# pyMatchXRD: Estudo de Estruturas

Vinícius dos Passos de Souza*
(Dated: May 14, 2025)

Neste trabalho, apresentamos um estudo sobre a determinação de estruturas cristalinas a partir de padrões de difração de raios X. Inicialmente, realizamos uma revisão das abordagens existentes na literatura e, em seguida, propomos estratégias próprias para abordar esse problema. Mostramos que é possível obter uma estrutura compatível com um determinado padrão de difração; no entanto, ressaltamos que a estrutura resultante nem sempre corresponde à real do material analisado, uma vez que o processo se assemelha a um ajuste ("fit") dos parâmetros estruturais aos dados experimentais.

## I. INTRODUÇÃO

A difração de raios-X (DRX) estabeleceu-se como uma das técnicas mais importantes na caracterização de materiais cristalinos. Como descrito em literatura fundamental [1], existe uma relação direta entre a estrutura atômica de um material e seu padrão de difração, o que torna esta técnica poderosa para identificação estrutural. Contudo, o processo inverso - determinar a estrutura cristalina a partir do padrão de difração - representa um desafio significativo, motivando o desenvolvimento contínuo de novas abordagens computacionais.

Os esforços de pesquisa concentram-se em três objetivos principais:

1. **Identificação de fases**: Determinar a composição multifásica de materiais (e.g., $TiO_2$ e $Ti_3C_2$) e quantificar suas proporções relativas;
2. **Indexação cristalográfica**: Determinar os parâmetros da célula unitária $[a, b, c, \alpha, \beta, \gamma]$;
3. **Solução estrutural**: Resolver a posição atômica em materiais monofásicos.

As abordagens convencionais empregam dois métodos distintos: (i) técnicas de *search-match*, que comparam os padrões experimentais com bancos de dados cristalográficos (COD, ICSD, PDF); e (ii) métodos de refinamento estrutural, como os implementados nos softwares Profex e TOPAS, que combinam algoritmos de gradiente descendente com métodos estocásticos (e.g., Monte Carlo). A Figura 1 ilustra esses dois paradigmas de análise.

Embora eficazes, esses métodos enfrentam limitações intrínsecas, particularmente a tendência de convergir para mínimos locais que não representam a solução estrutural verdadeira. Recentemente, técnicas de *machine learning* emergiram como alternativa promissora [2]. Redes neurais convolucionais, quando treinadas com grandes conjuntos de dados, podem identificar fases cristalinas diretamente a partir de espectros de DRX, enquanto abordagens híbridas combinam aprendizado de

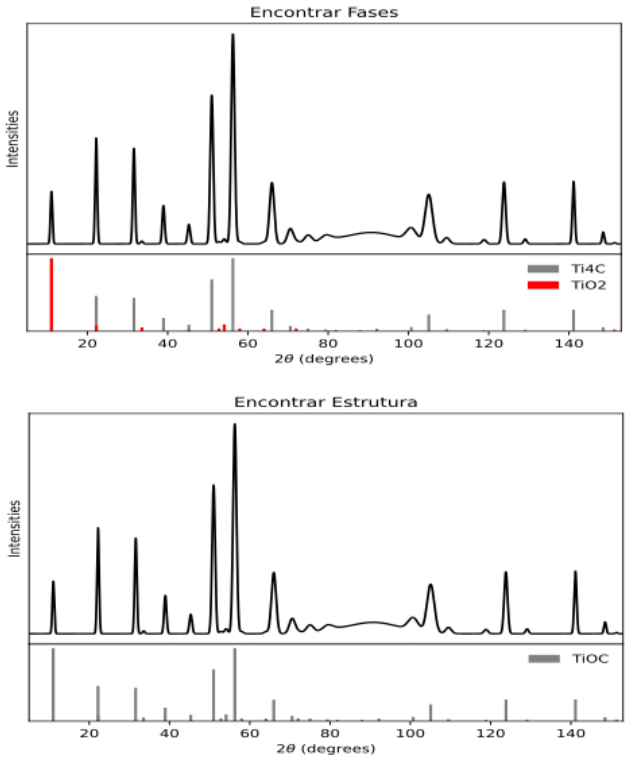


FIG. 1. Abordagens convencionais para análise de dados de DRX: (A) identificação de fases por *search-match*; (B) refinamento estrutural via otimização numérica.

máquina com métodos tradicionais para determinação estrutural [3].

