Universidad Nacional De Colombia
Facultad de Ciencias

Parcial 2: Cadenas de Markov y el diseño de nuevos materiales

Juan Sebastián Rueda Segura
Sara Sofía Rivera Sanabria
Santiago Rocha Pachón

Freddy Rolando Hernández Romero
Cadenas de Markov
Bogotá, Colombia
26 de diciembre de 2025

[display] 0.5ex [
]

Tabla de contenidos

Capítulo 1

Calentamiento en un modelo 2D sencillo

Todo a continuación se realizó en un cuaderno de Jupyter accesible mediante Colab aquí.

1.1. Descripción

El sistema a considerar para este ejemplo "juguete" es una red 4×4 en dos dimensiones, en la que los átomos de tierras raras (**R**) se ubican en el cuadrado 'interno' y los de hierro (**Fe**) en el 'externo' (ver 1.1).

El objetivo es encontrar la mejor posición de la red para sustituir un **único** átomo de hierro por uno de titanio (**Ti**), es decir, aquella en la que, al hacer la sustitución, se minimice la energía total. Esto se realizará usando dos métodos: fuerza bruta, es decir, cálculo exhaustivo de las energías de todas las posibles sustituciones y recocido simulado.

1.2. Fuerza bruta - Tarea 1

Para resolver el problema mediante fuerza bruta, implementamos un sistema basado en clases que modela la física del material de nuestro interes. La energía total del sistema se calcula mediante el potencial de Morse, que describe la interacción entre pares de átomos según la fórmula:

$$U(r) = D_0 \left[e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)} \right] \tag{1.1}$$

donde r es la distancia entre dos átomos, D_0 es la profundidad del pozo de potencial, α controla el ancho de la función, y r_0 es la distancia de equilibrio. La energía total del sistema se obtiene sumando las contribuciones de todos los pares únicos de átomos:

$$E_{\text{total}} = \sum_{i=1}^{N} \sum_{j>i}^{N} U_{ij}(r_{ij}) \tag{1.2}$$

La implementación utiliza una clase `MorseParameterDatabase` que almacena los parámetros D_0, α y r_0 específicos para cada tipo de interacción atómica: Fe-Fe, Fe-R, R-R, Fe-Ti, R-Ti, Ti-Ti. Tambien usa una clase `Atom` que representa cada átomo individual con su tipo (tanto el elemento atómico como si es fijo o no) y posición en la red, y una clase `CrystalLattice2D` que gestiona la estructura completa de la red, implementada como una matriz bidimensional.

El algoritmo de fuerza bruta implementado en la clase `BruteForceOptimizer` evalúa sistemáticamente todas las configuraciones posibles. Dado que la red contiene doce posiciones de hierro susceptibles de ser reemplazadas por titanio, el algoritmo calcula la energía total para cada una de estas doce configuraciones. Este enfoque exhaustivo garantiza encontrar el óptimo global, aunque su complejidad computacional lo hace inviable para sistemas más grandes.

Los resultados de la búsqueda por fuerza bruta se presentan en la tabla 1.1, donde se observa que hay cuatro posiciones óptimas correspondientes a las 'esquinas' del retículo, es decir, las coordenadas $(0,0), (0,3), (3,0)$ y $(3,3)$, que minimizan la energía del sistema al obtener el valor -9.744382 eV. Estas configuraciones sirven como referencia para validar el recocido simulado que se presenta en la siguiente tarea (1.3).

1.3. Recocido simulado - Tarea 2

El recocido simulado es un algoritmo inspirado en el proceso de enfriamiento controlado explorando el espacio de configuraciones mediante una cadena de Markov cuya distribución estacionaria es la distribución de Boltzmann a temperatura T:

$$\pi_{f,T}(s) = \frac{1}{Z_{f,T}} \exp\left(-\frac{f(s)}{T}\right) \tag{1.3}$$

donde $Z_{f,T}$ es la constante de normalización y $f(s)$ es la función de energía del estado s (que en lo que concierne al documento, es la misma 1.2 en función de s). A medida que la temperatura decrece, esta distribución concentra progresivamente más probabilidad en los estados de menor energía.

La implementación utiliza el criterio de aceptación de Metropolis, que constituye el núcleo del algoritmo. En cada iteración, se propone una transición a un estado vecino generado aleatoriamente. Si la nueva configuración tiene menor energía, se acepta incondicionalmente. Si tiene mayor energía, se

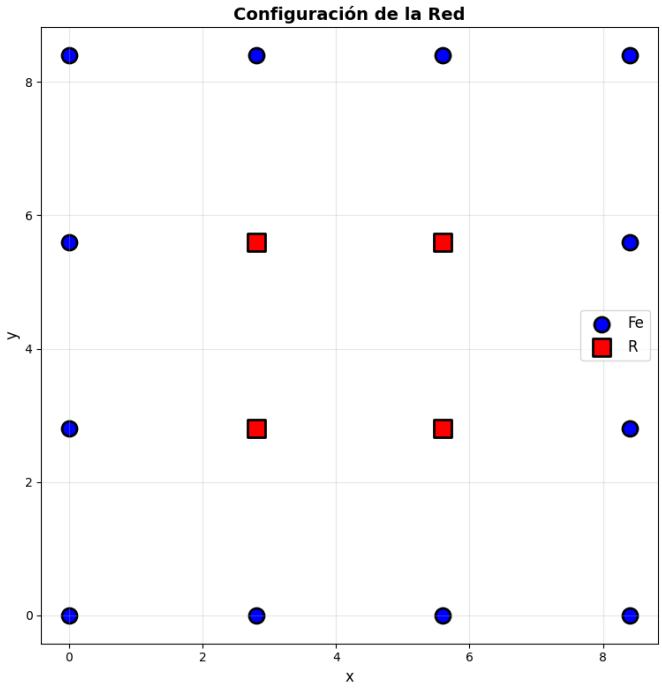

Figura 1.1: Configuración inicial

```python
lattice = create_4x4_lattice()
print("Red cristalina 4x4 creada.")
print(f"Total de átomos: {len(lattice.atoms)}")
print(f"Sitios disponibles: {sum(sum(1 for a in row if not a.fixed) for row in lattice.atoms)}")

brute_force = BruteForceOptimizer(lattice)
best_pos_bf, best_energy_bf, all_energies = brute_force.optimize()

lattice.set_ti_position(best_pos_bf)
lattice.visualize(f"Configuración Óptima (Fuerza Bruta)\nPosición: {best_pos_bf},
                                                    Energía: {best_energy_bf:.6f}")
```

Celda de código 1.1: `_metropolis`, núcleo de la implementación del algoritmo MH.

Cuadro 1.1: Energías calculadas para todas las posiciones posibles de Ti

Posición (x, y)	**Energía (eV)**	**Diferencia (eV)**
(0, 0)	-9.744382	0.000000
(0, 1)	-9.157522	0.586860
(0, 2)	-9.157522	0.586860
(0, 3)	-9.744382	0.000000
(1, 0)	-9.157522	0.586860
(1, 3)	-9.157522	0.586860
(2, 0)	-9.157522	0.586860
(2, 3)	-9.157522	0.586860
(3, 0)	-9.744382	0.000000
(3, 1)	-9.157522	0.586860
(3, 2)	-9.157522	0.586860
(3, 3)	-9.744382	0.000000

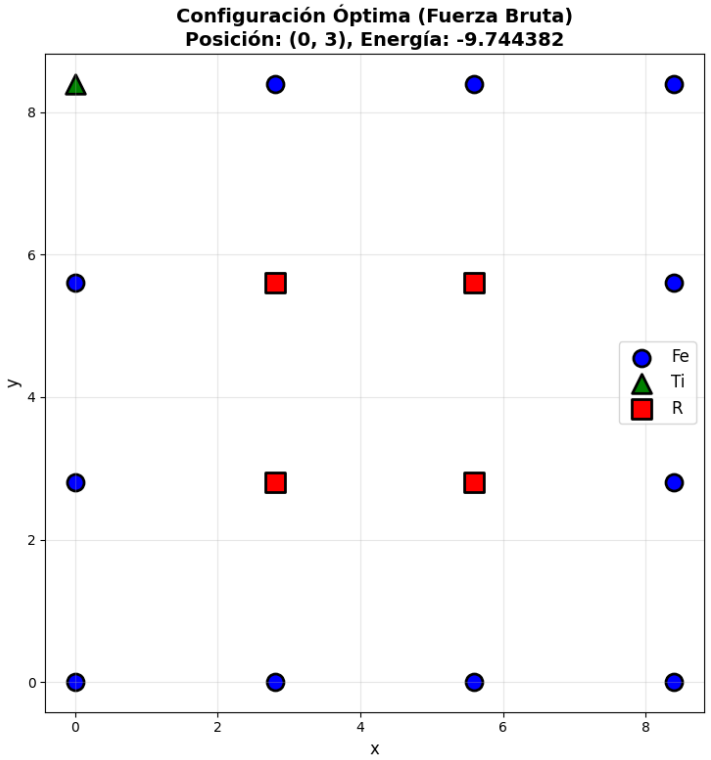

Figura 1.2: Una de las configuraciones óptimas encontrada por fuerza bruta. El átomo de Ti (verde) se ubica en una de las cuatro posiciones que minimiza la energía total del sistema.

acepta con una probabilidad calculada en terminos de la energia y temperatura. Esta regla permite escapar de mínimos locales a temperaturas altas, mientras que a temperaturas bajas el sistema se vuelve progresivamente más selectivo, aceptando principalmente transiciones que reducen la energía.

El esquema de enfriamiento implementado es de tipo geométrico, donde la temperatura se actualiza según $T_{n+1} = \alpha \cdot T_n$ con $\alpha = 0.95$. Los parámetros elegidos fueron temperatura inicial $T_0 = 10.0$, temperatura final $T_f = 0.01$, y diez iteraciones por nivel de temperatura.

Esta configuración representa un balance entre velocidad de convergencia y probabilidad de encontrar el óptimo global. Ahora, para verificar la robustez del algoritmo, se ejecutaron tres corridas independientes desde posiciones iniciales aleatorias. La figura 1.3 muestra la evolución típica de la energía durante la última ejecución, donde se observa que el sistema acepta transiciones ascendentes a temperaturas altas y se estabiliza en el óptimo a temperaturas bajas. La tasa de éxito observada, definida como la proporción de ejecuciones que convergen al óptimo global identificado por fuerza bruta, fue del 100 %, lo cual valida la efectividad del calendario de enfriamiento seleccionado.

