TP Integrador- Programación avanzada
UNaB 2022 cuatr.1
Jueves 30 de junio de 2022
Silvana Ojea

ALGORITMOS GENETICOS

Los algoritmos genéticos son mecanismos de búsqueda basados en las leyes de la selección natural de Darwin. Combinan la supervivencia de los individuos mejor adaptados junto con operadores de búsqueda genéticos como la mutación y el cruce, de ahí que sean comparables a una búsqueda biológica. Son algoritmos de optimización imitando los principios básicos de la naturaleza. Estos algoritmos se utilizan con éxito para gran variedad de problemas que no permiten una solución eficiente a través de la aplicación de técnicas convencionales.

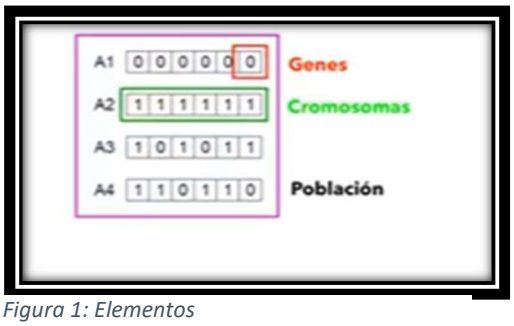


Figura 1: Elementos

Población: Subconjunto de las posibles soluciones.
Cromosoma: individuo o genotipo. Una de las soluciones del problema. Contiene la información de la solución del problema en forma codificada
Gen: es la posición de un elemento del cromosoma.
Alelo: posibles valores de un gen.
Locus: posición del gen en el cromosoma.

Operadores genéticos: Selección, Cruza, Mutación

Selección: proceso que sirve para elegir los individuos de la población mejor adaptados, para que actúen de progenitores de la siguiente generación. La cantidad de individuos está predeterminada por el tamaño de la población. En los algoritmos genéticos, la selección es un conjunto de reglas que sirven para elegir a los progenitores de la siguiente generación. Estos progenitores se reproducirán (cruzamiento genético) y generarán descendencia. Obtiene los mejores cromosomas de la población inicial
Cruza: durante esta fase, se seleccionan al azar dos cromosomas (individuos) padres y un punto de fraccionamiento también al azar y se obtienen dos hijos permutando los fragmentos de ambos padres. Es decir, los genes de los dos padres se mezclan entre sí para dar lugar a los diferentes hijos.
Mutación: se considera un operador básico, que proporciona un pequeño elemento de aleatoriedad en los individuos de la población, cambia uno o más genes del cromosoma, es el responsable del aumento o reducción del espacio de búsqueda dentro del algoritmo genético y del fomento de la variabilidad genética de los individuos de la población.

Parámetros: Tamaño de población, Probabilidad de cruza, Probabilidad de mutación.

El algoritmo tiene la siguiente secuencia:

1. Comenzar con una población inicial, la cual puede ser generada de manera aleatoria.
2. Calcular el fitness de cada individuo, permite valorar la aptitud de los individuos, dando una puntuación en términos de porcentaje.
3. Crear una nueva población:

   Aplicar el operador de selección con base en el fitness de la población, se seleccionan aquellos cromosomas que tengan más oportunidad de seguir, o sea aquellos que tienen el fitness más alto.

   Aplicar los operadores genéticos de cruce y mutación a la población actual para generar a la población de la siguiente generación.

   Aceptación, localizar la nueva descendencia como nueva población.
4. Reemplazar, usar la descendencia para correr el algoritmo.
5. Ir al paso 2 hasta que la condición parar se satisfaga.
6. Cuando se cumple la condición parar, se devuelve al mejor individuo encontrado.

A cada iteración de este proceso se le denomina una generación. El conjunto entero de generaciones se denomina una ejecución. Al final de una ejecución existen a menudo uno o varios cromosomas altamente adecuados en la población, y que pueden ser elegidos como solución al problema.

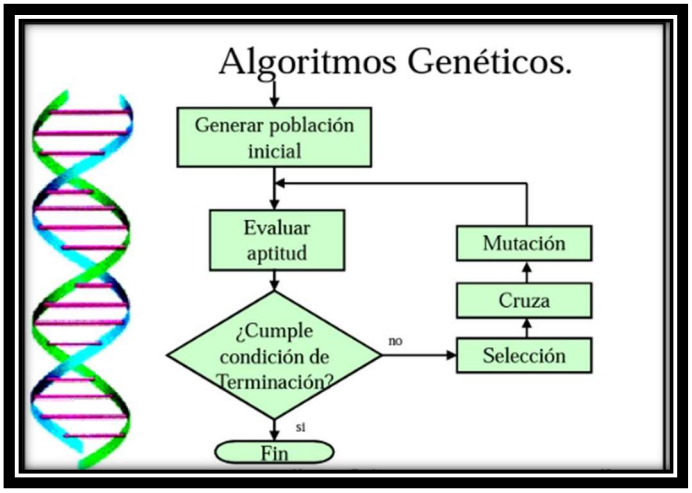


Figura 2: Algoritmos Genéticos

PROBLEMA DE LAS n REINAS

Consiste en encontrar el numero total de formas de ubicar n reinas en un tablero nxn sin que ninguna se ataque. Este problema tiene dos versiones, una que consiste en encontrar una solucion valida para un valor n dado, y la otra es encontrar todas las soluciones posibles para un valor n.