Estudos demonstram que essas técnicas alcançam acurácias de até 95% (tipicamente 80%), porém apresentam desafios significativos:

- **Requisitos de dados**: Necessitam de bancos com masi que 60 mil estruturas para treinamento, demandando recursos computacionais substanciais;
- **Falta de generalização**: Modelos são específicos para famílias de materiais, limitando sua transferibilidade para novos sistemas.

Este trabalho propõe o uso de Algoritmos Genéticos (GA) como alternativa para superar essas limitações. O

* Federal University of the ABC, Santo André, Brazil.

GA destaca-se por sua capacidade de: (i) explorar eficientemente espaços de parâmetros complexos, evitando mínimos locais; (ii) adaptar-se a diferentes classes de materiais sem exigir extensos conjuntos de treinamento; e (iii) integrar-se com métodos existentes para melhorar a confiabilidade das soluções estruturais.

## II. MÉTODOS

A aplicação do algoritmo genético à determinação estrutural por difração de raios-X baseia-se em dois cálculos fundamentais:

1. **Determinação dos ângulos de Bragg**: Para uma dada célula unitária, os ângulos $\theta_{hkl}$ dos picos de difração são calculados pela lei de Bragg:

$$\sin\theta_{hkl} = \frac{\lambda}{2d_{hkl}}. \tag{1}$$

2. **Cálculo das intensidades**: As intensidades relativas dos picos são determinadas pelo fator de estrutura:

$$I_{hkl} = |F_{hkl}|^2. \tag{2}$$

Esses cálculos definem dois problemas de otimização distintos: (i) determinação dos parâmetros da célula unitária e (ii) determinação das posições atômicas, detalhados nas subseções seguintes.

### A. Otimização da Célula Unitária

Primeiro podemos determinar a célula unitária que melhor descreve os ângulos com picos, para isso, podemos criar um algoritmo, no qual os indivíduos são representados por:

1. Parâmetros cristalográficos: $a$, $b$, $c$, $\alpha$, $\beta$, $\gamma$
2. Sistema cristalino: hexagonal, tetragonal, triclínica, ortorrômbica, cúbica, monoclínica ou trigonal

Essa construção foi feitas, para podemos aplicar restrições geométricas reduzem a dimensionalidade do problema. Por exemplo, para sistemas cúbicos:

$$a = b = c \quad \text{e} \quad \alpha = \beta = \gamma = 90^\circ \tag{3}$$

Assim, o algoritmo opera através das seguintes etapas:

- Geração aleatória inicial da população, com sistemas cristalinos diferentes e diferentes valores dos parametros;
- Seleção dos 50% melhores indivíduos, chamados de "elite";
- Reprodução da "elite", gerando crianças, as quais sofrem mutação e compoem nova geração, mantendo, o melhor 10%;
- Continuar processo, mas caso tenhamos estagnação, o fitness não se alterou nas últimas gerações, mantemos o melhor indivíduo e repopulamos.

Uma vez que propomos mecanismo a ser usado, precisamos definir como calcular o quão próximo cada indivíduo pode estar do ideal. Para isso, criamos uma função "compute_fitness", na qual usamos uma função (`generate_hkls`, criada por mim) que calcula os ângulos de difração para uma dada célula unitária. A comparação entre os picos calculados e experimentais é realizada através de uma função de custo definida como:

$$C = \left(\frac{\vec{u}\cdot\vec{v}}{|\vec{u}|\cdot|\vec{v}|}\right)^p, \tag{4}$$

onde:

- $\vec{u}$ é o vetor experimental (1 para ângulos com pico, 0 caso contrário)
- $\vec{v}$ é o vetor calculado para a célula proposta
- $p$ é um expoente de penalização ajustável

Assim, vamos seguir as etapas utilizando essa funçao para avaliar o desempenho de cada célula achada, podendo mudar sua estrutura e parametros e com espaço de parametros discretizado, para estar para os parametros a,b,c com variação de 0.01 entre cada ponto, num espaço de [1,25]. Enquanto para os ângulos temos variação de $0.1^{\text{o}}$ num espaço de $[50^{\text{o}},125^{\text{o}}]$.