La comparación entre ambos métodos revela que, aunque la fuerza bruta garantiza encontrar el óptimo global, su costo computacional crece factorialmente con el número de posiciones disponibles, haciéndolo inviable para sistemas realistas. El recocido simulado, por otro lado, ofrece una alternativa escalable que puede aplicarse a redes más grandes, sacrificando la garantía de optimalidad por una alta probabilidad de encontrar soluciones de muy buena calidad en tiempo razonable.

1.4. Análisis físico - Tarea 3

Para esta última tarea establecemos el Ti en la posición óptima encontrada por fuerza bruta. Obtenemos el objeto átomo de Ti usando el índice guardado. Filtramos solo los átomos R de la lista completa de átomos. Para cada átomo R, calculamos la distancia al Ti, la guardamos y la imprimimos. Asi, calculamos el promedio de todas estas distancias.

Este análisis numérico nos permite responder ¿Cuál es la característica principal del sitio óptimo? y verificar la conclusión de la tesis de Skelland: “el Titanio prefiere sitios alejados de las tierras raras” [1]. Si vemos que el sitio óptimo está efectivamente lejos de los R, esto confirmará la teoría física detrás del comportamiento del material.

```python
n_runs = 3
sa_results = []
lattice = create_4x4_lattice()

for run in range(n_runs):
    print(f"\n--- Ejecución {run + 1} de {n_runs} ---")

    sa_optimizer = SimulatedAnnealingOptimizer(
        lattice,
        initial_temp=10.0,
        final_temp=0.01,
        cooling_rate=0.95,
        iterations_per_temp=10
        )

    best_pos_sa, best_energy_sa = sa_optimizer.optimize()
    sa_results.append((best_pos_sa, best_energy_sa))

    if run == n_runs - 1:
        sa_optimizer.plot_convergence()

print(f"\nRecocido Simulado ({n_runs} ejecuciones):")
for i, (pos, energy) in enumerate(sa_results, 1):
    match = "MATCH" if np.isclose(pos, best_pos_bf)[0] else "NOT MATCH"
    print(f"  Ejecución {i}: Posición {pos}, Energía {energy:.6f} {match}")

success_rate = sum(1 for pos, _ in sa_results if np.isclose(pos, best_pos_bf)[0]) / n_runs * 100
print(f"\nTasa de éxito: {success_rate:.1f}%")
lattice.set_ti_position(sa_results[-1][0])
lattice.visualize(f"Configuración Final (Recocido Simulado)\nPosición: {sa_results[-1][0]},
                  Energía: {sa_results[-1][1]:.6f}")
```

Cuadro 1.2: Resultados de tres ejecuciones independientes del recocido simulado

Ejecución	Posición Final	Energía Final	Estado
1	(0, 3)	-9.744382	✓
2	(0, 3)	-9.744382	✓
3	(0, 3)	-9.744382	✓
Tasa de éxito: 100 %			

```python
lattice.set_ti_position(best_pos_bf)
ti_atom = lattice.atoms[best_pos_bf[0]][best_pos_bf[1]]
print(f"\nPosición óptima del Ti: {best_pos_bf}")
print("\nDistancias desde el Ti a los átomos R:")

r_atoms = []
for i in range(lattice.shape[0]):
    for j in range(lattice.shape[1]):
        atom = lattice.atoms[i][j]
        if atom is not None and atom.atom_type == 'R':
            r_atoms.append(atom)

distances_to_r = []

for r_atom in r_atoms:
    dist = ti_atom.distance_to(r_atom)
    distances_to_r.append(dist)
    print(f"  R en {r_atom.position}: distancia = {dist:.4f}")

avg_distance = np.mean(distances_to_r)
min_distance = np.min(distances_to_r)
max_distance = np.max(distances_to_r)

energies_dict = {lattice.fe_sites[i]: all_energies[i] for i in range(len(lattice.fe_sites))}

position_distances = {}
for pos in lattice.fe_sites:
    lattice.set_ti_position(pos)
    temp_ti = lattice.atoms[pos[0]][pos[1]]
    temp_distances = [temp_ti.distance_to(r_atom) for r_atom in r_atoms]
    avg_dist = np.mean(temp_distances)
    position_distances[pos] = avg_dist

sorted_positions = sorted(position_distances.items(), key=lambda x: x[1], reverse=True)

print(f"\n{'Posición':<15} {'Dist. Promedio Ti-R':<25} {'Energía':<15} {'Nota'}")
for pos, avg_dist in sorted_positions:
    energy = energies_dict[pos]
    marker = " <- ÓPTIMA" if pos == best_pos_bf else ""
    print(f"{str(pos):<15} {avg_dist:<25.4f} {energy:<15.6f}{marker}")

lattice.set_ti_position(best_pos_bf)
```

El análisis de la configuración óptima revela información importante sobre las preferencias energéticas del titanio en la estructura estudiada. La tabla 1.3 muestra las distancias desde el átomo de titanio en una de sus posiciones óptimas, $(0, 3)$, hasta cada uno de los cuatro átomos de tierra rara ubicados en el centro de la red.

Para establecer si existe una correlación sistemática entre la distancia a los átomos R y la estabilidad energética, se calculó la distancia promedio Ti-R para todas las configuraciones posibles.

En efecto, los resultados obtenidos muestran que las posiciones que maximizan la distancia a los átomos R tienden a minimizar la energía total. Este comportamiento valida cuantitativamente la conclusión cualitativa de Skelland y sugiere que en estructuras cristalinas más complejas, el titanio debería preferir ubicarse en regiones alejadas de concentraciones de tierras raras.

Los resultados de este análisis sencillo en dos dimensiones proporcionan una base sólida para abordar problemas más complejos en tres dimensiones, donde el mayor número de átomos hacen indispensable el uso del recocido simulado, ya que la búsqueda exhaustiva se vuelve computacionalmente costosa.

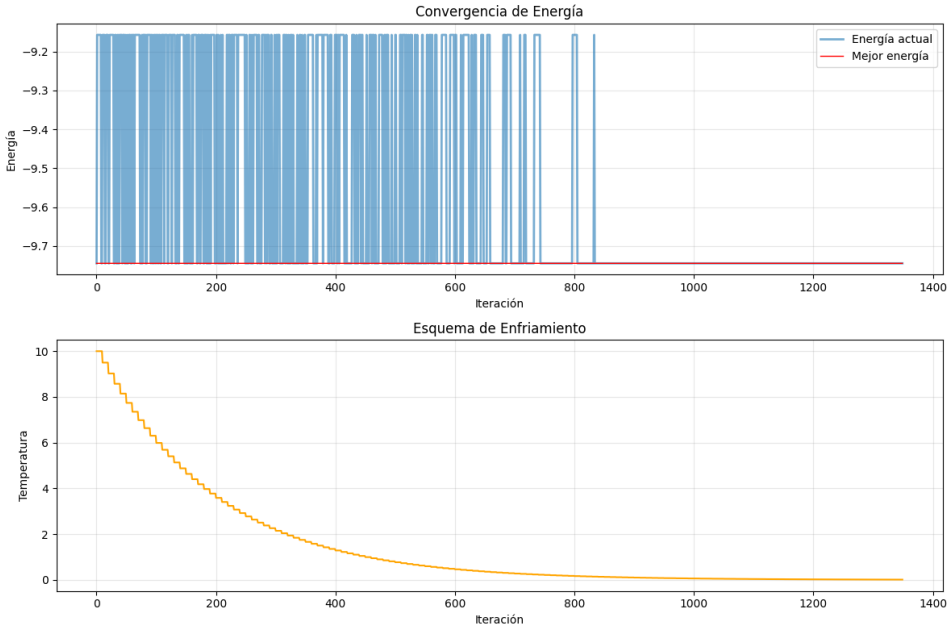

Figura 1.3: Convergencia SA. Evolución de la energía actual con mejor energía encontrada y esquema de enfriamiento exponencial.

Cuadro 1.3: Distancias desde un átomo de Ti en $(0, 3)$ hasta los átomos R

Átomo R en posición	**Distancia (u.a.)**
(1, 1)	6.2610
(1, 2)	3.9598
(2, 1)	7.9196
(2, 2)	6.2610
Promedio	**6.1003**
Mínima	**3.9598**
Máxima	**7.9196**

Cuadro 1.4: Correlación entre distancia promedio Ti-R y energía del sistema

Posición Ti	**Distancia Prom. Ti-R (Å)**	**Energía (eV)**
(0, 3)	6.1003	-9.744382 ← ÓPTIMA
(3, 3)	6.1003	-9.744382
(0, 0)	6.1003	-9.744382
(3, 0)	6.1003	-9.744382
(1, 3)	4.6552	-9.157522
(2, 3)	4.6552	-9.157522
(0, 2)	4.6552	-9.157522
(3, 2)	4.6552	-9.157522
(0, 1)	4.6552	-9.157522
(3, 1)	4.6552	-9.157522
(1, 0)	4.6552	-9.157522
(2, 0)	4.6552	-9.157522

Capítulo 2

Un desafío combinatorio en 2D

2.1. Descripción

A continuación, el sistema a trabajar es una red bidimensional de tamaño 10×10 con:

- 84 átomos de Fe.
- 16 átomos de R cuyas posiciones en sistema coordenado (x, y) cumplen que $x, y \in \{3, 4, 5, 6\}$ (iniciando con $x = 0, y = 0$).

Esta red se puede visualizar mucho mejor en la figura (a) de 2.3.

Por otro lado, es de interés reemplazar 8 átomos de Ti por 8 de Fe para minimizar la energía total de la red. Una posible configuración óptima [1] se muestra en la figura (b) de 2.3.

2.2. Espacio de estados y movimientos

Tal como se especifica en la guía, un estado es un conjunto [2] con cardinalidad 8 cuyos elementos son tomados de $A := \{0, 1, \ldots, 9\} \times \{0, 1, 2, 7, 8, 9\} \cup \{0, 1, 2, 7, 8, 9\} \times \{0, 1, \ldots, 9\}$. Formalmente, el espacio de estados, S, se define como

$$S = \{\{(a_1, b_1), (a_2, b_2), \ldots, (a_{10}, b_{10})\} : a_i, b_i \in A \quad \forall i \in \{1, 2, \ldots, 10\}\} \tag{2.1}$$

[1] Se verá más adelante que en realidad no lo es, pues tiene una energía de -121. COMPLETAR, mucho mayor a la mínima encontrada.

