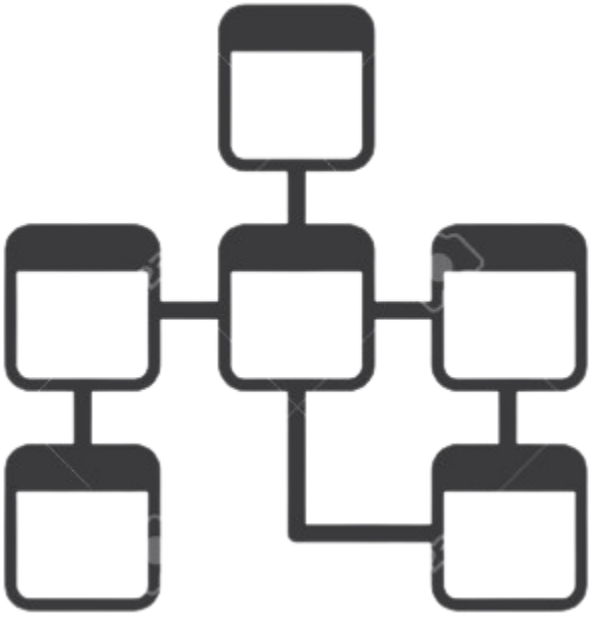
Instituto Tecnológico de Culiacán

Ingeniería en Tecnologías de la Información y Comunicaciones



Materia: Tópicos de IA

Problemas Clásicos de Optimización Combinatoria

Maestro: Zuriel Dathan Mora Félix

Alumno: Daniel Alejandro De Los Santos Cuevas

1. **Problema de Programación de Trabajos (JSSP - Job Shop Scheduling Problem)**

Descripción:

El problema consiste en programar las actividades de un conjunto de recursos (H) encargados de ejecutar una serie de tareas (I) independientes entre sí, asegurando que cada una se complete antes de su fecha límite ($\mathrm{D_i}$). Para llevar a cabo cada tarea, es necesario que varios recursos participen simultáneamente, siempre que posean las habilidades requeridas. Además, la cantidad de tareas que pueden asignarse simultáneamente a un mismo recurso está restringida por su capacidad ($\mathrm{M_{hk}}$).

Variables:

I: Índice de cada tarea. *i* = 1,..., *N* donde *N* es el número total de tareas.

I: Conjunto de tareas, *I* = {i: 1,..., *N*}.

K: Periodo de tiempo, *k* = 1,..., *K* donde *K* es el horizonte de programación.

h: Índice de cada máquina, *h* = 1,..., *M* donde *M* es el número total de máquinas.

H: Conjunto de máquinas, *H* = {*h*: 1,..., *H*}.

d_i: Tiempo proceso de la tarea *i*.

D_i: Fecha de entrega comprometida para la tarea *i*.

b_i: Fecha inicio programada de la tarea *i*.

c_i: Fecha finalización programada de la tarea *i*.

T_i: Retraso de la tarea *i*.

Υ_{ih} $\begin{cases} 1 \text{ si es necesario utilizar el recurso } h \\ 0 \text{ si no necesita usar el recurso } h \end{cases}$

Υ_i Vector que define las necesidades de utilización de recursos que tiene la operación *i*.

δ_{ihk} $\begin{cases} 1 \text{ si la tarea } i \text{ se programa en el recusro } h \text{ en instante } k \\ 0 \text{ en caso contrario} \end{cases}$

M_{hk} Capacidad de la máquina *h* en el instante *k*.

Restricciones:

El problema está sujeto a restricciones de capacidad de los recursos, ya que su disponibilidad es limitada. También enfrenta restricciones de simultaneidad, dado que los recursos requeridos para cada tarea deben asignarse en los mismos periodos de tiempo. Además, existen restricciones en la relación entre variables, las cuales contribuyen a definir completamente el problema.

Función Objetivo:

Los objetivos de la programación de la producción buscan evaluar la calidad de una solución al problema planteado. Para ello, se establecen criterios que permiten identificar soluciones óptimas o satisfactorias. Los criterios más comunes incluyen la minimización del tiempo total de fabricación, la reducción de los periodos de inactividad de las máquinas, la optimización de los costos asociados a la programación (como los derivados del almacenamiento de productos en proceso) y el cumplimiento de las fechas de finalización de las tareas. En algunos casos específicos, pueden emplearse criterios más detallados, como la reducción del impacto de los cuellos de botella generados por ciertos recursos. En este último caso, la función a minimizar corresponde a la suma ponderada de los cuadrados del retraso de cada tarea.

