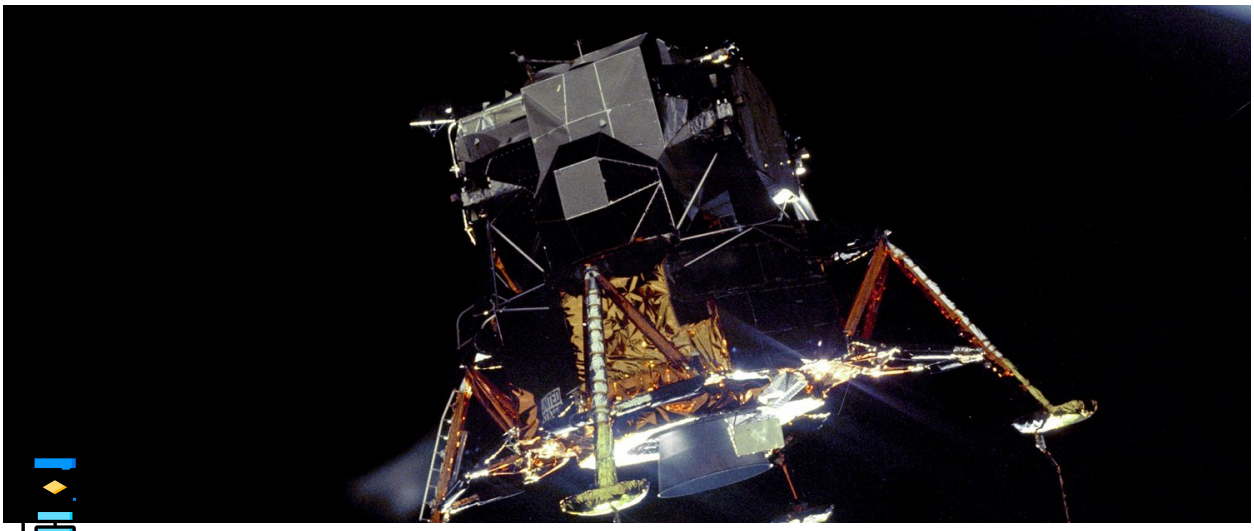


Sistemas de Búsqueda y Razonamiento

Unidad 1: Introducción

- Problemas de búsqueda
- Espacios estado
- Técnicas de búsqueda

Unidad 2: Técnicas no informadas

- Las listas de estados
- Técnica primero en amplitud
- Técnica primero en profundidad

Unidad 3: Técnicas informadas

- El valor de la calidad
- Búsqueda última
- Técnica para la mejora
- Técnica Avaox

- Técnica A-Estrella

Unidad 4: Técnicas adversarias

- La competencia y la búsqueda
- Técnica MinMax
- Poda alfa y beta

Unidad 5: Técnicas locales

- La vecindad de un estado
- El óptimo local
- Técnica para ascenso de colina
- Técnica beam
- Técnica tabú
- Técnica por templado simultáneo

Unidad 6: Problemas con restricciones

- El concepto de problema con restricciones.
- Variables, dominios y restricciones.
- Satisfacción de restricciones usando búsqueda exahustiva

ANTECEDENTES

Tenemos un problema que debe ser resuelto, y ese problema tiene una representación muy especial:

1. En cualquier momento se encuentra en uno de n estado.

   Al conjunto de n estados lo llamamos Q, y es como a continuación se indica:

$$Q = q1, q2, \ldots, qn$$

2. Estando en el estado q, es posible cambiarse a otro estado q, usando una acción aj, para $j = 1, 2, \ldots m$.

   Al conjunto con las m acciones le llamamos A, y es como a continuación se indica:

$$A = a1, a2, \ldots, am$$

3. Y luego, existe una función de transición $S : QxA \rightarrow Q$ (es decir que es una función que toma un par formado por un estado $q1 = Q$ y una acción $ajEA$ y regresa un estado $qxEQ$) que indica a qué estado qk cambia el problema estando actual/e en el estado qi, y usando la acción aj.
4. De todos los estados en Q, uno es el estado de inicio, el estado en donde el problema inicia al "conocer" el proceso de búsqueda. Al estado de inicio normalmente se le denota como $q0$ o qm.
5. Y de todos los estados en Q, uno o más son estados meta, es decir estados cuyas características nos hacen considerar al problema como resuelto $(qmetac_Q)$. Así que un problema de búsqueda es una tupla como la siguiente: $P = Q, A, q0, qmeta, S$. Aquí P es el problema que se debe resolver, pero si bien esta es una representación formal del problema, quizá sea más fácil visualizar al problema gráficamente, y por eso, lo más común es representar al problema como un grafo dirigido (dígrafo) en donde:

   $() \rightarrow nodo.$

   - El estado $q1EQ$ se representa como un nodo $(q1)$.
   - El estado $q0$ se representa como: $\rightarrow (q1)$.
   - Si $q1Eqmeta$ se representa como un nodo con doble circulo $((q1))$.
   - Y la transición $S(q1, q2) = qk$ se representa como una arista que sale de $q1$ y llega a qk, y se etiqueta como ak: $(q1) \rightarrow (qk)$ donde → lleva etiqueta de qj.

     No siempre es necesario etiquetar la acción con su nombre.
   - Si por alguna razón el nombre de la acción es prescindible (no importa o es obvio) puede evitarse el nombre.

- Eso de que exista una acción que cambia al problema de $q1$ a $q2$ y o viceversa, se llaman acciones complementarias.
- Se puede simplificar la representación gráfica, en lugar de una arista de $(q1) \rightarrow (q2)$ y luego de regreso otra arista de$(q2) \rightarrow (q1)$, es decir; $(q1) \leftrightarrows (q2)$ puede usarse una sola arista bidireccional. $(q1) \longleftrightarrow (q2)$.

¿De qué está hecho un estado?	Depende del problema, pero digamos que contienen una o más variables de estado, que son las variables con las que se controla el proceso de búsqueda. Por ejemplo el nombre del estado actual.
¿Puede una acción etiquetarse con algo que no sea su nombre específico?	Sí, en ocasiones una acción se etiqueta como el costo de la acción, la distancia de la acción , etc.

BÚSQUEDA CIEGA O NO INFORMADA

Búsqueda Ciega.pdf

Estas técnicas son exhaustivas en el espacio del problema, pueden ser inconvenientes por eso pero eso sí, son muy seguras. Entre los tópicos que aquí estudiamos están los siguientes:

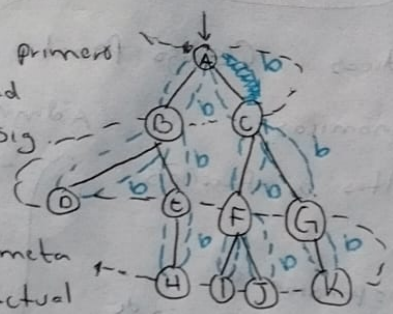
1. Búsquedas exhaustivas
2. La búsqueda primero en amplitud
3. La búsqueda primero en profundidad
4. Adaptando el AGB a primero en amplitud
5. Adaptando el AGB a primero en profundidad

---- Amplitud
--- Profundidad

12/02/24

Y si el problema a resolver tuviera la sig. forma

Mientras que primero en Profundidad procede de la sig. manera:



"...buscar a la meta en la rama actual antes de buscarlo en la sig. rama..."

[Backtracking permite Cambiar la rama de búsqueda porque la actual ya se agotó]

- Lista Abierta: Contiene estados que no se han explorado y que son candidatos a ser el siguiente estado.

- Lista cerrada contiene estados que ya han sido explorados y no se toman en cuenta para evitar ciclarse.

Si hubiera que "recorrer", "explorar" todo el problema en busca de la solución. Primero en Amplitud, es una técnica que procede así:

"...buscar a la meta en el nivel actual antes de buscarlo en el siguiente..."