### B. Otimização das Posições Atômicas

Com a célula unitária determinada, podemos agora olhar para outro problema, no qual podemos definir indivíduos por

- Posições atômicas $(x, y, z)$;
- Grupos de simetria permitidos;
- Tipos atômicos.

Nos quais temos como fixo a célula unitária e os tipos de átomo possíveis, fornecidos pelo usuário, o qual conhece os elementos que usou para fazer o material e no qual, para podermos fazer processo parecido com o feito para a célula unitária, precisamos definir a função de custo que devemos usar, a qual aqui definimos como

$$C = \sum_i |I_{\text{calc}}(\theta_i) - I_{\text{exp}}(\theta_i)|^2, \tag{5}$$

no qual temos a subtração das intensidades para cada pico, o que é geralmente feito para comparar ajuste e dados experimentais, o que é mais ou menos o que estamos fazendo.

Tendo isso, podemos ter mesmas etapas que para célula unitária, no entanto aqui devemos ter mutações e reproduções melhor deifinidas, para isso, geramos as funções:

1. **mutate()**:
   - Translação aleatória de átomos
   - Troca de espécies atômicas
   - Duplicação/remoção de átomos
2. **crossover()**:
   - Combinação de posições atômicas de pais
   - Herança de grupos de simetria

Adotou-se uma estratégia de repopular parcialmente quando a melhor solução permanece inalterada por múltiplas gerações, mantendo apenas o melhor indivíduo e reinicializando o restante da população. Sendo que assim como no último método, discretizamos espaço para que átomos estejam dentro da célula unitária, ou seja, [0,1) com variação de um ponto de no mínimo 0.01. Após achar esses átomos é aplicada uma operação de simetria, dada as possíveis para a célula unitária usada.

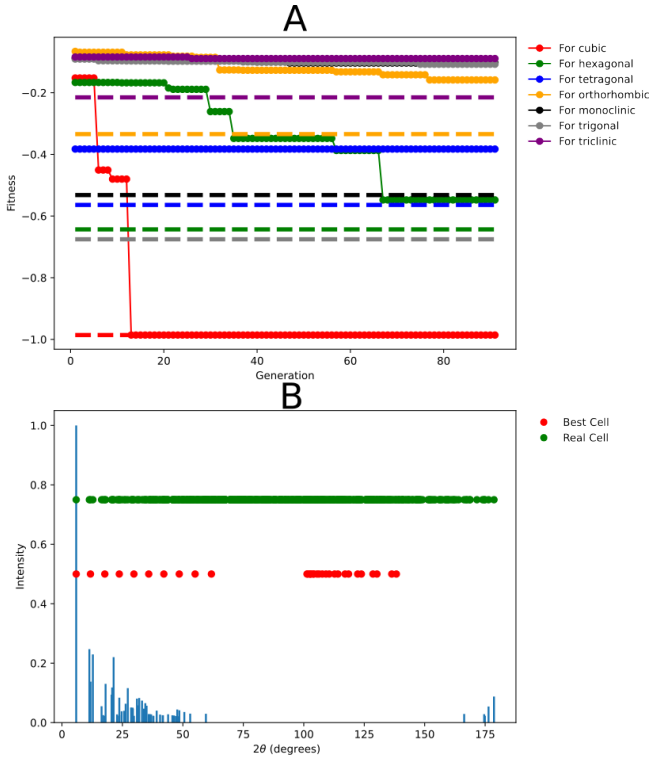


FIG. 2. Execução do algoritmo com população de 70 indivíduos por 40 gerações. Em (A), o valor de fitness por geração para diferentes padrões de difração, com a linha tracejada indicando o valor esperado para a célula correta. Em (B), comparação entre a célula real e a solução obtida para um padrão ortorrômbico.

