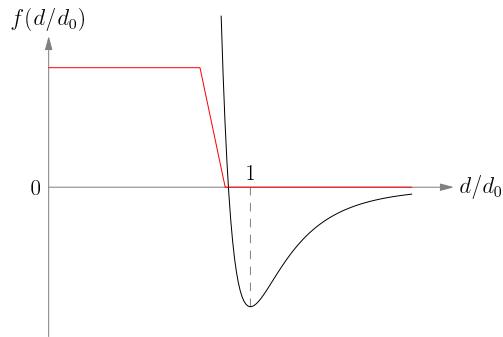


图 2.9 实际使用的对势（浅色）和 Lennard-Jones 势（深色）的对比

对于任意原子  $i_0j_0$ ，由等效原子的定义可知必然存在一对称操作  $T_{i_0j_0}$  将其变换为原子  $i_00$ ，于是有

$$\sum_{i_1j_1 \neq i_0j_0} c(i_0j_0, i_1j_1) = \sum_{i_1j_1 \neq i_0j_0} c(i_00, T_{i_0j_0}(i_1j_1)) = \sum_{i_1j_1 \neq i_00} c(i_00, i_1j_1), \quad (2.9)$$

因此进一步有

$$\sum_{\{i_0j_0, i_1j_1\}} c(i_0j_0, i_1j_1) = \frac{1}{2} \sum_{i_0} n_{i_0} \sum_{i_1j_1 \neq i_00} c(i_00, i_1j_1) = \sum_{\{i_0j_0, i_1j_1\}} \delta_{j_0} n_{i_0} c(i_0j_0, i_1j_1), \quad (2.10)$$

其中

$$\delta_j = \begin{cases} 1 & (j = 0) \\ 0 & (j \neq 0) \end{cases} \quad (2.11)$$

为 Kronecker 记号。

由此又有

$$\sum_{\{i_0j_0, i_1j_1\}} c(i_0j_0, i_1j_1) = \sum_{\{i_0j_0, i_1j_1\}} c(i_1j_1, i_0j_0) = \sum_{\{i_0j_0, i_1j_1\}} \delta_{j_1} n_{i_1} c(i_0j_0, i_1j_1); \quad (2.12)$$

两式平均即得最终用到的关系

$$C = \sum_{\{i_0j_0, i_1j_1\}} c(i_0j_0, i_1j_1) = \frac{1}{2} \sum_{\{i_0j_0, i_1j_1\}} (\delta_{j_0}n_{i_0} + \delta_{j_1}n_{i_1})c(i_0j_0, i_1j_1)。 \quad (2.13)$$

注意如果只将不对称单元内原子的  $j$  设为 0，则上式中  $(\delta_{j_0}n_{i_0} + \delta_{j_1}n_{i_1})$  一项对于不对称单元外的  $\{i_0j_0, i_1j_1\}$  对将恒为零：例如在图 2.8 中，有

$$C = \frac{1}{2} \left( (2+4)c(00, 10) + 2c(00, \{01, 11, 12, 13\}) + 4c(10, \{01, 11, 12, 13\}) \right)。 \quad (2.14)$$

由此可见其重要性：对于不对称单元外的原子对，可以不经 SAP 直接跳过细检测；这样即使完全不使用 SAP，我们也可以将细检测的次数从  $n(n-1)/2$  降低到  $mn - m(m-1)/2$ ，即从  $O(n^2)$  量级降低到  $O(mn)$  量级。

## 2.2 晶体学细碰撞检测

利用第 2.1.2–2.1.3 小节的算法，我们可以高效地筛除晶胞中明显不发生碰撞或因等效点系对称性而可以跳过细碰撞检测的原子对。本节将首先讨论晶胞中的细检测，然后构造一种评估晶胞中原子重叠状况的函数，最后对这些算法和评估函数处理原子重叠问题的有效性和性能进行测评。

### 2.2.1 键长计算和最邻近向量问题

在讨论晶胞中一对原子之间的“键长”时，我们指的一般是由这对原子周期性平移所得的两套格子之间的最短距离；由平移对称性可知，后者正是由晶胞原点周期性平移所得的格子到两原子之间位移向量的最短距离（图 2.10）。由此容易注意到，晶胞中键长的计算和最邻近向量问题（closest vector problem，简称 CVP；参考 Micciancio、Goldwasser 2002）之间有紧密的联系，因为后者正是要寻找给定格子中离给定向量最近的格点。

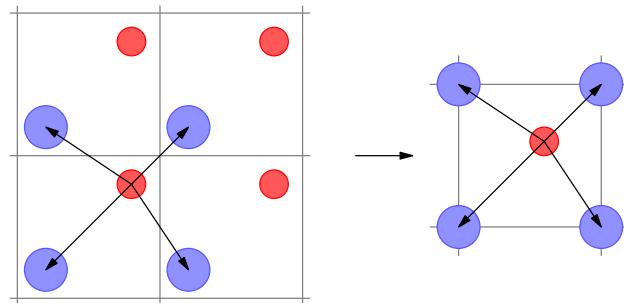
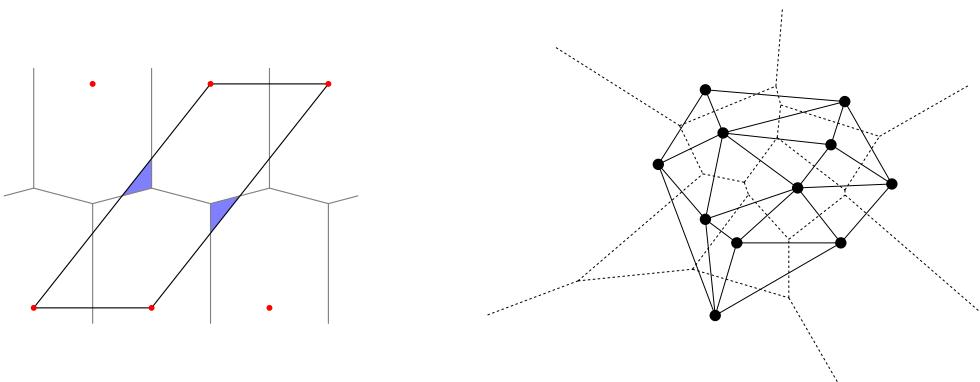


图 2.10 键长计算和 CVP 的等价性

CVP 在当代密码学研究中意义重大<sup>3</sup>，在很大程度上是因为其计算复杂度随着所用向量空间的维数急剧增长，即使用量子计算机求解也一样（Micciancio、Regev 2009）；而如果对所用的向量空间能找到一个尽量短且接近正交的基向量组，其计算会容易很多。晶体学中所用向量空间的维数常常不超过 3，因此 CVP 的计算复杂度并不构成严重问题，但我们仍须进行晶格归约，因为本文所用的细碰撞检测算法假定离位移向量最近的格点一定不会在包围此向量的晶胞之外，而这意味着我们须要选取合适的晶胞（其一反例见图 2.11<sup>4</sup>）。



在选取合适的晶胞之后便可以计算键长：首先计算两原子之间的位移向量，然后计算包围位移向量的晶胞中所有格点和位移向量的距离里最短的一个。另一种常用算法（Attfield 等 1999, Grudinin、Redon 2010）是构造一个原子在原子对所属晶胞的所有紧邻晶胞中的副本，然后计算这一原子以及上述所有副本到另一原子之间的距离中最短的一个。显然，这里的算法明显快于后一种算法：对于三维晶格，后一种算法须要计算距离 27 次，而这里的算法只需 8 次。

此外本人注意到对于分数坐标下  $l^1$  范数

$$|\Delta x| + |\Delta y| + |\Delta z| < 1/2 \quad (2.15)$$

的位移向量，Kshirsagar (2015) 直接以位移向量的长度作为键长，但本人未能找到其正确性的证明。SHELX (Sheldrick 2010) 中也采用了这一算法，但其说明中似乎未对位移向量的  $l^1$  范数进行明确的限制。

<sup>3</sup> 事实上，*decryt* 这一名字除了“decrypt your crystal structure”的表层含义之外，还暗示着晶体学（crystallography）和密码学（cryptography）两个领域之间因 CVP 而形成的联系。

<sup>4</sup> 有必要指出的是，其中按最近格点对平面进行划分得到的正是 Voronoi 图（参考第 2.3.1 小节）。

## 2.2.2 一种原子重叠评估函数

如第 1.4 节所述，正空间法中出现原子重叠问题的原因在于 Bragg  $R$  因子小的晶体模型仍可能发生原子重叠，而原始的全局最优化中也并未考虑原子重叠；因此，如果以某种方式在最优化中考虑原子重叠，就可以自动地排除存在原子重叠的模型。一种在全局最优化中考虑原子重叠问题的思路是修改最优化问题的约束条件，从而使得生成的晶体模型总是那些不发生原子重叠的；然而这样会使最优化的操作变得十分繁琐，其中最重要的原因是在上述约束条件下可行模型的集合将被分割为许多互相孤立的碎片，这将和最优化算法中对自变量随机地进行连续微扰的做法发生冲突，使得最优化的效率显著降低。

考虑到上述问题，本人采用了修改目标函数的做法：既然只用  $R$  因子不能很好地排除存在原子重叠的晶体模型，那就在目标函数中明确地加入一个对存在重叠的模型进行惩罚的评估函数，从而使得最优化算法自动地倾向于生成无重叠的解模型。在最优化算法中，目标函数往往被看成某种物理系统的势能函数，因此优化目标函数就是降低系统的势能；仿照这一思路，这里的评估函数被设计为基于一个两体势

$$C = \sum_{\{k_0, k_1\}} c(k_0, k_1), \quad (2.16)$$

其中  $k$  为晶胞中原子的编号；对势函数  $c$  被定义为

$$c(k_0, k_1) = f(d(k_0, k_1) / d_0(k_0, k_1)), \quad (2.17)$$

其中  $d$  和  $d_0$  分别为原子对的实际键长和正常键长。 $f$  是一个分段线性函数，其控制节点为

$$f(0) = f(0.75) = 1, \quad f(0.875) = f(+\infty) = 0; \quad (2.18)$$

其形态被设计为和 Lennard-Jones 势的相似，但斜率更加平缓（图 2.9）。注意到我们只在两原子间键长为正常键长的 0.875 倍时开始考虑它们的相互作用，各原子的碰撞检测半径（参考第 2.1.2 小节）可以按下式计算：

$$r(k) = 0.875q(k)r_0(k). \quad (2.19)$$

上述的两体势  $C$  显然满足等效点系对称性（参考第 2.1.3 小节），因此可以利用 (2.13) 式加速计算。对于总原子数  $n$  较大的晶胞，因为可能有更多的原子对发生重叠， $C$  的平均值也会比较大。为了处理这一问题，实际算法中使用的评估

函数被定义为

$$B = \min(C/n, 1), \quad (2.20)$$

其取值上限被强行设定为 1 的原因是当  $C = n$  时，平均每个原子会和两个原子全面重叠，而此时可以认为相应的晶体模型是极不合理的。注意到  $B \in [0, 1]$ ，如果能找到 Bragg  $R$  因子的一个上界  $\nu$ ，就可以将目标函数定义为

$$E = \mu B + (1 - \mu)R/\nu \quad (\mu \in [0, 1]), \quad (2.21)$$

这样对组合因子  $\mu$  的任何值都有  $E \in [0, 1]$ 。事实上，如果计算的衍射谱和实际谱都使用归一化的峰强<sup>5</sup>，那么可以得到

$$R = \sum_i |I_{\text{obs},i} - I_{\text{calc},i}| / \sum_i I_{\text{obs},i} = \sum_i |I_{\text{obs},i} - I_{\text{calc},i}| = 2D, \quad (2.22)$$

其中  $D \in [0, 1]$  为看作离散概率测度的两衍射谱之间的总变差距离 (Levin 等 2008)；于是可知  $\nu = 2$ ，并得到目标函数的实际公式

$$E = \mu B + (1 - \mu)D = \mu B + (1 - \mu)R/2. \quad (2.23)$$

### 2.2.3 原子重叠排除机制的测评

本小节涉及的测试代码可从 *decryst* 的项目主页自由获取：<https://gitea.com/CasperVector/decryst>；测试数据（含测试结构的晶胞参数、等效点系组合和缩放因子）以及图 1.1 所对应的结构数据<sup>6</sup>可以从文献 (Liu 2017) 的补充材料中获取。因为 SAP 算法的常数因子较大（从下文中可见），为了保证结果的一致性，这里只在具有最小  $R/ar$  值（参考第 2.1.1 节）的单个坐标轴上使用了 SAP 算法。

为了测试第 2.1.2–2.1.3 小节算法的有效性，本人从美国矿物学家晶体结构数据库 (American Mineralogist Crystal Structure Database, 简称 AMCSD；参考 Downs、Hall-Wallace 2003) 中获取了若干个不同复杂度的结构，并对每个结构通过为独立原子赋予随机坐标的方式生成了 10000 个晶体模型。对于每个结构，本人计算了在使用或不使用 (2.13) 式时（分别对应于  $N_{\text{max}}$  和  $N_{\text{orig}}$ ）每个模型所需细检测次数的上限，并统计了每个模型实际的细检测次数 ( $N_{\text{real}}$ ) 和碰撞原子对数 ( $N_{\text{eff}}$ ) 的平均值和标准差，如表 2.1 所述。从表中可见，(2.13) 式和 SAP 都

<sup>5</sup> 许多从粉末衍射数据求解晶体结构的软件使用的正是归一化的峰强；本人开发的 *decryst* 也是如此，因此后者可以使用 (2.23) 式。

<sup>6</sup> AMCSD 数据库结构代号：0005558。

极大地减少了测试结构中需要细检测的原子对数。

**表 2.1 碰撞检测算法的有效性测评：**ID 为测试结构在 AMCSd 中的代号； $n$  和  $m$  分别为晶胞中的总原子数和独立原子数； $N_{\max}$  和  $N_{\text{orig}}$  分别为每个模型在用或不用 (2.13) 式时所需的细检测次数； $N_{\text{real}}$  和  $N_{\text{eff}}$  分别为每个模型实际的细检测次数和碰撞原子对数。

ID	$n$	$m$	$N_{\text{orig}}$	$N_{\max}$	$N_{\text{real}}$	$N_{\text{eff}}$
0005558	24	5	276	105	50(3)	8(3)
0015840	64	8	2016	476	212(8)	10(4)
0009272	64	8	2016	476	218(24)	43(22)
0009563	90	10	4005	845	350(22)	23(8)
0002630	126	14	7875	1659	550(9)	29(8)
0000427	152	20	11476	2830	581(32)	40(9)
0000447	160	4	12720	630	227(11)	8(3)

为了测试上文中算法的性能，对于上述的每个测试结构，本人在不同条件下执行了一个测试程序，每个条件下测试了 250 组晶体模型，每组由 250 个通过为独立原子赋予随机坐标的方式生成的模型构成：

- ( $t_{\text{void}}$ ) 不进行任何碰撞检测，只生成随机的晶体模型；该条件下度量的是测试程序本身的时间开销。
- ( $t_{\text{bn}}$ ) 无 SAP，只使用 (2.13) 式筛选原子对，细检测步骤被设为空指令；该条件下度量的是只使用 (2.13) 式进行粗检测的时间开销。
- ( $t_{\text{bN}}$ ) 无 SAP，只使用 (2.13) 式筛选原子对，但正常执行细检测步骤；该条件下度量的是只使用 (2.13) 式进行完整碰撞检测的时间开销。
- ( $t_{\text{Bn}}$ ) 使用 (2.13) 式和 SAP 筛选原子对，但细检测步骤被设为空指令；该条件下度量的是同时使用 (2.13) 式和 SAP 进行粗检测的时间开销。
- ( $t_{\text{BN}}$ ) 完全的碰撞检测；该条件下度量的是使用 (2.13) 式和 SAP 进行完整碰撞检测的时间开销。
- ( $t_{\text{orig}}$ ) 不使用 (2.13) 式或 SAP，但正常执行细检测步骤；该条件下度量的是逐对算法的时间开销。

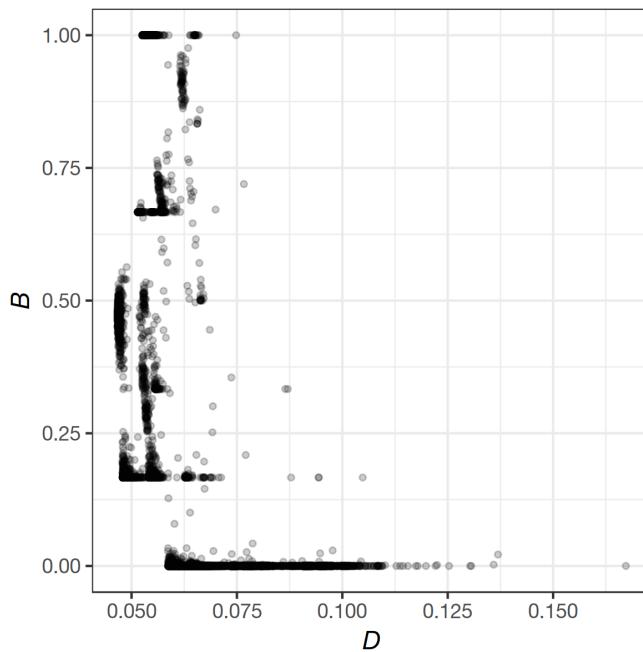
对以上每种条件，本人在 Intel i7-3720QM CPU 上收集了测试程序在每组模型上所花费时间的平均数和标准差，并将其除以每组的模型数，结果如表 2.2 所述。从表中可见，对于各个复杂度的测试结构，第 2.1.2–2.1.3 小节的算法都极大地降低了碰撞检测所花费的总时间；而且随着晶胞中原子数的增加，这些算法和逐对算法相比的优势体现得更加显著。有必要强调的是，从表中也可以注意到，虽然 SAP 是  $O(n \log n)$  的算法，但是其常数因子较大（参考第 2.1.2 小节中归并

排序和快速排序的比较), 因此对于较小的晶胞而言只使用(2.13)式进行粗检测可能在性能上会更有利。

**表 2.2** 测试程序在每个模型上花费的时间 ( $\mu\text{s}$ )。 $t_{\text{void}}$ : 只生成模型;  $t_{\text{bn}}$ : 使用(2.13)式;  $t_{\text{bN}}$ : (2.13)式和细检测;  $t_{\text{Bn}}$ : (2.13)式和 SAP;  $t_{\text{BN}}$ : (2.13)式、SAP 和细检测;  $t_{\text{orig}}$ : 使用逐对算法。

ID	$t_{\text{void}}$	$t_{\text{bn}}$	$t_{\text{bN}}$	$t_{\text{Bn}}$	$t_{\text{BN}}$	$t_{\text{orig}}$
0005558	0.88(7)	1.19(7)	7.4(2)	8.0(2)	10.9(3)	17.6(5)
0015840	1.8(1)	3.2(1)	31.8(8)	30.6(7)	44(1)	135(2)
0009272	2.3(2)	3.6(1)	31.4(4)	34.3(6)	47(2)	130(2)
0009563	2.76(7)	5.2(2)	54.0(8)	57.3(6)	78(1)	257(3)
0002630	3.6(1)	8.0(2)	105(1)	90(1)	121(1)	496(5)
0000427	4.2(1)	12(1)	180(6)	91(2)	124(3)	727(7)
0000447	3.8(1)	5.6(2)	46.2(6)	83(2)	98(2)	886(11)

为了测试第 2.2.2 小节中原子重叠评估函数的有效性, 本人利用同一小节中的目标函数  $E$  对  $\text{PbSO}_4$  结构进行了求解: 对于组合因子  $\mu = 0, 0.01, 0.02, \dots, 0.5$  的每个值<sup>7</sup>, 本人利用全局最优化算法生成了 1000 个随机的解模型, 并统计了它们的  $(D, B)$  分布, 如图 2.13 所示。由图可见正确解的  $D = R/2$  值并不是最低的, 而  $D$  值最低的解中都有严重的原子重叠问题, 这正好解释了第 1.4 节提到的通过最小化  $R$  因子求解  $\text{PbSO}_4$  结构得到的多数是错误解的现象。



**图 2.13**  $\mu = 0, 0.01, 0.02, \dots, 0.5$ , 每个  $\mu$  值所对应 1000 个解模型的总体  $(D, B)$  分布

<sup>7</sup> 上限取 0.5 是考虑到我们的最终目标是结构测定。

考虑到  $D$  和  $B$  都是原子坐标的连续函数，且两者都在和晶胞兼容的对称操作之下保持不变，因为正确解的  $D = R/2 = 0.0642$ 、 $B = 0$ ，根据上图可以假定所有正确<sup>8</sup>的解模型都满足  $D < 0.12$ 、 $B < 0.05$ 。为了验证这一判据，本人生成了 100 个随机的满足判据的解模型，其中  $\mu \in [0, 0.5]$  为随机选取；在人工检查每个解之后，可以发现其中 99 个是正确的，而唯一错误解（图 2.14）的  $D = 0.0814$ （所有 100 个解中最大，次大值为 0.0705）。本人由此假定一个解正确的充分必要条件为其  $D < 0.075$ 、 $B < 0.05$ ，并统计了每个  $\mu$  值所对应 1000 个解中正确解的比例，如图 2.15 所示。由图可见，上述的评估函数在最优化的过程中有效地排除了  $\text{PbSO}_4$  的不合理晶体模型。