ALGORITMO GENERAL DE BÚSQUEDA O AGB

- Búsqueda en cualquier tipo de gráfo
- Directo para programarlo

- Lista de estados Abierta
- Lista de estados Cerrada

Los ciclos son algo a evitar

- Hacer abierto [Ini.]
- Hacer cerrado [ ]
- Hacer lista = false
- while lista = false
  - Hacer actual = Siguiente(Abierta)
  - if actual == meta
    - lista = true
    - agregar(actual, Cerrado)
- else
  - expandir(actual, abierta, cerrado)

Aunque las tres funciones del AGB anterior son:

- `Siguiente():` que de la lista de abiertos extrae el siguiente estado a visitar con el problema.
- `Agregar():` que agrega el estado que se le indique al final de la lista que se le indique.
- `Expandir():` que toma a todos los hijos del estado actual y los agrega al final de Abiertos (menos aquellos que ya están Cerrados o que ya estaban antes en Abiertos).

Estas funciones toman parámetros o argumentos, por ejemplo

`siguiente( Abiertos )`: de aquí sale el siguiente estado a visitar.

`agregar( lista, estado ):` Agregar estado al final de la lista.

`expandir( actual, Abiertos, Cerrados ):` Tomar todos los hijos de actual y agregarlos en Abiertos a menos que ya estén Cerrados o en Abiertos.

Primero en amplitud o primero en profundidad ¿Cuál es la diferencia?

Ya conocemos que primero en amplitud explora el espacio estado del problema por niveles o generaciones, y que primero en profundidad lo hace por las ramas, pero ¿Qué significa esto para el AGB?

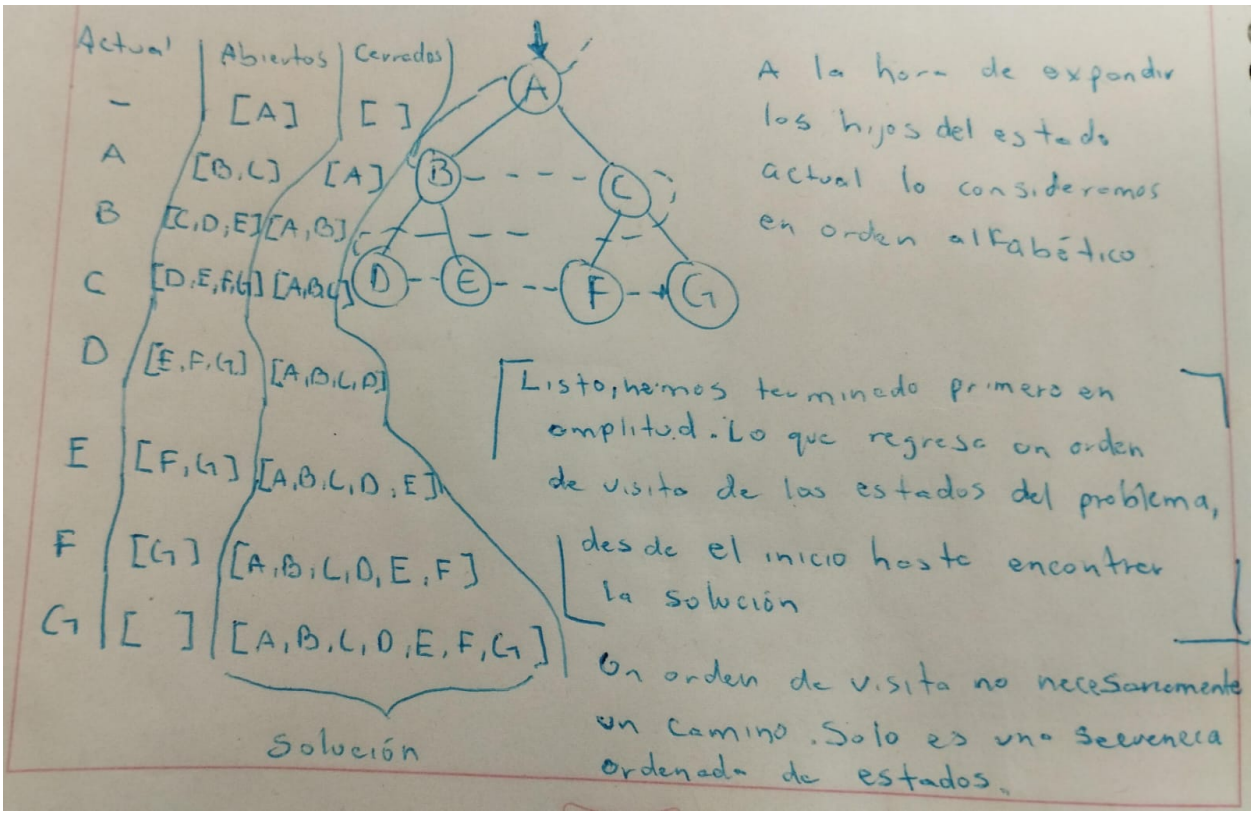
Por un lado, puesto que son búsquedas ciegas la función siguiente() no tiene forma de preferir un estado en Abiertos, todos se ven igual para ella. Pero por otro lado, primero en amplitud explora por niveles y primero en profundidad por ramas ¿Cómo unimos esto? Y lo más sorprendente de todo es que todo esto se une de la siguiente manera:

- Para implementar primero en amplitud la lista de estados Abiertos se debe comparar como una cola (FIFO).
- Para primero en profundidad la lista Abiertos se comportan como una pila (LIFO).

- Para primero en amplitud `siguiente()` extrae de Abiertos el primer estado que fue agregado.
- Y para primero en profundidad `siguiente()` extrae de Abiertos el último estado que fue agregado.

Uno de los beneficios más grandes de implementar primero en profundidad de esta manera es que el proceso de **backtracking** ya está incorporado en la misma lógica de la función `siguiente()`.

Primero en Amplitud:



La línea punteada muestra, el recorrido primero en amplitud encima del grafo del problema.

BÚSQUEDA INFORMADA

Búsqueda Informada.pdf

En esta unidad vamos a estudiar las técnicas de búsqueda que normalmente se consideran las mejores. Estas técnicas a diferencia de las anteriores no son exahustivas en el espacio del problema, pues mas bien si guian por un camino específico en el problema usando para ellos algún tipo de información. Entre los tópicos en esta unidad tenemos los siguientes:

1. La naturaleza de la información
2. La búsqueda primero por lo mejor
3. La búsqueda avara
4. La búsqueda A-Estrella (A*)
5. Acoplando el AGB a primero por lo mejor
6. Acoplando el AGB a la búsqueda avara
7. Acoplando el AGB a la búsqueda A-estrella

La naturaleza del informacion

Acabamos de establecer uno de los aspectos mas importantes de las tecnicas de busqueda informada, que poseen informacion para ayudarse a guiarse por un camino. Pero ¿esta informacion que significa, que es, quien la provee? Esto lo respondemos afirmando que existen dos tipo de naturalezas en la informacion que podemos usar en una busqueda:

1. La informacion puede ser un costo de llevar a cabo una accion.
2. La informacion puede ser un valor heurıstico de llevar a cabo una accion.

La búsqueda primero por lo mejor

La búsqueda primero por lo mejor, también conocida como búsqueda de mejoría, es una estrategia de búsqueda informada que explora un árbol de búsqueda expandiendo en cada paso el nodo que parece más prometedor según una función de evaluación. Esta función de evaluación evalúa la calidad de un nodo en función de alguna medida de "bueno", sin tener en cuenta el costo total hasta el momento. La búsqueda primero por lo mejor es rápida pero puede no encontrar la solución óptima si se utilizan heurísticas inadecuadas.

Ejemplo:
En un problema de ruta, como encontrar la ruta más corta entre dos puntos en un mapa, la búsqueda primero por lo mejor podría expandir primero los nodos que están más cerca del destino según alguna heurística de distancia.

La búsqueda avara

La búsqueda avara es otra estrategia de búsqueda informada que, al igual que la búsqueda primero por lo mejor, selecciona el nodo más prometedor en cada paso. Sin embargo, a diferencia de la búsqueda primero por lo mejor, la búsqueda avara prioriza los nodos que parecen estar más cerca de la solución final según una función de evaluación heurística. Esto significa que la búsqueda avara no considera el costo total hasta el momento y puede terminar en soluciones subóptimas si la heurística subestima el costo real.