[2] No es una 8-tupla ya que los átomos de Ti son indistinguibles entre sí.

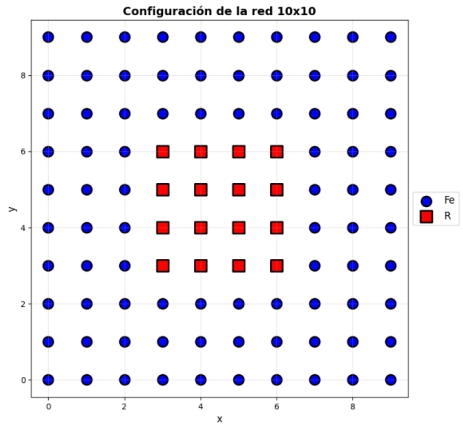

Figura 2.1: *
(a) Red 10×10 básica a considerar.

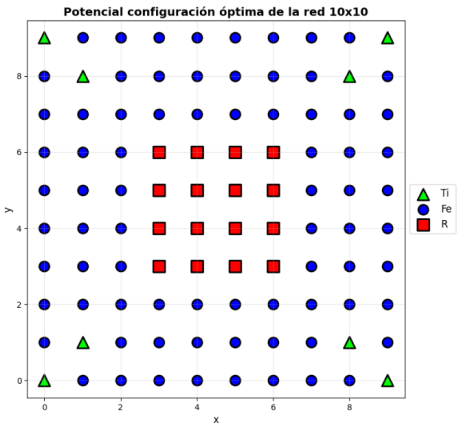

Figura 2.2: *
(b) Red 10×10 con un posicionamiento de los ocho átomos de Ti posiblemente óptimo.

Figura 2.3: Redes 10×10 a considerar con (a) y sin (b) átomos de Ti.

Así, si, por ejemplo, la simulación en el tiempo n tiene una configuración igual a la de la figura (b) de 2.3, quiere decir que la cadena de Markov se encuentra en el estado $s = \{(0,0),(1,1),(0,9),(1,8),(8,8),(9,9),(9,0),(8,1)\} \in S$.

Asimismo, siguiendo también la guía, un movimiento o potencial [3] actualización de un estado a otro, partiendo de una configuración en la que ya se han posicionado los 8 átomos de Ti, será intercambiar alguno de los 8 átomos de Ti (elegido uniformemente) por alguno de los 76 átomos de Fe (elegido también uniformemente pero con un número aleatorio no necesariamente igual).

Esto quiere decir que los estados vecinos serán aquellos que difieran en un único elemento, esto es, un único átomo de Ti posicionado en un lugar distinto o que sean el mismo (esto ya que, como se vió en 1.3, la función de actualización admite que la cadena se mantenga en el mismo estado). Formalmente,

$$s \sim s' \quad \text{si } s = s' \text{ ó existen } (a,b) \in s \backslash s' \text{ y } (c,d) \in s' \backslash s \text{ tales que } s \backslash \{(a,b)\} = s' \backslash \{(c,d)\}$$

Con estos ingredientes en mano, es posible adaptar el código hecho en el capítulo anterior (1) para esta situación.

[3] No es exactamente lo mismo que evaluar la función de actualización, como se verá en 2.3

2.3. Implementación del recocido simulado

En primer lugar, se creó una clase `TbyTLattice` cuya estructura es muy similar a la de `CrystalLattice2D` usada en 1.2 (usa igualmente las clases `Atom` y `MorseParameterDatabase`), con las siguientes modificaciones:

- Se adaptó todo el código de `CristalLattice2D` para que `atoms` fuera una lista de listas de átomos y poder acceder a ellos de forma más intuitiva utilizando la visualización del retículo como matriz de posiciones.
- Se añadió el método `ti_sites_random_init` para inicializar la red con una cantidad dada de átomos de Ti, y se eliminó `add_atom`.
- `set_ti_position` se renombró por `swap_ti_position`, que admite tanto la adición de un átomo de Ti en una posición de un átomo de Fe como el movimiento mencionado en 2.3: el intercambio de un átomo de Ti ya presente en el retículo por uno de Fe.
- Se modificó `calculate_total_energy` para realizara el cálculo exhaustivo o, a partir de una energía previa a un intercambio de un átomo de Fe por uno de Ti, hiciera un cálculo de la energía tras el intercambio más optimizado.

2.3.1. Justificación del cálculo de la energía optimizado

Con respecto al último punto, el método ahora admite `current_energy`, es decir, si se provee, el método aprovecha la energía del retículo que ya se había calculado previo al intercambio de un átomo de Ti por uno de Fe de la siguiente manera: si s son las ocho posiciones actuales de los átomos de Ti y s' es la misma s pero intercambiando el n-ésimo átomo de Ti por el m-ésimo átomo de Fe (es decir, reemplazando la posición del átomo n de Ti por la posición del átomo m de Fe), entonces la diferencia entre la energía de los dos retículos correspondientes se reduce a lo siguiente: asumiendo $n > m$ sin pérdida de generalidad y denotando $U_{ij} := U_{ij}(r_{ij})$ el potencial en s y $U'_{ij}(r_{ij}) := U'_{ij}$ el potencial en s',

$$\Delta E = E(s') - E(s) = \sum_{i=1}^{N}\sum_{j>i}^{N} U'_{ij} - \sum_{i=1}^{N}\sum_{j>i}^{N} U_{ij}$$

$$\left(\sum_{i=1,\, i\neq m,n}^{N}\ \sum_{j>i,\, j\neq m,n}^{N} U_{ij} + \sum_{j\neq m}^{N} U'_{mj} + \sum_{j\neq n,m}^{N} U'_{nj}\right) - \sum_{i=1}^{N}\sum_{j>i}^{N} U_{ij}$$

el primer sumando, ya que para todos los átomos distintos a los ubicados en n y m, el cálculo del potencial sigue siendo el mismo (las distancias y los

tipos de átomos no se han alterado). Por otro lado, en la doble suma del segundo, se están excluyendo todas las parejas en las que alguno de los dos átomos sea el n-ésimo de Ti o el m-ésimo de Fe y se están añadiendo estas en dos sumas: una con el m-ésimo de Fe y otra con el n-ésimo de Ti **salvo** el potencial U_{nm} que ya se está considerando en la primera suma como U_{mn}. Continuando,

$$= \left(\sum_{j \neq m}^{N} U'_{mj} + \sum_{j \neq n,m}^{N} U'_{nj} \right) - \left(\sum_{j \neq m}^{N} U_{mj} + \sum_{j \neq n,m}^{N} U_{nj} \right)$$

haciendo la misma separación que se hizo en el segundo sumando, en el primero. Así,

$$= \left(\sum_{j \neq m}^{N} U'_{mj} + \sum_{j \neq n,m}^{N} U'_{nj} \right) - \left(\sum_{j \neq m}^{N} U_{mj} + \sum_{j \neq n,m}^{N} U_{nj} \right)$$

vale la pena notar que, al hacer el intercambio, $U_{nm} = U'_{nm}$, pues la distancia y los átomos en cuestión no cambian, por lo que se puede sumar y restar sin afectar el resultado, quedando la diferencia en lo siguiente:

$$= \left(\sum_{j \neq m}^{N} U'_{mj} + \sum_{j \neq n,m}^{N} U'_{nj} \right) + U_{nm} - \left(\sum_{j \neq m}^{N} U_{mj} + \sum_{j \neq n,m}^{N} U_{nj} \right) - U_{nm}$$

$$= \left(\sum_{j \neq m}^{N} U'_{mj} + \sum_{j \neq n}^{N} U'_{nj} \right) - \left(\sum_{j \neq m}^{N} U_{mj} + \sum_{j \neq n}^{N} U_{nj} \right)$$

de modo que

$$E(s') = E(s) + \Delta E = E(s) - \left(\sum_{j \neq m}^{N} U_{mj} + \sum_{j \neq n}^{N} U_{nj} \right) + \left(\sum_{j \neq m}^{N} U'_{mj} + \sum_{j \neq n}^{N} U'_{nj} \right) \tag{2.2}$$

lo que permite calcular la energía en tiempo lineal ($\mathcal{O}(4n) = \mathcal{O}(n)$) en vez de cuadrático, como se hacía en el cálculo exhaustivo de la energía ($\mathcal{O}(n(n-1)/2) = O(n^2)$)).

2.3.2. Eficiencia y precisión del cálculo de la energía optimizado

Para cuantificar qué tan rápido es un método con respecto al otro y confirmar que el método optimizado daba lo mismo que el exhaustivo, se ejecutó una celda de código[4] que reemplazaba alguno de los ocho átomos

[4]Se puede encontrar dos celdas después de la definición de la función `create_nxn_lattice` en el cuaderno de Jupyter. Se decidió solamente mostrar los resultados de su ejecución al considerarlos mucho más relevantes.

de Ti por un átomo de Fe, pasando por todos los átomos de Fe posibles y calculaba la energía tras el intercambio por el método exhaustivo y el optimizado.

Durante la iteración por los posibles sitios de Fe, se calcula la diferencia absoluta entre el resultado de los dos métodos, y, si es mayor a 10^{-20} [5] , suma 1 a la cuenta de las discordancias; y además, se registra el tiempo que toma el cálculo de la energía con ambos métodos.

Tras la ejecución de este ciclo, se evidenció que no habían habido ninguna discrepancia entre el método exhaustivo y el optimizado. Además, al promediar y hallar el máximo, mínimo y desviación estándar para cada uno de los métodos, se confirmo que el segundo método es en efecto más eficiente que el primero. En conclusión, el método optimizado toma un poco menos de $\frac{1}{10}$ del tiempo que toma el exhaustivo.

Así, teniendo todos los ingredientes en mano, se pudo proceder a elegir el esquema de enfriamiento y encontrar la posición óptima de los átomos de Ti.

2.4. Utilización del recocido simulado

2.4.1. Calendario de enfriamiento

A pesar de que se sugiría estimar una constante C tal que el esquema de enfriamiento estuviera dado por $T(n) = \frac{C}{\log n}$ [6], por el éxito que se tuvo en la rejilla 4×4 (1) y por la lentitud con la que convergía el esquema logarítmico, se decidió probar las dos alternativas.