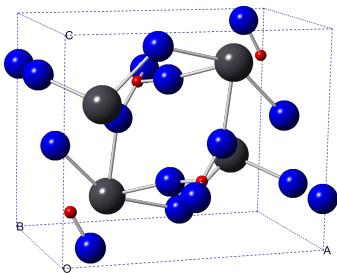


图 2.14 一个指标几乎合理的错误解 ( $D = 0.0814$ ,  $B = 0$ )

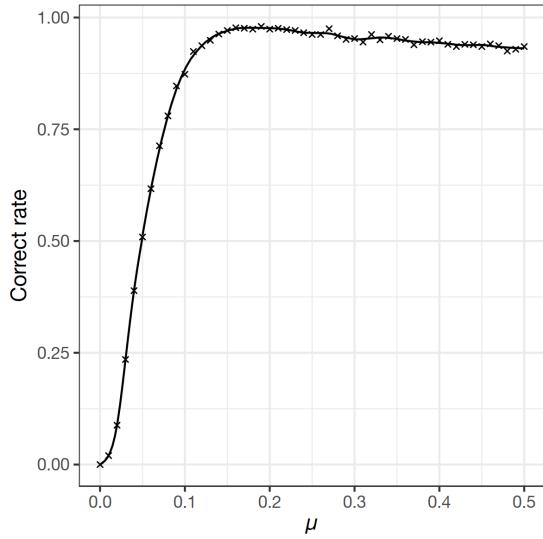


图 2.15 不同  $\mu$  值所对应解的实验正确率

## 2.3 讨论和小结

### 2.3.1 关于细碰撞检测正确性的讨论

如第 2.2.1 小节所述，本文中计算键长的算法假定已经选取合适的晶胞，从而使得离位移向量最近的格点一定不会在包围此向量的晶胞之外。然而，能这样做的条件是对每一个晶格都至少存在一个满足要求的晶胞，但本人未能找到这一条件的严格证明。希望理论工作者能证明或证伪这一条件，并（如果条件是正确的）给出一种构造合适晶胞的算法；另一方面，如果这一条件是错误的，本

<sup>8</sup> 所有由对称操作所联系的模型都视为同一模型，而只在原子坐标上有细微差别的模型也视为等价。

文中的细检测算法将不得不搜索更多的近邻晶胞以寻找离位移向量最近的格点，但本文中的粗检测算法将不需改动。

考虑到在二维情形下，晶胞出现类似于图 2.11 中问题的前提是其一对对边在正交投影下没有重叠（图 2.16），而晶格归约中的 Seeber 算法（Engel 1986）正好会排除这种情形，后者可能会在上述问题的研究中有一定的启发意义。另一种或许具有可行性的研究思路是利用 Voronoi 图（Berg 等 2008, pp. 147–171）和 Delaunay 三角化（Berg 等 2008, pp. 191–218）之间的对偶关系（图 2.12）：本人猜想，格点的 Delaunay 三角化（Caroli、Teillaud 2016）可能满足离任意点最近的格点是其一包围单纯形的顶点；在此基础之上，通过适当拼接这些单纯形，或许就可以得到满足要求的晶胞。此外在二维情形下，Delaunay 三角化会最小化所有三角形中的最大内角，而这似乎也和晶格归约有一定的内在关联。

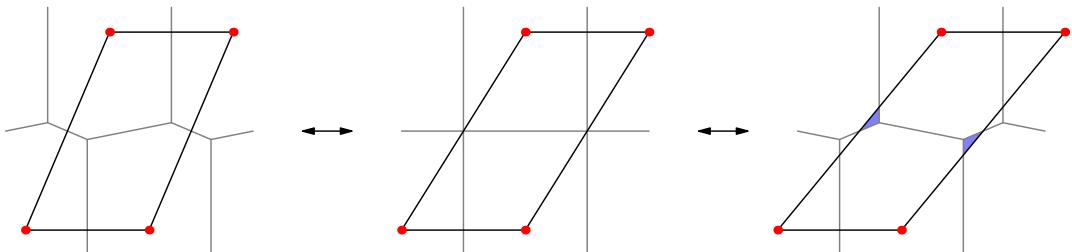


图 2.16 格点的 Voronoi 图随晶胞参数的变化

最后有必要指出，如果第 2.2.1 小节中对  $l^1$  范数小于  $1/2$  的位移向量计算键长的简便算法被证明是正确的，那么我们将能大幅度地进一步减少计算短距离原子对键长所花费的时间。尽管上述简便算法尚无严格证明，但本人倾向于认为其是正确的，只是需要一定的限制条件，包括合适的晶胞选择<sup>9</sup>（例如图 2.11 中的晶胞便会使上述简便算法出错）。另外，如果在某些应用场景下，对于原子对没有进行除了 (2.13) 式之外的筛选，那么长距离原子对将不能被充分筛选，因此在这些应用场景中上述算法带来的性能提升将较为有限。

### 2.3.2 其它讨论

(2.13) 式并未彻底利用特殊位置的对称性：例如在图 2.8 中因为  $c(00, 10) = c(00, 11)$ 、 $c(00, 12) = c(00, 13)$ ，总的两体势事实上可以被进一步归约为

$$C = c(00, 01) + 4c(00, 10) + 2c(00, 12) + 2c(10, \{01, 11, 12, 13\})。 \quad (2.24)$$

<sup>9</sup> 本人在某些限制条件下对随机产生的晶胞进行了测试（具体代码参考第 3.2.2 小节中的说明），未能找到这一简便算法的反例。随机测试显然不能取代严格证明，但本人基于此测试猜想这一算法应该正确。

如果这样的归约能自动进行，那么  $C$  的计算将能进一步简化；然而这样很可能将需要一个比 (2.13) 式中的  $(\delta_{j_0}n_{i_0} + \delta_{j_1}n_{i_1})/2$  复杂得多的原子对多重度计算公式，而后者本身的计算开销未必能被其对  $C$  的简化所带来的性能提升所补偿。

在表 2.2 中，尽管“0000447”是总原子数  $n$  最大的结构，但其  $t_{BN}$  却小于  $t_{BN}$ ，这是因为其独立原子数  $m$  太小，导致其晶体模型中绝大多数原子对被 (2.13) 式筛除。此外尽管如第 2.1.2 小节所述，SAP 算法并不特别适用于成键关系大部分已知的结构，但如果这类结构中的成键信息因故不须在碰撞检测中考虑时，其也可以用 SAP 来处理：在表 2.2 中，“0015840”（其中唯一的分子晶体）的测试结果和“0009272”的结果相差并不大，这在两结构的  $n$ 、 $m$  值都完全相同的背景之下不难理解。

更加复杂的两体势，例如 Bushmarinov 等 (2012) 所用的势或者甚至吸引势，原则上也是可以在实际的评估函数中使用的。第 2.2.2 小节中的对势  $c$  只仿照了 Lennard-Jones 势中的排斥部分，因为本人认为我们的目标是排除原子重叠，而不是对原子之间的相互作用进行精确的建模。虽然如此，从第 2.2.3 小节可见，目前使用的评估函数应该至少对包括  $\text{PbSO}_4$  在内的一些简单结构是有效的，而其在求解更复杂结构时的应用将在第 4 章中展示。

对于很复杂的结构，特别是独立原子数很大的结构，期望只使用 Bragg  $R$  因子和原子重叠评估函数就能成功求解是不太现实的：在这些结构的晶体模型中，完全有可能存在一些  $R$  因子合理且无明显原子重叠，但在成键类型、配位数、配位多面体形态、键价等等方面有问题的模型。相应地，我们须要在结构测定中应用更多的结构验证 (structure validation；参考 Spek 2003) 手段，而第 4 章将给出一些通过人工方式进行结构验证的例子；但为了实现使结构测定更加自动化的目标，我们最终须要做的显然是将更多的结构验证技术自动化地引入结构测定中。考虑到在结构验证中很多测试都以结构中的成键关系为基础，可以认为第 2.1.2–2.1.3 小节的算法应该会在这些方面的研究中发挥基础性的作用。

### 2.3.3 本章小结

针对晶体结构测定中的原子重叠问题，本人基于轴对齐包围盒 (AABB) 模型提出了一种具有通用性的晶体学碰撞检测算法框架。为了处理成键关系总体未知的结构，本人基于上述框架修改了 sweep and prune (SAP) 算法，使之能以  $O(n \log n)$  的时间复杂度检测晶胞中的成键关系。考虑到在求解成键关系总体未知的结构时常常使用等效点系组合 (EPC) 法，本人也提出了一种利用等效点系

对称性极大降低碰撞检测复杂度的算法，这一算法即使对于小晶胞也可以显著降低碰撞检测的时间开销。基于以上算法，本人提出了一种针对晶胞中原子重叠状况的评估函数，其可以用于在正空间法的全局最优化步骤中实时地排除存在原子重叠的晶体模型；此外，上述算法对于晶胞中配位多面体、原子键价等等的计算也具有重要的意义。

即得易见平凡，仿照上例显然；留作习题答案略，读者自证不难。

反之亦然同理，推论自然成立；略去过程 *Q.E.D.*，由上可知证毕。

——《西江月·证明》（摘自互联网）



## 第3章 *decryst*: 高效的粉晶结构测定软件

### 3.1 *decryst* 中的增量计算

如第 1.4 节所述，针对成键关系总体未知的结构，为了提升排除原子重叠的效率，本人基于第 2 章中的机制开发了 *decryst*。本人在开发 *decryst* 的过程中，通过应用增量计算的思想，使其全局最优化步骤的性能得到了大幅度的提升（参考第 3.1.2 小节）；除此之外，通过增量生成等效点系组合，本人极大地降低了 *decryst* 的内存需求（参考第 3.1.3 小节）。此外，在讨论这些问题之前，第 3.1.1 小节将简要回顾 *decryst* 中应用的和增量计算相关的技术。

#### 3.1.1 增量计算和惰性计算

增量计算的思想在计算机科学技术的发展中有着深远的影响：于 70 年代后期出现在 Unix 中的 `make` 程序（Feldman 1979）会检测软件项目中发生变更的模块，然后执行命令根据依赖关系重新构建所有受影响的模块（图 3.1），从而避免重复构建未发生变更的模块；于 2016 年推出的视频游戏 *Doom* 中，通过在渲染每一帧画面时巧妙地利用为前一帧画面计算的数据，制作方将游戏的画面品质和图形性能提升到了新的水平（Courrèges 2016）。有必要指出，增量计算和惰性计算是一脉相承的，因为合理地避免多余的计算可以说正是惰性计算的核心思想；受此启发，在 *decryst* 中，除了增量计算之外，本人也应用了其它的可归类为惰性计算的技术，而本小节将对这些技术进行简短的总结。

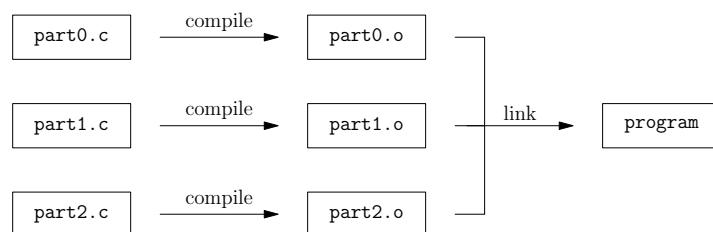


图 3.1 `make` 如何构建一个示例程序：在 `program` 构建完成之后，如果 `part0.c` 发生变更，为了再次构建 `program`，`make` 只须重新编译 `part0.c` 然后重新链接所有 `.o` 文件

首先，本人认为代码中处理对象的分离（separation of concerns）和惰性计算的思想是符合的：例如全局最优化的代码不应该关心结构中所涉及化学元素的名称和符号，更不应该关心在绘制结构图时使用什么颜色的球体来表示其中的原子。通过对各模块所处理对象的细致分离，我们不仅会使软件的架构更加清

晰，也会减少各模块代码所从事的无用工作，因此在增强软件可维护性的同时也提升了软件的性能。在 *decryst* 的架构设计中，这一原则的一些具体体现可以参考第 3.2.2 小节。

其次，CPU 缓存（Bryant、O’Hallaron 2011）可以认为是一种通过硬件实现的惰性计算技术，这一技术使得对连续内存区域的访问被定向到访问效率远快于内存的缓存，从而提升程序性能。相应地，利用 CPU 缓存的方法是使用对缓存友好的算法和数据结构，从而尽量利用连续的内存区域：例如在可行的前提下，尽量使用快速排序算法而非归并排序算法（参考第 2.1.2 小节），使用数组而非链表，等等。

根据增量计算避免重新计算未改变值的思想，对于值不发生改变的函数，在首次求值之后便不须重新求值了；而如果将这些函数的求值提前到程序的开头，我们所做的就是建立查找表。对于计算开销很大的函数，使用查找表显然十分有利：例如在正空间法的全局最优化步骤中，对于 X 射线原子散射因子

$$f_X(hkl) = g_X(\sin \theta_{hkl}/\lambda) = c_X + \sum_i a_{X,i} e^{b_{X,i} (\sin \theta_{hkl}/\lambda)^2}, \quad (3.1)$$

注意到其求值须要多次用到计算较慢的指数函数和正弦函数，其计算开销显然很大；又因为其值只取决于衍射指标  $hkl$ 、元素 X 和在最优化步骤中不变的波长  $\lambda$ ，所以其特别适合使用查找表。因为上述原因，*decryst* 会在最优化程序的一开始建立包括原子散射因子在内若干函数的查找表。

*decryst* 使用模拟退火作为其全局最优化算法，而后者可以和物理退火相类比：既然保温可以让系统趋于平衡，如果在系统始终不太远离平衡的前提下逐渐降低温度，就可以期望使系统接近低温下的平衡态。显然，如果系统已经很接近平衡，就不必再保温，这样可以减少非必要的保温过程所花费的时间，而这正是自适应模拟退火的思路：不是在每个温度阶梯上保温固定的时间，而是在检测到系统接近平衡时自动降温。就目前而言，*decryst* 在其最优化步骤中使用的是 Lam、Delosme (1988) 的自适应模拟退火算法，因为对后者可以比较方便地进行并行化（参考第 3.2.1 小节）。

最后，注意到晶胞中常有的对称性，以及结构因子的计算开销在全局最优化总开销中的高权重（参考第 3.1.2 小节），*decryst* 中也利用以下关系来降低结构因子的计算开销（其中  $\mathbf{q}$  为散射向量）：

$$F_{\text{反演对称}}(\mathbf{q}) = \sum_i f_i(\mathbf{q})(e^{2\pi i \mathbf{q} \cdot \mathbf{x}_i} + e^{2\pi i \mathbf{q} \cdot (-\mathbf{x}_i)}) = 2 \sum_i f_i(\mathbf{q}) \cos(2\pi \mathbf{q} \cdot \mathbf{x}_i), \quad (3.2)$$

其中  $i$  为中心对称原子对的编号;

$$F_{\text{带心格子}}(\mathbf{q}) = \sum_{ij} f_i(\mathbf{q}) e^{2\pi i \mathbf{q} \cdot (\mathbf{x}_i + \mathbf{r}_j)} = \sum_i f_i(\mathbf{q}) e^{2\pi i \mathbf{q} \cdot \mathbf{x}_i} \sum_j e^{2\pi i \mathbf{q} \cdot \mathbf{r}_j}, \quad (3.3)$$

其中  $\mathbf{r}_j$  为可选原点的坐标。

最后有必要指出, 惰性计算技术或多或少地在其它晶体学软件中有所应用, 而且其对软件性能的影响是较为显而易见的, 但本人未能找到一个较为全面的对以上技术的总结。这样的现状所造成的一个不良的结果是部分晶体学软件对这些技术的忽视, 例如 *EPCryst* (Deng、Dong 2011) 在上述技术中只应用了查找表这一项技术, 这使得其在  $\text{PbSO}_4$  正确等效点系组合的全局最优化中比 *decryst* 慢了两个数量级<sup>1</sup>。正因如此, 本人希望本小节中的总结能对晶体学软件的设计者有所帮助。

### 3.1.2 目标函数的增量计算

因为 *decryst* 使用 (2.23) 式中的函数  $E$  作为全局最优化的目标函数 (参考第 2.2.2 小节), 为了加速最优化, 我们事实上须要加速 Bragg  $R$  因子和原子重叠评估函数  $B$  的计算; 又因为计算  $R$  因子时大部分时间用在计算结构因子上, 我们显然须要加速结构因子的计算。考虑到两个相邻的最优化循环中晶体模型之间的差别只在于某一个独立原子位置的移动, 根据 Fourier 变换的可加性有

$$F'(\mathbf{q}) - F(\mathbf{q}) = \sum_i f_i(\mathbf{q}) (e^{2\pi i \mathbf{q} \cdot \mathbf{x}'_i} - e^{2\pi i \mathbf{q} \cdot \mathbf{x}_i}), \quad (3.4)$$

其中  $i$  是被移动原子的编号。根据这一关系, 我们可以利用前一晶体模型的结构因子数据增量地计算后一晶体模型的结构因子: 只须对被移动的原子重新计算  $e^{2\pi i \mathbf{q} \cdot \mathbf{x}_i}$  项的值即可。由此可知, 如果晶胞中有  $m$  个独立原子, 而这些原子在各最优化循环中被轮流移动, 那么晶胞中所有原子结构因子的重新计算将被  $m$  个最优化循环分担 (而非原先的在单个最优化循环中完成), 于是结构因子的计算性能将是原先算法的  $m$  倍。

类似地, 对于  $B$  所依赖的两体势  $C$  (参考 2.16 式), 显然其中对势函数  $c(k_0, k_1)$  的值只在  $\{k_0, k_1\}$  对中至少一个原子被移动时发生改变; 因此  $C$  的值也可以增量地计算, 只须对上述原子对重新进行碰撞检测和处理即可。又注意到利用等效点系对称性 (参考第 2.1.3 小节), 我们只须对含独立原子的原子对进

<sup>1</sup> 根据本人的测试, 在 Intel i7-3720QM CPU 上, *EPCryst* 耗时多于 1000 s, 而 *decryst* 即使在不用增量计算的前提下耗时也少于 10 s。

行碰撞检测，所以最终细碰撞检测次数的上限是  $(n - 1) + (m - 1)(n' - 1)$ ，其中  $n$ 、 $m$ 、 $n'$  分别为晶胞的总内原子数、独立原子数和被移动独立原子的等效原子数。在平均意义下，上述碰撞检测的时间复杂度是  $O(n)$  量级，因此我们不再需要专门使用类似于 sweep and prune（参考第 2.1.2 小节）的粗碰撞检测算法以避免逐对碰撞检测的  $O(n^2)$  复杂度，这在上述 SAP 算法常数因子较大的背景之下尤其值得注意；相应地，因为不再须要计算原子的投影区间，我们也就不再须要使用原子缩放因子，而是只需要对缩放因子。此外有必要特别提请注意的是，当最优化循环的次数很大时，增量求和的浮点数误差积累可能会造成严重的问题，而类似于 Kahan (1965) 算法的增量浮点数求和算法难以应用，因为被求和数的平均值渐进地趋于零 (Higham 1993)。就目前而言，*decryst* 通过在增量求和中使用定点数来绕过这一问题，因为定点数算术是没有误差的。

最后，本人在 Intel i7-3720QM CPU 上就增量计算对全局最优化步骤性能的影响进行了测评。除了可以从项目主页 <https://gitea.com/CasperVector/decryst> 获取的数据之外，其它测试数据可以从文献 (Liu 2018) 的补充材料中获取。本人从 AMCSd 数据库 (Downs、Hall-Wallace 2003) 中获取了若干个不同复杂度的结构，并对这些结构计算了目标函数：具体而言，对于每一个测试结构，本人测试了 100 组晶体模型，每组由 100 个通过为独立原子赋予随机坐标的方式生成的模型构成。本人度量了在使用或不使用增量计算（分别对应于  $t_{\dots,\text{incr}}$  和  $t_{\dots,\text{orig}}$ ）时对每组模型计算整个目标函数  $E$  或只是 Bragg  $R$  因子所花费时间（分别对应于  $t_{E,\dots}$  和  $t_{R,\dots}$ ）的平均值和标准差，并将其除以每组的模型数，结果如表 3.1 所述。

**表 3.1** *decryst* 中增量计算的性能测评（时间单位为  $\mu\text{s}$ ）：ID 为测试结构在 AMCSd 中的代号， $n$  和  $m$  分别为晶胞中的总原子数和独立原子数， $N_{\text{refl}}$  为有记录的衍射峰数； $t_{E,\dots}$  和  $t_{R,\dots}$  分别为计算每个模型  $E$ 、 $R$  所需的时间； $t_{\dots,\text{incr}}$  和  $t_{\dots,\text{orig}}$  分别为用或不用增量计算时所需的时间。

ID	$n$	$m$	$N_{\text{refl}}$	$t_{R,\text{incr}}$	$t_{R,\text{orig}}$	$t_{E,\text{incr}}$	$t_{E,\text{orig}}$
0005558	24	5	94	8.9(3)	43.9(5)	12.5(7)	44.5(7)
0009272	64	8	80	39.7(9)	241(3)	50.6(6)	242(3)
0009563	90	10	76	15.0(2)	168(3)	30.0(7)	174(2)
0002630	126	14	73	11.3(3)	244(3)	30.7(3)	252(2)
0000427	152	20	374	27.4(5)	639(7)	53.6(7)	663(8)
0000447	160	4	32	27.0(9)	87(2)	59(1)	102(2)

