**Proyecto Parte 1**

**Taller # 2**

1. **Manejo memoria**

Lo primero **que deben hacer** es ilustrarse sobre el problema en el que trabajaremos: sus generalidades, conceptos, estructuras etc. Esa información la encuentran en:

**Objetivos del Taller**

1. Comprender los desafíos de manejar grandes espacios de estados en memoria RAM.
2. Explorar técnicas para reducir el uso de memoria en problemas computacionales intensivos.
3. Implementar estrategias prácticas para optimizar el manejo de datos en sistemas con alta demanda de memoria.

**1.1 Introducción**

**Conceptos clave:**

* + Explicación del crecimiento exponencial del espacio de estados y su impacto en el uso de memoria.
  + Ejemplo práctico: Problemas de partición de mínima información (MIP) y su relación con la explosión combinatoria.
  + Limitaciones de la memoria RAM en sistemas convencionales.

**Discusión inicial:**

* + ¿Qué estrategias conocen para manejar grandes volúmenes de datos en memoria?
  + Introducción a la idea de optimización de memoria mediante técnicas como:
    - Representación compacta de datos.
    - Uso de estructuras de datos eficientes.
    - Procesamiento en paralelo y en bloques.

**1.2 Estrategias de Manejo de Memoria**

**Técnicas para optimizar el uso de memoria:**

* 1. **Representación compacta de datos:** Uso de estructuras de datos como matrices dispersas (sparse matrices) para representar grandes espacios de estados.

Ejemplo: *Representar solo los estados alcanzables en lugar de todo el espacio de estados.*

* 1. **Procesamiento en bloques:** Dividir el espacio de estados en bloques más pequeños que puedan procesarse de manera independiente.

Ejemplo: *Procesar subconjuntos de estados en iteraciones.*

* 1. **Paralelización:**
* Uso de herramientas como multiprocessing en Python para dividir el trabajo entre múltiples núcleos.
* Introducción a CUDA para procesamiento en GPU.
  1. **Técnicas de reducción de memoria:** Uso de algoritmos heurísticos para evitar la exploración completa del espacio de estados.

Ejemplo: *Algoritmos de Monte Carlo para muestreo de estados.*

**Análisis de complejidad:** Comparación de costos computacionales y de memoria entre diferentes enfoques.

* Ejemplo práctico: Comparar el uso de memoria entre una matriz completa y una matriz dispersa.

**1.3 Implementación de Estrategias**

**Ejercicio 1: Representación compacta de datos**

* Implementar una matriz dispersa para representar un espacio de estados de .
* Usar la biblioteca scipy.sparse, scipy.linalg, scipy.optimize en Python para manejar grandes matrices.

**Ejercicio 2: Procesamiento en bloques**

* Dividir un espacio de estados en bloques y procesarlos de manera iterativa.
* Implementar un algoritmo que procese solo los bloques necesarios.

**Ejercicio 3: Paralelización**

* Usar multiprocessing para dividir el procesamiento de estados entre múltiples núcleos.
* Implementar un ejemplo básico de paralelización con Python.

**Ejercicio 4: Reducción de memoria**

* Estrategias como el algoritmo de Monte Carlo pueden utilizarse para muestrear estados en lugar de procesar todo el espacio.
* Comparar el uso de memoria y tiempo de ejecución con un enfoque exhaustivo.

**1.4 Discusión y conclusiones**

**Reflexión grupal:**

* ¿Qué técnicas fueron más efectivas para reducir el uso de memoria?
* ¿Qué desafíos encontraron al implementar las estrategias?
* Discusión sobre el uso de hardware especializado (GPU, clústeres de computación).

**Materiales que pueden ser utilizados**

1. Lenguaje Python.
2. Algunas bibliotecas que pueden consultar:
   * CPU:
   * Numpy
   * Scipy
   * multiprocessing
   * (opcional, para quienes tengan acceso a GPU).
   * Pycuda
   * Numba
   * CuPy
   * Matmul
   * JAX
3. Acceso a ejemplos de datos o simulaciones para trabajar durante el taller.
4. **Estrategia Q-Nodes**

En este punto deben abordar el estudio de la estrategia Q-Nodes, entender el uso de las estructuras de datos, y la lógica de la aplicación empleando la documentación que se anexa que contiene la explicación del proceso, arquitectura del aplicativo y diagramas del mismo, así como todo el software que está en el repositorio github del curso.

Luego de revisado el material y comprendida la estrategia, desarrollar lo siguiente:

1. Cuando se haya el par candidato, se genera una partición candidata que luego se evalúa al final de todo el proceso para escoger la que genera la menor pérdida. Lo que ustedes deben hacer es que inmediatamente se genera un par candidato (y por ende una partición candidata) evaluar la pérdida de esa partición candidata, y si esa pérdida es CERO, terminar el proceso y reportar esa como la partición respuesta que genera la menor perdida.
2. En los procesos de iteración, hay cálculos que se repiten una y otra vez. La idea es que estos cálculos sólo se calculen una vez y se puedan reutilizar.
3. Genere nuevamente las pruebas desarrolladas en el Taller #1 y compárelas con los resultados obtenidos en valores de pérdida y tiempo, con estas modificaciones a la estrategia Q-Nodes.