Vale la pena mencionar que en ambos casos, se tomó la sucesión de N's constante igual a 100, es decir, $\{N_i\}_{i \in \mathbb{N}} = \{100 \ \forall i \in \mathbb{N}\}$.

Esquema logarítmico

Como se mencionó, se pretendía estimar una constante $C \in \mathbb{R}$ tal que, tomando el esquema de enfriamiento $T(n) = \frac{C}{\log n}$ para $2 \leq n \leq 1,002$ (es decir, enfriando por 1000 periodos de tiempo), se llegara a una configuración que minimizará la energía de la red.

[5] Una salvedad que vale la pena hacer aquí es que, aunque tras un único intercambio de un átomo de Ti y un átomo de Fe la energía seguía dando lo mismo comparando 20 cifras decimales, para estimar la constante C, por ejemplo, se evidenció que, a pesar de que en varias ejecuciones se llegaba a alguna de las configuraciones que minimizaban la energía, su energía difería entre sí por $\pm 1 \cdot 10^{-12}$. Esto se debe, posiblemente, a que al realizar millones de cálculos (ya que se hacían $1,000$ iteraciones del algoritmo) usando `numpy` o las mismas librerías nativas de Python y no un sistema algebráico de cómputo que conserva precisión indefinida a medida que se hacen las operaciones, se 'perdió' precisión decimal, en algunos casos más que otros.

[6] Se tomó log como el logaritmo natural.

En suma, para evitar datos atípicos, por cada constante C se realizó el procedimiento anterior cinco veces y se promedió la energía mínima obtenida.

Esto se realizó ejecutando el código mostrado en 2.1, que almacenó los datos en archivos .csv que luego fueron leídos y se promedió la energía por cada constante tres celdas después. En la tabla 2.2 se presentan los resultados de la ejecución.

Esto quiere decir que, tras $1,000$ iteraciones del recocido (cada una tomando $N = 100$ pasos), si $C \leq 0.25$, entonces converge a la configuración mínima global. Esto implica, ya que se toma $2 \leq n$, que la temperatura inicial tendría que ser menor o igual a $0.3607K$, lo cual puede que funcione para este modelo por su ausencia de mínimos locales y la posibilidad de posicionar el átomo elegido de Ti para intercambiar en cualquiera de las 76 posiciones posibles, pero posiblemente no serviría en general, pues tiene muy poca exploración.

Además, es posible ver en la gráfica 2.4 que, a pesar de que se alcanza el mínimo tras algunas decenas de miles de iteraciones, la energía sigue fluctuando (en un rango más pequeño, pero lo sigue haciendo). Esto se debe, principalmente, a lo lento que decrece $\frac{C}{\log(x)}$, pues su derivada es $-\frac{C}{x\log^2(x)}$. Por otro lado, si se escoge una constante muy pequeña como 0.001 que da una temperatura inicial de 0.0014, es visible en la gráfica de la convergencia de la energía 2.5 que no se le permite pasar a ninguna configuración que no minimice la energía actual. Es por esto que se decidió experimentar con la convergencia de la energía mínima usando un calendario de enfriamiento logarítmico.

Esquema exponencial

Siguiendo el mismo procedimiento descrito en el esquema logarítmico, y usando la misma elección de $\alpha = 0.95$ hecha en 1 para formar el calendario de enfriamiento, se estimó una constante $C \in \mathbb{R}$ tal que tomando $T(n) = C \cdot (0.95)^n$ para $2 \leq n \leq 1,002$, tras $1,000$ iteraciones, se llegara al mínimo de energía encontrado con el esquema logarítmico ($-123.91169923373\,eV$).

Esto se logró modificando manualmente el calendario en el método `run` de `SimulatedAnnealingTbyT` y usando el código de 2.1 y de tres celdas después. A continuación, se muestran los resultados.

Como se puede observar, en constantes con centenas, ya se había alcanzado la energía mínima tras las $1,000$ iteraciones del recocido. En otras palabras, para temperaturas iniciales menores o iguales a $225.625\,K$, el esquema de enfriamiento exponencial con $\alpha = 0.95$ ya convergía a la mínima energía.

Esto se debe, mayoritariamente, a que, además de haber un único máximo global, la derivada de 0.95^x es $-d \cdot 0.95^x$, donde $d \in \mathbb{R}^+$ alguna constante pequeña. Esto implica que, si bien la temperatura desciende más lentamente

Cuadro 2.1: Tiempos de ejecución para cada uno de los métodos que permiten calcular la energía de un retículo $n \times n$.

Método	Medio $\pm$ desviación estándar	Máximo	Mínimo
Optimizado	0.005073 ± 0.000777	0.010486	0.004369
Exhaustivo	0.063965 ± 0.007356	0.115862	0.056803
Diferencia porcentual (Opt. vs. Exh.)	92.0691 %	90.9496 %	92.3085 %

```
    lattice: TbyTLattice = create_nxn_lattice()
    #*Potential constants for logarithmic scheme
    # cooling_constants: list[float] = [(500 - 50*n) for n in range(5, 10)]
    # cooling_constants += [25, 10, 5, 1]
    # cooling_constants += [0.75, 0.5, 0.25, 0.1, 0.075, 0.05, 0.025, 0.001]
    # cooling_constants = [0.000075, 0.000050, 0.000025, 0.000010]

    #*Potential constants for exponential scheme
    cooling_constants = [500, 250, 100, 50, 10, 1, 0.5, 0.05, 0.005]

    print(len(cooling_constants), cooling_constants)
    history: pd.DataFrame = pd.DataFrame(columns=['C', 'Minimum energy', 'Best Ti sites',
    'Total time'])

    #*Simulate for all cooling constants
    for c in cooling_constants:
        print(f"############## C: {c} ##############")
        for _ in range(1):
            sa_optimizer = SimulatedAnnealingTbyT(
                    lattice,
                    cooling_constant=c,
                    n_cooldowns=100
                    )
            best_ti_sites, minimum_energy, time = sa_optimizer.run()
            history = pd.concat([history, pd.DataFrame([{"C": c,
            "Minimum energy": minimum_energy, "Best Ti sites": best_ti_sites,
            "Total time": time}])], ignore_index=True) #*the dict must be
            #*in a list to preserve the list of Ti sites in a single row, col position

    history
    history.to_csv("P2/C_estimation.csv")
```

Celda de código 2.1: Código usado para almacenar la información sobre cinco ejecuciones del recocido simulado para cada valor único de C y estimar la mejor elección de dicha constante.

Cuadro 2.2: Valores promedio de la energía mínima encontrada tras correr 1000 iteraciones del recocido simulado cinco veces para cada constante C usando el esquema logarítmico.

C	Energía mínima promedio
500	-123.70515847521
450	-123.66881470085
400	-123.69987450945
350	-123.77392074812
300	-123.72224143372
250	-123.75721555617
200	-123.67575313103
150	-123.74900457290
100	-123.71693176721
50	-123.69046978475
25	-123.79129641812
10	-123.80010366215
5	-123.81191381237
1.0	-123.87983884963
0.75	-123.89875646046
0.50	-123.90965976169
0.25	-123.91169923373
0.10	-123.91169923373
0.075	-123.91169923373
0.050	-123.91169923373
0.025	-123.91169923373
0.001	-123.91169923373
0.00075	-123.91169923373
0.00050	-123.91169923373
0.00025	-123.91169923373
0.00010	-123.91169923373

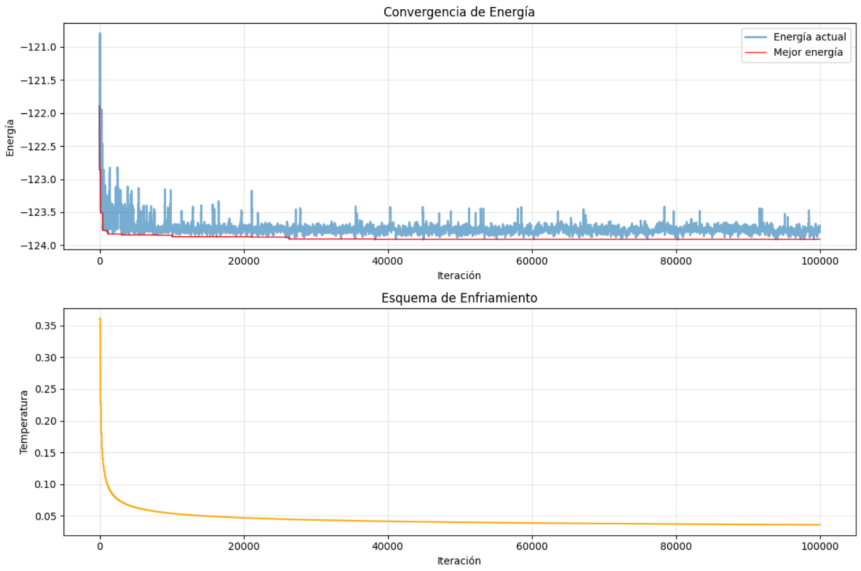

Figura 2.4: Energía de cada iteración a lo largo de $1,000$ pasos de ejecutar el algoritmo de recocido simulado usando el esquema de enfriamiento $\frac{0.25}{\log(n)}$

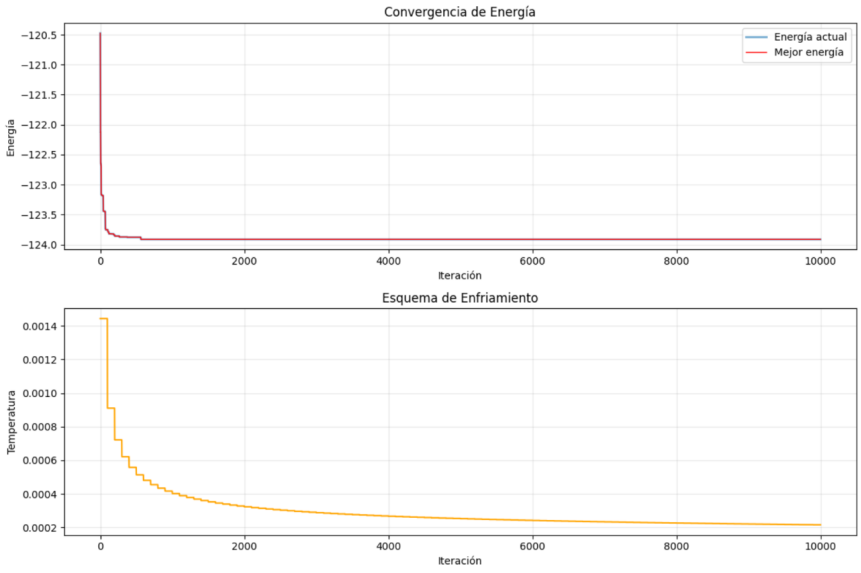

Figura 2.5: Energía de cada iteración a lo largo de $1,000$ pasos de ejecutar el algoritmo de recocido simulado usando el esquema de enfriamiento $\frac{0.001}{\log(n)}$

Cuadro 2.3: Tabla con los valores promedio de la energía mínima encontrada tras correr 1000 iteraciones del recocido simulado cinco veces para cada constante C usando el esquema exponencial.