从表中可见，对于各个复杂度的测试结构， $R$  因子和  $B$  的计算性能都因增量计算的应用而得到了极大（有时多于一个数量级）的提升。有必要指出，表中

“0009272”的测试结果和“0000447”的 $t_{\dots,\text{orig}}$ 可能显得反常，但事实上前者可以从相应结构并非带心格子以及其中用到的Wyckoff位置均没有中心对称性（参考第3.1.1小节）得到解释，而后者可以从相应结构中和 $n$ 相比很小的 $m$ （参考第2.1.3小节）以及和其它结构相比很小的 $N_{\text{refl}}$ 得到解释。

### 3.1.3 等效点系组合的增量生成

考虑生成 $\text{Al}_2\text{O}_3$ ( $R\bar{3}c$ , 莱方表示下 $Z=2$ )等效点系组合(EPC)的问题：其空间群具有 $2a$ 、 $2b$ 、 $4c$ 、 $6d$ 、 $6e$ 、 $12f$ 等共6种Wyckoff位置，其中 $2a$ 、 $2b$ 位置因为坐标完全固定所以不能被多次占据，否则必然发生极其严重的原子重叠。在这样的约束下，其总共有6个可行EPC： $a_1b_1/d_1$ ， $a_1b_1/e_1$ ， $c_1/a_1c_1$ ， $c_1/b_1c_1$ ， $c_1/d_1$ 和 $c_1/e_1$ （其中 $a_1b_1/d_1$ 是 $[\text{Al}^{3+}: 2a \times 1 + 2b \times 1; \text{O}^{2-}: 4d \times 1]$ 的简写，其余类似； $c_1/e_1$ 为正确EPC）。对于同样空间群的 $\text{A}_2\text{B}_3$ ， $\text{A}_2\text{B}_3\text{C}_3$ ， $\text{A}_2\text{B}_3\text{C}_3\text{D}_3$ ，…，这一系列结构的可行EPC数随着元素数的增长接近于指数增长：6, 16, 38, 78, 142, 236, 366, …；对于Wyckoff位置种类较多的空间群，特别是以 $Pmmm$ 为代表的一些正交晶系空间群，可行EPC数过大的问题尤为突出。

在Deng、Dong(2009)原先提出的EPC生成算法中，每个元素的可行EPC被单独生成，然后这些单元素EPC被组合成总的EPC，这样的做法显然须要存储各元素可行EPC的列表以便在生成总EPC时使用；此外，*EPCryst*(Deng、Dong 2011)将单元素可行EPC和总可行EPC的列表存储在内存中，因此其在可行EPC数很大时可能遇到内存溢出的问题。为了应对这一问题，*decryst*使用了增量的方式生成可行EPC，并直接将得到的每个总可行EPC输出到文件，从而避免内存溢出，而本小节将对增量生成EPC的算法进行讨论。

首先考虑 $\text{Al}_2\text{O}_3$ 中 $\text{Al}^{3+}$ 可行EPC的生成：如图3.2所示，在人工搜索 $\text{Al}^{3+}$ 的可行EPC时，一般的顺序是 $a_0b_0 \rightarrow a_1b_0 \rightarrow a_0b_1 \rightarrow a_1b_1 \rightarrow \dots$ ，而这样的顺序事实上正是对一棵以EPC为叶节点的树进行深度优先搜索(depth-first search；参考Cormen等2009, pp. 603–612)的顺序；当搜索到恰好完全符合晶胞内 $\text{Al}^{3+}$ 原子数的节点时（例如图中的 $a_1b_1$ ），就输出该节点的EPC。显然，一个首先需要关心的问题是如何确定搜索的限度：例如，为什么图中在搜索到 $a_1b_0$ 之后紧接着搜索到的是 $a_0b_1$ ，而不是 $a_2b_0$ ？事实上，在对树进行搜索时，往往可以根据所关心问题的数学性质进行剪枝(pruning)，即自动排除一些可以严格证明不需要搜索的子树<sup>2</sup>。

<sup>2</sup> 在计算机科学中，一个孤立的根节点也被认为是一棵树(Cormen等2009, p. 176)，因此这里的“子树”也包含单一叶节点的情形。

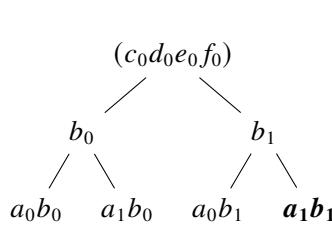


图 3.2  $\text{Al}_2\text{O}_3$  中  $\text{Al}^{3+}$  原子 EPC 的生成被建模为深度优先树搜索（粗体为搜索到的可行 EPC）

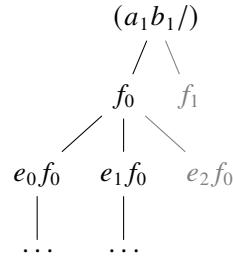


图 3.3 EPC 树搜索中根据晶胞化学式进行的剪枝（浅色为被剪掉的子树）

首先如上文所述，为了避免原子重叠，所有的固定 Wyckoff 位置（例如  $\text{Al}_2\text{O}_3$  所属空间群的  $2a$ 、 $2b$  位置）分别最多只能被占据一次，因此这类位置被占据多次的子树都可以剪掉，而这就是图 3.2 中在  $a_1b_0$  之后紧接着搜索到  $a_0b_1$  的原因。除此之外，根据晶胞化学式也可以进行剪枝：例如在图 3.3 中，因为  $\text{Al}_2\text{O}_3$  的晶胞化学式不允许  $6e$  位置被占据多次，且不允许  $12f$  位置被占据，所以相应 EPC 为  $a_1b_1/e_n f_0$  ( $n > 1$ ) 和  $a_1b_1/f_n$  ( $n > 0$ ) 的子树都可以剪掉；值得注意的是，在使用这一剪枝策略之后，*decryst* 使用的 EPC 生成算法不再须要像 Deng、Dong (2009) 原先的算法那样预先计算单个元素 EPC 中各 Wyckoff 位置可占据次数的上限。在 *decryst* 使用的 EPC 生成算法中，以上两个策略是进行剪枝的基础。

在实际生成单元素可行 EPC 时，因为固定 Wyckoff 位置的约束是对固定位置各自总占据数的约束，在剪枝时必须考虑其它元素 EPC 中各固定位置被占据的次数：例如在图 3.4 中，尽管被生成的是  $\text{O}^{2-}$  的可行 EPC，程序仍须考虑  $\text{Al}^{3+}$  的 EPC；因为后者占据  $a_1b_1$ ，所以程序可以将所有占据了  $2a$ 、 $2b$  的子树剪掉。注意到用户可能须要根据自己的晶体学知识对 EPC 施加额外的限制，在生成 EPC 时须要支持额外设置各 Wyckoff 位置占据数的上下限，包括对单个元素占据数的限制和对所有元素占据数之和的限制；显然，这里设置的上限也可以通过剪枝实现，其剪枝策略和上述策略完全类似，只是在生成单元素可行 EPC 时剪枝策略中不须要考虑其它元素单独的占据数限制。

总 EPC 的生成也可以建模为树搜索，只是此时树中的节点须要换成单元素可行 EPC。例如在搜索  $\text{Al}_2\text{O}_3$  的总可行 EPC 时（图 3.5），程序首先会搜索  $\text{Al}^{3+}$  的可行 EPC，于是找到  $a_1b_1$ ；其接着在  $\text{Al}^{3+}$  占据  $a_1b_1$  的前提下搜索  $\text{O}^{2-}$  的可行 EPC，但因  $6d$ 、 $6e$  位置占据数上限为零的约束<sup>3</sup>而找不到，于是回溯到对  $\text{Al}^{3+}$  可

<sup>3</sup> 这显然和  $\text{Al}_2\text{O}_3$  实际的 EPC 不符；这样设定纯粹是为了方便演示。

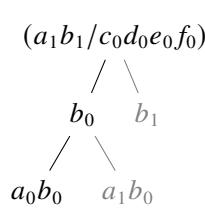


图 3.4 单元素 EPC 的生成 (注意剪枝时须考虑其它元素 EPC 中各 Wyckoff 位置被占据的次数)

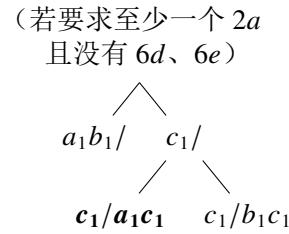


图 3.5 总 EPC 的生成 (注意其中以可行的单元素 EPC 作为树中的节点)

行 EPC 的搜索。类似于以上的步骤，程序最终会搜索到指定约束下所有的总可行 EPC。至此，我们还没有考虑的问题只剩占据数的下限约束，其中单元素下限约束显然可以在搜索单元素可行 EPC 时通过剪枝实现；总占据数的下限约束因为无法在搜索单元素可行 EPC 时实现，所以只能在生成总可行 EPC 时通过对结果的额外筛选实现，例如图中  $c_1/b_1c_1$  就是因为  $2a$  位置至少占据一次的约束而被筛除的。

有必要指出，由本小节可见，增量计算不仅可以用于降低算法的时间复杂度，而且在特定的条件下也可以用于降低算法的空间复杂度。此外，*decryst* 使用的 EPC 生成算法在性能上未必优于 *EPCryst* 中的算法，但正空间法的性能瓶颈并不在 EPC 的生成上，因此用其性能上的降低来换取整套软件在内存需求上的极大缩减还是很有利的。最后，本人将本小节的算法归类为增量算法，其理由有两条：第一，这一算法不再使用一个分离的步骤专门生成单元素可行 EPC，而是利用树搜索模型动态地生成总可行 EPC；第二，在这一算法的具体实现中，*decryst* 使用的是迭代式而非递归式的树搜索，因此程序会从前一 EPC 计算后一 EPC，而这正是增量式的计算。

## 3.2 *decryst* 设计简介

### 3.2.1 *decryst* 中的并行化

增量计算的基础是数据之间依赖关系的相对独立性，例如图 3.1 中 `part0.c` 变更后其余 `.c` 文件不需重新编译的根本原因在于数据的变更只影响反向依赖（以及反向依赖的反向依赖，等等）。实际工作中遇到的依赖关系往往更加复杂多变，例如像图 3.6 中所示的情形；幸运的是，*decryst* 中许多关键计算环节（例如第 3.1.2 小节中分析的那些）的依赖关系可以归纳为（图 3.7）一组互相独立的计算结果（图中的  $f(x_i)$ ）被集中汇总（图中的“ $\oplus$ ”操作）。事实上，这种相对独立

性在很大程度上也是并行化的基础：例如要在 make 程序构建软件项目时启用了并行选项，那么 make 会根据依赖关系并行地执行互相独立的计算步骤。对于图 3.7 中的依赖模式，其中被汇总的各项数据在一定前提下是互相独立的，因此可以并行地计算；这正是目前在并行和分布式计算中被广泛应用的 MapReduce (Dean、Ghemawat 2004) 框架的基础，而 *decryst* 中的并行化也是基于这一模式。

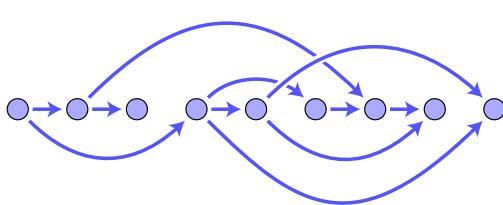


图 3.6 复杂依赖关系示例

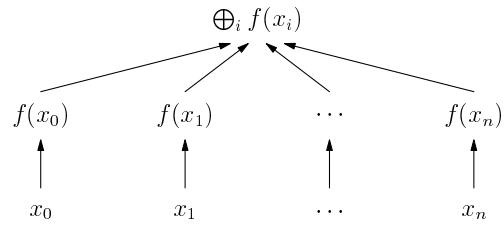


图 3.7 MapReduce 式依赖关系

类似于 *EPCryst* (Deng、Dong 2011)，*decryst* 也使用等效点系组合 (EPC) 法处理已指标化的数据，并把求解流程分为生成 EPC 列表、对各 EPC 进行统计分析、对每个 EPC 进行全局最优化和导出解模型等 4 个主要步骤。因为各 EPC 上的任务互相独立，所以 *decryst* 可以并行地处理这些任务（图 3.8）。Deng、Dong (2011) 提到，*EPCryst* 在 EPC 数较小（例如少于 100 个）且每个 EPC 的维数不超过 10 时特别适用；相比之下，因为 *decryst* 中增量计算的应用（参考第 3.1 节）以及 EPC 任务的并行化，其可以处理上千个 EPC，每个 EPC 的维数可高达 20 以上。考虑到 EPC 法可以极大缩减搜索空间，以及成键关系大体未知的结构 EPC 数往往很大，可以认为 EPC 任务的大规模并行化将为这类结构的求解带来前所未有的机遇。

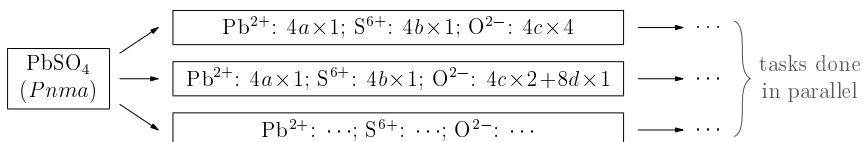


图 3.8 EPC 的并行处理

对于维数很高的 EPC，*decryst* 也可以使用并行的全局最优化，其中应用的是 Chu 等 (1999) 的并行模拟退火算法，而后者又是基于 Lam、Delosme (1988) 的自适应模拟退火算法。Lam 算法使用连续降温，其中降温速率由若干统计参数动态控制，而这些参数是根据相应的一套统计估计函数 (statistical estimators) 周期性地（每  $\tau$  个最优化循环）计算得到的；此外为了尽量发挥最佳性能，最优

化中的扰动规模也是动态控制的，以使随机晶体模型的接受率保持在 0.44 左右。Chu 等 (1999) 注意到模拟退火的一种物理图像是对 Boltzmann 分布的抽样，而多个抽样进程之间可以通过周期性的状态混合来实现关联（图 3.9）；具体而言，在温度  $T$  下选取某一进程中最近状态的概率为

$$p_j = e^{-E_j/T} / \sum_j e^{-E_j/T}, \quad (3.5)$$

其中  $E_j$  为进程  $j$  中最近的目标函数值。利用对各进程最近状态的周期性混合，Boltzmann 抽样的操作可以被分配到多个进程上进行，由此便可实现对模拟退火的并行化。

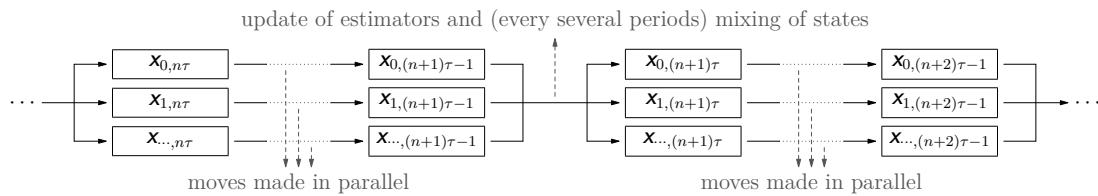


图 3.9 模拟退火的并行化

考虑到 *decryst* 需要处理的实际需求，本人对原先并行模拟退火算法中的一些技术细节进行了细微的改动。首先，因为晶胞具有周期边界，*decryst* 中使用的搜索空间不是无限的空间，而是具有周期边界的超立方体，即多维环面（参考第 2.1.1 小节）；相应地，最优化中的移动步长不是从带有随机符号的指数分布生成，而是从带有随机符号的周期指数分布（wrapped exponential distribution）生成。其次，因为 *decryst* 会随机地初始化独立原子的坐标，所以原先算法中用于消除初态痕迹的在无限温度下进行的多次初始循环不再需要；此外因为 *decryst* 中有专门的统计分析步骤，所以用于计算统计估计函数值的多次初始循环也不再需要。最后，为了修复或改善 *decryst* 在实际测试中的一些问题，本人还对原先算法本身进行了几项调整：

- 本人发现在最优化的初期，原先的算法有一定概率遇到降温过快的问题；为了解决这一问题，*decryst* 硬性设定了模拟退火时两个相邻最优化循环中温度之比的上限，从而避免降温过快。本人也注意到，在最优化问题的维数很大时，特定统计参数的估计函数值可能被计算为浮点数中的 Inf (infinity) 或 NaN (not a number)，导致算法异常终止；为了解决这一问题，相应估计函数被适当修改<sup>4</sup>，以尽量消除计算得到的 Inf 和 NaN。此外

<sup>4</sup> 具体而言，是文献 (Lam、Delosme 1988, p. 10) 中的  $A$ 、 $D$ ；同一页中的  $\hat{\mu}$  从代数上看也可能出现类似问题，因此也有类似的修改。

有必要指出，上述避免降温过快的机制在其中也有所帮助，因为降温过快是出现 Inf 和 NaN 的一种触发条件。

- 为了避免扰动规模  $\theta$  在接近零时变化过于剧烈，*decryst* 中使用了基于倍数的动态控制（其中  $\rho_0$  为最近  $\tau$  个最优化循环所产生随机晶体模型的接受率）

$$\theta_i = \theta_{i-\tau}(\rho_0 - 0.44 + 1), \quad (3.6)$$

而不是原先基于加和的动态控制

$$\theta_i = \theta_{i-\tau} + (\rho_0 - 0.44). \quad (3.7)$$

- 在并行化中，每个进程的降温速率须为串行降温速率乘以并行进程数：例如假设串行温度序列为  $T_0, T_1, T_2, \dots$ ，则原算法中双进程时每个进程的温度序列将为  $T_0, T_2, T_4, \dots$ 。然而注意到一些温度值会在降温中被跳过，例如上述例子中的  $T_1, T_3, T_5, \dots$ ，而这样可能造成统计估计函数值和实际值偏差过大，本人因此改为将串行温度序列分配到各进程，例如双进程时两个进程的温度序列将分别为  $T_0, T_2, T_4, \dots$  和  $T_1, T_3, T_5, \dots$ 。
- 假设最优化问题的维数为  $n$ ，则每个最优化进程在进行  $n$  个最优化循环之后会将  $n$  个自变量的更新顺序打乱，从而消除各自变量上扰动之间的关联。这样的顺便带来的一个效果是每个进程中更新统计估计函数值的周期（即  $\tau$  除以进程数得到的值）不再必须是  $n$  的倍数。
- 原先的并行化实现混合地使用了服务器-客户端模式和点对点模式来实现计算节点之间的通信。考虑到点对点模式在这一算法中对性能影响并不大，本人在 *decryst* 的全局最优化步骤中统一采用了服务器-客户端模式，以本机为进行汇总操作的客户端、计算节点为进行独立计算的服务器端，利用 ZeroMQ 库进行通信（参考第 3.2.2 小节）。事实上，这样也更接近于 MapReduce 的做法。

### 3.2.2 *decryst* 的软件架构

*decryst* 是自由、开源的软件，可以从 <https://gitea.com/CasperVector/decryst> 获得。考虑到本人能力和精力有限，难以承担“重复发明轮子”的成本，本人在开发 *decryst* 时尽量将其设计得简洁、灵活，从而可以关注其核心功能的开发。本人衷心希望其中应用的各种技术能够被晶体学同行研究学习，并最终在更多的晶体学软件中得到实际的应用，由此促进晶体学软件在自动化水平和性

能上的整体进步。*decryst* 由若干个部分组成（图 3.10），各部分的基本分工和代码组织将在本节中进行简单的介绍，而 *decryst* 的具体使用方法将在第 4 章中进行展示。

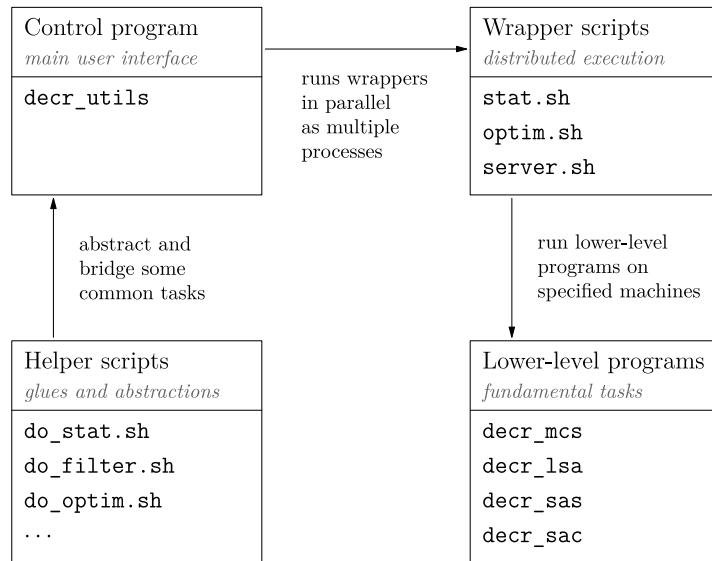


图 3.10 *decryst* 的软件架构

*decryst* 中最基础的是一套底层程序（lower-level programs），其中每一个程序实现一项底层功能：统计分析，全局最优化，等等。为了尽量发挥最佳性能，底层程序以 C 语言写成，其代码位于 *decryst* 目录树中的 `src` 目录：