## III. RESULTADOS E DISCUSSÕES

Com base nos algoritmos e métodos utilizados, o primeiro passo foi definir quais funções de custo adotar. No caso da determinação da célula unitária, é necessário escolher uma função que favoreça a identificação correta da célula, ainda que esta não seja única ou exata. O objetivo é encontrar uma célula que contenha todos os picos observados experimentalmente. No entanto, é possível que células incorretas também apresentem esses mesmos picos, e, portanto, é necessário penalizar soluções que apresentem picos em excesso — evitando, assim, a seleção de células com parâmetros exageradamente altos, que resultariam em alta densidade de picos em regiões amplas do espectro.

Para isso, exploramos diferentes valores do parâmetro $p$ na função de custo. Quando $p = 1$, observamos que o algoritmo é capaz de encontrar a célula correta em alguns casos, como na estrutura cúbica e uma solução bem próxima como na hexagonal, no qual obtemos célula (1.5,1.5,15), sendo que era desejada (3,3,15). A Figura 2 (A) ilustra o comportamento do fitness ao longo das gerações, onde buscamos minimizar essa métrica. A linha tracejada representa o valor esperado para a célula correta.

Observamos, contudo, que em alguns casos o algoritmo encontra mínimos menores que o valor real, como ilustrado na Figura 2 (B). Ao compararmos os picos dos padrões gerados pela célula real e pela solução do algoritmo, percebemos que ambos apresentam alta densidade de picos, dificultando a identificação da estrutura correta. Para contornar essa limitação, propomos uma modificação no algoritmo: durante a execução, o programa deve armazenar todas as células que conseguem explicar todos os picos observados. Assim, o usuário pode avaliar e escolher, entre essas soluções, aquelas que fizerem mais sentido físico ou químico. Além disso, os valores do coeficiente $p$ afetam o nosso perfil, fazendo com que quanto maior o número de picos que não foram achados maior a penalidade, podendo ser visto no Apêndice B.

Por outro lado, ao analisarmos a etapa de otimização estrutural, deparamo-nos com um desafio importante. Mesmo quando o perfil de difração gerado por uma estrutura ajustada apresenta excelente concordância com os dados experimentais, a estrutura resultante pode ser significativamente diferente da estrutura original. Isso demonstra que, embora possamos obter um bom ajuste, isso não garante a identificação da estrutura correta.

A facilidade em obter uma estrutura que reproduz bem o padrão de difração é análoga à de ajustar curvas: curvas simples (com poucos picos) são mais fáceis de ajustar corretamente, enquanto padrões complexos (com muitos picos) aumentam significativamente a di-

ficuldade de se obter a estrutura exata. A Figura 3 exemplifica essa situação, mostrando ajustes visualmente satisfatórios para estruturas distintas.

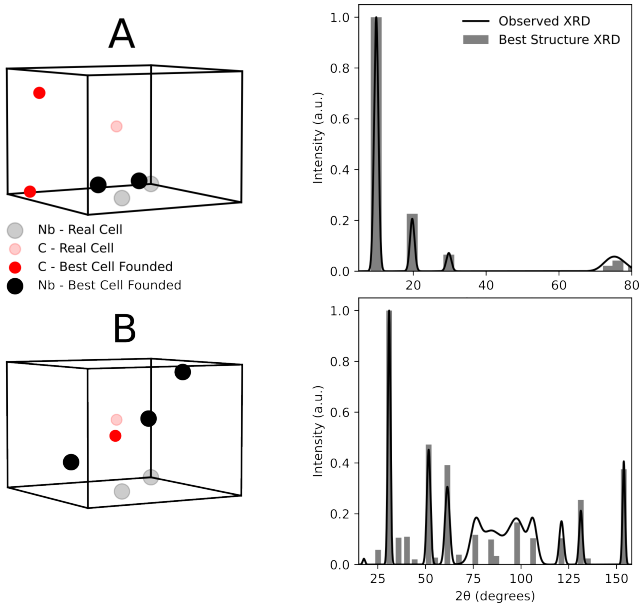


FIG. 3. Comparação entre estruturas geradas e estruturas reais com seus respectivos padrões de difração de raios X. Em ambos os casos, os mesmos átomos e algoritmo foram utilizados, variando apenas a célula unitária: (A) hexagonal (3,3,18) e (B) cúbica (5,5,5).