Para la resolución del problema se hace uso de uno de los pilares de la POO, el principio de abstracción, como la propia palabra lo indica, lo que implica es que la clase debe representar las características de la entidad hacia el mundo exterior, pero ocultando la complejidad que llevan

aparejada. O sea, nos abstrae de la complejidad que haya dentro dándonos una serie de atributos y comportamientos (propiedades y funciones) que podemos usar sin preocuparnos de qué pasa por dentro cuando lo hagamos.
El problema requiere que no exista ataque entre reinas, por ello existen tres restricciones que se deben considerar que son todas las posibilidades de ataque que este caso es en filas, columnas y diagonales.
Para ello una de las formas es utilizar un algoritmo recursivo (backtraking), donde se puede representar el tablero como un vector, la idea es que el vector se indexe por columna, e indique en cada posición "columna", la fila en la que está ubicada una reina. Entonces como la representación de soluciones es vectorial, sólo se requiere restricciones para filas y diagonales.
La restricción, $R_i \neq R_J$ (filas) garantiza valores distintos para las filas, luego la restricción $R_i + i \neq R_J + j$ (diagonal 1), evita posiciones en las diagonales derecha-arriba hacia izquierda-abajo, y por último la restricción $R_i - i \neq R_J - j$ (diagonal 2), evita posiciones en las diagonales izquierda-arriba hacia derecha-abajo.

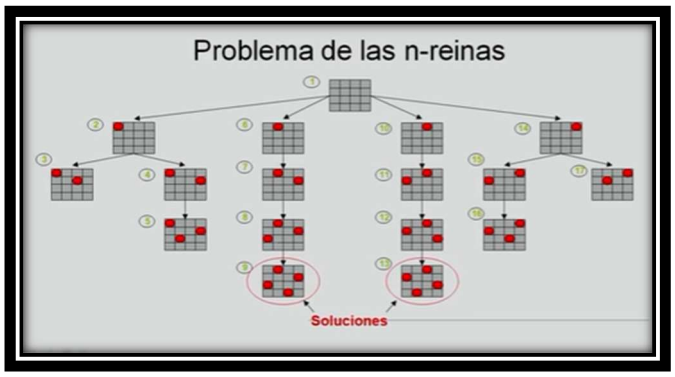


Figura 3:Backtracking como árbol de búsqueda

PATRONES DE DISEÑO

Los patrones de diseño son soluciones habituales a problemas que ocurren con frecuencia en el diseño de software. Son como planos prefabricados que se pueden personalizar para resolver un problema de diseño recurrente en tu código. El patrón no es una porción específica de código, sino un concepto general para resolver un problema particular. un patrón es una descripción de más alto nivel de una solución.
El patrón Observer es un patrón de diseño de software en el que un objeto, llamado Sujeto (Observable), administra una lista de dependientes, llamados Observadores, y les notifica automáticamente cualquier cambio de estado interno llamando a uno de sus métodos.
El patrón Observer sigue el concepto de publicación/suscripción. Un suscriptor, se suscribe a un editor. El editor luego notifica a los suscriptores cuando es necesario.
El observador almacena el estado que debe ser coherente con el tema. El observador solo necesita almacenar lo que es necesario para sus propios fines.

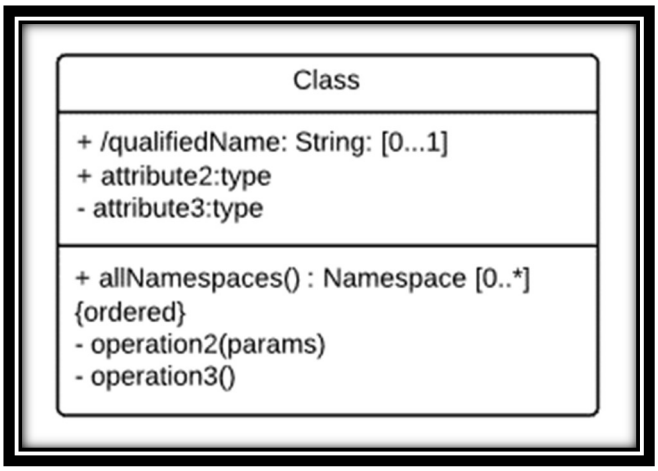
- Interfaz de sujeto: (Interfaz observable) La interfaz que el sujeto debe implementar.
- Sujeto Concreto: (Observable) El objeto que es el sujeto.