- **decr\_mcs** (**M**onte **C**arlo **s**tatistical **a**nalysis):  
实现了对单个等效点系组合 (EPC) 的统计分析。
- **decr\_lsa** (**l**ocal **s**imulated **a**nnealing):  
实现了对单个 EPC 的串行模拟退火。
- **decr\_sas** (**s**imulated **a**nnealing **s**erver):  
实现了单 EPC 并行模拟退火的服务器端。
- **decr\_sac** (**s**imulated **a**nnealing **c**lient):  
实现了单 EPC 并行模拟退火的客户端。

建立在底层程序之上的是一个上层的主控程序（control program），其和底层程序一起构成了 *decryst* 的核心。主控程序调用各底层程序并和它们协作，此外一些较为复杂但对软件总体性能影响不大的功能（特别是 EPC 的生成）也是在主控程序中实现的。为了最大化其灵活性，主控程序以 Python 语言写成，其代码位于 *decryst* 目录树中的 `python` 目录：

- **decr\_utils**: 即主控程序本身；为了方便安装，本人有意将程序代码集中

在同一文件里，但对其各部分代码进行了较为清晰的组织。

- `wyck.json`: 各空间群及其中各 Wyckoff 位置的相关数据。
- `asf.json`: 各元素的原子散射因子数据（参考第 3.1.1 小节）。
- `rad.json`: 各元素的“原子”半径（含离子半径）数据。

包装脚本（wrapper scripts）根据给定的控制参数在指定的计算节点<sup>5</sup>上执行相应的底层程序。主控程序在调用底层程序时，事实上是通过并行地调用包装脚本，从而使得相应的底层程序在指定的节点上执行，由此实现并行和分布式计算。包装脚本是普通的 Unix shell 脚本，其代码位于 `decryst` 目录树中的 `doc/scripts` 目录：

- `stat.sh`: 用于在节点上执行 `decr_mcs`。
- `optim.sh`: 用于在节点上执行 `decr_lsa`。
- `server.sh`: 用于在节点上执行 `decr_sas`。
- `ssh.sh`: 以上脚本均调用该脚本以实现对远程机器的访问，其具体实现会调用 Unix 中的 `ssh`（secure shell）程序。

助手脚本（helper scripts）连接不同的计算步骤，并提供上层的抽象：例如 `do_rank.sh` 会根据统计分析或全局最优化的结果，利用 Deng、Dong (2011) 提出的品质因数（figure of merit，简称 FOM）对 EPC 进行排序，从而为各 EPC 的筛选以及在后续处理中排名的确定提供依据。和包装脚本类似，助手脚本也是 shell 脚本，而其代码也位于 `decryst` 目录树中的 `doc/scripts` 目录；因为助手脚本的功能较为多样，且部分脚本之间还有互相调用的关系，所以此处不再列出其完整列表。本人有意将包装脚本和助手脚本放在 `doc` 目录下，原因是它们是 `decryst` 灵活性的一个主要来源：在 `decryst` 目前总共约 5000 行（不计数据文件）的代码中，包装脚本和助手脚本只用了约 100 行来实现，但在求解实际结构时用户直接调用的主要是助手脚本，而包装脚本又是 `decryst` 实现分布式计算的真正基础<sup>6</sup>。虽然这两类脚本已经可以直接应用于实际的结构求解，但是本人希望用户能自己读懂其中的代码，从而可以在现有脚本的基础之上进一步定制 `decryst`，使之在结构测定的自动化中最大限度地发挥作用。

`decryst` 的软件文档位于其目录树中的 `doc` 目录：

- `scripts` 子目录下是示例脚本，即上述的包装脚本和助手脚本。

<sup>5</sup> 因为 `decryst` 中客户端在统计分析和全局最优化时的计算负担一般并不太大，所以一台机器可以同时充当客户机和计算节点。

<sup>6</sup> 如下文所述，ZeroMQ 只在并行模拟退火中使用，而且事实上并行模拟退火中也使用了 `server.sh` 来实现对服务器端的远程控制。

- `usage` 子目录下是软件的用法说明，用户应首先阅读其中的 `guide.txt`。
- `examples` 子目录下是软件用法说明中涉及的示例数据。

*decryst* 还包含了几个测评程序 (benchmark programs)，包括第 2.2.3 和 3.1.2 小节中测评部分所涉及的一些程序，以及一些对软件正确性的内部测试。因为须要对代码的内部结构和微观性能进行较为准确的测试，测评程序以 C 语言写成，其代码位于 *decryst* 目录树中的 `bench` 目录：

- `bench_bump` 实现了第 2.2.3 小节中的性能测试。
- `bench_cryst` 实现了第 3.1.2 小节中的性能测试。
- `bench_metric` 主要是对第 2.3.1 小节中若干猜想的尝试性验证。
- `bench_rbtree` 是对 *decryst* 在碰撞检测中所用红黑树 (red-black tree; 参考 Cormen 等 2009, pp. 308–338) 实现正确性的测试，因为红黑树是一个在实现时容易出细节错误的数据结构。

*decryst* 为类 Unix 系统设计，而且一般不需要太多改动就能在类似于 Cygwin 的模拟环境中运行<sup>7</sup>；除此之外，其正确编译和安装还需要一个满足 POSIX 标准的开发环境，而 `decr_sas`、`decr_sac` 两个底层程序还需要 ZeroMQ 库。为了能运行本文中的示例，用户还需要 Python、ssh 和满足 POSIX 标准的 Unix 工具；更加详细的安装说明可以参考 *decryst* 根目录下的 README 文件。

### 3.3 讨论和小结

#### 3.3.1 关于 *decryst* 性能的讨论

对于具有反演对称性的晶胞，*decryst* 可以利用 (3.2) 式加速结构因子的计算，从而提升最优化的性能：注意到此时只须计算半数原子所对应结构因子的实部，计算结构因子时的性能显然是不用此公式时性能的 4 倍；因此在使用 *decryst* 求解结构时，如果晶胞原点可选，应尽量使原点位于一个反演中心上。除此之外，对于菱方晶系的结构，因为使用六方坐标系时晶胞内总原子数是使用菱方坐标系时的 3 倍，所以为了尽量减少 *decryst* 在计算原子重叠评估函数时须要进行碰撞检测的原子对数，我们应当尽量采用菱方坐标系。

在对测量得到的衍射数据进行指标化以及确定空间群时 (Altomare、Giacovazzo 等 2008)，根据系统消光的规律对不符合实际结果的空间群进行筛选是必不

---

<sup>7</sup> 如上文所述，虽然 *decryst* 的核心部分应该不难移植到 Windows 平台，但是其中的包装脚本和助手脚本是普通的 Unix shell 脚本，因此也须要调用传统的 Unix 工具；这些工具在 Windows 下通常是由 Cygwin 等模拟环境提供，因此 *decryst* 在 Windows 下一般需要这些模拟环境。

可少的步骤；相应地，在此步骤之后得到的指标化数据，例如 FullProf (Rodríguez-Carvajal 2001) 产生的 .hkl 文件中，往往包含了零强度  $hkl$  的数据。由 (2.22) 式可知在计算 Bragg  $R$  因子时，如果应发生系统消光的一些  $hkl$  实际测得的强度确实为零，那么在计算中忽略这些  $hkl$  不会对得到的  $R$  值产生影响。因此，在将指标化数据转换到方便 *decryst* 读取的格式时，我们可以利用一些自动化工具删除数据中符合消光规律的零强度  $hkl$ ，从而一定程度地提升最优化的性能；类似地，在适当的条件下，可以酌情忽略高角度的  $hkl$ 。

考虑  $R\bar{3}c$  空间群（菱方坐标系，忽略反演对称）的对称操作：

$$(x, y, z), (z, x, y), (y, z, x), \\ (\bar{z} + \frac{1}{2}, \bar{y} + \frac{1}{2}, \bar{x} + \frac{1}{2}), (\bar{y} + \frac{1}{2}, \bar{x} + \frac{1}{2}, \bar{z} + \frac{1}{2}), (\bar{x} + \frac{1}{2}, \bar{z} + \frac{1}{2}, \bar{y} + \frac{1}{2}) \quad (3.8)$$

注意到上述坐标中有大量重复的表达式，例如  $x$ 、 $y$ 、 $z$ 、 $\bar{x} + \frac{1}{2}$ 、 $\bar{y} + \frac{1}{2}$ 、 $\bar{z} + \frac{1}{2}$  分别出现了 3 次，我们似乎可以通过将表达式分组，然后在其自变量发生改变时只求值一次，后续用到时直接使用求出的值。在 *decryst* 的全局最优化步骤中，考虑到两个相邻的最优化循环里晶体模型之间的差别一般只在于单个独立原子中一个坐标的改变，这样似乎会比较有利：除了在计算结构因子时的便利之外，在使用 sweep and prune (SAP，参考第 2.1.2 小节) 时，利用上述分组，我们可以只对受坐标变动影响的那些投影区间重新进行碰撞检测。然而，在每次计算目标函数时，原子坐标事实上是在被计算一次之后便用于计算许多  $hkl$  的结构因子，因此原子坐标的计算开销在目标函数的计算开销中所占比例不大；此外如 3.1.2 小节所述，*decryst* 事实上并未使用 SAP，因此以上性能调优技术带来的性能提升非常有限，所以最终未被采用。

### 3.3.2 关于浮点数求和的讨论

如第 3.1.2 小节所述，*decryst* 目前在增量求和时通过在内部使用定点数的方式来绕过浮点数求和的误差积累问题；这样做并不完美，因为定点数精度也有限，在加法超过其上界时会溢出到下界（相应地，减法可能会溢出到上界）。为了防止溢出，*decryst* 对浮点数  $x_{\text{float}}$  到定点数  $x_{\text{fix}}$  之间的换算常数  $c \approx x_{\text{fix}}/x_{\text{float}}$  进行了限制，以保证求和结果的定点数表示即使在最坏情形下也不会发生溢出。然而，因为

$$c \approx M_{\text{fix}} / \sum_i |x_{\text{float},i}|_{\max} \quad (3.9)$$

(其中  $M_{\text{fix}}$  为定点数所用整数类型绝对值的上界)，所以  $c$  的值在被求和数的个数很大时会很小，从而增大换算时的截断误差；这在晶胞内总原子数  $n$  很大时尤其值得注意，因为此时被求和数的个数是  $n(n - 1)/2$ 。

另外一种可能可行的思路是直接使用增量的浮点求和，但周期性地从头求和，从而避免求和误差的无界积累；事实上，*decryst* 在并行模拟退火（参考第 3.2.1 小节）中进行状态混合时，在很大程度上已经是从头计算晶体模型的目标函数了。然而，在各种条件下如何妥善确定从头求和的周期，还有这一周期和 Lam、Delosme (1988) 的算法中统计估计函数的更新周期以及 Chu 等 (1999) 的算法中进行状态混合的周期之间的关系应当如何把握，这些问题在应用上述思路之前都需要仔细考虑。

使用高精度的浮点数可以大大减轻浮点数求和中的误差积累问题，但在 *decryst* 中仍无法完全根治这一问题，因为其中浮点数增量求和的误差积累是无上界的，只是在用高精度浮点数时每次求和的相对误差被很大程度地缩小了。当然，如果将高精度浮点数和以上周期性从头求和的思路同时使用，对误差积累进行有效控制仍是很有希望做到的；类似地，对于上文所述晶胞内总原子数  $n$  很大的情形，如果以高精度的整数为定点数的基础，截断误差对最终计算结果的影响应该也是可以控制在一个满意的范围之内的。

### 3.3.3 关于等效点系组合生成算法的讨论

如第 1.3 节所述，在早期的正空间法工作中，晶体学家以极其具有技巧性的方式对等效点系组合 (EPC) 进行筛选。例如在陆学善、梁敬魁 (1965) 关于  $\text{V}_2\text{Ga}_5$  ( $P4/mbm$ ,  $Z = 2$ ) 的工作中，作者首先根据晶胞化学式和各 Wyckoff 位置的多重度列出了可能的多重度组合：

- |                     |                                 |
|---------------------|---------------------------------|
| A. $8 + 4 + 2;$     | D. $4 + 4 + 2 + 2 + 2;$         |
| B. $8 + 2 + 2 + 2;$ | E. $4 + 2 + 2 + 2 + 2 + 2;$     |
| C. $4 + 4 + 4 + 2;$ | F. $2 + 2 + 2 + 2 + 2 + 2 + 2.$ |

因为固定 Wyckoff 位置只有 4 个，所以组合 E、F 首先被排除；接下来，因为  $c$  方向空间有限， $2a$ 、 $2b$  互斥且  $2c$ 、 $2d$  互斥，所以组合 B、D 也被排除。考虑到  $c$  方向只能容纳一个原子， $4e$ 、 $4f$  不可用，在多重度为 4 的 Wyckoff 位置中只能选择  $4g$  或  $4h$ ，而这导致组合 C 在  $\{220\}$  平面一定会发生原子重叠，所以组合 C 也被排除。对于组合 A，在排除等价 EPC 之后，作者根据特定衍射峰的强度对剩余的可行 EPC 进行了最终的筛选。有必要指出，在不考虑除固定位置约束之

外的原子重叠因素时,  $V_2Ga_5$  总共有 428 个可选 EPC, 因此仅通过人工计算将其 EPC 筛选至只剩一个已经可谓是一项惊人的工作。

以上的技巧并未完全在 *decryst* 中得到应用。首先, 以上先生成多重度组合再生成 EPC 的思路并未在 *EPCryst* (Deng、Dong 2011) 或 *decryst* (参考第 3.1.3 小节) 中采用, 这是因为以上思路实现起来明显更加复杂, 但在算法的时间和空间复杂度上似乎并没有很大的改善。本人认为这并不说明以上思路已经过时, 而是说明其在当前的应用场景下没有充分发挥优势; 在其它一些应用场景下, 这一思路仍有可能体现其独特的价值。此外, 以上筛选中用到了涉及多于一个 Wyckoff 位置的占据数约束, 例如“ $2a$ 、 $2b$  位置总共最多占据一次”, 而 *decryst* 目前尚不支持这样的约束。

此外有必要指出, 利用第 3.1.3 小节中剪枝的思路, 从原则上来看应该是可以在生成 EPC 时根据原子重叠等因素对其进行实时筛选的, 关于这一点的详细讨论可以参考第 4.3.1 小节。

### 3.3.4 其它讨论

对于在正空间法中主要以原子坐标作为全局最优化中自变量的结构测定软件, 例如 *FOX* (Favre-Nicolin、Černý 2002)、*FraGen* (Li 等 2012) 等, 利用类似于第 3.1.2 小节所述的方法应该不难实现目标函数的增量计算, 其中也包括比 (2.23) 式更为复杂的一些目标函数 (例如考虑了配位数、键价、配位多面体形状等等因素的那些)。然而, 增量计算对于主要面向分子晶体的正空间法软件可能并不太适用, 因为这一类软件中大量使用键长、键角等等作为最优化的自变量, 而这常常会导致单个自变量的变化影响到所有原子的坐标。

尽管多数晶体学软件并不采用等效点系组合法, 可完美并行的任务仍是广泛存在的: 例如第 4 章中将介绍须要多次执行全局最优化步骤, 并在其所得解中挑选最优解的复杂结构; 显然, 各次最优化之间是完全独立的, 因此类似于第 3.2.1 小节中那样可以完美并行。最优化算法的并行化也具有重要意义: 例如结构精修中最优化算法的并行化将可以明显缩短用户等待精修结果的时间, 而本人注意到 Eremenko 等 (2017) 近期的工作已经在往这一方向努力; 此外, 一些最优化算法本身就比较容易并行化, 例如 *FOX* 中采用的并行回火算法。

本人最近也注意到, 基于 Chu 等 (1999) 算法中 Boltzmann 状态混合的思路, Lou、Reinitz (2016) 近期提出了一种新的并行模拟退火算法。新算法在实现上明显比原算法要简单, 但却有更高的并行效率; 但为了提升并行效率, 其不再使

用自适应的模拟退火，这使其串行性能并不优于原算法。在未来版本的 *decryst* 中，本人有可能采用上述新算法来实现并行的全局最优化；当然，并行回火等算法也将在考虑之列。

### 3.3.5 本章小结

基于第 2 章中的原子重叠排除机制，本人开发了 *decryst* 这一套利用正空间法和等效点系组合（EPC）法从指标化的粉末衍射数据求解晶体结构的软件。受 Unix 中 `make` 程序的启发，本人在 *decryst* 中首次将增量计算的思想以一种具有通用性的方式应用于全局最优化步骤中，使其性能得到了明显的提升；*decryst* 也使用一种增量算法生成 EPC，这不仅极大降低了其内存需求，而且为在生成 EPC 时对其进行实时筛选做了准备。根据 *decryst* 中计算步骤的相关属性，本人在其中加入了对并行和分布式计算的支持，使之可以通过同时利用多个处理器实现对晶体结构测定的进一步加速；考虑到同一结构的各 EPC 互相独立，而且 EPC 数在多数有意义的情形下都相当大，*decryst* 中统计分析和全局最优化任务的并行化将为求解成键关系总体未知的结构带来前所未有的机遇。

‘Ο βίος βραχύς, ἡ δὲ τέχνη μακρή.

(*Vita brevis, ars longa.*)

— Greek aphorism



## 第 4 章 *decryst* 的使用方法和常用技巧

本章将以 AMCSd 数据库 (Downs、Hall-Wallace 2003) 中若干个不同晶系和复杂度的结构为例, 循序渐进地演示 *decryst* 的基本用法和常用技巧; 目前 *decryst* 中自带了  $\text{Al}_2\text{O}_3$  (AMCSd 代号: 0009325)、 $\text{As}_2\text{S}_3$  (0010733)、 $\text{PbSO}_4$  (0005558) 和“0009563”等 4 个结构作为示例。此外, 因为有一位用户在 *decryst* 的试用阶段建议再增加几个不同晶系和复杂度的示例, 本人在 AMCSd 中搜索  $a$ 、 $b$ 、 $c$  均介于 8–16 Å 的结构, 在第一页结果中去除过于简单或过于复杂的结构后选取了“0000158”“0000219”“0000220”“0000428”“0000589”等 5 个结构。各代号所对应的化学式见相应小节; 在本章的演示中, 以上所有结构在求解时均忽略 H 原子<sup>1</sup>, 所有时间测量均在 Intel i7-3720QM CPU 上进行。

### 4.1 *decryst* 的基本用法

在本节中, 第 4.1.1 小节将介绍 *decryst* 中主控程序 (参考第 3.2.2 小节) 的输入格式, 第 4.1.2 小节将演示主控程序的用法, 以及用 *decryst* 进行结构测定的基本流程。在此基础之上, 第 4.1.3 小节将讨论对全局最优化步骤中得到的评价指标相近的解进行手工 (目视) 筛选的技术, 并演示结合自动化工具使用 *decryst* 的基本方法。第 4.1.4 小节将初步演示和讨论对等效点系组合 (EPC) 进行筛选的方法。

#### 4.1.1 *decryst* 中主控程序的输入格式

如第 3.2.2 小节所述, *decryst* 中面向用户的核心接口由主控程序 `decr_utils` 提供。因此为了使用 *decryst* 进行结构测定, 我们最先须要编写的是 `decr_utils` 的输入文件 (称为 **decr** 文件, 详细格式参考 `doc/usage/format.txt`)。首先以  $\text{Al}_2\text{O}_3$  结构为例, 其 `decr` 文件如下所示<sup>2</sup>:

```
# (Lines beginning with '#' are comments and are ignored.)
# ID of space group, (a, b, c), (alpha, beta, gamma).
167a    4.7602 4.7602 12.9933   90 90 120
# Combination factor 'mu', and some other control parameters.
```

<sup>1</sup> 有必要指出, 在结构精修中利用 X 射线衍射数据处理 H 原子是可行且有很强实际意义的, 相关研究可参考文献 (Cooper 等 2010)。

<sup>2</sup> AMCSd 中的衍射数据不含半高宽 (FWHM) 字段, 因此本章所有 `decr` 文件中 FWHM 值均为粗略的估计值。

```

0.25 0.75 0.875 0.5 1.0
# Global EPC constraints ('--' for the defaults).
--

# Atom species, number of atoms, EPC constraints, already
# guessed EPCs with unknown coordinates ('--' for none).
Al3+    12      --      --
O2-     18      --      --
# Pairwise zoom factors.
1.6:0-0
# Wyckoff positions and coordinates for already solved atoms.
#0@c1    0        0        0.14784

# Data for individual reflections:
# 2-theta, FWHM, hkl, multiplicity, intensity.
25.59   0.2     0 1 2   6       61.91
35.17   0.2     1 0 4   6       96.62
# (... further data lines left out for brevity ...)

