N reinas

Luis Fernando Franco Flores

Introducción

El problema de las N-reinas es un desafío clásico de optimización, que consiste en colocar N reinas en un tablero de ajedrez de N x N de manera que ninguna reina ataque a otra. Este es un problema típico de la inteligencia artificial y los algoritmos de optimización combinatoria.

Una de las técnicas utilizadas para resolver este tipo de problemas es la **búsqueda tabú**. La búsqueda tabú es un algoritmo de optimización basado en la exploración de soluciones vecinas y el uso de una memoria para evitar ciclos de soluciones ya visitadas. A lo largo de este documento, se explora cómo el algoritmo de búsqueda tabú se aplica para resolver el problema de las N-reinas, utilizando un enfoque iterativo para reducir las colisiones entre reinas.

¿Qué es la Búsqueda Tabú?

La **búsqueda tabú** (Tabu Search) es un algoritmo de optimización que se basa en la exploración de soluciones vecinas a partir de una solución actual. A diferencia de los métodos clásicos de búsqueda local, la búsqueda tabú evita ciclos de soluciones mediante el uso de una "memoria" llamada **lista tabú**, que registra las soluciones recientes para que no se repitan. Esto permite que el algoritmo salga de óptimos locales y explore nuevas áreas del espacio de soluciones.

El algoritmo de búsqueda tabú se caracteriza por:

- **Exploración local**: Se buscan soluciones vecinas a la solución actual.
- **Memoria tabú**: Se evita volver a soluciones cercanas ya visitadas mediante la lista tabú.
- **Aceptación flexible de soluciones**: Se aceptan soluciones peores bajo ciertas condiciones, lo que ayuda a escapar de óptimos locales.

Pasos del Algoritmo

El algoritmo de búsqueda tabú para el problema de las N-reinas sigue los siguientes pasos:

1. **Inicialización**: Se comienza con una solución inicial (configuración de reinas en el tablero).

2. **Generación de Vecindario**: Se generan todas las posibles configuraciones vecinas (tableros con una reina intercambiada) a partir de la solución actual.
3. **Evaluación de Vecinos**: Se evalúan las soluciones vecinas en función del número de colisiones (conflictos entre reinas).
4. **Selección de la Mejor Solución**: Se selecciona la vecina con el menor número de colisiones que no esté en la lista tabú.
5. **Actualización de la Lista Tabú**: La solución seleccionada se agrega a la lista tabú.
6. **Enfriamiento y Terminación**: El proceso se repite durante un número fijo de iteraciones o hasta que se encuentra una solución óptima (sin colisiones).

Funcionamiento Detallado del Algoritmo

El código implementa el algoritmo de búsqueda tabú de la siguiente manera:

1. **Entrada de Datos**: El usuario ingresa el número de reinas y la configuración inicial del tablero.
2. **Contar Colisiones**: La función contar_colisiones evalúa cuántos pares de reinas se atacan entre sí en la configuración actual del tablero. Este es el valor que se busca minimizar.
3. **Generación de Vecindario**: La función generar_vecindario genera todas las posibles soluciones vecinas, intercambiando dos reinas en diferentes posiciones.
4. **Evaluación y Selección**: Para cada vecino, se calcula el número de colisiones y se selecciona el vecino con el menor número de colisiones que no esté en la lista tabú.
5. **Lista Tabú**: Una vez seleccionada una solución vecina, esta se agrega a la lista tabú. Si la lista alcanza el tamaño máximo (tabu_tenure), se elimina el elemento más antiguo.
6. **Iteración**: Este proceso se repite durante un número máximo de iteraciones (max_iter). Si se encuentra una solución sin colisiones, el algoritmo se detiene antes de alcanzar el número máximo de iteraciones.

Ventajas de la Búsqueda Tabú

La búsqueda tabú presenta varias ventajas clave para la resolución de problemas complejos como el de las N-reinas:

1. **Escape de Óptimos Locales**: Al evitar soluciones repetidas mediante la lista tabú, el algoritmo es capaz de escapar de óptimos locales y explorar una mayor parte del espacio de soluciones.
2. **Flexibilidad**: Es un algoritmo altamente flexible que puede adaptarse a una amplia gama de problemas de optimización combinatoria.
3. **Mejor Exploración**: Aunque no garantiza encontrar la solución óptima, permite encontrar soluciones cercanas a la óptima en un tiempo razonable.
4. **Control de la Diversidad**: La lista tabú ayuda a controlar la diversidad de las soluciones durante la búsqueda, lo que mejora la exploración del espacio de soluciones.

Conclusión

El algoritmo de **búsqueda tabú** es una técnica poderosa para resolver problemas de optimización combinatoria como el problema de las N-reinas. Gracias a su capacidad para escapar de óptimos locales y su uso de la memoria tabú para evitar soluciones repetidas, el algoritmo es capaz de explorar el espacio de soluciones de manera más eficiente que los enfoques tradicionales.

Aunque el algoritmo no garantiza una solución óptima, en la mayoría de los casos es capaz de encontrar una solución suficientemente buena en un número razonable de iteraciones. Su flexibilidad y eficiencia lo convierten en una opción atractiva para abordar problemas complejos de optimización, como el caso del problema de las N-reinas.