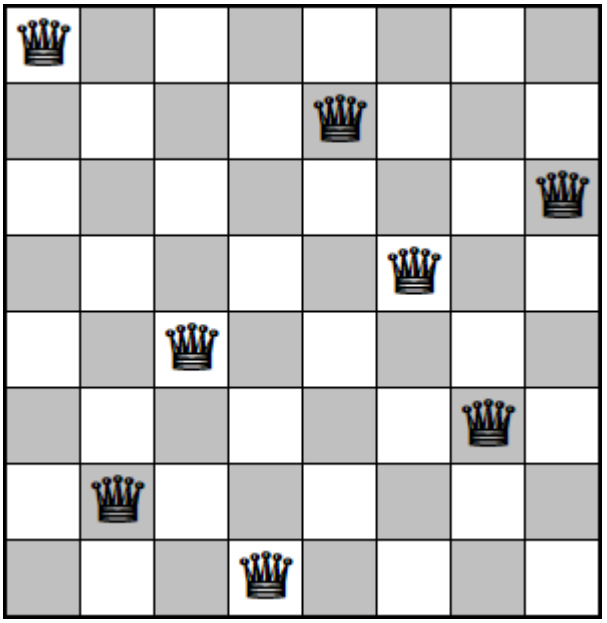
Trabajo Práctico 5: Búsqueda local

1. Implementar un algoritmo de *Hill Climbing* (versión canónica) para resolver el problema de las $N-$reinas. Tener en cuenta lo siguiente:



   - El algoritmo deberá ser capaz de encontrar solamente una solución para tableros de diferentes tamaños.
   - Una posible estructura para representar el tablero consiste en un arreglo de tamaño N, donde cada posición hace referencia a una columna del tablero; y cada valor hace referencia a una fila.
   - Se define una función objetivo $H(e)$ la cual contabiliza la cantidad de pares de reinas amenazadas para un tablero e.
   - Se deberá definir una variable que establezca el número máximo de estados que podrán ser evaluados.
   - El programa deberá devolver el tablero solución (únicamente la estructura que representa el tablero), junto a la cantidad de estados que tuvo que recorrer el algoritmo para llegar a la misma, y el tiempo empleado. En caso de alcanzar el máximo de estados evaluados, devolver la mejor solución encontrada y el valor correspondiente de la función H.

2. Implementar el algoritmo *Simulated Annealing* para resolver el problema del ejercicio 1.

3. Implementar un algoritmo genético para resolver el problema del ejercicio 1. Además de la implementación en código del mismo, se deberán incluir detalles respecto a:

   a) Definición de los individuos de la población.

   b) Estrategia de selección.

   c) Estrategia de reemplazo.

d) Operadores.

4. Ejecutar 30 veces cada uno de los algoritmos implementados en los ejercicios (1), (2) y (3), para el caso de 4, 8 y 10 reinas (opcional: 12 y 15 reinas). Para cada uno de los algoritmos:

   a) Generar una tabla con los resultados obtenidos y guardarla en formato `.csv` (*comma separated value*).

   b) Calcular:

      i) El número (porcentaje) de veces que se llega a un estado de solución óptimo.

      ii) El tiempo de ejecución promedio y la desviación estándar para encontrar dicha solución (se puede usar la función `time.time()` de `python`).

      iii) La cantidad de estados previos promedio, y su desviación estándar, por los que tuvo que pasar para llegar a una solución.

   c) Realizar gráficos de caja y bigotes (boxplots) que muestren la distribución de los tiempos de ejecución de cada algoritmo, y la distribución de la cantidad de estados previos visitados.

5. Para cada uno de los algoritmos, graficar la variación de la función $H()$ a lo largo de las iteraciones. Para ello, considerar sólo una ejecución en particular.

6. Indicar, según su criterio, cuál de los tres algoritmos implementados resulta más adecuado para la solución del problema de las $N-$reinas. Justificar.

7. Forma de entrega:

   a) Dentro del repositorio `ia-uncuyo-2024`, crear una carpeta con el nombre `tp5-busquedas-locales`.

   b) Dentro de la carpeta `tp5-busquedas-locales`, crear una nueva carpeta `code` para el proyecto desarrollado en `python`.

   c) Dentro de la carpeta `tp5-busquedas-locales`, colocar un archivo con el nombre `tp5-Nreinas.csv` que contenga la tabla generada en el ejercicio $4a$.

   d) Dentro de la carpeta `tp5-busquedas-locales`, colocar un archivo con el nombre `tp5-reporte.md` con la evaluación de desempeño de los algoritmos probados, respondiendo los ejercicios (4), (5) y (6). Organice el reporte siguiendo la estructura:

      - Introducción: descripción general del problema.
      - Marco teórico: descripción teórica y general de los algoritmos puestos a prueba.
      - Diseño experimental: descripción de los experimentos.
      - Análisis y discusión de resultados: presentar los resultados obtenidos en los experimentos, realizando un breve análisis de dichos resultados, e incluyendo gráficas o tablas que los resuman.
      - Conclusiones: conclusiones finales de los resultados obtenidos.

   e) Dentro de la carpeta `tp5-busquedas-locales`, crear una nueva carpeta `images`, que incluya todos los gráficos e imágenes utilizados en el reporte final.