Portanto, o algoritmo desenvolvido é eficaz em encontrar células que descrevem adequadamente o padrão experimental. No entanto, identificar a célula inicial correta é um problema complexo, que poderia ser mitigado caso houvesse conhecimento prévio de parâmetros como o número de átomos por célula ou a simetria da estrutura. Na ausência dessas informações, o problema permanece altamente desafiador.

## IV. CONCLUSÃO

Neste projeto, tratamos de um problema relevante e crescente na ciência dos materiais: a determinação estrutural a partir de padrões de difração de raios X. Mostramos parte da complexidade envolvida nessa tarefa e propusemos uma abordagem baseada em algoritmos genéticos.

Como resultado, conseguimos determinar os parâmetros de células unitárias para sistemas cúbicos, hexagonais e tetragonais com sucesso. Entretanto, para outras simetrias, enfrentamos maiores dificuldades, principalmente devido ao fato de que a estrutura real pode conter mais picos do que os observados experimentalmente, levando o algoritmo a priorizar outras soluções com menos picos e melhor ajuste.

Concluímos que, embora o ajuste de padrões de difração seja uma ferramenta poderosa, ele não garante a identificação única da estrutura. O algoritmo desenvolvido permite obter uma das possíveis estruturas candidatas, e, se aplicado de forma iterativa (descartando soluções anteriores), pode fornecer uma sequência de estruturas plausíveis. No entanto, a confirmação da estrutura correta exige informações adicionais, já que múltiplas estruturas diferentes podem reproduzir satisfatoriamente o mesmo conjunto de picos.


[1] M. D. Graef and M. E. McHenry, *Structure of Materials: An Introduction to Crystallography, Diffraction and Symmetry* (Cambridge University Press, Cambridge, 2007).

[2] L. Chen, B. Wang, W. Zhang, S. Zheng, Z. Chen, M. Zhang, C. Dong, F. Pan, and S. Li, Crystal structure assignment for unknown compounds from x-ray diffraction patterns with deep learning, Journal of the American Chemical Society **146**, 8098 (2024), pMID: 38477574, https://doi.org/10.1021/jacs.3c11852.

[3] G. Guo, J. Goldfeder, L. Lan, *et al.*, Towards end-to-end structure determination from x-ray diffraction data using deep learning, npj Computational Materials **10**, 209 (2024).

[4] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson, and G. Ceder, Python materials genomics (pymatgen): A robust, open-source python library for materials analysis, Computational Materials Science **68**, 314 (2013).

[5] S. K. Lam, A. Pitrou, and S. Seibert, Numba: a llvm-based python jit compiler, in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15 (Association for Computing Machinery, New York, NY, USA, 2015).

## Appendix A: Cálculo de Padrões de Difração a partir da Estrutura Cristalina

Para realizar todas as análises discutidas neste trabalho, é essencial dispor de uma forma de calcular padrões de difração a partir de estruturas cristalinas. Diversos pacotes em *Python* permitem realizar essa tarefa, como o *pymatgen* [4], que, a partir de um arquivo CIF (Crystallographic Information File), fornece o padrão de difração correspondente.

No entanto, os testes preliminares realizados mostraram que o *pymatgen* não permite uma edição suficientemente flexível das estruturas e de sua representação gráfica, o que motivou o desenvolvimento de um código próprio para esse fim.

Este código foi construído com base nos princípios fundamentais da cristalografia, conforme apresentado em obras clássicas da área [1]. O cálculo do padrão de difração envolve essencialmente três etapas principais:

1. **Indexação**: Determinação dos vetores de difração a partir dos parâmetros da célula unitária, permitindo calcular os ângulos dos planos (hkl) via a Lei de Bragg:

$$\sin\theta_{hkl} = \frac{\lambda}{2d_{hkl}}. \tag{A1}$$

2. **Aplicação das Simetrias**: A partir das posições atômicas e das simetrias do grupo espacial, aplicam-se operações que completam a estrutura, permitindo representar corretamente todos os átomos equivalentes da célula unitária.

3. **Cálculo do Fator de Estrutura**: Com as posições atômicas e os vetores de difração determinados, calcula-se o fator de estrutura, cuja magnitude ao quadrado determina a intensidade dos picos:

$$F_{hkl} = \sum_j f_j \cdot \exp\left(2\pi i \vec{r}_j \cdot \vec{G}_{hkl}\right), \tag{A2}$$

onde $f_j$ é o fator de espalhamento atômico do átomo $j$, $\vec{r}_j$ sua posição, e $\vec{G}_{hkl}$ o vetor de difração.

