UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Centro de Ciencias Básicas

Departamento: Ciencias de la Computación

Área Académica: Inteligencia Artificial

Metaheurísticas I

Ingeniería en Computación Inteligente

5° “A”

“Examen Parcial I: Optimización por Enjambre de Partículas (PSO) para Simular la Planificación de Carga de 100,000 Criticas de Cine”

Alumnos:

Cesar Eduardo Elías del Hoyo

José Luis Sandoval Pérez

Diego Emanuel Saucedo Ortega

Carlos Daniel Torres Macías

Profesor: Francisco Javier Luna Rosas

Aguascalientes, Ags. Septiembre, 22 de 2024

Optimización por Enjambre de Partículas (PSO) para Simular la Planificación de Carga de 100,000 Criticas de Cine

Introducción

El problema que se aborda en este proyecto es la **asignación de carga de trabajo** para el procesamiento de 100,000 reseñas de cine mediante un sistema de procesamiento **paralelo**. Dado que el procesamiento secuencial sería ineficiente y consumiría demasiado tiempo, se ha optado por un enfoque de procesamiento paralelo, donde las reseñas se distribuyen entre **8 procesadores homogéneos**. El objetivo principal es minimizar el **tiempo total de procesamiento** mediante la **Optimización por Enjambre de Partículas (PSO)**, un algoritmo metaheurístico ampliamente utilizado para resolver problemas de optimización de alta dimensionalidad.

Análisis del Algoritmo PSO

El **PSO** es un algoritmo bioinspirado que imita el comportamiento colectivo de los enjambres. En este caso, cada **partícula** representa una **posible solución** al problema de distribución de carga entre los procesadores. Estas partículas exploran el espacio de soluciones a través de iteraciones, ajustando su **posición** (solución actual) y **velocidad** (cambio en la solución) con base en su propia experiencia y la experiencia del grupo.

El proceso de PSO implica los siguientes pasos fundamentales:

1. **Inicialización**:
    - Se generan partículas con posiciones aleatorias que representan posibles asignaciones de tareas.
    - Cada partícula tiene una velocidad inicial aleatoria.
    - Se definen coeficientes para controlar la inercia, el aprendizaje individual y el aprendizaje grupal
2. **Evaluación de la función objetivo**:
    - La función objetivo mide el tiempo total de procesamiento para una asignación dada de reseñas. El objetivo es minimizar el tiempo al distribuir de manera equilibrada la carga entre los procesadores.
3. **Actualización de partículas**:
    - En cada iteración, la velocidad de las partículas se actualiza en función de:
        - **La mejor posición personal** que una partícula ha encontrado.
        - **La mejor posición global** encontrada por todo el enjambre.
    - La nueva posición de cada partícula se calcula sumando su velocidad actual a su posición anterior.
4. **Criterio de convergencia**:
    - El algoritmo termina cuando se alcanza el número máximo de iteraciones o las partículas convergen hacia una solución.

Implementación del Algoritmo PSO para la Planificación de Carga de Críticas de Cine

El propósito de este proyecto es implementar el Algoritmo de Optimización por Enjambre de Partículas (PSO) para simular la planificación de carga de un conjunto de 100,000 críticas de cine en un sistema de procesamiento paralelo. La implementación se basa en los principios presentados en el artículo "Observations on Using Genetic Algorithms for Dynamic Load-Balancing" y tiene como objetivo realizar una comparación entre el tiempo secuencial y el tiempo paralelo al procesar las críticas en 8 procesadores homogéneos. Se incluye una tabla y una gráfica que muestran el rendimiento con diferentes cantidades de críticas.

Contexto del Problema

El procesamiento de críticas de cine es una tarea computacionalmente costosa, donde cada crítica requiere 1 segundo de procesamiento por procesador. Realizar este procesamiento de forma secuencial implicaría que un único procesador gestione todas las críticas, lo que resulta ineficiente para grandes volúmenes de datos. Por esta razón, se propone una planificación de carga en paralelo utilizando 8 procesadores homogéneos, donde el PSO optimiza la distribución de las críticas para minimizar el tiempo total de procesamiento.

El desafío principal consiste en simular la planificación para diferentes cantidades de críticas (desde 10,000 hasta 100,000, con incrementos de 10,000 críticas), evaluando el tiempo de procesamiento secuencial versus el paralelo.