C	Energía mínima promedio
500.0	-123.89331691916
250.0	-123.91169923374
100.0	-123.91169923373
50.0	-123.91169923373
10.0	-123.91169923374
1.0	-123.91169923373
0.5	-123.91169923374
0.05	-123.91169923374
0.005	-123.91169923373
500.0	-123.91169923373
250.0	-123.91169923373
100.0	-123.91169923373
50.0	-123.91169923373
10.0	-123.91169923373
1.0	-123.91169923373
0.5	-123.91169923373
0.05	-123.91169923373
0.005	-123.91169923374

que usando un calendario logarítmico en las primeras iteraciones del algoritmo, apenas ocurren estas, la temperatura decrece siempre más rápido en el esquema exponencial y se acerca mucho más rápido a 0 que el logarítmico.

Esto implica que, inicialmente, se le da una libertad de exploración bastante grande al recocido, pero esta va decreciendo 'rápidamente' a medida que pasan las iteraciones y se enfría de tal manera que al estar cerca a 0, ya se ha alcanzado el mínimo global. Esta situación se puede ver más claramente en la figura 2.6 En suma, aunque se intentaron algunas combinaciones de los dos calendarios, ya fuese sumando ambos, tomando uno o el otro antes o después de su intersección, se evidenció que el exponencial, con una temperatura inicial suficientemente alta, resulta convergiendo siempre a la energía mínima sin restringir demasiado la exploración, lo cual puede resultar bastante útil para 3.

2.4.2. Configuración óptima

Independientemente del calendario de enfriamiento, a partir de alguna constante C, se observó que habían varias configuraciones con la mínima energía encontrada, que fue de $-123.91169923373\,eV$. Se evidenció que todas eran reflexiones diagonales, verticales u horizontales entre sí, y se pueden 'construir' de acuerdo a la siguiente estructura:

1. Siguiendo la diagonal, al lado de un átomo 'esquinero' de tierras raras (en el cuadrado interno de átomos de R) deben haber dos átomos de titanio seguidos en las posiciones (a_1, a_1) y (a_2, a_2), siendo este último el más cercano a alguno de los límites de la red.

2. Hay cuatro átomos de Ti que tienen una coordenada igual a a_2 y la otra difiere en 2 o en 4 posiciones de a_2. Es decir, tanto horizontal como verticalmente, del átomo de Ti en (a_s, a_2) se 'extiende' una fila Fe-Ti-Fe-Ti de átomos.

3. Uno de los dos átomos de Ti que faltan por mencionar, A^*, se añade en alguna de las dos filas (la vertical o la horizontal), conservando, de nuevo, una de sus componentes igual a a_2 y la otra difiriendo por 2 del último átomo de la fila.

4. El último átomo de Ti se añade una posición antes o después (de tal manera que se mantenga en el mismo 'perímetro' del resto de los átomos, o si se prefiere, rodeando a las tierras raras) del anterior átomo mencionado sobre la misma diagonal que siguieron los átomos de Ti en (a_1, a_1) y (a_2, a_2).

Este procedimiento se ejemplifica y posiblemente se comprende mucho mejor en la figura 2.11.

Esto implica que, al haber cuatro pares de posiciones posibles para los dos átomos de Ti que se colocan en el primer paso, y dos filas en las que colocar otro átomo de Ti en el tercer paso, hay 8 configuraciones que tienen energía $-123.91169923373\,eV$.

Contraste con la parte 1

En configuraciones como la de la figura 2.12, aunque sigue habiendo un átomo de Ti en las 'esquinas' del perímetro de átomos de Fe que rodean a los de tierras raras, no se mantiene la tesis de la parte 1 (1.2) de que "el Titanio prefiere sitios alejados de las tierras raras" [1], pues la mayoría de los átomos de Ti deciden ubicarse a dos posiciones o en el segundo perímetro más alejado del 'cuadrado' de tierras raras en vez del más alejado, como inicialmente se pensaba al mostrar la figura 2.3 (b).

En suma, todas las parejas de átomos en el perímetro B dejan exactamente un átomo de Fe de por medio, salvo por las parejas que interactúan en diagonal (las del paso 1 y 4 en la construcción mencionada anteriormente).

El porqué físico de este hecho va más allá de nuestro conocimiento e intención. Sin embargo, explorando algunas configuraciones similares se encontró que la energía solo va aumentando: por ejemplo, en la figura 2.16 (a), se movieron los átomos de titanio que interactuaban entre sí diagonalmente y dio una energía de $-123.78103583819\,eV$; en la figura 2.16 (b), se formaron dos 'perímetros' alrededor del 'cuadrado' de tierras raras y dio una energía de $-123.79142286069\,eV$; y en la figura 2.16 (c) se intentó modificar la potencialmente óptima descrita en 2.3 (b), que tiene una energía de $-121.74765625321\,eV$, acercando algunos de los átomos de titanio a los de las esquinas del 'cuadrado' de tierras raras y dió una energía de $-121.67758837184\,eV$.

Inconformes con este resultado y algo curiosos, se decidió realizar algunas pocas simulaciones para ver qué ocurría a medida que la rejilla se hacía más grande, y lo que se obtuvo fue bastante interesante:

- En la figura 2.21 (a), se muestra la configuración obtenida para una red 12×12. Como se puede observar, ahora hay **más átomos de Ti aún más cerca de los de R**, aunque preservan el hecho de acumularse hacia alguna de las esquinas de la red, como si formaran una especie de 'barrera' contra los átomos de R para los átomos de Fe de una esquina.

- En la figura 2.21 (b), se ve el resultado del recocido para una red 14×14, que empieza a diferir bastante de los resultados en 10×10 y 12×12, pues ahora, los átomos de Titanio se agrupan en tres filas **diagonales**, asemejándose a una nave del juego *Space invaders* (referencia aquí) o a una pica de un juego de cartas, ambas rotadas 90 grados en el sentido de las manecillas.

- En la figura 2.21 (c), se observa que el patrón de la 'nave' o la 'pica' se mantiene, solo que esta vez, se encuentran rotadas 180 grados. Además, y notoriamente, los átomos de Ti no se acumulan en las esquinas de la red, sino que están directamente bajo los átomos de R.

- Siguiendo lo ocurrido en la (c), en la figura 2.21 (d), se observa de nuevo el patrón de la 'nave' o la 'pica', solo que esta vez alineada con ambos ejes positivos. Además, se preserva el comportamiento de 'libertad' de los átomos para decidir dónde acumularse.

Esto puede indicar que el titanio busca agruparse en tantas capas puedan, dejando algunos átomos de Fe entre ellos y, potencialmente, siguiendo figuras geométricas con alguna razón física por detrás.

2.4.3. Fases de exploración y explotación

Finalmente, mencionar que se pueden identificar dos fases (principalmente) en la convergencia de la energía: una de exploración y otra de explotación. En la primera, se evidencia una variación comparativamente grande de la energía, mientras que en la segunda, esta se estabiliza y se mantiene dentro de un rango o definitivamente no cambia salvo que encuentre un valor menor.

Previamente, ya se había mostrado cómo se veía la convergencia de energía para algunos calendarios de enfriamiento logarítmicos (2.4 y 2.5). Sin embargo, se modificó el método `plot_convergence` para que mostrara dos 'sombras', una azul y una naranja, que corresponden a las fases de explotación y exploración, respectivamente.

El criterio para mostrar dichas fases se basó en que tan pequeña era la distancia absoluta que había de la temperatura en cada iteración al 0 absoluto. Para el caso del esquema exponencial, se usó un umbral de 0.1, pues se evidenciaba que a partir de ese, la fase de explotación convergía en no muchas más iteraciones a una totalmente restrictiva que no permitía cambio de posiciones de los átomos de Ti salvo que minimizara la energía aún más. Esto para una constante $C = 10$, pues para constantes demasiado pequeñas (10^{-4} por ejemplo), no hay una fase de exploración sino solo de explotación, y lo opuesto ocurre si la constante es demasiado grande ($1,000$, por ejemplo). Se puede observar claramente en 2.22, al menos para la elección de dicha C, la fase de exploración dura aproximadamente 100 iteraciones y la de explotación el resto.

En contraste, para el caso logarítmico, a pesar de haber escogido un umbral de 0.05, aún más pequeño que en el caso exponencial, la fase de exploración no dura más de 25 iteraciones. Esto debido a que, por lo explicado en 2.4.1, la temperatura decrece muy rápidamente en las primeras iteraciones, pero muy lento después. Es por ello que no se ve una estabilización

de la energía en un rango o ausencia de movimiento alguno en la fase de explotación y es la razón de que la fase de exploración sea tan corta.