2. Problema de las N Reinas (N-Queens Problem)

Descripción:

El problema de las *n*-reinas consiste en ubicar *n* reinas en un tablero de ajedrez de *n* × *n* de manera que ninguna de ellas pueda atacarse entre sí. Esto implica que no deben coincidir en la misma fila, columna o diagonal. Se produce una colisión cuando dos reinas pueden capturarse mutuamente. El objetivo es encontrar una disposición en la que se seleccionen *n* celdas para colocar las reinas, minimizando el número total de colisiones.

Variables:

x_i representa la columna donde se ubica la reina en la fila *i*, es decir, $x_i \in$ {1, 2,…, *N*}.

Restricciones:

1. **Cada reina en una fila distinta:** El índice de la fila está determinado, por lo que no hay restricciones en filas.
2. **Cada reina en una columna distinta:** Todas las variables deben ser diferentes:
$$x_i \neq x_j \qquad \forall i \neq j$$
3. **Evitar ataques en diagonal:** Una reina en la posición (i, x_i) no debe compartir la misma diagonal con otra reina $((j, x_i)$, lo que implica:
$$|x_i - x_j| \neq |i - j|, \qquad \forall i \neq j$$

Representación y soluciones:

En la imagen 2.1 se muestra una matriz 4x4 en la cual existen 4 colisiones: {(1, 2), (3, 4), (1, 3), (2, 4)}

	Columna1	Columna2	Columna3	Columna4
Reina1			♕1	
Reina2				♕2
Reina3	♕3			
Reina4		♕4		

Imagen 2.1

Este problema puede resolverse mediante diferentes enfoques, como la enumeración exhaustiva de todas las posibles configuraciones y la evaluación de colisiones. En este caso, sería necesario analizar factorial de *n* posibles soluciones.

Para este caso, existen un total de 2 soluciones, mostrados en la imagen 2.2 y 2.3

Reina	R1	R2	R3	R4
Ubicación en la columna	2	4	1	3

Imagen 2.2

Reina	R1	R2	R3	R4
Ubicación en la columna	3	1	4	2

Imagen 2.3

3. Árbol de Expansión Mínima (MST - Minimum Spanning Tree)

Descripción:

El Árbol de Expansión Mínima (MST) es un subconjunto de un grafo no dirigido y conectado que une todos sus vértices con el menor peso total posible en sus aristas. En otras palabras, es una estructura en forma de árbol que abarca todos los nodos del grafo minimizando la suma de los pesos de las conexiones.

Algunas de sus características principales son:

- Contiene exactamente una arista menos que el número total de vértices del grafo.
- No presenta ciclos.

- La adición de una arista genera un ciclo, mientras que la eliminación de una rompe la conexión del árbol.
- Puede haber múltiples árboles de expansión mínima para un mismo grafo, pero todos tienen el mismo peso total.

En la imagen 3.1 se presenta un grafo con todas las posibles conexiones entre los nodos. Mediante la solución que presenta un algoritmo para este tipo de problemas que lleva por nombre “Kruskal” se puede determinar cuál es el recorrido de menor valor o de menor peso. Se empieza seleccionando la arista con menor valor, una vez marcada todas las aristas con el menor valor, se selecciona la nueva arista con menor valor y así sucesivamente. La excepción es no generar un ciclo o bucle.