```

其中指定各  $hkl$  所对应衍射数据的那些行以及指定空间群和晶胞参数的一行意义应该很明显，唯一需要注意的是空间群的代号须要根据 `wyck.json` 进行设定，例如上述示例中 167a 表示采用  $R\bar{3}c$  空间群的六方表示。晶胞参数之后的一行数据指定结构测定中使用的一些参数（详见 `format.txt`），其中最重要的是第 1 个和第 4 个：前者是组合因子  $\mu$ （参考第 2.2.2 小节），而后者是 Lorentz–极化因子公式<sup>3</sup>中的参数  $p$ ：

$$LP = \frac{(1-p) + p \cos^2 2\theta}{\sin 2\theta \sin \theta}. \quad (4.1)$$

$\mu$  等参数之后的一行数据指定对晶胞中总 EPC 的额外约束（参考第 3.1.3 小节），其写法形如 0-3 1-2，其中每个字段对应于一个 Wyckoff 位置 ( $a$ 、 $b$ 、…，其顺序根据 `wyck.json` 中指定空间群 Wyckoff 位置的列表确定) 总占据数的下限和上限。下限和上限都可省略，表示不对相应值设定额外约束；此外，`decr_utils` 会自动设定所有固定 Wyckoff 位置的占据数上限不超过 1。最后，如果所有 Wyckoff 位置的总占据数均无额外约束，那么设定这一约束的行可以被简写为 “--”，正如上述  $\text{Al}_2\text{O}_3$  的例子所示。

### 晶胞化学式和对单元素 EPC 的约束由形如

<sup>3</sup> 这一公式和国际晶体学表（Prince 2004, p. 620）中的公式是兼容的：事实上，对前者而言的  $p$  正是对后者而言的  $A/(1+A)$ 。

Al3+	12	--	--
------	----	----	----

的数据行指定。其中第 1 个字段是元素的代号，其半径<sup>4</sup>和散射因子数据分别会从 `rad.json` 和 `asf.json` 中读取；两者所需元素代号不同时可使用形如 `S6+;S` 的写法，而“混合元素”可用形如 `0.6Fe2+,0.3Fe3+` 的写法来指定。第 2 个字段是晶胞中相应元素的总原子数。第 3、4 个字段分别指定相应元素占据数的下限和上限，写法形如 `a2b1`（注意这和第 3.1.3 小节中所用 EPC 表达式的相似性）；其中未写出的 Wyckoff 位置均视作未设定额外约束，而所有 Wyckoff 位置均无额外约束时相应字段可写为“--”。

对缩放因子（参考第 2.1.2 小节，默认为 1）由形如

1.1:0,1-1,2	1.2:2-2
-------------	---------

的数据行<sup>5</sup>指定，其中每个字段中“：“之前为缩放因子的值，之后为被赋予相应缩放因子的原子对：`0,1-0,1`会被展开为 `0-0`、`0-1` 和 `1-1`。这里元素的编号根据晶胞化学式部分各元素被定义的顺序而定，从零开始计数：例如在上述的 Al2O3 示例中，`0-0` 表示的是 Al3+-Al3+ 原子对。已知位置的独立原子<sup>6</sup>由形如

0@c1	0	0	0.14784
------	---	---	---------

的数据行指定。在上述的 Al2O3 示例中，以上数据行表示第 1 种元素（编号为 0，即 Al3+）占据 1 组 *c* 位置，其中一个原子的坐标为 `(0,0,0.14784)`。

关于 *decryst* 所用各类输入文件的格式，最后须要指出一些小但值得注意的技术细节。首先，所有输入文件都只支持 Unix 中常用的 LF (`\n`) 换行符，而不支持 Windows 中常用的 CRLF (`\r\n`) 换行符，因此用户在 Windows 上编写的输入文件一般须要经过 `dos2unix` 等工具的转换。其次，输入文件中的空行（包括文件末尾的空行，例如上述 Al2O3 示例中的最后一行）具有特殊作用，不可随意添加或删除。最后，数据行中空白字符（空格和制表符 `\t`）可以混用，但建议在不同类型的行中有规律地使用空白字符，从而方便利用文本处理工具进行自动化的修改（参考第 4.2.1 和 4.2.3 小节）。

## 4.1.2 基本求解流程：以 Al2O3 和 As2S3 为例

如第 3.2.1 小节所述，*decryst* 将结构测定划分为生成 EPC 列表、对各 EPC 进行统计分析、对每个 EPC 进行全局最优化和导出解模型等 4 个主要步骤。这

<sup>4</sup> 中性原子采用共价半径（Cordero 等 2008），而离子采用离子半径（Shannon 1969）。

<sup>5</sup> 即输入文件中除了空行和（以“#”开头的）注释行之外的那些行。

<sup>6</sup> 只有部分坐标已知的独立原子也可以用这种数据行表示，详见 `format.txt`。

里先继续以  $\text{Al}_2\text{O}_3$  的结构测定为例；在编写完 `decr` 文件之后，用以下命令即可生成 EPC 列表：

```
$ decr_utils comb /path/to/Al203.txt | tee combs.list
```

其输出（称为 **combs** 文件，详见 `format.txt`）应如下所示：

```
> 0@a1b1,1@d1.cr      0@a1b1,1@d1
> 0@a1b1,1@e1.cr      0@a1b1,1@e1
> ...
```

其中两个字段分别为该 EPC 所对应的文件名和表达式。有必要特别提请注意的是，`combs` 文件中的 EPC 只考虑未分配到任何 Wyckoff 位置的原子，而且 `decr` 文件中已知 Wyckoff 位置的两种原子是被分开计数的：例如要是 `decr` 文件中指定了某元素的占据数上限约束为 `c3`、下限约束为 `c1`，且该元素还有一个坐标已知的 `c1` 占据，那么该元素在 `combs` 文件列出的 EPC 中最多会占据 1 组 `c` 位置。此外，如果所有原子均已在 `decr` 文件中分配到确定的 Wyckoff 位置，那么在此条件下只可能生成一个“空的”EPC，而该 EPC 会被显示为 `nil`。有了 `combs` 文件，我们就可以实际生成各 EPC 所对应的用于 *decryst* 中底层程序的输入文件<sup>7</sup>（称为 **crysts** 文件<sup>8</sup>）：

```
$ decr_utils dump /path/to/Al203.txt < combs.list | tee crysts.list
```

其输出（称为 **crysts** 文件，详见 `format.txt`）应如下所示：

```
> 0 0 - - - - 0@a1b1,1@d1.cr
> 1 1000 100 0.01 - - - 0@a1b1,1@e1.cr
> ...
```

在 `crysts` 文件中，最后一个字段是 `cryst` 文件的文件名（可以注意到相应的文件已在当前目录生成），第一个字段是相应 EPC 的维数；第 2–4 个字段分别为统计分析中的样本数和 Lam 算法（参考第 3.2.1 小节）中的  $\tau$ 、 $\lambda$  参数<sup>9</sup>（后者取更小的值时，最优化会更精细但更耗时）；第 5–7 个字段分别为相应 EPC 所产生晶体模型目标函数值  $E$  的平均值、标准差和已知最佳值（目前均显示为“-”，表示这些参数的值未知）。

<sup>7</sup> 这里的 EPC 文件名和下文中各 EPC 的全局最优化参数都是可以修改的；分两步生成 `cryst` 文件是为了在分步求解复杂结构时（例如在使用重原子法时，参考第 4.2 节）方便管理各步骤所需的输入文件。

<sup>8</sup> 本文中的各个示例不涉及其格式，因此本文不对其进行讨论，但其细节可参考 `format.txt`；其余未注明“详见 `format.txt`”等字样的输入文件也类似。

<sup>9</sup> 其中零维 EPC 的样本数显示为 0（实际为 1）是出于方便自动化的考虑；这些 EPC 的  $\tau$  和  $\lambda$  显示为“-”，表示这两个参数无意义，因为这些 EPC 不需要全局最优化。

*decryst* 在对 EPC 进行统计分析和全局最优化时支持并行和分布式计算，而相应的设定是通过 **hosts** 文件进行的，其详细格式参考 `format.txt` 及其同一目录中的 `cmdline.txt`。`doc/example/hosts.conf` 是 *decryst* 自带的一个和 `doc/scripts` 目录中脚本兼容的示例 hosts 文件，用户可根据自身需求对其进行修改。此外，为了能正确调用包装脚本（参考第 3.2.2 小节），`scripts` 目录须要在搜索路径 `$PATH` 中，而且 `decr_mcs`、`decr_lsa` 和 `decr_sas` 须要安装到所有远程机器上。在正确进行以上配置之后，我们便可以对上一步生成的 `cryst` 文件进行统计分析：

```
$ decr_utils stat stat.sh /path/to/hosts.conf < crysts.list |
  tee crysts.stat
```

其输出应大致如下：

```
> 0 0 - - - 0.385611 0@a1b1,1@d1.cr
> 2 1000 200 0.01 0.530822 0.165276 0.045932    0@c1,1@e1.cr
> ...
```

可见平均值、标准差、最佳值已被实际值取代；对于零维 EPC，只有最佳值有意义，因此平均值和标准差仍显示为“-”。在获得统计分析的结果之后，我们便可以进行全局最优化：

```
$ decr_utils optim optim.sh /path/to/hosts.conf < crysts.stat |
  tee crysts.optim | sort -k7g
```

其输出应大致如下：

```
> 2 1000 200 0.01 0.530822 0.165276 0.0298237    0@c1,1@e1.cr
> 1 1000 100 0.01 0.473024 0.157376 0.209828    0@a1b1,1@e1.cr
> ...
```

注意其中最佳值已被替换为全局最优化步骤所得解模型的相应值，以及当前目录中生成了用于记录全局最优化过程的 `.log` 文件（称为 **crlog** 文件），如 `0@c1,1@e1.log`。此外因为零维 EPC 不需要全局最优化，所以 `crysts` 文件中的相应行会被 `decr_utils` 的 `optim` 子命令原样输出。

如上述输出（已用 `sort` 命令排序）所示，`0@c1,1@e1` 是 Al2O3 唯一可行的 EPC。最后，我们可以将全局最优化的结果和原 `decr` 文件合并，以产生各解模型所对应的 `decr` 文件和 CIF 文件：

```
$ decr_utils merge /path/to/Al203.txt < combs.list
```

其输出应如下所示：

```
> 0@a1b1,1@d1.cr
> 0@a1b1,1@e1.cr
> ...
```

正确解所对应的 decr 文件和 CIF 文件分别是 0@c1,1@e1.txt 和 0@c1,1@e1.cif，后者可以直接在可视化软件中查看。以上求解过程（除去人工步骤之后）耗时不到 1 s。

$\text{As}_2\text{S}_3$  ( $P2_1/c$ ) 的 decr 文件如下所示（注意其中因为缺少  $\text{As}^{3+}$ 、 $\text{S}^{2-}$  的原子散射因子数据而采用了电中性 As、S 原子的数据，参考第 4.1.1 小节）：

```
# sg, abc, angles; ctl, extra; limits.
14a      4.256 9.577 12.191      90 109.75 90
0.25 0.75 0.875 0.5 1.0
--

# rad/asf, n, limits, comb.
As3+;As 8      --      --
S2-;S 12      --      --

# 2theta, fwhm, hkl, mult, val.
12.04  0.1      0 1 -1  4      12.34
18.02  0.1      0 1 -2  4      15.82
# (... further lines omitted ...)
```

仿照以上步骤，其结构可以用完全类似的方法进行求解，除去人工步骤之后耗时约 15 s。在求解的过程中，可以注意到正确 EPC 0@e2,1@e3 耗时远多于其它所有 EPC，因为其维数（15）明显高于其它 EPC（均不超过 12）；此外， $\text{As}_2\text{S}_3$  有时一次全局最优化不能得到正确的解模型，需要多次求解。对于这类 EPC，并行的全局最优化可以用于缩短单个 EPC 求解所需的时间，例如对  $\text{As}_2\text{S}_3$  的 0@e2,1@e3 EPC 可以使用以下命令：

```
$ grep 0@e2,1@e3 < crysts.list |
decr_utils poptim server.sh hosts.conf | tee 0@e2,1@e3.log
```

需要注意的是 decr\_utils 的 poptim 子命令只接受单行的 crysts 文件，而且输出的是 crlog 文件而非 crysts 文件，因此用户须要自行管理其输出。

#### 4.1.3 实际求解：以“0000220”和“0000589”结构为例

“0000220”结构 ( $Ia\bar{3}d$ ) 的 decr 文件如下所示（注意其中一价原子被表示为  $\text{Li}^{1+}$ 、 $\text{Na}^{1+}$  和  $\text{F}^{1-}$ ，这和 wyck.json 是一致的，参考第 4.1.1 小节）：

```

# sg, abc, angles; ctl, extra; limits.
230      12.122 12.122 12.122    90 90 90
0.25 0.75 0.875 0.5 1.0
-- 

# rad/asf, n, limits, comb.
Li1+    24      --      --
Na1+    24      --      --
Al3+    16      --      --
F1-     96      --      --
# zooms.
1.4:2-2

# 2theta, fwhm, hkl, mult, val.
17.92   0.2      2 1 1    24      11.87
20.73   0.2      2 2 0    12      100.00
# (... further lines omitted ...)

```

该结构可以使用第 4.1.2 小节中的方式求解：其总 EPC 数为 28，单个 EPC 维数为 2–3，除去人工步骤之后求解总耗时约 5 s，可直接得到目标函数值  $E = 0.0278$  的正确解；正确 EPC 为 0@d1, 1@c1, 2@a1, 3@h1，而其余 EPC 所得解的  $E$  值至少为 0.11。

“0000589”结构 (*Pm $\bar{3}m$* ) 的 decr 文件如下所示：

```

# sg, abc, angles; ctl, extra; limits.
221      10.358 10.358 10.358    90 90 90
0.25 0.75 0.875 0.5 1.0
-- 

# rad/asf, n, limits, comb.
Li1+    1      --      --
K1+     6      --      --
Fe2+    24     --      --
S2-;S  26     --      --
Cl1-    1      --      --

# 2theta, fwhm, hkl, mult, val.
8.54    0.2      1 0 0    6      56.41
12.08   0.2      1 1 0    12     38.10
# (... further lines omitted ...)

```

该结构的总 EPC 数为 2416，单个 EPC 维数为 5–9。其依然可以按照上述各结构的方式求解，所得结果通常大致如下<sup>10</sup>：

```
> [...] 0.677702 0.0638962 0.0456528 0@b1,1@f1,2@m1,3@e1g1h1,4@a1.cr
> [...] 0.676874 0.0680791 0.0469421 0@b1,1@e1,2@m1,3@f1g1h1,4@a1.cr
> [...] 0.682297 0.0653026 0.0740492 0@a1,1@e1,2@m1,3@f1g1h1,4@b1.cr
> [...] 0.686445 0.060813 0.0749921 0@a1,1@f1,2@m1,3@e1g1h1,4@b1.cr
> [...] 0.697594 0.0678327 0.133365 0@a1,1@f1,2@m1,3@e1g1j1,4@b1.cr
> ...
```

从  $E$  值可见只有前 4 个 EPC 可能是合理的，又由  $(1/2, 1/2, 1/2)$  平移使  $Pm\bar{3}m$  空间群中  $a/b$ 、 $c/d$ 、 $e/f$ 、 $i/j$ 、 $k/l$  位置互变并保持其它 Wyckoff 记号不变可知这 4 个 EPC 是两对等价 EPC。在可视化软件中查看这两对 EPC 生成的解，可见在  $0@b1,1@f1,\dots$  和  $0@a1,1@e1,\dots$  的解中  $\text{Cl}^-$  均被  $\text{S}^{2-}$  配位（图 4.1(c,d)），这显然是错误的。在  $0@a1,1@e1,\dots$ （图 4.1(a)）和  $0@a1,1@f1,\dots$ （图 4.1(b)）的解中，即使不考虑后者  $\text{K}^+-\text{Fe}^{2+}$  键长过短的问题，只根据  $E$  值也可以知道前者是唯一合理的解。除去人工步骤之后，“0000589”求解总耗时约 5 min。

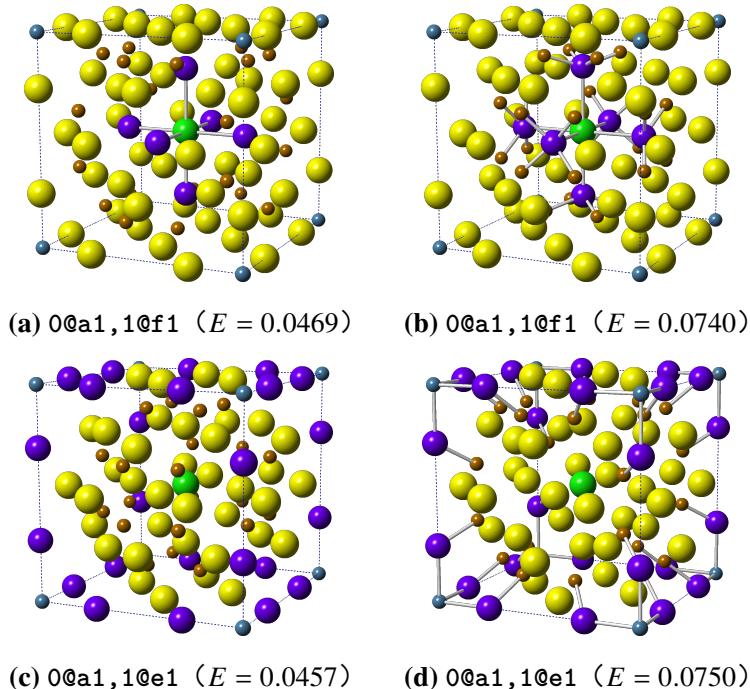


图 4.1 “0000589”的典型解（均已转换到使  $\text{Cl}^-$  在  $b$  位置）

至此我们已经求解了 4 个测试结构。显然，如果只是手动输入（或复制、粘贴）第 4.1.2 小节中的命令并加以修改，这样会是十分低效且容易出错的。在观

<sup>10</sup> 因为所使用算法中随机因素的缘故，实际结果中  $E \approx 0.075$  的 EPC 数可能发生一定程度的浮动；在使用 *decryst* 默认的参数时，其一般为 1–2。

察上述求解过程后，我们不难将其一般模式抽象为如下的 shell 脚本<sup>11</sup>（此处命名为 `solve.sh`）：

```
#!/bin/sh
decr_utils comb "$1" > combs.list
decr_utils dump "$1" < combs.list > crysts.list
decr_utils stat stat.sh hosts.conf < crysts.list > crysts.stat
decr_utils optim optim.sh hosts.conf < crysts.stat > crysts.optim
sort -k7g < crysts.optim > crysts.tmp && mv crysts.tmp crysts.optim
decr_utils merge "$1" < combs.list
```

这样只须将 `hosts.conf` 复制到当前目录，便可通过以下命令求解：

```
$ /path/to/solve.sh /path/to/some-cryst.txt
```

利用助手脚本（参考第 3.2.2 小节）可以实现更加复杂的自动化结构测定，相关内容将在第 4.2 节展示。

#### 4.1.4 EPC 筛选：以“0000428”和“0000219”结构为例

“0000428”结构 ( $R\bar{3}$ ) 的 `decr` 文件如下所示：

```
# sg, abc, angles; ctl, extra; limits.
148a    12.868 12.868 9.821      90 90 120
0.25 0.75 0.875 0.5 1.0
-- 

# rad/asf, n, limits, comb.
Pb2+    18      --      --
Cr6+;Cr3+    3      --      --
O2-     24      --      --
Cl1-    18      --      --
# zooms.
2:1-1

# 2theta, fwhm, hkl, mult, val.
12.01   0.1     1 0 1   6      20.45
13.76   0.1     1 1 0   6      45.28
# (... further lines omitted ...)
```

该结构的总 EPC 数为 88，单个 EPC 维数为 7–10，使用第 4.1.3 小节中的 `solve.sh` 脚本求解时单次执行耗时约 25 s。虽然这个结构看似简单，但是其求解并不十

<sup>11</sup> 限于主题和篇幅，本文不试图对 shell 脚本编程进行详尽的解释；不熟悉相关背景的用户可以参考文献 (Robbins、Beebe 2005)。

分直接，其中最重要的是直接使用 `solve.sh` 时结果的重复性出人意料地低：各 EPC 的排名并不稳定，且各 EPC 所得解的  $E$  值浮动也较大；根据本人的经验，这种情况通常是在维数大约为 15 的 EPC 中开始出现。无论如何，在多次执行 `solve.sh` 之后，不难发现排名最靠前的是以下 3 对等价 EPC ((0, 0, 1/2) 平移可使  $a/b$ 、 $d/e$  位置互变并保持其它 Wyckoff 记号不变)：

- 0@f1, 1@a1, 2@c1f1, 3@f1、0@f1, 1@b1, 2@c1f1, 3@f1:

典型  $E$  值为 0.057–0.060、0.064–0.066、0.075–0.100 或更大。

- 0@f1, 1@a1, 2@b1c2d1, 3@f1、0@f1, 1@b1, 2@a1c2e1, 3@f1:

典型  $E$  值为 0.097–0.099、0.103–0.121 或更大。

- 0@f1, 1@a1, 2@b1c2e1, 3@f1、0@f1, 1@b1, 2@a1c2d1, 3@f1:

典型  $E$  值为 0.113–0.114、0.120–0.125、0.162–0.168 或更大。

事实上，只须执行几次脚本便可注意到 1@a1, 2@c1f1, … 和 1@b1, 2@c1f1, … 所得解的最佳  $E$  值明显低于其它两对 EPC，因此正确的 EPC 无疑是这一对。收集其各  $E$  值的解，可发现  $E \geq 0.075$  的解较为多样（例如图 4.2(c) 中的解），但  $E \in [0.064, 0.066]$  的解（图 4.2(b)）和  $E \leq 0.060$  的解（图 4.2(a)）基本固定。考虑到后两个解从成键上看没有明显的问题，正确解应该是  $E$  值最小的。