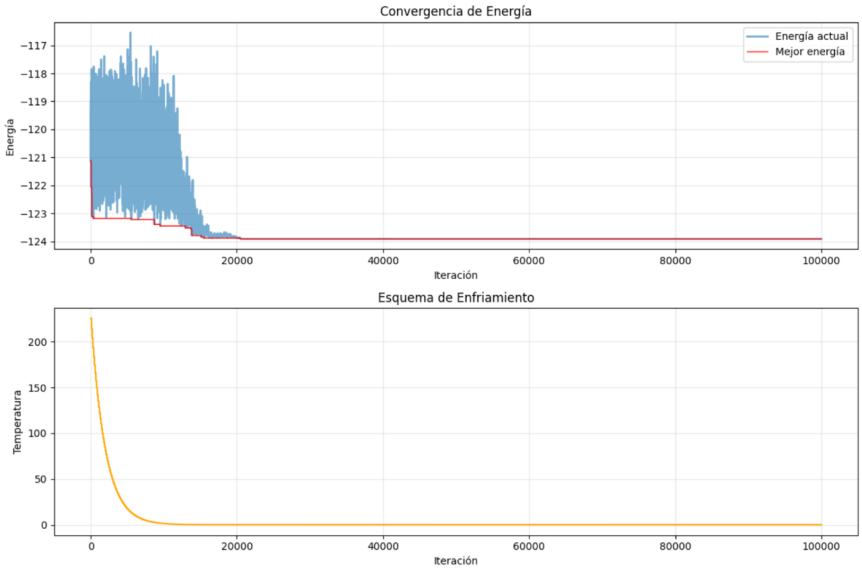

Figura 2.6: Energía de cada iteración a lo largo de $1,000$ pasos de ejecutar el algoritmo de recocido simulado usando el esquema de enfriamiento $250 \cdot 0.95^n$.

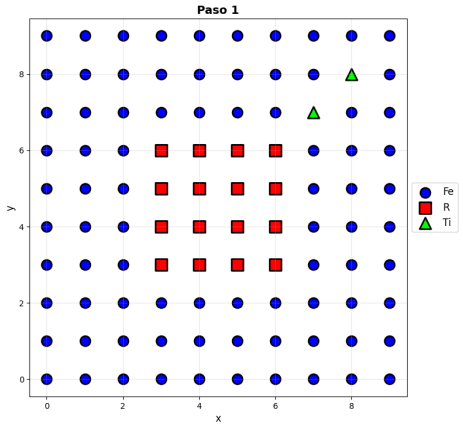

Figura 2.7: *
(a) Colocación de los primeros dos átomos en alguna de las 'esquinas' del 'cuadrado' de tierras raras.

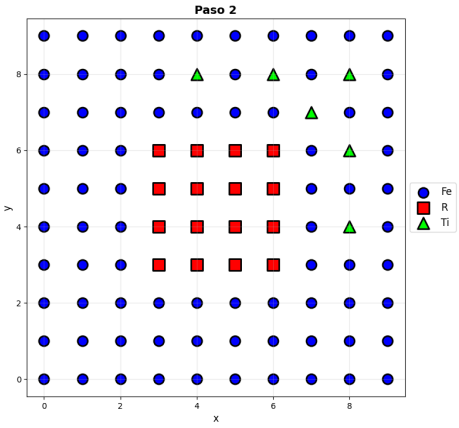

Figura 2.8: *
(b) Colocación de las 'filas' Fe-Ti-Fe-Ti vertical y horizontalmente.

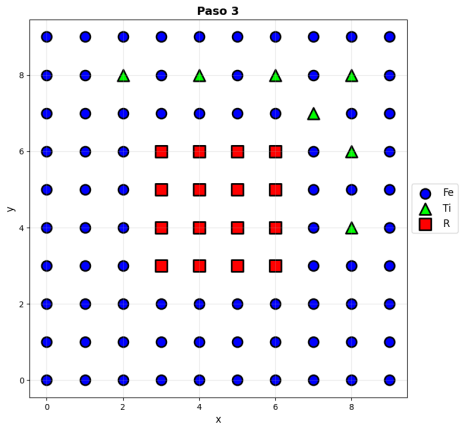

Figura 2.9: *
(c) Colocación del penúltimo átomo de Ti en la fila vertical u horizontal.

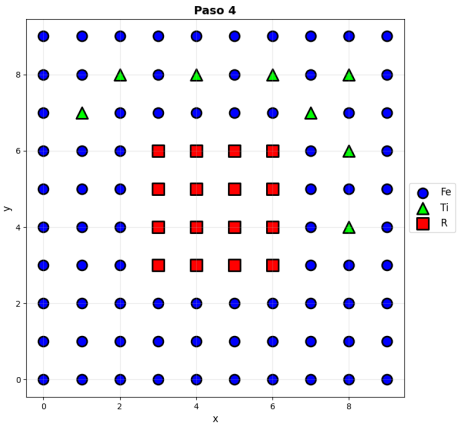

Figura 2.10: *
(d) Colocación del último átomo de Ti diagonalmente al lado del colocado en (c).

Figura 2.11: Ejemplificación de cómo se colocan los átomos de Ti para obtener una de las configuraciones que minimizan la energía.

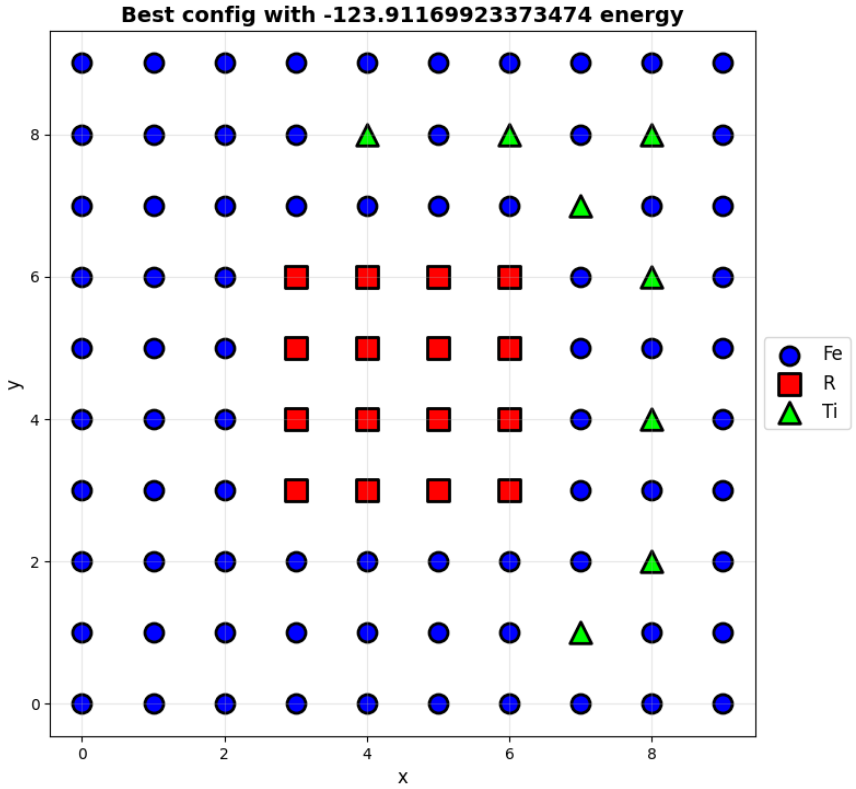

Figura 2.12: Una de las configuraciones con la mínima energía posible resultante de ejecutar el recocido con el calendario de enfriamiento $T(n) = 10 \cdot (.95)^n$ durante 200 iteraciones.

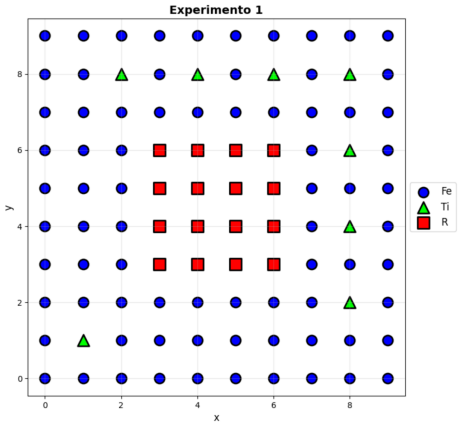

Figura 2.13: *
(a) Energía de
$-123.78103583819\,eV$.

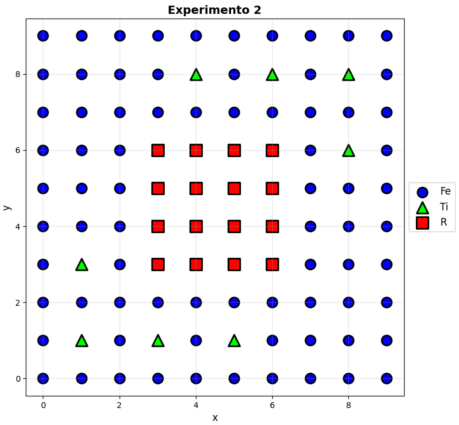

Figura 2.14: *
(b) Energía de
$-123.79142286069\,eV$.

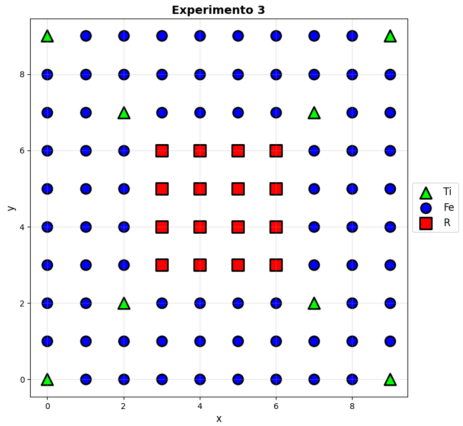

Figura 2.15: *
(c) Energía de
$-121.67758837184\,eV$

Figura 2.16: Otras configuraciones que, se pensaban, podían seguir una estructura similar a la planteada inicialmente o a la dada por el algoritmo y minimizar la energía.

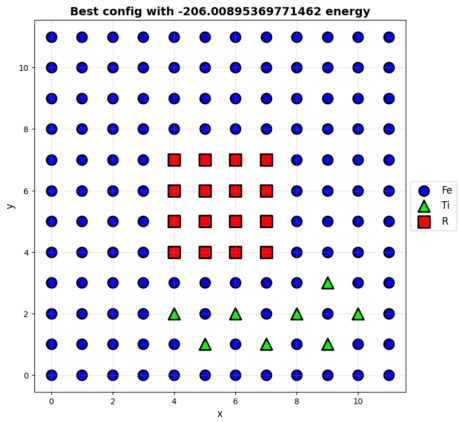

Figura 2.17: *
(a) Configuración que minimiza la energía de una red 12×12.