Esses conceitos são complexos e mereceriam uma discussão mais aprofundada, mas como o foco deste trabalho não é o desenvolvimento do algoritmo em si, limitamo-nos a uma descrição concisa dos procedimentos. O código implementado está disponível na pasta `pyXRD` no repositório deste trabalho GitHub - pyMatchXRD.

### 1. Indexação

A indexação começa com a definição dos vetores da célula unitária no espaço real: $\vec{a}, \vec{b}$ e $\vec{c}$. Esses vetores são construídos a partir dos parâmetros da rede $(a, b, c)$ e dos ângulos $(\alpha, \beta, \gamma)$:

$$\vec{a} = a \cdot \hat{x}, \tag{A3}$$
$$\vec{b} = b \cdot (\cos\gamma\, \hat{x} + \sin\gamma\, \hat{y}), \tag{A4}$$
$$\vec{c} = c \cdot (c_x\, \hat{x} + c_y\, \hat{y} + c_z\, \hat{z}), \tag{A5}$$

onde:

$$c_x = \cos\alpha,$$
$$c_y = \frac{\cos\beta - \cos\alpha\cos\gamma}{\sin\gamma},$$
$$c_z = \sqrt{1 - c_x^2 - c_y^2}.$$

A partir desses vetores, calculamos os vetores da rede recíproca:

$$\vec{a}^* = \frac{\vec{b} \times \vec{c}}{V}, \tag{A6}$$
$$\vec{b}^* = \frac{\vec{c} \times \vec{a}}{V}, \tag{A7}$$
$$\vec{c}^* = \frac{\vec{a} \times \vec{b}}{V}, \tag{A8}$$

com $V = \vec{a} \cdot (\vec{b} \times \vec{c})$, o volume da célula unitária.

O vetor de difração correspondente a um plano $(hkl)$ é então:

$$\vec{G}_{hkl} = h\vec{a}^* + k\vec{b}^* + l\vec{c}^*, \tag{A9}$$

e a distância interplanar $d_{hkl}$ é:

$$|\vec{G}_{hkl}| = \frac{1}{d_{hkl}}. \tag{A10}$$

Substituindo na Lei de Bragg, obtemos:

$$\theta_{hkl} = \arcsin\left(\frac{\lambda}{2}|\vec{G}_{hkl}|\right). \tag{A11}$$

Para esse processo, foi criada a função `generate_hkls`, que recebe o comprimento de onda $\lambda$, os limites de $h, k, l$ e os parâmetros da célula. Considerando o custo computacional, essa função foi acelerada com o uso da biblioteca `numba` [5]. O qual possibilita fazer cálculos "Just in Time" (jit), os quais possibilitam acelerar a velocidade de laços, que é exatamente o motivo de usarmos.

### 2. Aplicação de Simetrias

Os arquivos CIF normalmente trazem apenas as posições atômicas reduzidas, sobre as quais devem ser aplicadas operações de simetria. Essas simetrias estão codificadas pelos grupos espaciais e definem a repetição das posições dentro da célula.

Utilizamos os dados do *pymatgen* armazenados em arquivos JSON para obter os operadores de simetria associados a cada grupo espacial. Por exemplo, o grupo *P1* (número 1) possui apenas a identidade:

$$(x, y, z),$$

enquanto o grupo 4 possui:

$$(x, y, z) \quad \text{e} \quad (-x, -y, z + \tfrac{1}{2}).$$

As operações são aplicadas através da função `POS`, que calcula todas as posições equivalentes e reduz os valores ao intervalo $[0, 1)$ em coordenadas fracionárias, garantindo que estejam dentro da célula unitária.