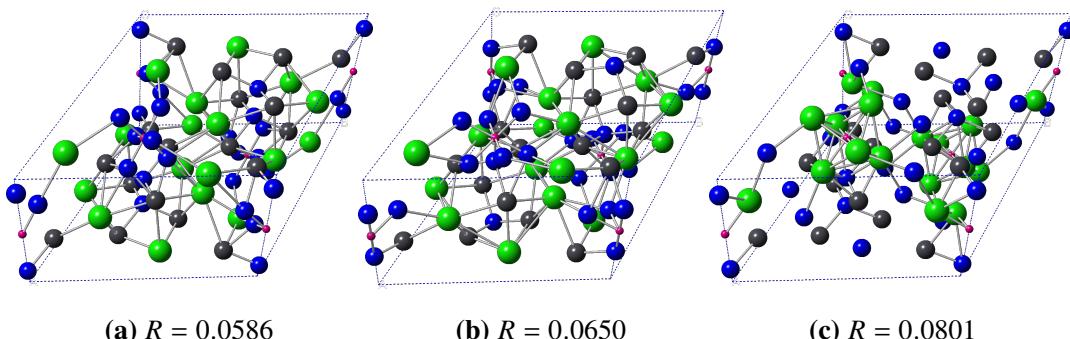


图 4.2 “0000428”的几个解，其中 (c) 只是相应  $E$  值区间多种典型解中的一个

“0000219”结构（原数据采用的是 *Pbna* 表示，求解前已转换<sup>12</sup>到 *Pbcn*）的 *decr* 文件如下所示：

```
# sg, abc, angles; ctl, extra; limits.
60      8.536 9.404 9.973      90 90 90
0.25 0.75 0.875 0.5 1.0
```

<sup>12</sup> 可以使用 Bilbao 晶体学服务器 (Aroyo 等 2006) 上的 SETSTRU 工具在同一空间群的不同表示之间转换，此外须特别注意 *hkl* 和晶胞参数都需要转换。顺便值得一提的是，用此工具还可以对独立原子的 Wyckoff 位置进行判断，从而方便计算测试结构的晶胞化学式，以及寻找独立原子在 *decryst* 所用不对称单元内等效原子的坐标。

```
--  
  
# rad/asf, n, limits, comb.  
P5+;P 8      --      --  
Fe3+ 12      --      --  
O2- 44      --      --  
# zooms.  
2:0-0 1.2:1-1 1.5:0-1  
  
# 2theta, fwhm, hkl, mult, val.  
14.01 0.1    1 1 0 4      3.05  
16.61 0.1    1 1 1 8      18.09  
# (... further lines omitted ...)
```

容易发现，原封不动地使用上述 `solve.sh` 脚本求解须花费很长时间（约 1.5 h），这不难理解：观察 `crysts.list` 可见虽然总 EPC 数不大（284），但是单个 EPC 的维数并不低（14–23），这导致每个 EPC（特别是那些接近或超过 20 维的）须要花费很长的时间来求解。在这样的情况下，我们可以根据统计分析的结果对 EPC 进行筛选：中断求解<sup>13</sup>，并执行以下命令

```
$ awk '{ print 1 " " (1 / $7 + 10 * $6 / $5) "\t" $0 }' \  
< crysts.stat > crysts.rank0
```

来根据 *EPCryst* (Deng、Dong 2011) 中使用的经验公式计算各 EPC 的品质因数 (FOM)。观察 `crysts.rank0` 可发现各 EPC 的 FOM 并没有非常明显的聚类现象，因此我们先尝试对前 1/4 的 EPC 求解：

```
$ sed 70q < crysts.rank0 | cut -f 2-3 > crysts.stat  
$ decr_utils optim optim.sh hosts.conf < crysts.stat > crysts.optim  
$ sort -k7g < crysts.optim > crysts.tmp && mv crysts.tmp crysts.optim  
$ decr_utils merge /path/to/0000219.txt < combs.list
```

观察 `crysts.optim`，可见最佳 EPC 通常大致如下（注意 (0, 1/2, 0) 平移可使 *Pbcn* 空间群中 *a/b* 位置互变并保持其它 Wyckoff 记号不变）：

```
> [...] 0.637264 0.0639186 0.0296934 0@d1,1@a1d1,2@c1d5.cr  
-> [...] 0.638173 0.0653523 0.0484107 0@d1,1@b1d1,2@c1d5.cr  
-> [...] 0.637305 0.0693133 0.0834741 0@d1,1@c1d1,2@b1d5.cr  
-> [...] 0.638933 0.0675984 0.0847271 0@d1,1@c1d1,2@a1d5.cr  
> ...
```

<sup>13</sup> EPC 的维数对统计分析耗时的影响较小，因此我们来得及获取统计分析的结果。

于是可知只有  $0@d1,1@a1d1,2@c1d5$ 、 $0@d1,1@b1d1,2@c1d5$  这对等价 EPC 是正确的；在可视化软件中查看  $E$  值为 0.0297 和 0.0484 的两个解模型（图 4.3），可见后者主要是部分  $\text{O}^{2-}$  原子在  $c$  方向上发生了违背成键规律的较大偏移。加入以上 EPC 筛选之后，求解所需时长可以压缩到约 45 min（只压缩了约 1/2 的时间，因为筛选出的 EPC 多为 18–22 维）。

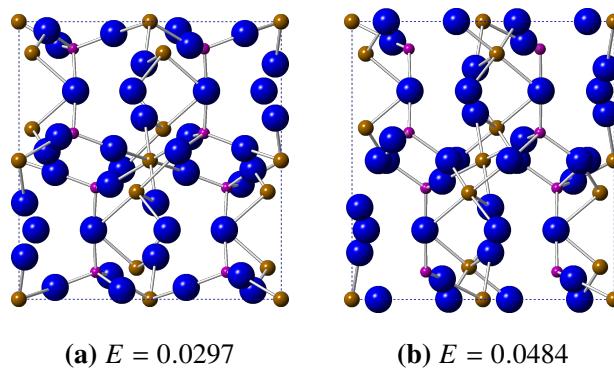


图 4.3 “0000219”典型解在  $a$  方向上的投影（均已转换到使部分  $\text{Fe}^{3+}$  在  $a$  位置）

这里有必要强调，本章所有求解均是在启用了 Intel i7-3720QM CPU 的所有 8 个处理器线程的前提下进行的：在求解其它结构时这或许不是必需，但在求解“0000219”时可以看到，因为 EPC 任务是完美并行的（参考第 3.2.1 小节），所以我们利用并行节省了大约  $0.75 \times 7 = 5.2$  h 的时间，这是不可忽视的。可能有读者会疑惑，在硬件性能突飞猛进的当代，从软件方面提升计算效率是否仍有必要？事实上，根据近年来 CPU 性能的增长趋势（图 4.4），CPU 的单线程浮点性能大约是每 12 年提升一个数量级，所以数倍的性能提升仍然具有重要意义，何况在单线程性能增长放缓的背景之下大力发展并行和分布式计算本来就是科学计算（乃至通用计算）的大势所趋。

## 4.2 *decryst* 的高级用法和常用技巧

在本节中，第 4.2.1 小节将演示如何利用 *decryst* 的自动化机制实现正空间法中的重原子法，而重原子法是之后各小节中结构测定的基础。第 4.2.2 小节将演示如何实现对重原子、轻原子等效点系组合（EPC）的分步确定，以及在此之后如何自动化地对同时涉及轻、重原子的 EPC 进行准确的求解。第 4.2.3 小节将演示对 EPC 的复杂筛选：其中不仅会用到之前的 EPC 筛选方法，而且会在结构测定的各个步骤中多次利用原子重叠评估函数（参考第 2.2.2 小节）对 EPC 进行精细的筛选。

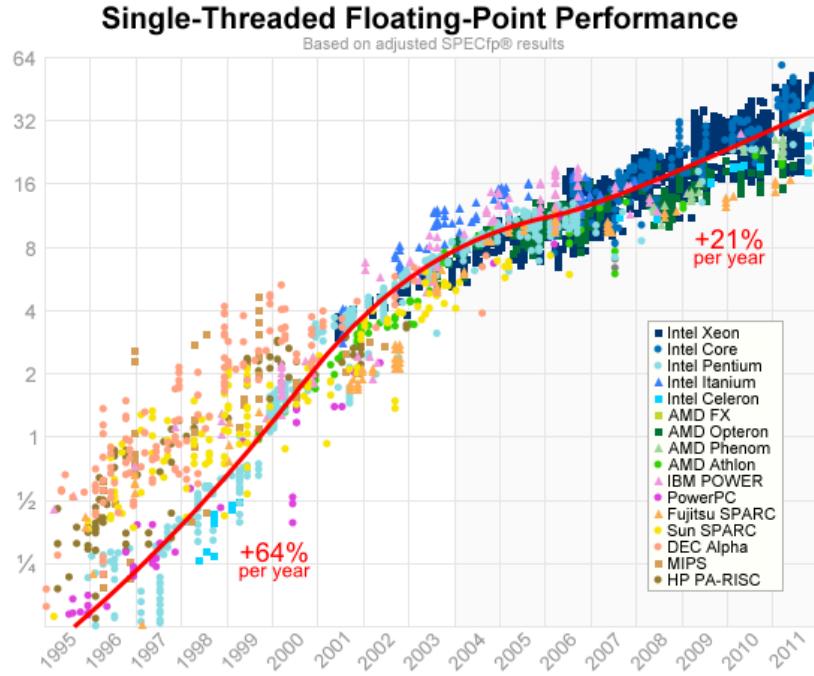


图 4.4 CPU 单线程浮点性能在 1995–2011 年间的增长（图引自文献（Preshing 2012））

#### 4.2.1 重原子法：以 $\text{PbSO}_4$ 为例

$\text{PbSO}_4$ （原数据采用的是  $Pbnm$  表示，求解前已转换到  $Pnma$ ）的 *decr* 文件（假设文件名为 *ps0.txt*）如下所示：

```
# sg, abc, angles; ctl, extra; limits.
62      8.4720 5.3973 6.9549      90 90 90
0.25 0.75 0.875 0.5 1.0
-- 

# rad/asf, n, limits, comb.
Pb2+    4      --      --
S6+;S   4      --      --
O2-     16     --      --
# zooms.
1.4:1-0 2.8:1-1
0.9:2-1

# 2theta, fwhm, hkl, mult, val.
16.49   0.2    1 0 1   4      2.70
20.83   0.2    0 1 1   4      79.73
# (... further lines omitted ...)
```

$\text{PbSO}_4$  结构可以直接使用第 4.1.3 小节中的 *solve.sh* 脚本求解，耗时约 8 s；然而考虑到  $\text{PbSO}_4$  中  $\text{Pb}^{2+}$  和  $\text{S}^{6+}$  均明显地重于  $\text{O}^{2-}$ ，该结构的简单性使其在演

示重原子法时成为一个完美的例子。为了使用重原子法求解  $\text{PbSO}_4$  结构，我们首先须要使 *decryst* 忽略  $\text{O}^{2-}$  原子，这可以通过注释实现：

```
$ sed -i '/^02-/ s/^/#/; /:2-/ s/^/#/' pso.txt
```

注意原 *decr* 文件中以 02- 开头的行只有指定晶胞中  $\text{O}^{2-}$  个数的行，且该文件中包含 2: 字符串的数据行只有指定和  $\text{O}^{2-}$  相关对缩放因子的行，因此以上命令能够精准地注释掉所有涉及  $\text{O}^{2-}$  的数据行。由于重原子法须要分步求解重、轻原子，为了避免不同步骤中的文件互相覆盖，我们可以在不同目录中生成不同步骤所需的文件：

```
$ mkdir ps
$ decr_utils comb pso.txt | awk '{ print "ps/pso," $0 }' |
    decr_utils dump pso.txt > ps/index.list
```

在 ps 目录中生成 *cryst* 文件和 *crysts* 文件之后，便可以对各 EPC 进行统计分析和全局最优化：

```
$ PYTHONUNBUFFERED=1 decr_utils stat stat.sh hosts.conf < ps/index.list |
    tee ps/index.tmp | do_rank.sh 0 > ps/index.rank0
$ grep -v '^#' < ps/index.rank0 | cut -f 2-3 |
    PYTHONUNBUFFERED=1 decr_utils optim optim.sh hosts.conf |
    tee ps/index.tmp | do_rank.sh 1 > ps/index.rank1
```

`PYTHONUNBUFFERED` 环境变量强制 Python 将每一行结果分别输出而不是将若干行结果凑到一起输出。助手脚本 `do_rank.sh` 的作用是在 *crysts* 文件中添加两列以记录各 EPC 的品质因数（FOM，基于 *EPCryst* (Deng、Dong 2011) 中的经验公式）以及是否零维，例如以下数据行（为了方便解释，此处明确区分了代码中的空格和制表符）

```
> 0 0.522904 *ps/pso,0@b1,1@a1.cr
> 2 1000 200 0.01 0.396994 0.140564 0.129636 *ps/pso,0@c1,1@a1.cr
```

将被变换为

```
> 0 3.82479 *...-0.522904 *ps/pso,0@b1,1@a1.cr
> 1 11.2546 *...0.396994 0.140564 0.129636 *ps/pso,0@c1,1@a1.cr
```

`do_rank.sh` 的命令行参数决定了其对数据行排序的方式：为 1 时只根据 FOM 排序，而为 0 时将零维 EPC 排在前面，然后才是按 FOM 排序。注意其中使用 `grep` 命令去掉了 `index.rank0` 中用 “#” 注释掉的行，从而避免对这些行计算上述的临时字段。此外，以上操作中有规律地使用了空白字符（参考第 4.1.1 小节），从

而方便了之后使用 `cut` 命令去掉上述临时字段；这一技巧事实上已经在第 4.1.3 小节中使用。

在求解完重原子的坐标之后，我们就可以导出结果了：

```
$ do_filter.sh < ps/index.rank1 | decr_utils merge pso.txt > /dev/null
```

`decr_utils` 在导出 `decr` 文件时会保留原有注释，于是上一步中 `pso.txt` 的片段

```
#02-      16      --      --
# Pairwise zoom factors.
1.4:1-0 2.8:1-1
#0.9:2-1
```

在正确 EPC 所对应的 `decr` 文件（`ps/pso,0@c1,1@c1.cr`）中将被变换为

```
#02-      16      --      --
# Pairwise zoom factors.
1.4:1-0 2.8:1-1
#0.9:2-1
0@c1      0.314226      0.25      0.33363
1@c1      0.43269  0.25      0.805449
```

因此为了求解  $O^{2-}$  原子的位置，我们只须去掉所有涉及  $O^{2-}$  的注释：

```
$ sed -i '/^#02-/ s/^#//; /:2-/ s/^#//' pso.txt
$ do_filter.sh < ps/index.rank1 | sed 's/\.cr$/\.txt/' |
    xargs -n 100 sed -i '/^#02-/ s/^#//; /:2-/ s/^#//'
```

其中 `do_filter.sh` 的作用是提取 `rank` 文件中未被注释的 `cryst` 文件名，例如

```
> #0 3.82479      ...   - - 0.522904      ps/pso,0@c1,1@a1.cr
> 1 11.2546      ...  0.396994 0.140564 0.129636  ps/pso,0@c1,1@a1.cr
```

将被变换为

```
> ps/pso,0@c1,1@a1.cr
```

接下来我们须要产生  $O^{2-}$  原子的 EPC：

```
$ mkdir pso
$ do_filter.sh < ps/index.rank1 | sed 's/\.cr$/\.txt/' | while read f; do
    decr_utils comb "$f" | awk '{
        print "pso/" "$(basename $f | sed -r 's@^.+@'"$f"'@')"" $0
    }' | decr_utils dump "$f" >> pso/index.list; done
```

以上第 2 行命令的作用是将从重原子 EPC 所对应 `decr` 文件生成的 `cryst` 文件冠以相应重原子 EPC 的前缀：例如从 `ps/ps0,0@c1,1@c1.txt` 生成的 `2@c1d1 cryst` 文件将被保存到 `ps0/ps0,0@c1,1@c1,2@c1d1.cr`；通过这一操作，我们事实上得到了（包含所有原子的）总 EPC 的完整表达式。 $O^{2-}$  原子的统计分析和全局最优化步骤和之前如出一辙，不再赘述：

```
$ PYTHONUNBUFFERED=1 decr_utils stat stat.sh hosts.conf < ps0/index.list |
    tee ps0/index.tmp | do_rank.sh 0 > ps0/index.rank0
$ grep -v '^#' < ps0/index.rank0 | cut -f 2-3 |
    PYTHONUNBUFFERED=1 decr_utils optim optim.sh hosts.conf |
    tee ps0/index.tmp | do_rank.sh 1 > ps0/index.rank1
$ do_filter.sh < ps0/index.rank1 | decr_utils merge ps0.txt > /dev/null
$ rm ps0/index.tmp ps0/index.rank1
```

除去人工步骤之后，以上求解过程总耗时约 5 s。

#### 4.2.2 实际 EPC 的搜索：以“0000158”结构为例

第 4.2.1 小节中求解  $PbSO_4$  结构所用的命令行操作具有很强的典型性：许多代码在其它结构的求解中只需很少修改就可以使用，这些代码被抽象为了助手脚本；在使用助手脚本之后， $PbSO_4$  结构的求解可以用以下脚本实现：

```
#!/bin/sh
sed -i '/^02-/ s/^/#/; /:2-/ s/^/#/' ps0.txt
mkdir ps
echo ps0.txt | do_dump.sh ps
do_stat.sh ps && do_optim.sh ps && do_merge.sh ps0.txt ps
sed -i '/^#02-/ s/^##/; /:2-/ s/^##/' ps0.txt
do_filter.sh < ps/index.rank1 | sed 's/\.cr$/.txt/' |
    xargs -n 100 sed -i '/^#02-/ s/^##/; /:2-/ s/^##/' |
mkdir ps0
do_filter.sh < ps/index.rank1 | sed 's/\.cr$/.txt/' | do_dump.sh ps0
do_stat.sh ps0 && do_optim.sh ps0 && do_merge.sh ps0.txt ps0
rm ps/index.tmp ps0/index.tmp
```

可能有读者会感到疑惑：脚本一般是用于自动化地运行具有重复性的任务，但晶体结构在解出之后一般不用重复求解，那么这样的求解脚本有什么意义？事实上，因为实际结构的求解往往是相当复杂的，其步骤和参数往往需要多次调整；因此，求解脚本不仅可以方便用户在调整求解步骤和参数时对求解的全程进行宏观的把握，而且可以作为增强求解过程可重复性的“实验记录”，从而为其它

结构的求解提供参考。事实上，本节中后两种结构的求解都是以上述  $\text{PbSO}_4$  求解脚本为基础的。

“0000158”结构( $C2/c$ )的decr文件(假设文件名为158.txt)如下所示：

```
# sg, abc, angles; ctl, extra; limits.
15a      10.129 8.306 8.533      90 112.19 90
0.25 0.75 0.875 0.5 1.0
--

# rad/asf, n, limits, comb.
Ca2+    4      --      --
S6+;S   8      --      --
Na1+    8      --      --
O2-     32     --      --
# zooms.
2.8:1-1
1.2:2,0-1

# 2theta, fwhm, hkl, mult, val.
14.24   0.1    1 1 0    4      44.67
18.92   0.1    2 0 0    2      23.65
# (... further lines omitted ...)
```

该结构的总EPC数为3706，单个EPC维数为9–19；注意到其中的原子可以分为重原子 $\text{S}^{6+}$ 、 $\text{Ca}^{2+}$ 和轻原子 $\text{Na}^+$ 、 $\text{O}^{2-}$ ，其可以使用重原子法求解以节约时间，因此我们首先求解重原子(EPC数为44，单个EPC维数为0–4)：

```
$ sed -ri '/^(O2-|Na1\+)/ s/^/#/; /:2,/ s/^/#/' 158.txt
$ mkdir cs
$ echo 158.txt | do_dump.sh cs
$ do_stat.sh cs && do_optim.sh cs && do_merge.sh 158.txt cs
```

观察`cs/index.rank1`，不难注意到`0@e1,1@f1`是唯一的重原子EPC：其 $E$ 值为0.20左右，其它非零维EPC的 $E$ 值都在0.30–0.40之间，而所有零维EPC的 $E$ 值都大于0.40。因此我们在求解轻原子时只采用`0@e1,1@f1`这一重原子EPC(下属EPC数为350，单个EPC维数为6–15)：

```
$ sed -i '2,$ s/^/#/' cs/index.rank1
$ sed -ri '/#(O2-|Na1\+)/ s/^#/;; /:2,/ s/^#/;;' 158.txt
$ do_filter.sh < cs/index.rank1 | sed 's/\.\cr$/\.txt/' |
    xargs -n 100 sed -ri '/#(O2-|Na1\+)/ s/^#/;; /:2,/ s/^#/;;'
$ mkdir no
```

```
$ do_filter.sh < cs/index.rank1 | sed 's/\.\cr$/\.\txt/' | do_dump.sh no
$ do_stat.sh no && do_optim.sh no && do_merge.sh 158.txt no
```

观察 no/index.rank1 可见各 EPC 的  $E$  值没有明显的聚类现象，所以求解似乎失败了，但事实并非如此：在“0000158”中，重原子  $S^{6+}$ 、 $Ca^{2+}$  和轻原子  $Na^+$ 、 $O^{2-}$  的电子数相差并不悬殊，因此对重、轻原子分别求解得不到很满意的结果并不难理解。然而从直觉上不难理解分别求解得到的（包含所有原子的）总 EPC 排名和直接对所有原子求解得到的总 EPC 排名仍会存在一定的相关性，因此我们可以对排名靠前的总 EPC 再次进行求解（这里取了前 30 个；此外注意 do\_dump.sh 生成的 cryst 文件名包含了总 EPC 的完整表达式，参考第 4.2.1 小节）：

```
$ sed -i '31,$ s/^/#/' no/index.rank1
$ mkdir csno
$ do_filter.sh < no/index.rank1 | sed 's@^[\^,]\+\@\@; s/\.\cr//' |
    awk '{ print "csno/158," $0 ".cr\t" $0 }' |
    decr_utils dump 158.txt > csno/index.list
$ do_stat.sh csno && do_optim.sh csno && do_merge.sh 158.txt csno
```

观察 csno/index.rank1 可见正确 EPC  $0@e1,1@f1,2@f1,3@f4$  是唯一合理的 EPC：其  $E$  值最多为 0.05 左右，而其它 EPC 的  $E$  值都至少为 0.13 左右。注意到该 EPC 的维数为 19，我们须要多次进行全局最优化，然后挑选最优解：

```
$ mkdir fin
$ l=$(sed 1q < csno/index.rank1 | sed 's/csno/fin/; s/\cr//')
$ name=$(echo "$l" | cut -f 3 | sed 's@.*@\@')
$ for i in $(seq 0 9); do
    cp csno/"$name".cr fin/"$name-$i".cr
    echo "$l-$i".cr >> fin/index.rank0; done
$ do_optim.sh fin && do_merge.sh 158.txt fin
$ rm */index.tmp
```

fin/index.rank1 中排名最靠前的一般会是 3 个左右的  $E$  值在 0.025–0.030 的解，这些就是正确解；除去人工步骤之后，以上求解过程总耗时约 5 min。

### 4.2.3 EPC 的复杂筛选：以“0009563”结构为例

“0009563”结构 ( $P\bar{3}c1$ ) 的 decr 文件（假设文件名为 0009563.txt）如下所示（为了方便解释，此处明确区分了代码中的空格和制表符）：

```
#msg,abc,angles;ctl,extra;limits.
165      #12.19#12.19#10.14      #90#90#120
```

```

0.25 0.75 0.875 0.5 1.0
-- 

#_rad/asf,_n,_limits,_comb.
Rb1+ #12    #--    #--
Ge4+ #18    #--    #--
Ti4+ #6     #--    #--
O2- #54    #--    #--

#_zooms.
1.5:1,2-1,2

#_2theta,_fwhm,_hkl,_mult,_val.
16.97 #0.2   #1_1_1 #12      #10.71
17.49 #0.2   #0_0_2 #2       #7.14
#(..._further_lines_omitted...)