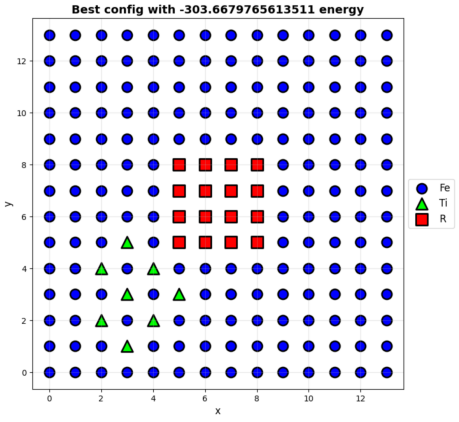

Figura 2.18: *
(b) Configuración que minimiza la energía de una red 14×14.

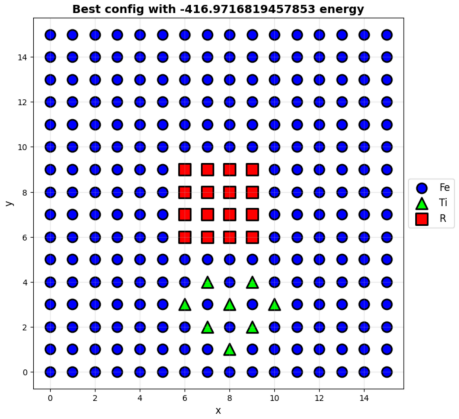

Figura 2.19: *
(c) Configuración que minimiza la energía de una red 16×16.

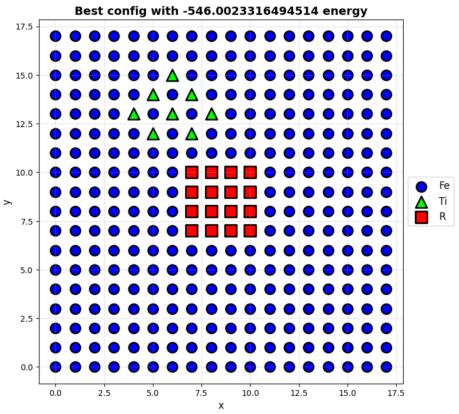

Figura 2.20: *
(d) Configuración que minimiza la energía de una red 18×18.

Figura 2.21: Algunas configuraciones óptimas para cuatro tamaños de rejillas diferentes, obtenidas ejecutando el recocido simulado con esquema de enfriamiento $T(n) = C \cdot (0.95)^n$ $(n \geq 2)$ durante 200 iteraciones (a) y (b) y 300 (c) y (d), recordando que $N_i = 100$.

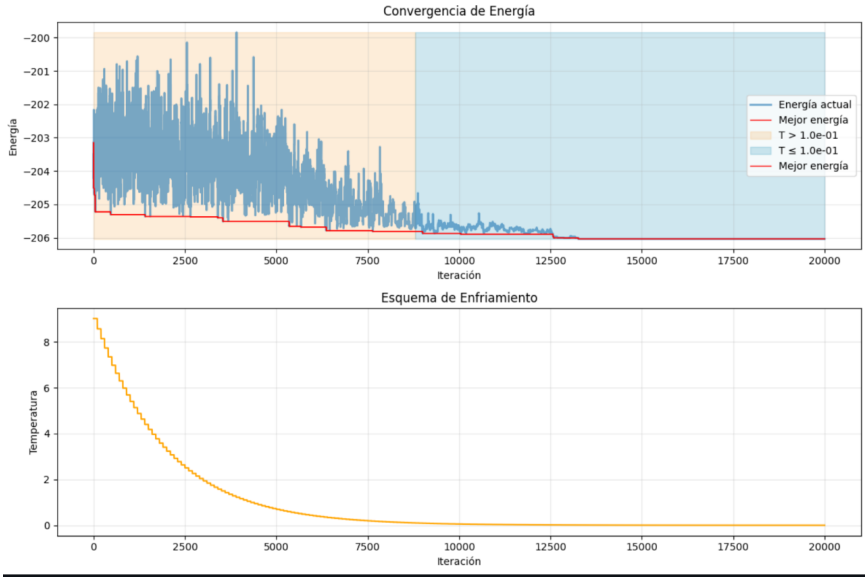

Figura 2.22: Convergencia de energía y temperatura de una ejecución del recocido simulado usando el calendario $T(n) = 10 \cdot (0.95)^n$ durante 200 iteraciones.

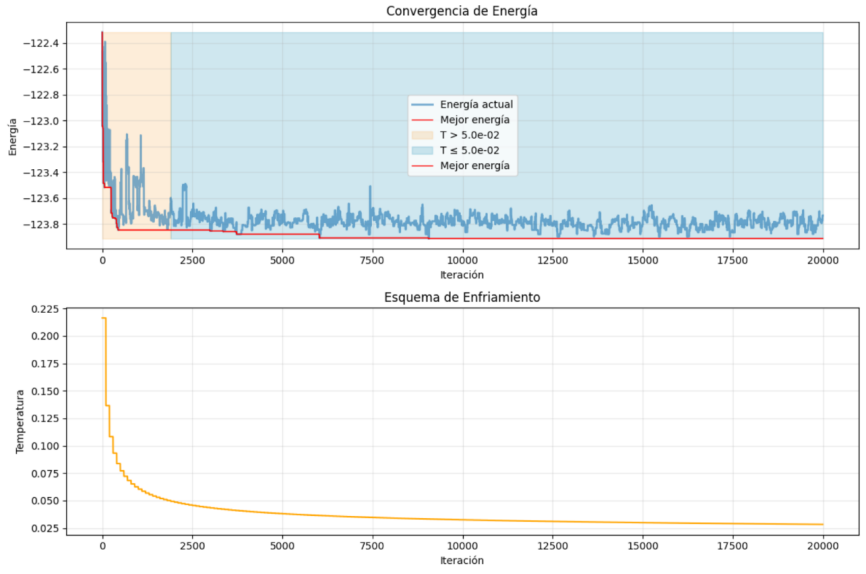

Figura 2.23: Convergencia de energía y temperatura de una ejecución del recocido simulado usando el calendario $T(n) = \frac{10}{log(n)}$ durante 200 iteraciones.

Capítulo 3

El desafío 3D de la tesis de Skelland

3.1. Descripción

Por último, vamos a resolver el problema tridimensional real de la tesis. A diferencia de los anteriores casos, acá ya no podemos reutilizar tanto las clases pasadas puesto que se tenía como supuesto que manejábamos una red reticular, por lo que obtener posiciones era más sencillo. En este caso se nos dan las coordenadas de los átomos de Fe y de Nd por lo que calcularemos las distancias directamente con estas posiciones con la distancia euclidiana en $\mathbb{R}^3$. Primero visualicemos cómo lucen los átomos en la estructura mencionada anteriormente.

Observamos que como se cambió el archivo que contenía las coordenadas de los átomos, ya no posee forma de supercelda.

3.2. Implementación 3D

Para adaptar el algoritmo utilizado previamente en el apartado 2, debemos redefinir la distancia calculada entre dos átomos, pues ahora trabajamos con coordenadas y no con puntos de un retículo en el plano y cómo vamos a almacenar las posiciones de los átomos.

3.2.1. Modelamiento de la supercelda

Utilizaremos 3 diccionarios, uno para almacenar las posiciones de los átomos de Fe, otro para los de Nd y el último para los de Ti, cuyas llaves son las coordenadas y los valores son los átomos posicionados en esa coordenada. Los átomos de Titanio se inicializaran aleatoriamente, escogiendo 8 posiciones de los átomos de Hierro dadas en el archivo *txt*. Cabe resaltar que hay posiciones duplicadas, por lo que simplemente obviaremos estas y

utilizaremos las 90 únicas. A la hora de llamar la función `swap_positions` se podrá hacer más eficiente el intercambio usando esta implementación

Para el cómputo de la energía se pudo reutilizar la forma optimizada del punto anterior ya que dio muy buenos resultados y no se necesitó cambiar demasiadas cosas, simplemente sustituir pasar como parámetro la posición al átomo.

Un ejemplo de una inicialización de 8 átomos de Titanio sería:

3.2.2. Recocido Simulado

Procediendo de manera análoga al punto anterior, la lógica descrita en la sección 2 se puede reutilizar, pues es la misma, solo cambian pequeños detalles de implementación específicos para trabajar en 3D.

La energía de la configuración inicial, sin átomos de Titanio es de -46.21 eV, por lo que esperamos minimizar aún más este valor. Siguiendo los esquemas de enfriamiento sugeridos en el apartado previo, es decir, tomando $c \times 0.95^n$ obtenemos los siguientes resultados:

En los tres casos vemos que converge a la energía de -55.981237 eV, por lo que es un buen candidato a ser esta la mínima.

Para finalizar, vemos el esquema de enfriamiento y la convergencia de energía con respecto al número de pasos. Concluimos, al igual que en el caso pasado, que usar este esquema de enfriamiento es bastante óptimo puesto que siempre resulta convergiendo a la energía mínima sin restringir demasiado la exploración y ya que el problema 2 era menos intenso computacionalmente que el 3, por lo que hacer pruebas era mucho más eficiente y al fin y al cabo terminó dando el resultado minimizante esperado. La toma de la sucesión $\{N\}_n$ como constante igual a 100 es clave para que el algoritmo no se demore cantidades de tiempo exorbitantes (se tardó alrededor de 30 minutos para todas las c) pero que permita hallar la energía mínima y que no se quede estancado en un mínimo local.

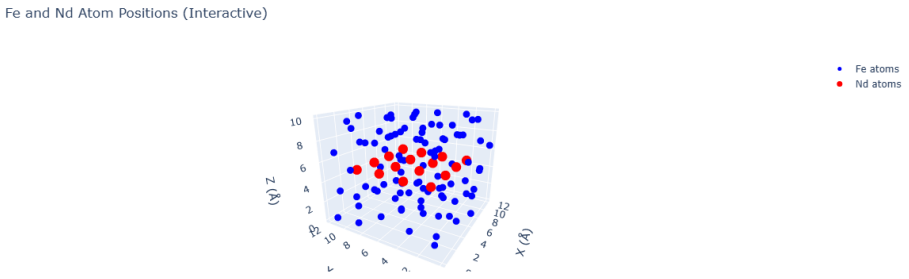

Figura 3.1: Visualización de átomos en la celda 3x3x2

```python
    def swap_ti_position(self, prev_atom: Atom, new_atom: Atom):
        """
        Si `prev_atom` y `new_atom` no son None, intercambia el tipo de átomo en dichas
        posiciones siempre y cuando:
        - En `prev_atom` y en `new_atom` hay un átomo movible, y en `prev_atom` hay un
        átomo de Ti.
        - Ambas componentes de de las posiciones son menores que la dimensión de un lado
        del retículo.
        Por otro lado, si `prev_atom` es None, añade `new_atom.position` a `ti_sites` si
        es una posición valida.
        """

        if prev_atom is None:
            pass

        elif prev_atom.atom_type != "Ti":
            print(f"At prev_pos there must be a Ti atom but there is a {prev_atom.atom_type}")

        else:
            del self.ti_sites[tuple(prev_atom.position)]
            del self.fe_sites[tuple(new_atom.position)]
            prev_atom.atom_type = "Fe"
            new_atom.atom_type = "Ti"
            self.ti_sites[tuple(new_atom.position)] = new_atom
            self.fe_sites[tuple(prev_atom.position)] = prev_atom

```

Cuadro 3.1: Valores promedio de la energía mínima encontrada tras correr 1000 iteraciones del recocido simulado para cada constante C usando el esquema exponencial.