Implementación del Algoritmo

1. Inicialización de Parámetros

El PSO comienza con la definición de un conjunto de 30 partículas, cada una de las cuales representa una posible manera de distribuir las críticas de cine entre los procesadores. Las partículas son inicializadas de manera aleatoria, y cada una tiene una posición (que define la distribución de las críticas) y una velocidad (que controla los cambios en la posición de las partículas a lo largo de las iteraciones).

```python
# Inicializar parámetros
num_particles = 30
num_dimensions = 1  # Cambiado a 1 dimensión
max_iterations = 100
w = 0.5  # Inercia
c1 = 1.5  # Coeficiente cognitivo
c2 = 1.5  # Coeficiente social

NUM_PROCESADORES = 8
```

Python

2. Función de Evaluación (Fitness)

La función de fitness evalúa la calidad de cada asignación de carga calculando el tiempo total de procesamiento para cada distribución propuesta. En el escenario secuencial, un único procesador realiza el procesamiento de todas las críticas, lo que lleva un tiempo proporcional al número de críticas. En el escenario paralelo, las críticas se distribuyen entre los 8 procesadores, y el tiempo de procesamiento depende de la máxima carga asignada a un solo procesador. El objetivo del PSO es minimizar esta máxima carga para reducir el tiempo total.

```python
def porcentaje_balanceados(particula):
    promedio_procesadores = np.mean(particula)
    th_H = 1.2 * promedio_procesadores
    th_L = 0.8 * promedio_procesadores

    balanceado = 0
    for x in particula:
        if x>th_H and x<th_L:
            balanceado += 1
    return balanceado/NUM_PROCESADORES

def objective_function(particula):
    # tomamos el maximo de toda la particula
    maxspan = max(particula)
    # calculamos el promedio de uso de procesadores
    promedio_uso = np.mean(particula/maxspan)
    # determinamos el limite de sobrecarga de procesador
    porcentaje_carga = porcentaje_balanceados(particula)

    return (1/maxspan) + promedio_uso + porcentaje_carga
```

Python

3. Actualización de las Partículas

Durante cada iteración, las partículas actualizan su posición y velocidad en función de:

- La mejor solución individual encontrada por la partícula hasta el momento (mejor posición personal).

- La mejor solución global encontrada por todo el enjambre (mejor posición global).

La actualización permite que las partículas exploren el espacio de soluciones en busca de una distribución más equitativa de las críticas entre los procesadores. A medida que las iteraciones avanzan, las partículas convergen hacia una solución óptima o cercana a lo óptimo.

```
# PSO loop
for iteration in range(max_iterations):
    for i in range(num_particles):
        # Actualizar velocidad
        r1, r2 = np.random.rand(2)
        # actualizamos cada conjunto de velocidades
        for index in range(NUM_PROCESADORES):
            velocities[i][index] = (w * velocities[i][index] +
                                    c1 * r1 * (personal_best_positions[i][index] - particles[i][index]) +
                                    c2 * r2 * (global_best_position[index] - particles[i][index])).astype(int)

            # Actualizar posición
            particles[i][index] += velocities[i][index]

        # ajustar el que sea el total de resenas
        resto_resenas_particula = num_resenas - np.sum(particles[i])

        if resto_resenas_particula > 0:
            indice_max = np.argmax(particles[i])
            particles[i][indice_max] += resto_resenas_particula
        elif resto_resenas_particula < 0:
            indice_min = np.argmin(particles[i])
            particles[i][indice_min] += resto_resenas_particula

        # Evaluar nueva posición
        score = objective_function(particles[i])

        # Actualizar mejor posición personal
        if score < personal_best_scores[i]:
            personal_best_scores[i] = score
            personal_best_positions[i] = particles[i]

    # Actualizar mejor posición global
    global_best_position = personal_best_positions[np.argmax(personal_best_scores)]
```

4. Simulación del Tiempo Secuencial vs. Paralelo

La planificación simula tanto el procesamiento secuencial como el paralelo para diferentes cantidades de críticas. Para el procesamiento secuencial, el tiempo total es directamente proporcional al número de críticas. Para el procesamiento paralelo,

el tiempo total se calcula dividiendo las críticas entre los 8 procesadores y midiendo el tiempo de procesamiento del procesador con la mayor carga.