```

本小节的详细测试结果（包括图 4.5 所对应的结构数据）可以从文献（Liu 2018）的补充材料中获取。注意到“0009563”中的金属原子均明显重于 O<sup>2-</sup>，使用重原子法求解是很合适的，因此本人首先生成了重原子的 cryst 文件（EPC 数为 1451 个，单个 EPC 维数为 5–8）：

```

$ mkdir rgt-0
$ sed '/^O2-/ s/^/#/; s/^0\.25 /0.99 /' < 0009563.txt > 9563.txt
$ echo 9563.txt | do_dump.sh rgt-0
$ mkdir rgt-1
$ sed -i 's/^0\.99 /0.25 /' 9563.txt
$ echo 9563.txt | do_dump.sh rgt-1

```

因为 EPC 数很大，我们须要尽量对其进行筛选，从而减少（包含所有原子的）总 EPC 备选范围。受陆学善、梁敬魁（1965），Reddy 等（1965）所做工作的启发，本人生成了两套 cryst 文件，其中组合因子  $\mu$ （参考第 4.1.1 小节）的值分别为<sup>14</sup> 0.99 和默认的 0.25；通过对  $\mu = 0.99$  的 cryst 文件进行全局最优化，我们就可以筛除那些不可能产生无原子重叠晶体模型的 EPC。此外还须要强调的是 0009563.txt 中在行首出现且后面跟着空格的数值只有  $\mu$  值，这是以上命令可以精准地对  $\mu$  值进行更改的基础。

我们接下来会看到，即使事先筛除不可能产生无原子重叠晶体模型的 EPC，全局最优化仍可能得到发生重叠的解模型；为了筛除这种解模型，本人定义了一个在 crysts 文件中加入原子重叠评估函数值  $B$  的 shell 函数：

<sup>14</sup> 考虑到部分化学上很不合理的 EPC 可能以大概率产生原子重叠评估函数值  $B = 1$  的晶体模型，不将  $\mu$  值设定为 1 可以避免这些 EPC 的全局最优化过早结束。

```
$ do_bump_sh() {
    while read l; do
        name=$(echo "$l" | cut -f 3 | sed 's/\.\cr$/.txt/'")
        echo 'nil.cr nil' | decr_utils dump "$name" > /dev/null
        sed 's/0\.25 0\.75 0\.875/0.99 0.75 0.875/' < nil.cr |
            decr_mcs 0 | cut -d ' ' -f 3 | tr '\n' ' '
        echo "$l"; done; }
```

其中 nil EPC 的意义可以参考第 4.1.1 小节。接下来本人按照上文所述的思路对之前产生的 cryst 文件进行了统计分析和全局最优化：

```
$ do_stat.sh rgt-0 && do_optim.sh rgt-0
$ mv -f rgt-0/index.rank1 rgt-0/index.tmp
$ awk '{ print ($9 > 0.12 ? "#" : "") $0 }' \
    < rgt-0/index.tmp > rgt-0/index.rank1
$ do_filter.sh < rgt-0/index.rank1 | sed 's@.*@rgt-1@' > rgt-1/index.tmp
$ grep -Ff rgt-1/index.tmp < rgt-1/index.list > rgt-1/index.tmpp
$ mv -f rgt-1/index.tmpp rgt-1/index.list
$ do_stat.sh rgt-1 && do_optim.sh rgt-1 && do_merge.sh 9563.txt rgt-1
$ do_bump_sh < rgt-1/index.rank1 | awk '{ print
    ($10 > 0.2 || $1 > 0.02 ? "#" : "") $0 }' > rgt-1/index.rank2
```

以上命令先筛选了  $\mu = 0.99$  时解模型  $E > 0.12$  的 EPC，剩下 183 个（13%）；在恢复默认  $\mu$  值后又筛选了解模型  $E > 0.2$  的 EPC，剩下 26 个。本人在观察 index.rank2 之后发现其中的 EPC 分为一组解模型  $B \leq 0.0027$  的和一组解模型  $B \in [0.056, 0.112]$  的，并由此筛选了所有解模型  $B > 0.02$  的 EPC，剩下 20 个（11%）。除去人工步骤之后，重原子 EPC 求解耗时约 2.5 min。事实上，重原子求解得到了正确的结果：正确 EPC 0@g1,1@f1g1,2@b1d1 的  $E$  值（0.058）是最小的；事后本人在可视化软件中查看重原子 EPC 的解模型，确认了所有因  $B$  值过大而被筛选的解模型中都发生了很明显的原子重叠。

仿照以上步骤，本人对 O<sup>2-</sup> 原子的 EPC 进行了求解：

```
$ mkdir rgto-0 && sed -i '/^#02-/ s/^#//; s/^0\.25 /0.99 /' 9563.txt
$ do_filter.sh < rgto-0/index.rank2 | sed 's/\.\cr$/.txt/' |
    xargs -n 100 sed -i '/^#02-/ s/^#//; s/^0\.25 /0.99 /'
$ do_filter.sh < rgto-0/index.rank2 |
    sed 's/\.\cr$/.txt/' | do_dump.sh rgto-0
$ mkdir rgto-1 && sed -i 's/^0\.99 /0.25 /' 9563.txt
$ do_filter.sh < rgto-1/index.rank2 | sed 's/\.\cr$/.txt/' |
    xargs -n 100 sed -i 's/^0\.99 /0.25 /'
```

```
$ do_filter.sh < rgt-1/index.rank2 |
    sed 's/\.\cr$/\.txt/' | do_dump.sh rgto-1
$ do_stat.sh rgto-0 && mv -f rgto-0/index.rank0 rgto-0/index.tmp
$ awk '{ print ($9 > 0.99 ? "#" : "") $0 }' \
    < rgto-0/index.tmp > rgto-0/index.rank0
$ do_optim.sh rgto-0 && mv -f rgto-0/index.rank1 rgto-0/index.tmp
$ awk '{ print ($9 > 0.05 ? "#" : "") $0 }' \
    < rgto-0/index.tmp > rgto-0/index.rank1
$ do_filter.sh < rgto-0/index.rank1 |
    sed 's@.*@rgto-1@' > rgto-1/index.tmp
$ grep -Ff rgto-1/index.tmp < rgto-1/index.list > rgto-1/index.tmpp
$ mv -f rgto-1/index.tmpp rgto-1/index.list
$ do_stat.sh rgto-1 && do_optim.sh rgto-1 && do_merge.sh 9563.txt rgto-1
$ do_bump_sh < rgto-1/index.rank1 |
    awk '{ print ($1 > 0.02 ? "#" : "") $0 }' > rgto-1/index.rank2
```

在这一步骤中，本人首先基于之前筛选得到的重原子 EPC 生成了 O<sup>2-</sup> 的 EPC（共 4455 个，维数主要为 10–13）。在筛除不可能产生无原子重叠晶体模型的 EPC 时为了节省时间，本人先在统计分析后筛除了  $E$  的最佳值大于 0.99 的 EPC，剩下 2186 个（49%）；在全局最优化之后又筛除了解模型  $E > 0.05$  的 EPC，剩下 110 个（5%）。

考虑到 O<sup>2-</sup> 对“0009563”的衍射谱影响有限，而其 EPC 维数较高，我们可以逐步缩小备选 EPC 的范围。在上一步骤中，本人已经筛选了解模型  $B > 0.02$  的 EPC，剩下 50 个；接下来，本人又重复进行了 2 次全局最优化，并分别选取了  $E$  值排名前 20 和前 10 的 EPC：

```
$ i=0; for n in 21 11; do
    [ "$i" -eq 0 ] && c=rgto-1/index.rank2 ||
        c=fin-$((i - 1))/index.rank1
    d=fin-"$i"; mkdir "$d"
    do_filter.sh < "$c" | sed 's@.*@rgto-1@' > "$d"/index.tmp
    grep -Ff "$d"/index.tmp < rgto-1/index.rank0 |
        sed 's/rgto-1/"$d"/' > "$d"/index.rank0
    cp $(cat "$d"/index.tmp) "$d" && do_optim.sh "$d"
    sed -i "$n",\$ s/^#/ "$d"/index.rank1
    do_merge.sh 9563.txt "$d"; i=$((i + 1)); done
```

事后本人观察上述步骤中产生的 index.rank1 文件，确认了这 10 个 EPC 通常都是在其中排名靠前的。本人对这些 EPC 中的每一个进行了 10 次全局最优化，并筛选了所得解模型中  $B > 0.02$  的那些，最后剩下了 90 个解模型：

```

$ c="$d"/index.rank1; d=fin-"$i"; mkdir "$d"
$ do_filter.sh < "$c" | sed 's@.*@rgto-1@' > "$d"/index.tmp
$ grep -Ff "$d"/index.tmp < rgto-1/index.rank0 |
    sed 's/rgto-1/'"$d"'/' > "$d"/index.list
$ cat "$d"/index.list | while read l; do
    name=$(echo "$l" | sed 's@.*@@; s@\.\cr$@@')
    l=$(echo "$l" | cut -f 1-2); for i in $(seq 0 9); do
        cp rgto-1/"$name".cr "$d"/"$name-$i".cr
        printf '%s\t%s\n' "$l" "$d"/"$name-$i".cr >> "$d"/index.rank0
    done; done
$ do_optim.sh "$d" && do_merge.sh 9563.txt "$d"
$ do_bump_sh < "$d"/index.rank1 |
    awk '{ print ($1 > 0.02 ? "#" : "") $0 }' > "$d"/index.rank2
$ rm -f 9563.txt nil.cr */index.tmp

```

除去人工步骤之后,  $O^{2-}$  原子 EPC 的求解耗时约 25 min。本人首先按 EPC 分组查看了上述的 90 个解, 发现除正确 EPC  $0@g1, 1@f1g1, 2@b1d1, 3@f1g4$  之外的 EPC 生成的解在成键关系上都有明显的问题: 难以解释的不均匀配位数 (图 4.5(c)), 有孤立  $O^{2-}$  的前提下金属原子被金属原子配位 (图 4.5(d)), 等等。又考虑到这一 EPC 产生了上述 100 个解中排名最靠前的 10 个, 其无疑是“0009563”唯一合理的 EPC; 在这 10 个解中, 本人找到了 2 种互不等价的合理解: 一种 (图 4.5(a)) 和数据库中的记录基本一致, 其原子坐标只需轻微的精修; 另一种 (图 4.5(b)) 一眼看去没有明显的问题, 但事实上可以注意到其中  $Ge^{4+}$  原子的配位多面体有很明显的畸变。

## 4.3 讨论和小结

### 4.3.1 关于动态剪枝的讨论

如第 3.1.3 小节所述, Wyckoff 位置种类较多的空间群对于原子数稍多的结构往往会产生极大量的可行 EPC; 其内存溢出的潜在问题可以用增量生成 EPC 解决, 但其使结构测定的时间复杂度发生爆炸性增长仍是一个严重的问题。陆学善、梁敬魁 (1965) 在求解  $V_2Ga_5$  结构时筛选 EPC 的思路对我们有一定的启发: 其根据对晶胞几何属性的分析得出了各个 Wyckoff 位置之间的互斥关系。第 3.3.3 小节指出, 可以通过在 *decryst* 中引入涉及多个 Wyckoff 位置的占据数约束来应用 Wyckoff 位置的互斥关系; 然而, 这样的互斥关系很多时候难以手工进行推导。

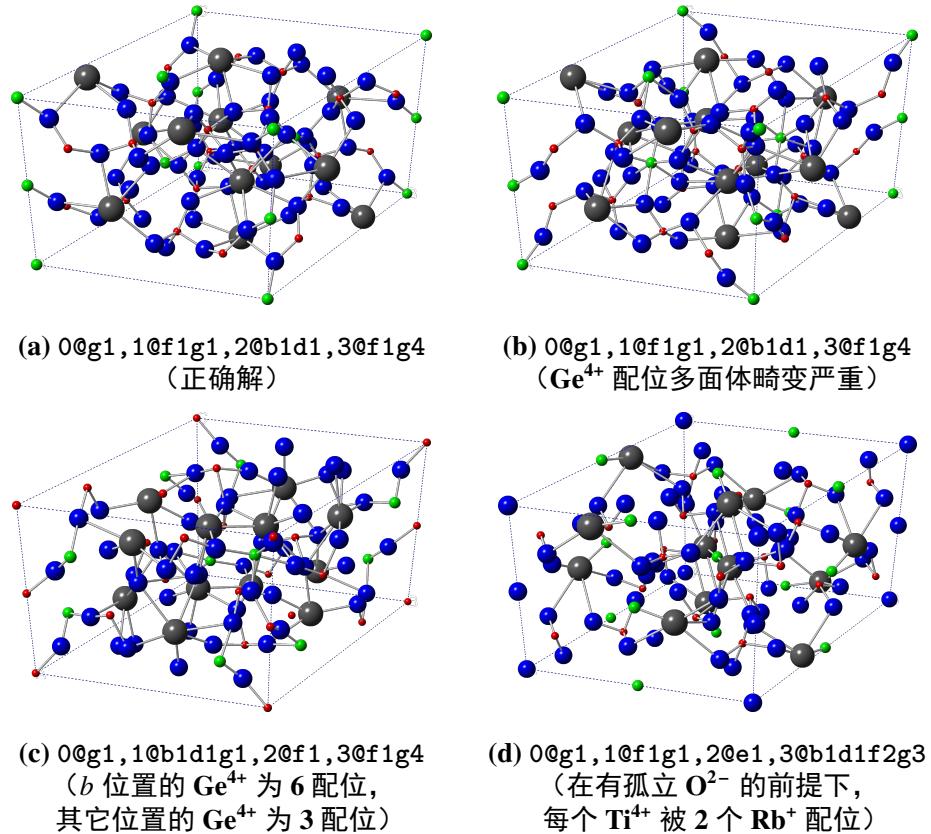


图 4.5 “0009563”的一些解

事实上，如果从深度优先树搜索的角度考虑，不难注意到生成 EPC 时的互斥约束也可以放到剪枝的框架之下理解：例如对于 V<sub>2</sub>Ga<sub>5</sub>，因为 a、b 位置互斥，在其中一个位置有占据时另一位置被占据的子树在搜索时就被剪掉了。在此背景之下，我们可以沿用 4.2.3 小节中利用原子重叠评估函数进行 EPC 筛选的思路，不再手工推导互斥关系，而是动态地对互斥关系进行检测：在树搜索中实时地对当前节点所代表的（部分或全部原子的）EPC 进行针对原子重叠的统计分析或全局最优化，并将不可能产生无碰撞晶体模型的子树剪掉。

如果要实现这样的动态剪枝，那么 EPC 的生成和（针对原子重叠而非衍射谱的）统计分析或全局最优化将必须在同一步骤中进行，因此 decrust 的设计和实现必须经过大幅度的修改：在搜索过的每个节点上加入统计分析会明显增加树搜索的复杂度，而全局最优化更是如此，因此动态剪枝必须是可选的，而且在启用动态剪枝时还必须考虑树搜索的并行化问题，此外参数取值的遍历顺序（例如  $a_1b_1 \rightarrow a_2b_1 \rightarrow \dots \rightarrow a_1b_2 \rightarrow \dots$  和  $a_1b_1 \rightarrow a_1b_2 \rightarrow \dots \rightarrow a_2b_1 \rightarrow \dots$ ）对算法复杂度的影响应该也是需要考虑的。不过有必要指出，在晶体学软件中类似的情况不是只在 decrust 里出现：例如为了将电荷反转法（参考第 1.2 节）应用于衍射峰之间常有明显重叠的粉末衍射数据，相应的算法（Baerlocher 等 2007）须

要实时地对衍射峰的划分方案进行调整，而分峰在倒空间法中原本通常是相对独立的步骤；事实上，UACIEM (Ma 等 2004) 就是一种动态分峰算法。

### 4.3.2 其它讨论

如第 4.2.3 小节所述，除了在求解之后用于筛除发生原子重叠的解模型以及在求解过程中实时排除发生重叠的晶体模型之外，本文中的原子重叠评估函数也可以用于在求解前筛除不可能产生无重叠晶体模型的 EPC。更复杂的评估函数，例如考虑键角、配位数、成键类型、原子链类型等等因素的那些 (Li 等 2012)，从原则上讲应该也可以用于类似用途，而且从实践上看肯定会有重要的意义：例如要是能实际使用这样的评估函数，我们在求解“0000428”结构时就可以很容易地拉开正确解和错误解在目标函数值上的差别，而在求解“0009563”结构时也能极大地简化最后的手工（目视）筛选步骤。

从本章中大部分测试结构的求解过程可见，我们如果能自动化地实现对等价 EPC 和等价晶体模型的检测，就能进一步提升结构测定的效率。前一问题大概可以通过按局部对称性 (site symmetry) 对 Wyckoff 位置进行分组（例如  $R\bar{3}$  空间群中  $d$ 、 $e$  位置的局部对称性均为  $\bar{1}$ ），然后为对称操作可能导致的组内互变关系（例如  $R\bar{3}$  空间群中  $(0, 0, 1/2)$  平移可以使  $a/b$ 、 $d/e$  位置互变）构造查找表的方式来处理。后一问题大概须要通过对某种特征性的指标进行聚类分析来实现，而 CALYPSO (Wang 等 2012) 中的成键特征矩阵 (bond characterisation matrix) 方法可能对这一指标的构造有一定的启发意义。

如第 4.2.3 小节所述，即使在求解前筛除不可能产生无重叠晶体模型的 EPC，并在求解中实时排除发生重叠的晶体模型，我们最后仍然可能得到发生重叠的解模型；本人认为，发生这种现象的根本原因是某些 EPC 不能产生  $R$  因子很小而且同时不发生原子重叠的晶体模型，或者至少只能以很小的概率得到这样的模型。出于这一原因，我们目前仍然须要在求解之后筛除发生重叠的解模型；另一种应该可行的思路是在目标函数中以一种非线性的方式组合  $R$  因子和原子重叠评估函数，从而使得目标函数值在趋于零时被  $R$  因子和评估函数值中较大的那个数值决定。

### 4.3.3 本章小结

*decryst* 的设计追求简洁、灵活，而本人也希望其中的技巧可以在更多的晶体学软件中得到应用。在现有自动化工具的配合下，用 *decryst* 能简单地实现相

当复杂的求解流程：利用重原子法的求解，统计分析和全局最优化后基于 Bragg *R* 因子的等效点系组合（EPC）筛选，求解前筛除必发生原子重叠的 EPC、最优化中实时排除存在原子重叠的晶体模型、求解后筛除仍发生原子重叠的 EPC，等等。为了演示 *decryst* 的基本用法和常用技巧，本人从美国矿物学家晶体结构数据库（AMCSD）中选取了若干个不同晶系和复杂度的测试结构，并成功地对它们进行了求解。

*Almost anything in software can be implemented, sold, and even used, given enough determination. There is nothing a mere scientist can say that will stand against the flood of a hundred million dollars. But there is one quality that cannot be purchased in this way, and that is reliability. The price of reliability is the pursuit of the utmost simplicity. It is a price which the very rich find most hard to pay.*