C	Energía mínima promedio
250	-55.981237
100	-55.981237
10	-55.981237

```python
def calculate_total_energy(self, prev_atom: Atom, new_atom: Atom, current_energy: float)
-> float:
    if current_energy is None:
        total_energy = 0.0
        n_total_atoms = len(self.atoms)

        # Calcular la sumatoria sum_{i} sum_{j>i} U_{ij}(r_{ij})
        for i in range(n_total_atoms):
            for j in range(i + 1, n_total_atoms):
                atom_i: Atom = self.atoms[i]
                atom_j: Atom = self.atoms[j]

                distance = atom_i.distance_to(atom_j)
                params = self.morse_db.get_parameters( # obtener alpha, r0 y D0
                    atom_i.atom_type,
                    atom_j.atom_type
                )

                if params: # si son atomos de Fe, R o Ti
                    energy = params.calculate_energy(distance)
                    total_energy += energy

        return total_energy

    else:
        #*Si son la misma posición
        new_pos = new_atom.position
        prev_pos = prev_atom.position

        if np.array_equal(new_pos, prev_pos):
            return current_energy

        new_energy: float = current_energy

        def recalculate_energy(current_energy: float, atom1: Atom, atom2: Atom,
        operation: str = "-") ->float:
            if np.array_equal(atom1.position, atom2.position):
                return current_energy
            new_energy: float = current_energy
            for atom in self.atoms:
                distance_to_1: float = atom1.distance_to(atom)
                distance_to_2: float = atom2.distance_to(atom)
                params_1: MorseParameters = self.morse_db.get_parameters(
                    atom.atom_type,
                    atom1.atom_type
                )
                params_2: MorseParameters = self.morse_db.get_parameters(
                    atom.atom_type,
                    atom2.atom_type
                )

                if operation == "-":
                    if np.array_equal(atom.position, atom1.position) == False:
                        new_energy -= params_1.calculate_energy(distance_to_1)
                    if np.array_equal(atom.position, atom2.position) == False:
                        new_energy -= params_2.calculate_energy(distance_to_2)
                elif operation == "+"35
                    if np.array_equal(atom.position, atom1.position) == False:
                        new_energy += params_1.calculate_energy(distance_to_1)
                    if np.array_equal(atom.position, atom2.position) == False:
                        new_energy += params_2.calculate_energy(distance_to_2)

            return new_energy

        new_energy = recalculate_energy(new_energy, prev_atom, new_atom, "-")
```

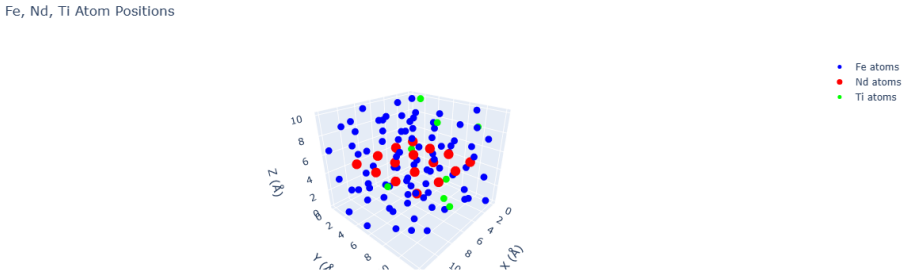

Figura 3.2: Visualización de átomos en la celda 3x3x2 con Titanio

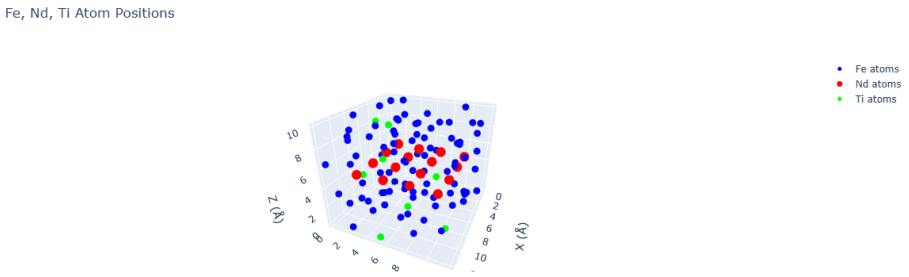

Figura 3.3: Posible configuración óptima de átomos de Titanio

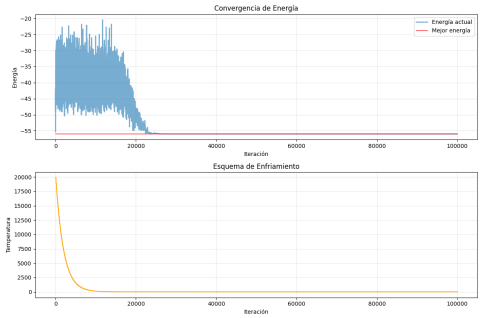

Figura 3.4: Energía de cada iteración a lo largo de $1,000$ pasos de ejecutar el algoritmo de recocido simulado usando el esquema de enfriamiento $250 \cdot 0.95^n$.

3.3. Comparación con los resultados de la Tesis

Lamentablemente, al actualizar las posiciones de los átomos en la segunda versión, ya no se trata de una supercelda y por ende no podemos apreciar las 3 "placas" que había antes para proyectarlas en el plano XY y verlo en 2D como en la Tesis. Es por eso que debemos visualizarlo como en la figura 3.4. Sin embargo, en el colab, podemos apreciar que rotando la estructura 3D se ven resultados similares a los de la Tesis: los átomos de Titanio se alejan entre sí y de los átomos de Nd. Esto se justifica más fuertemente haciendo el análisis cuantitativo propuesto en el documento. Lo implementamos de esta manera:

Con este código calculamos las distancias medias entre los (8) átomos de Titanio y los átomos de Hierro y Neodimio. La salida del código fue:

Como la distancia de equilibrio es menor a estos valores, concluimos que en efecto los átomos de Titanio se deben estar alejando entre sí y entre los átomos de Nd. No se pidió explícitamente en la tarea computar la interacción de Ti-Fe pero al calcular la distancia media vemos que soporta los resultados aún más, puesto que es la menor de las interacciones.

Por último, decidimos hacer una prueba haciendo una configuración "lógica", en el sentido de que si queremos maximizar la distancia entre los átomos de Titanio con otros de Titanio y con otros de Neodimio, si escogemos las 8 "esquinas", es decir, los puntos más alejados del centro de la región creada por los puntos, de las posiciones de los átomos, como se puede ver en la siguiente figura, obtenemos una energía de -44.7, valor alejado del hallado usando el recocido simulado.

Concluimos, así, que los resultados de la Tesis parecen verse reflejados en este trabajo, que si bien no tiene las mismas condiciones que las de la investigación hecha, nos permiten hallar soluciones que dependen de métodos estocásticos, que en este caso fueron bastante eficientes y satisfactorios.

```python
# Compute the average distance between Ti and the closest Nd atoms
distances = []
for ti_site in best_ti_sites.values():
    mind = np.inf
    for nd_site in supercell.nd_sites:
        dist = np.linalg.norm(ti_site.pos_arr - nd_site)
        if dist <= mind:
            mind = dist
    distances.append(mind)
average_distance = np.mean(distances)
print(distances)
print(f"Average distance between Ti and closest Nd atoms: {average_distance:.4f} Å")

# Compute the average distance between Ti and the closest Ti atoms
distances = []
for ti_site in best_ti_sites.values():
    mind = np.inf
    for other_ti_site in best_ti_sites.values():
        if not np.array_equal(ti_site.pos_arr, other_ti_site.pos_arr):
            dist = np.linalg.norm(ti_site.pos_arr - other_ti_site.pos_arr)
            if dist <= mind:
                mind = dist
    distances.append(mind)
average_distance = np.mean(distances)
print(distances)
print(f"Average distance between Ti and closest Ti atoms: {average_distance:.4f} Å")

# Compute the average distance between Ti and the closest Fe atoms
distances = []
for ti_site in best_ti_sites.values():fig:
    mind = np.inf
    for fe_site in supercell.fe_sites.values():
        dist = np.linalg.norm(ti_site.pos_arr - fe_site.pos_arr)
        if dist <= mind:
            mind = dist
    distances.append(mind)
average_distance = np.mean(distances)
print(distances)
print(f"Average distance between Ti and closest Fe atoms: {average_distance:.4f} Å")

```

Cuadro 3.2: Distancias promedio entre átomos de Titanio y Neodimio o Titanio o Hierro

Interacción	Distancia promedio (Å)
Ti-Ti	5.0571
Ti-Nd	4.2571
Ti-Fe	2.6100

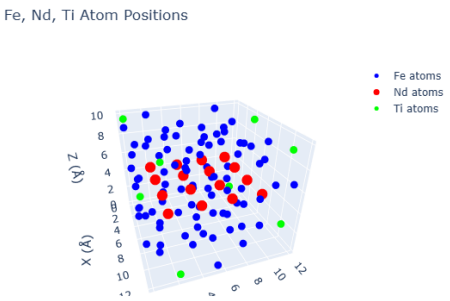

Figura 3.5: Configuración pensada a ser la minimizante de energía.

Capítulo 4

Bibliografía y referencias

1. Skelland C. *Impact of Chemical and Morphological Changes on the Phase Stability of Magnetic Materials*. Tesis Doctoral, Universidad de Exeter, 2021. link