### 3. Cálculo do Fator de Estrutura

Com os vetores $\vec{G}_{hkl}$ e as posições atômicas $\vec{r}_j$, o fator de estrutura é dado por:

$$F_{hkl} = \sum_j f_j \cdot \exp\left(2\pi i \vec{r}_j \cdot \vec{G}_{hkl}\right), \tag{A12}$$

onde $f_j$ é o fator de espalhamento atômico, calculado por:

$$f_j(\theta_{hkl}) = \sum_i a_i \cdot \exp\left(-b_i \cdot \frac{\sin\theta_{hkl}}{\lambda}\right) + c, \tag{A13}$$

com coeficientes $a_i, b_i$ e $c$ tabelados na International Tables for Crystallography.

A intensidade observada nos experimentos de difração de pó envolve ainda fatores adicionais:

- **Fator de Lorentz-Polarização**:

$$L_p(\theta_{hkl}) = \frac{1 + \cos^2 2\theta_{hkl}}{\sin^2\theta_{hkl}\cos\theta_{hkl}}.$$

- **Fator de Debye-Waller (opcional)**:

$$dw = \exp\left(-\frac{B(T)}{4} \cdot \left(\frac{\sin\theta_{hkl}}{\lambda}\right)^2\right).$$

  Esse fator pode ser negligenciado na ausência de dados confiáveis para o parâmetro $B(T)$.

- **Fator de Multiplicidade** ($\rho_{hkl}$): Número de planos equivalentes com o mesmo $\theta$.

A intensidade final para cada plano é então:

$$I(\theta_{hkl}) = |F_{hkl} \cdot dw|^2 \cdot \rho_{hkl} \cdot L_p(\theta_{hkl}). \tag{A14}$$

Esse conjunto de cálculos permite simular padrões de DRX a partir de estruturas cristalinas arbitrárias, oferecendo flexibilidade para estudos como os apresentados neste trabalho.

### Appendix B: Diferentes valores de p

Para verificar o efeito dos coeficientes $p$ no fitness, fizemos o teste de rodar para as mesmas estruturas o algoritmo para achar a célula unitária dados diferentes valores de $p$, para isso, obtemos a Figura 4, na qual podemos ver que independente da potência não obtivemos muita diferença. No entanto, esse fator aumenta a punição adicionada numa célula caso gere mais picos do que existem. O que é algo bom, porém como não sabemos a quantidade de picos presentes na célula real, não podemos aplicar essa punição sem pensar. Uma vez que assim como visto na Figura 2 (B), na qual podemos ver que a célula ideal possui muitos picos, bem mais do que os vistos, por isso podemos acabar punindo tanto essa resposta, que não a obteriamos. No entanto para o caso de células triclinicas, essas potências podem ser interessantes, uma vez que nos possibilitam limitar a quantidade de picos, já que nesses casos temos muitos picos unidos, assim teriamos como retirar casos extremos. Porém, ao final cabe a quem usar ver qual potência trás o resultado mais próximo do desejado.

### Appendix C: Outros Métodos Testados

Embora o método principal apresentado tenha sido o mais eficaz para a determinação da célula unitária, diversos outros algoritmos foram testados ao longo do desenvolvimento do projeto. A seguir, descrevo brevemente alguns desses métodos e as razões pelas quais foram descartados:

1. **PSO (Particle Swarm Optimization)**: Este algoritmo, amplamente utilizado em problemas de busca em espaços complexos, pode ser metaforicamente comparado ao lançamento de peixes em um rio, onde cada partícula (ou "peixe") explora o espaço em busca de mínimos locais, sendo guiada tanto por seu histórico individual quanto pela melhor solução global encontrada. No entanto, observou-se uma forte tendência à estagnação em mínimos locais, o que impediu a exploração eficiente do espaço de busca. No entanto, dados testes em que fixamos parametros, por exemplo, fixamos que temos célula cúbica, o algoritmo chegou perto do valor, no entanto a partir de hexagonal não foi obtido um bom resultado.

2. **Algoritmo Genético com Ilhas**: Trata-se de uma variação do algoritmo genético utilizado, onde populações independentes (ilhas) evoluem separadamente, cada uma associada a uma estrutura cristalina distinta. Quando uma ilha se encontra estagnada (sem melhora por várias gerações), ocorre uma migração entre ilhas. Após metade das gerações previstas, quatro ilhas são removidas, permitindo o refinamento das três melhores soluções.

Apesar de conceitualmente promissor, esse método apresentou um custo computacional elevado, devido à necessidade de manter populações suficientemente grandes em cada ilha para garantir diversidade genética.