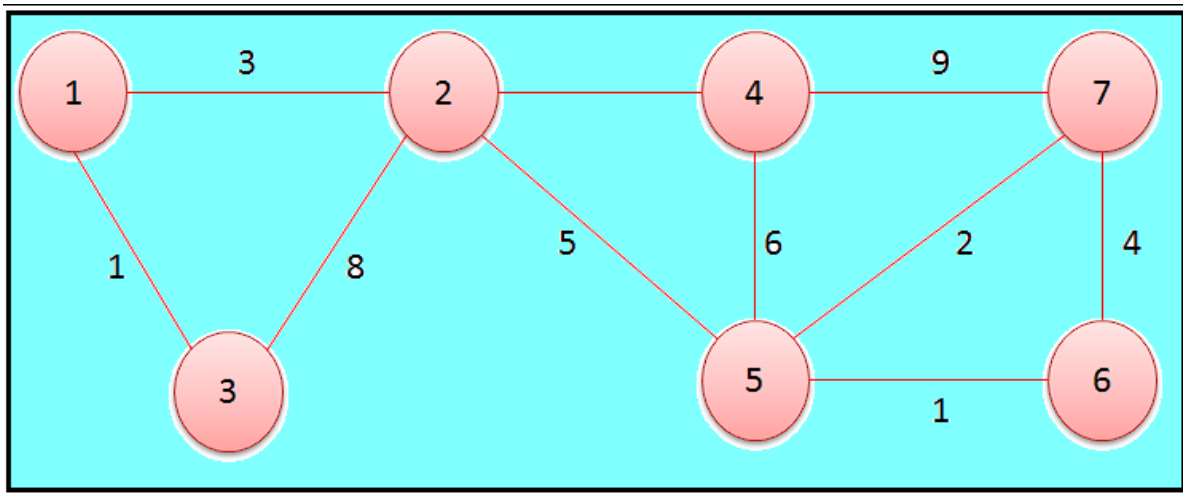


Imagen 3.1

En la imagen 3.2 se muestra la solución, siendo ese el camino que conecta todos los nodos con el valor mínimo o el menor peso.

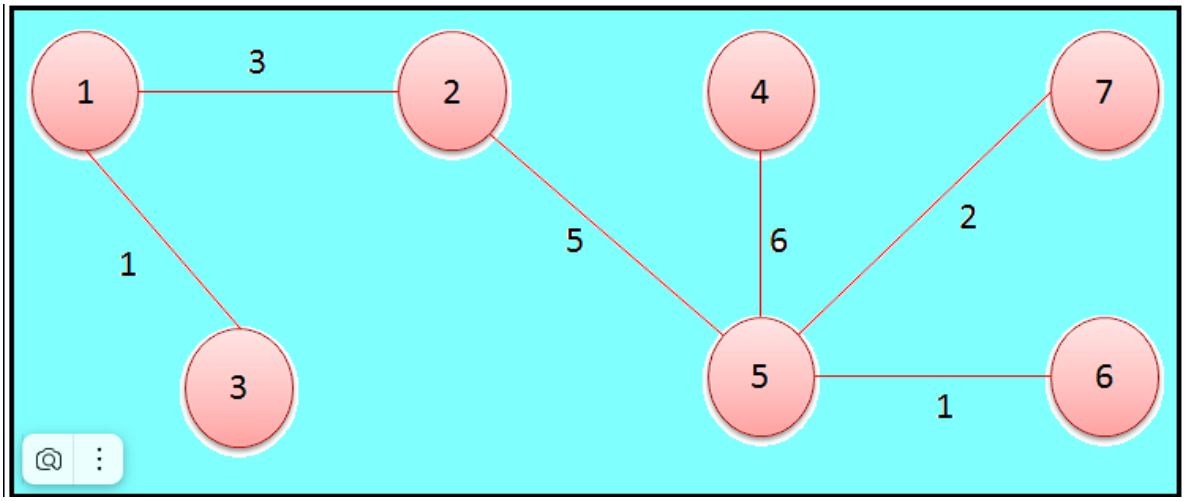


Imagen 3.2

4. Problema del Agente Viajero (TSP - Traveling Salesman Problem)

Descripción:

El TSP es uno de los desafíos más conocidos y complejos en el campo de la computación. Ha sido estudiado por diversas ramas de la ingeniería debido a sus múltiples aplicaciones, principalmente en la optimización de rutas. Su utilidad abarca desde la planificación de procesos secuenciales hasta la distribución logística, donde intervienen aspectos de transporte, con el objetivo de encontrar la ruta más eficiente en términos de distancia o costo. Resolver este problema contribuye a mejorar procesos y optimizar recursos en diferentes ámbitos, tanto científicos como industriales. Entre las disciplinas que lo abordan se encuentran la investigación de operaciones y áreas de la informática como la algoritmia y la teoría de grafos.

Variables:

$x_{ij} =$ 1 si el agente viaja de la ciudad *i* a la ciudad *j*, 0 en caso contrario.

$d_{ij} =$ Representa la distancia entre las ciudades *i* y *j*.

Función Objetivo:

$$min \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} x_{ij}$$

Restricciones:

1. Cada ciudad debe visitarse exactamente una vez:

$$\sum_{j=1}^{N} x_{ij} = 1$$

2. Debe evitarse la formación de subciclos.

En la imagen 4.1 se observa una solución a este tipo de problemas llamada “Optimización por Colonia de Hormigas”, mostrando la cantidad de iteraciones y el costo de solución el cuál irá bajando con cada iteración que se haga. Así como se ve en la imagen 4.2 y la conclusión del mejor resultado después de casi 1000 iteraciones que se muestra en la imagen 4.3.