— C. A. R. Hoare (1981)



## 第5章 结论和展望

本人提出了一种具有通用性的晶体学碰撞检测算法框架，按照此框架让 sweep and prune (SAP) 算法能以  $O(n \log n)$  的时间复杂度检测晶胞中的成键关系，并提出了一种在小晶胞中也适用的利用等效点系对称性极大降低碰撞检测复杂度的算法，最后基于以上算法提出了一种针对晶胞中原子重叠状况的评估函数。利用以上结果，我们能在正空间法的全局最优化步骤中高效地对晶胞中的成键关系进行检测，从而可以实时地排除存在原子重叠的晶体模型，以及实时地进行关于配位多面体、原子键价等等的计算。在此基础之上值得进一步探索的课题主要是上述晶体学碰撞检测算法框架在成键关系大部分已知的结构上的应用，以及细碰撞检测正确性的证明。

基于上述机制，本人开发了 *decryst* 这一套利用正空间法和等效点系组合 (EPC) 法从指标化的粉末衍射数据求解晶体结构的软件，其中首次以一种一般性的方式使用增量计算的思想来提升其计算性能，并且通过一种增量的算法生成 EPC 以降低其内存需求，此外 *decryst* 中也加入了对并行和分布式计算的支持。通过应用以增量计算为代表的惰性计算技术，以及并行和分布式计算，*decryst* 有着很高的性能；因为 EPC 互相独立且数量往往很大，*decryst* 对 EPC 任务的并行化将为求解成键关系总体未知的结构带来前所未有的机遇。在此基础之上值得进一步探索的课题主要是更加复杂的目标函数在 *decryst* 中的应用，以及对 *decryst* 中全局最优化算法的进一步改进。

*decryst* 的设计追求简洁、灵活，而本人也希望其中的技巧可以在更多的晶体学软件中得到应用；在现有自动化工具的配合下，用 *decryst* 能简单地实现相当复杂的求解流程。本人以从美国矿物学家晶体结构数据库 (AMCSD) 中若干个不同晶系和复杂度的测试结构为例，演示了 *decryst* 的基本用法和常用技巧。在目前工作的基础之上，主要值得进一步探索的课题是利用增量 EPC 生成算法中剪枝的思路动态地排除不可能生成无碰撞晶体模型的 EPC，以及对等价 EPC 和等价晶体模型的自动排除。

*Finally, the number of Unix installations has grown to 10, with more expected.*

— Ken Thompson and Dennis Ritchie (1972)



## 参考文献

- 曹禺 (1936)。“我如何写《雷雨》” [N]。大公报·文艺, 1936-01-19。
- 梁敬魁 (2011)。粉末衍射法测定晶体结构 [M]。第 2 版。北京: 科学出版社, 258–305。
- 陆学善, 梁敬魁 (1965)。“ $V_2Ga_5$  的晶体结构” [J]。物理学报, 21(5): 997–1007。DOI: 10.7498/aps.21.997。
- G. 肖盖 (2009)。拓扑学教程: 拓扑空间和距离空间、数值函数、拓扑向量空间 [M]。史树中, 王耀东译。第 2 版。北京: 高等教育出版社, 26。
- 钟锡华 (2003)。现代光学基础 [M]。北京: 北京大学出版社, 66。
- A. Altomare, C. Cuocci, C. Giacovazzo *et al.* (2013). “EXPO2013: a kit of tools for phasing crystal structures from powder data” [J]. *J. Appl. Cryst.* 46(4): 1231–1235. DOI: 10.1107/S0021889813013113.
- A. Altomare, R. Caliandro, C. Cuocci *et al.* (2008). “Direct methods and simulated annealing: a hybrid approach for powder diffraction data” [J]. *J. Appl. Cryst.* 41(1): 56–61. DOI: 10.1107/S0021889807054192.
- A. Altomare, C. Giacovazzo and A. Moliterni (2008). “Indexing and Space Group Determination” [M]. In: R. E. Dinnebier and S. J. L. Billinge, eds. *Powder Diffraction Theory and Practice*. Cambridge: Royal Society of Chemistry, 206–226.
- Y. G. Andreev, P. Lightfoot and P. G. Bruce (1997). “A General Monte Carlo Approach to Structure Solution from Powder Diffraction Data: Application to  $\text{Poly(ethylene oxide)}_2\text{LiN}(\text{SO}_3\text{CF}_3)_2$ ” [J]. *J. Appl. Cryst.* 30(3): 294–305. DOI: 10.1107/S0021889896013556.
- M. I. Aroyo, J. M. Perez-Mato, C. Capillas *et al.* (2006). “Bilbao Crystallographic Server I: Databases and crystallographic computing programs” [J]. *Z. Kristallog.* 221(1): 15–27. DOI: 10.1524/zkri.2006.221.1.15.
- M. Attfield, P. Barnes, J. K. Cockcroft *et al.* (1999). *Molecular Geometry: I. Interatomic Distances & Bond Lengths* [M/OL]. <http://pd.chem.ucl.ac.uk/pdnn/refine2/bonds.htm>, accessed on 2018-06-28.
- C. Baerlocher, F. Gramm, L. Massüger *et al.* (2007). “Structure of the Polycrystalline Zeolite Catalyst IM-5 Solved by Enhanced Charge Flipping” [J]. *J. Appl. Cryst.* 31(5815): 1113–1116. DOI: 10.1126/science.1137920.
- M. de Berg, O. Cheong, M. van Kreveld *et al.* (2008). *Computational Geometry: Algorithms and Applications* [M]. 3rd ed. Berlin: Springer.
- R. E. Bryant and D. R. O'Hallaron (2011). *Computer Systems: A Programmer's Perspective* [M]. 2nd ed. Boston: Pearson, 46.

- I. S. Bushmarinov, A. O. Dmitrienko, A. A. Korlyukova *et al.* (2012). “*Rietveld refinement and structure verification using ‘Morse’ restraints*” [J]. *J. Appl. Cryst.* 45(6): 1187–1197. DOI: 10.1107/S0021889812044147.
- M. Caroli and M. Teillaud (2016). “*Delaunay Triangulations of Closed Euclidean d-Orbifolds*” [J]. *Discrete Comput. Geom.* 55(4): 827–853. DOI: 10.1007/s00454-016-9782-6.
- R. Černý and V. Favre-Nicolin (2007). “*Direct space methods of structure determination from powder diffraction: principles, guidelines and perspectives*” [J]. *Z. Kristallog.* 222(3/4): 105–113. DOI: 10.1524/zkri.2007.222.3-4.105.
- K.-W. Chu, Y.-F. Deng and J. Reinitz (1999). “*Parallel Simulated Annealing by Mixing of States*” [J]. *J. Comput. Phys.* 148(2): 646–662. DOI: 10.1006/jcph.1998.6134.
- R. I. Cooper, A. L. Thompson and D. J. Watkin (2010). “*CRYSTALS enhancements: dealing with hydrogen atoms in refinement*” [J]. *J. Appl. Cryst.* 43(5): 1100–1107. DOI: 10.1107/S0021889810025598.
- B. Cordero, V. Gomez, A. E. Platero-Prats *et al.* (2008). “*Covalent radii revisited*” [J]. *Dalton Trans.* 37(21): 2832–2838. DOI: 10.1039/b801115j.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest *et al.* (2009). *Introduction to Algorithms* [M]. 3rd ed. Cambridge, MA: MIT Press.
- A. Courrèges (2016). *DOOM (2016) – Graphics Study* [EB/OL]. 2016-09-09. <https://www.adriancourreges.com/blog/2016/09/09/doom-2016-graphics-study/>, accessed on 2018-07-03.
- W. I. F. David, K. Shankland, L. B. McCusker *et al.*, eds. (2002). *IUCr Monographs on Crystallography, Volume 13: Structure Determination from Powder Diffraction Data* [M]. Oxford: Oxford University Press.
- W. I. F. David, K. Shankland, J. van de Streek *et al.* (2006). “*DASH: a program for crystal structure determination from powder diffraction data*” [J]. *J. Appl. Cryst.* 39(6): 910–915. DOI: 10.1107/S0021889806042117.
- J. Dean and S. Ghemawat (2004). “*MapReduce: Simplified Data Processing on Large Clusters*” [C]. In: E. A. Brewer and P. Chen, eds. *6th Symposium on Operating System Design and Implementation (OSDI 2004)*. San Francisco: USENIX Association, 137–150.
- X.-D. Deng and C. Dong (2009). “*SMEPOC: a computer program for the automatic generation of trial structural models for inorganic compounds with symmetry restriction*” [J]. *J. Appl. Cryst.* 42(5): 953–958. DOI: 10.1107/S0021889809034062.
- X.-D. Deng and C. Dong (2011). “*EPCryst: a computer program for solving crystal structures from powder diffraction data*” [J]. *J. Appl. Cryst.* 44(1): 230–237. DOI: 10.1107/S0021889810053835.

- C. T. J. Dodson and T. Poston (1991). *Graduate Texts in Mathematics, Volume 130: Tensor Geometry, the Geometric Viewpoint and Its Uses* [M]. Berlin: Springer.
- R. T. Downs and M. Hall-Wallace (2003). “*The American Mineralogist Crystal Structure Database*” [J]. *Am. Mineral.* 88(1): 247–250.
- P. Engel (1986). *Geometric Crystallography: An Axiomatic Introduction to Crystallography* [M]. Dordrecht: D. Reidel Publishing Company, 46–50.
- M. Eremenko, V. Krayzman, A. Gagin *et al.* (2017). “*Advancing reverse Monte Carlo structure refinements to the nanoscale*” [J]. *J. Appl. Cryst.* 50(6): 1561–1570. DOI: 10.1107/S1600576717013140.
- C. Ericson (2005). *Real-time Collision Detection* [M]. San Francisco: Morgan Kaufmann Publishers.
- M. Falcioni and M. W. Deem (1999). “*A biased Monte Carlo scheme for zeolite structure solution*” [J]. *J. Chem. Phys.* 110(3): 1754–1766. arXiv: cond-mat/9809085.
- V. Favre-Nicolin and R. Černý (2002). “*FOX, ‘free objects for crystallography’: a modular approach to ab initio structure determination from powder diffraction*” [J]. *J. Appl. Cryst.* 35(6): 734–743. DOI: 10.1107/S0021889802015236.
- S. I. Feldman (1979). “*Make – a program for maintaining computer programs*” [J]. *Softw.: Pract. Exper.* 9(4): 255–265. DOI: 10.1002/spe.4380090402.
- C. Giacovazzo (2001). “*Direct methods*” [M]. In: U. Shmueli, ed. *International Tables for Crystallography, Volume B: Reciprocal space*. 3rd ed. Dordrecht: Kluwer Academic Publishers, 210–234.
- S. Grudinin and S. Redon (2010). “*Practical modeling of molecular systems with symmetries*” [J]. *J. Comput. Chem.* 31(9): 1799–1814. DOI: 10.1002/jcc.21434.
- N. J. Higham (1993). “*The Accuracy of Floating Point Summation*” [J]. *SIAM J. Sci. Comput.* 14(4): 783–799. DOI: 10.1137/0914050.
- C. A. R. Hoare (1981). “*The Emperor’s Old Clothes*” [J]. *Commun. ACM*, 24(2): 75–83. DOI: 10.1145/358549.358561.
- International Union of Crystallography (1992). “*Report of the Executive Committee for 1991*” [J]. *Acta Cryst. A*48(6): 922–946. DOI: 10.1107/S0108767392008328.
- W. Kahan (1965). “*Further Remarks on Reducing Truncation Errors*” [J]. *Commun. ACM*, 8(1): 40. DOI: 10.1145/363707.363723.
- D. E. Knuth (1998). *The Art of Computer Programming, Volume 3: Sorting and Searching* [M]. 2nd ed. Reading, MA: Addison-Wesley.
- S. Kshirsagar (2015). *CVP.cpp* [CP/OL]. <https://github.com/krishkshir/crystalLattice/blob/master/CVP.cpp>, accessed on 2018-06-28.

- J. Lam and J.-M. Delosme (1988). *An Efficient Simulated Annealing Schedule: Implementation and Evaluation* [R]. Tech. Rep. 8817. New Haven, CT: Department of Electrical Engineering, Yale University.
- D. A. Levin, Y. Peres and E. L. Wilmer (2008). *Markov Chains and Mixing Times* [M]. Providence, RI: American Mathematical Society, 47–48.
- Y. Li, J.-H. Yu and R.-R. Xu (2012). “*FraGen: a computer program for real-space structure solution of extended inorganic frameworks*” [J]. *J. Appl. Cryst.* 45(4): 855–861. DOI: 10.1107/S002188981201878X.
- Y. Liu (2017). “*Real-time detection and resolution of atom bumping in crystallographic models*” [J]. *Acta Cryst. A* 73(5): 414–422. arXiv: 1708.03180.
- Y. Liu (2018). “*decryst: an efficient software suite for structure determination from powder diffraction*” [J]. *J. Appl. Cryst.* 51(4): 1237–1243. arXiv: 1801.08372.
- Z.-H. Lou and J. Reinitz (2016). “*Parallel simulated annealing using an adaptive resampling interval*” [J]. *Parallel Comput.* 53: 23–31. DOI: 10.1016/j.parco.2016.02.001.
- H.-W. Ma, J.-K. Liang, G.-Y. Liu et al. (2004). “*UACIEM: A new method to extract reliable intensities of nonequivalent systematical overlapping reflections from powder diffraction data*” [J]. *Powder Diffr.* 19(4): 333–339. DOI: 10.1154/1.1814979.
- D. Micciancio and S. Goldwasser (2002). *Complexity of Lattice Problems: A Cryptographic Perspective* [M]. Boston: Kluwer Academic Publishers.
- D. Micciancio and O. Regev (2009). “*Lattice-based Cryptography*” [M]. In: D. J. Bernstein, J. Buchmann and E. Dahmen, eds. *Post-Quantum Cryptography*. Berlin: Springer-Verlag, 147–191.
- L. Palatinus (2013). “*The charge flipping algorithm in crystallography*” [J]. *Acta Cryst. B* 69(1): 1–16. DOI: 10.1107/S2052519212051366.
- V. K. Pecharsky and P. Y. Zavalij (2009). *Fundamentals of Powder Diffraction and Structural Characterization of Materials* [M]. 2nd ed. New York: Springer.
- J. Preshing (2012). *A Look Back at Single-Threaded CPU Performance* [EB/OL]. 2012-02-08. <https://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance/>, accessed on 2018-07-18.
- E. Prince, ed. (2004). *International Tables for Crystallography Volume C: Mathematical, physical and chemical tables* [M]. 3rd ed. Dordrecht: Kluwer Academic Publishers.
- J. M. Reddy, A. R. Storm and K. Knox (1965). “*The crystal structure of  $V_2Gas$* ” [J]. *Z. Kristallog.* 121(6): 441–448. DOI: 10.1524/zkri.1965.121.16.441.
- H. M. Rietveld (1969). “*A profile refinement method for nuclear and magnetic structures*” [J]. *J. Appl. Cryst.* 2(2): 65–71. DOI: 10.1107/S0021889869006558.
- A. Robbins and N. Beebe (2005). *Classic Shell Scripting* [M]. Sebastopol, CA: O'Reilly.

- J. Rodríguez-Carvajal (2001). “Recent developments of the program FullProf” [J]. *Commission on Powder Diffraction Newsletters*, 26: 12–19.
- M. G. Rossmann and E. Arnold (2001). “Patterson and molecular-replacement techniques” [M]. In: U. Shmueli, ed. *International Tables for Crystallography, Volume B: Reciprocal space*. 3rd ed. Dordrecht: Kluwer Academic Publishers, 235–263.
- P. H. Salus (1994). *A Quarter Century of Unix* [M]. Reading, MA: Addison-Wesley, 52.
- K. Shankland, W. I. F. David and T. Csoka (1997). “Crystal structure determination from powder diffraction data by the application of a genetic algorithm” [J]. *Z. Kristallog.* 212(8): 550–552. DOI: 10.1524/zkri.1997.212.8.550.
- R. D. Shannon (1969). “Revised effective ionic radii and systematic studies of interatomic distances in halides and chalcogenides” [J]. *Acta Cryst.* A32(5): 751–767. DOI: 10.1107/S0567739476001551.
- D. Shechtman, I. Blech, D. Gratias *et al.* (1984). “Metallic Phase with Long-Range Orientational Order and No Translational Symmetry” [J]. *Phys. Rev. Lett.* 53(20): 1951–1953. DOI: 10.1103/PhysRevLett.53.1951.
- G. M. Sheldrick (2010). *Some crystallographic algorithms* [M/OL]. <http://shelx.uni-ac.gwdg.de/SHELX/cryst-alg.pdf>, accessed on 2018-06-28.
- A. L. Spek (2003). “Single-crystal structure validation with the program PLATON” [J]. *J. Appl. Cryst.* 36(1): 7–13. DOI: 10.1107/S0021889802022112.
- O. Tange (2011). “GNU Parallel: The Command-Line Power Tool” [J]. *;login: The USENIX Magazine*, 36(1): 42–47.
- K. Thompson and D. M. Ritchie (1972). *Unix Programmer’s Manual* [M]. 2nd ed. Murray Hill, NJ: Bell Telephone Laboratories, Inc., ii.
- Y.-C. Wang, J. Lv, L. Zhu *et al.* (2012). “CALYPSO: A method for crystal structure prediction” [J]. *Comput. Phys. Commun.* 183(10): 2063–2070. DOI: 10.1016/j.cpc.2012.05.008.
- M. M. Wolfson (1997). *An Introduction to X-Ray Crystallography* [M]. 2nd ed. Cambridge: Cambridge University Press, 231.



## 致谢

首先要感谢的是董成老师，因为如果没有他提出原子重叠的自动化处理这样一个极其适合我的课题，我很难如此顺利地进行研究生期间的科研工作并完成本文。不仅如此，尽管我和董成老师之间在不少技术问题上有明显的分歧，他不但没有阻止我投稿，而且主动要求在我的论文 (Liu 2017、2018) 中不署名，甚至批准了我的提前答辩申请；能这样做的导师在研究生导师中未必是多数，因此我衷心地感谢董成老师的高风亮节。

我也要感谢北大化学院的王颖霞老师和中科院高能所的王焕华老师，因为他们不仅分别从化学家和物理学家的角度让我入了晶体学的门，而且在晶体学之外的不少方面给了我很多帮助。还要感谢的是北大数学院的范后宏老师以及 *Tensor Geometry* (Dodson、Poston 1991) 的作者，因为前者开设的线性代数课程和后者关于同一主题的阐述使我对代数和几何之间关系的理解有了质的飞跃，而数形结合的思路正是解决原子重叠问题的数学基础。

感谢 Ken Thompson、Dennis Ritchie 等等 Unix 先驱以及 Daniel J. Bernstein、Laurent Bercot 等等 Unix 精神继承者，因为他们所提倡的 Unix 哲学 “do one thing and do it well” (Salus 1994) 使我不仅能以很高的效率完成 *decryst* 和其它的项目中的编程任务，而且能以更深刻的眼光来审视学习和生活中的问题，这样的影响是无价的。我也要感谢北大 Linux 俱乐部和其它一些开源社区，因为它们提供的平台使我接触了来自各家的技术观点，这也是无价的。

感谢 *decryst* 的测试用户，*decryst* 的文档质量在他们的意见和建议之下得到了极大的提升；感谢 *TeX/LaTeX* 社区的许多人，他们的工作极大地简化了本文的撰写。最后也要感谢 AMCSD 数据库 (Downs、Hall-Wallace 2003) 和 Bilbao 服务器 (Aroyo 等 2006) 的作者在开放获取的精神之下提供了宝贵的数据和一些有用的工具，以及感谢 *GNU Parallel* (Tange 2011) 作者的工作极大地方便了本文中一些数据的处理。本文中的相关工作受到了国家自然科学基金（批准号：21271183）和国家重点研发计划（批准号：2017YFA0302903）的资助。

我用一种悲悯的心情来写剧中人物的争执。我诚恳地祈望着  
看戏的人们也以一种悲悯的眼来俯视这群地上的人们。

——《〈雷雨〉序》(曹禺 1936)



## 作者简历和研究成果目录

### 作者简历

- 2008 年 9 月 – 2009 年 7 月在北京大学医学部药学院学习。
- 2009 年 9 月 – 2013 年 7 月在北京大学化学与分子工程学院获得学士学位。
- 2013 年 9 月 – 2014 年 7 月在家报考中国科学院物理研究所。
- 2014 年 9 月至今在中国科学院物理研究所攻读博士学位。

### 已发表或正式接受的学术论文

- Y. Liu (2017). “Real-time detection and resolution of atom bumping in crystallographic models” [J]. *Acta Cryst. A*73(5): 414–422. arXiv: 1708.03180.
- Y. Liu (2018). “decryst: an efficient software suite for structure determination from powder diffraction” [J]. *J. Appl. Cryst.* 51(4): 1237–1243. arXiv: 1801.08372.

### 参与的研究项目及获奖情况

- 参与国家自然科学基金项目（批准号：21271183）“层状过渡金属硫属化合物的合成、晶体结构和超导电性研究”。
- 参与国家重点研发计划项目（批准号：2017YFA0302903）“关联电子的量子效应及调控”。
- 2017 年 12 月获中国科学院物理研究所所长奖学金表彰奖。