3. **Algoritmo Genético sem Fixação de Estrutura**: Nesta abordagem, a estrutura cristalina não é previamente fixada, permitindo a busca em um espaço de seis dimensões (a, b, c, $\alpha$, $\beta$, $\gamma$). No entanto, essa liberdade excessiva resultou em convergência para soluções com parâmetros de rede excessivamente grandes (valores altos de $a$, $b$ e $c$), que produziam espectros com alta densidade de picos em uma região restrita do ângulo de difração, incompatíveis com os padrões experimentais.

Em resumo, nenhum dos métodos alternativos superou o desempenho do algoritmo genético com estrutura fixa, seja pela limitação na exploração do espaço de busca, alto custo computacional ou inadequação das soluções encontradas.

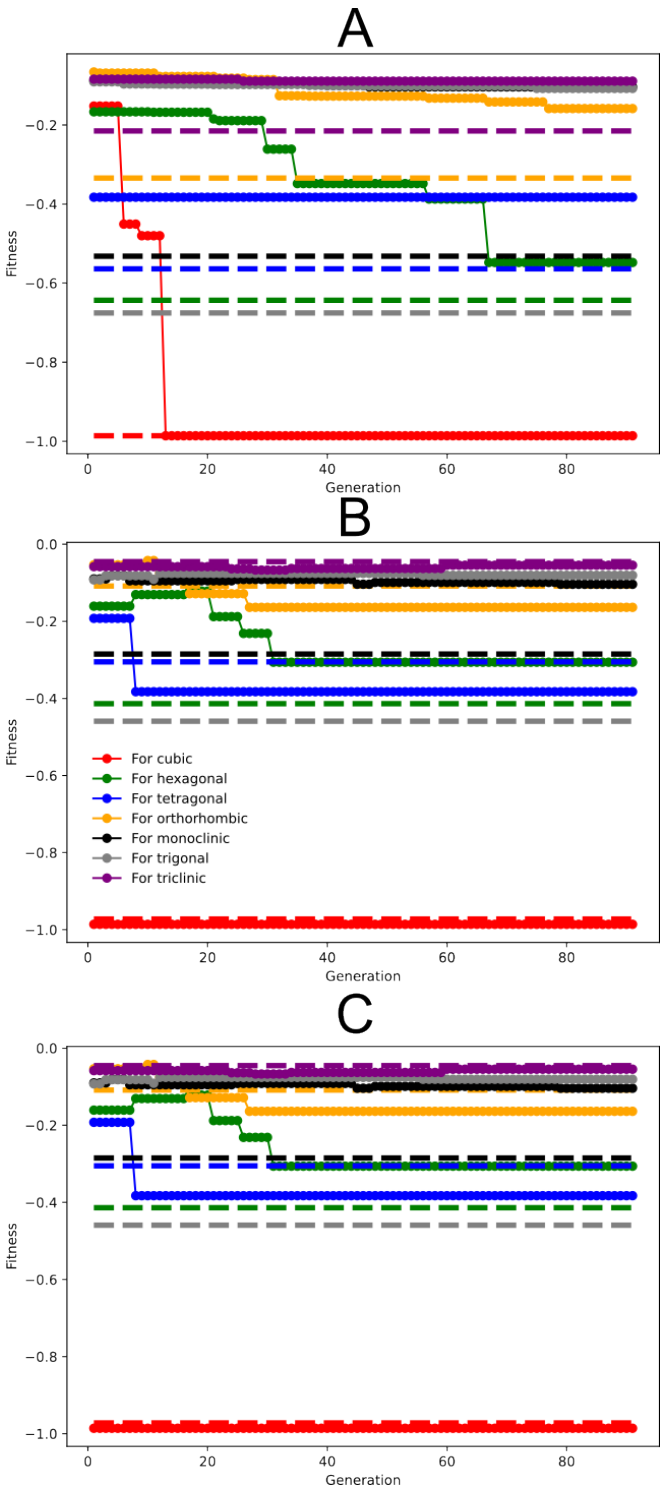


FIG. 4. Comparação entre resultados de algoritmo genético para célula unitária para função de custo com diferentes fatores $p$.