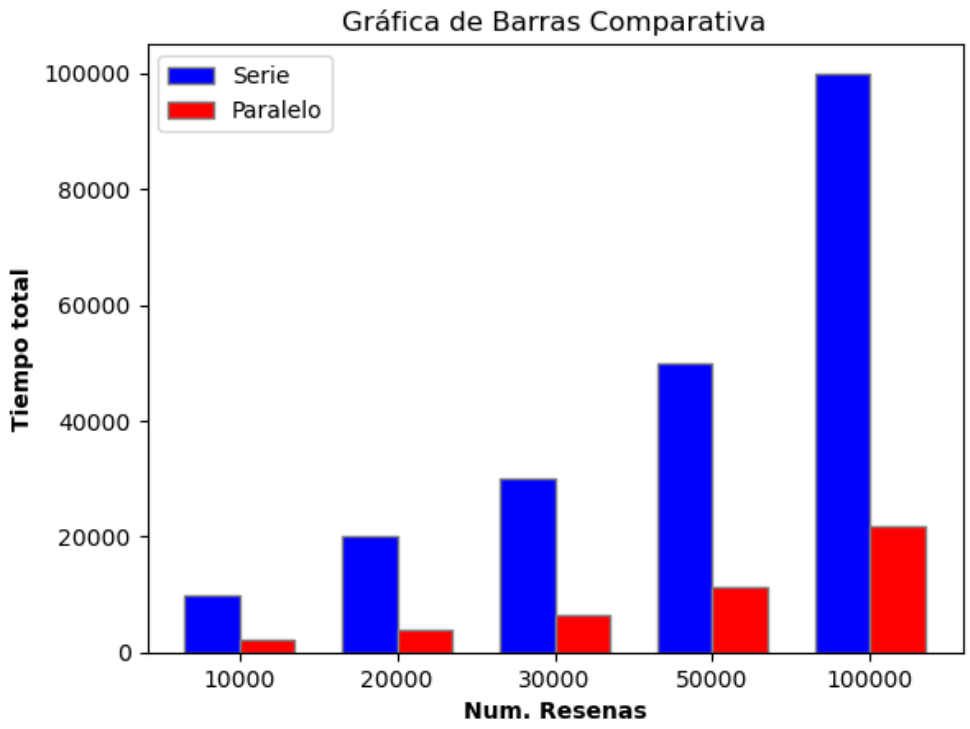
Se realiza la simulación para los siguientes valores de críticas: 10,000, 20,000, 30,000, 40,000, 50,000, 60,000, 70,000, 80,000, 90,000 y 100,000 críticas. En cada caso, se mide el tiempo de procesamiento tanto en el escenario secuencial como en el paralelo.

	Num.Resenas	P.Serie	P.Paralelo
0	10000	10000	2086
1	20000	20000	3947
2	30000	30000	6492
3	50000	50000	11346
4	100000	100000	21703

Comparación y Visualización

La implementación incluye una gráfica que comparan los tiempos de procesamiento secuencial y paralelo para las diferentes cantidades de críticas. Los resultados muestran una mejora significativa en el tiempo de procesamiento cuando se utiliza el enfoque paralelo con PSO, especialmente para volúmenes grandes de críticas.

- Gráfica: Visualiza la diferencia de tiempo entre el procesamiento secuencial y paralelo, mostrando cómo la distribución de carga optimizada por el PSO reduce el tiempo de ejecución conforme aumenta el número de críticas.



Conclusión

El uso del PSO para la planificación de carga en el procesamiento de reseñas de cine en un entorno paralelo con 8 procesadores homogéneos demuestra ser altamente efectivo. El algoritmo optimiza la distribución de trabajo, logrando una reducción significativa del tiempo de procesamiento en comparación con el procesamiento secuencial. La simulación para diferentes cantidades de críticas (de 10,000 a 100,000) revela que el enfoque paralelo es especialmente ventajoso para grandes volúmenes de datos, lo que hace que el PSO sea una herramienta poderosa para la optimización de tareas en sistemas distribuidos.

Referencias

El-Ghazali Talbi. (2009). METAHEURISTICS FROM DESIGN TO IMPLEMENTATION. John Wiley & Sons. https://www.researchgate.net/profile/Sura-Abdullah-5/post/What_is_the_proper_balance_between_Exploration_and_Selection_in_metaheuristics/attachment/5ff75b60e35e2b000103c944/AS%3A977419192246274%401610046303870/download/Metaheuristics.pdf

Tam, A. (2021, October 11). A gentle introduction to particle swarm optimization. MachineLearningMastery.com. https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/