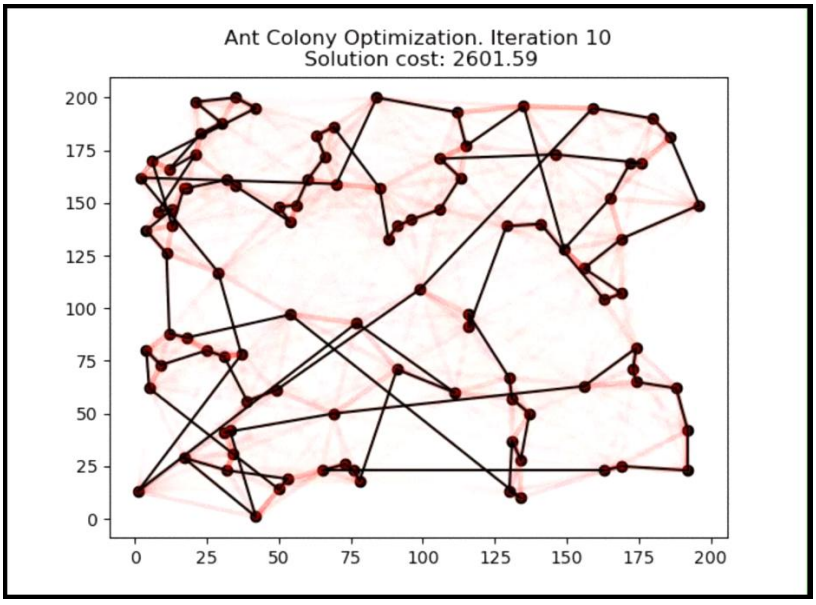


Imagen 4.1

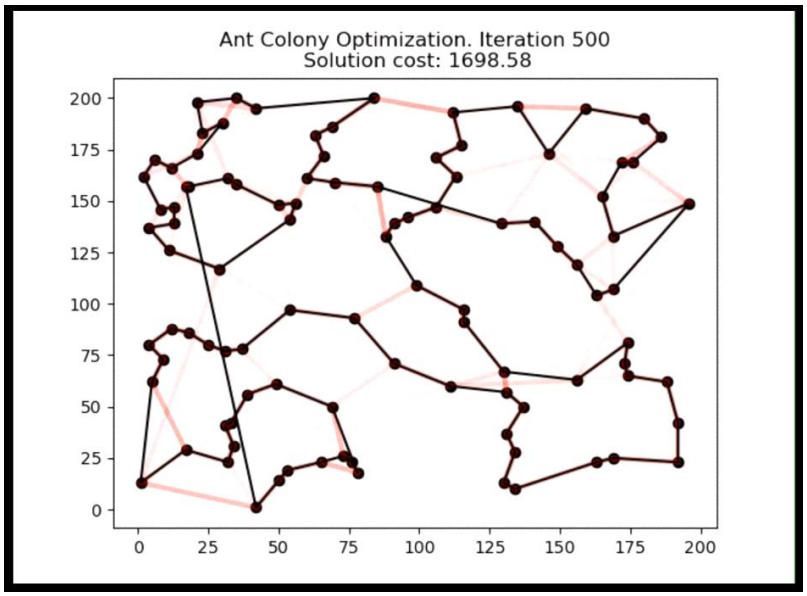


Imagen 4.2

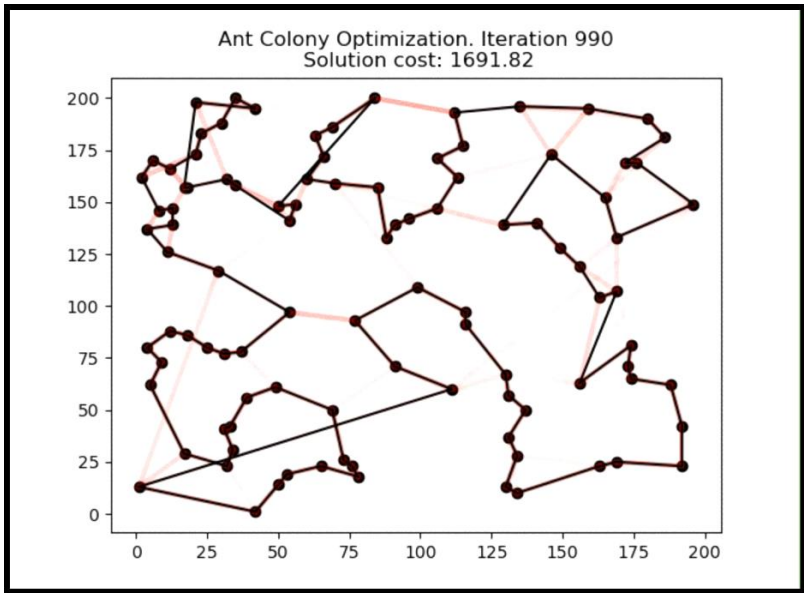


Imagen 4.3