- Interfaz del observador: la interfaz que debe implementar el observador.
- Observador Concreto: El objeto que es el observador. Puede haber un número variable de observadores que pueden suscribirse/darse de baja durante el tiempo de ejecución.

Para este problema de las n reinas se propone el patrón de diseño de programación orientada a objetos *observer,* que es un patrón de diseño de comportamiento que te permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda al objeto que están observando, por lo tanto, permite interactuar con una interfaz gráfica para mostrar por ejemplo el tablero de ajedrez.
Se quiere mostrar un tablero que se va actualizando hasta llegar a la o las soluciones del problema.
El concepto básico de la implementación es, el acceso a un objeto concreto se pone a disposición de muchos otros objetos.

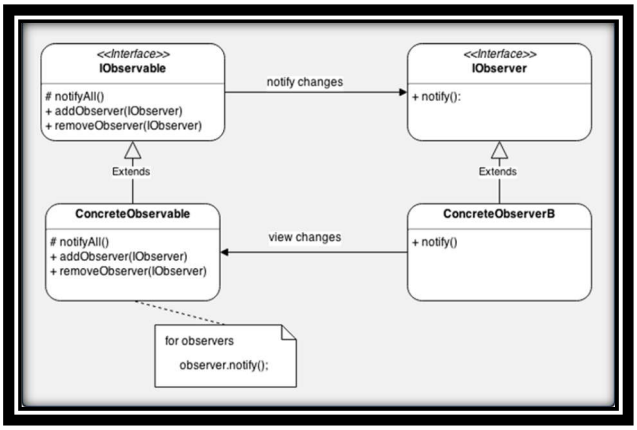
LENGUAJE UNIFICADO DE MODELADO (UML)

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Los diagramas estructurales muestran la estructura estática del sistema y sus partes en diferentes niveles de abstracción, mientras que los diagramas de comportamiento muestran el comportamiento dinámico de los objetos en el sistema.
Se muestran dos ejemplos de tipos de diagramas en UML, diagrama de clases y diagrama de secuencia pertenecientes a las categorías anteriormente mencionas respectivamente.



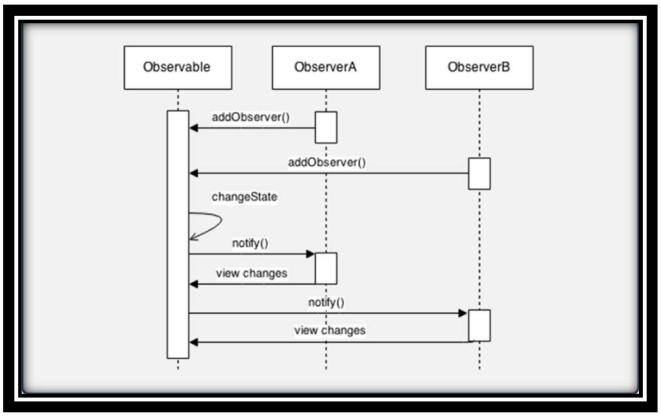
Partes de un diagrama de clases UML:
Sección superior: se escribe el nombre de cada clase o elemento que se desea clasificar.
Sección media: Describe los atributos de la clase, puntualizando de forma específica sus cualidades.
Sección inferior: Se describen las operaciones o metodología a implementar, describiendo en modo de lista vertical cada operación.

Figura 4: Partes de un diagrama UML



IObservable: Interface de los objetos que quieren ser observados, se definen los métodos mínimos a usar.
ConcreteObservable: Clase que desea ser observada, ésta implementa IObservable y debe implementar sus métodos.
IObserver: Interfaces que deben implementar todos los objetos que desean observar los cambios de IObservable.
ConcreteObserver: Clase concreta, atenta de los cambios de IObserver, hereda de IObserver y debe de implementar sus métodos.

Figura 5: Diagrama de clases del patrón Observer



El ObserverA se registra con el objeto Observable para ser notificado de algún cambio.
El ObserverB se registra con el objeto Observable para ser notificado de algún cambio.
Ocurre algún cambio en el estado del Observable.
Todos los Observers son notificados con el cambio ocurrido.

Figura 6: Diagrama de secuencia del patrón Observer

Fuentes:

https://www.youtube.com/watch?v=8Ia8Kx1CFXE
https://www.youtube.com/watch?v=p7ZtAJuQxr8
https://github.com/alirezaeftekhari/8-queens-genetic-algorithm/blob/main/main.py
https://www.campusmvp.es/recursos/post/los-conceptos-fundamentales-sobre-programacion-orientada-objetos-explicados-de-manera-simple.aspx
https://www.frba.utn.edu.ar/wp-content/uploads/2021/02/AlgoritmosGeneticos-compressed.pdf
https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-el-patron-observer/
https://refactoring.guru/es/design-patterns/observer
https://reactiveprogramming.io/blog/es/patrones-de-diseno/observer