Ejemplo:
En el mismo problema de ruta, la búsqueda avara podría expandir primero los nodos que están más cerca del destino según una heurística de distancia en línea recta, sin tener en cuenta la distancia real a través de los caminos disponibles.

*La búsqueda A (A-estrella)**

A* es un algoritmo de búsqueda informada que combina las ventajas de la búsqueda primero por lo mejor y la búsqueda avara. Utiliza una función de evaluación que considera tanto el costo real hasta el momento como una estimación heurística del costo restante hasta la meta. La función de evaluación de A* es $f(n)=g(n)+h(n)$f(n)=g(n)+h(n), donde $g(n)$g(n) es el costo real para llegar

al nodo nn y $h(n)h(n)$ es la estimación heurística del costo desde nn hasta la meta. A* garantiza encontrar la solución óptima si la heurística es admisible (nunca sobreestima el costo real) y consistente (satisface la desigualdad triangular).

Ejemplo:
En el problema de ruta, A* expandiría primero los nodos que tienen la menor suma de costo real y estimación heurística, lo que significa que consideraría tanto la distancia recorrida hasta el momento como la estimación de la distancia restante hasta el destino.

Acoplando el AGB a primero por lo mejor

- **Utilizando AGB para mejorar la calidad de las soluciones iniciales:** La búsqueda primero por lo mejor es rápida pero puede no encontrar la solución óptima. Al acoplar el AGB, podemos utilizarlo para generar y mejorar soluciones iniciales que se expandan en la búsqueda primero por lo mejor. Esto se logra codificando las soluciones como individuos en la población del AGB y utilizando operadores genéticos para evolucionar estas soluciones hasta que sean adecuadas para iniciar la búsqueda primero por lo mejor.
- **Ajustando la función de evaluación de la búsqueda primero por lo mejor:** La calidad de la búsqueda primero por lo mejor puede mejorar ajustando su función de evaluación. El AGB puede utilizarse para evolucionar parámetros de la función de evaluación, como pesos o coeficientes, para que se adapten mejor al problema específico que se está abordando. Esto puede conducir a una búsqueda más efectiva de soluciones de alta calidad.

Acoplando el AGB a la búsqueda avara

- **Generando y mejorando soluciones iniciales con AGB:** Al igual que con la búsqueda primero por lo mejor, el AGB puede utilizarse para generar y mejorar soluciones iniciales que se expandan en la búsqueda avara. Esto se realiza de manera similar codificando las soluciones como individuos en la población del AGB y aplicando operadores genéticos para mejorar gradualmente estas soluciones antes de iniciar la búsqueda avara.

- **Explorando el espacio de soluciones con AGB:** La búsqueda avara puede tener dificultades para explorar todo el espacio de soluciones, especialmente en problemas con múltiples óptimos locales. Al acoplar el AGB, podemos utilizar su capacidad para explorar de manera más exhaustiva el espacio de soluciones, lo que puede ayudar a la búsqueda avara a encontrar soluciones de mejor calidad.

Acoplando el AGB a la búsqueda A-estrella:

- *Ajustando la heurística de A con AGB:*Como se mencionó anteriormente, la búsqueda A* depende en gran medida de la calidad de su heurística. Al acoplar el AGB, podemos utilizarlo para ajustar automáticamente la heurística de A* mediante la evolución de parámetros relevantes. Esto puede mejorar la eficiencia y la calidad de las soluciones encontradas por A*.
- **Generando soluciones iniciales y explorando el espacio de soluciones con AGB:** De manera similar a cómo se aplica al acoplamiento con la búsqueda primero por lo mejor y la búsqueda avara, el AGB puede utilizarse para generar soluciones iniciales de alta calidad y explorar de manera más efectiva el espacio de soluciones cuando se combina con A*. Esto puede ayudar a A* a encontrar soluciones óptimas en problemas complejos.

BÚSQUEDA ADVERSARIA

Búsqueda Adversaria.pdf

En esta unidad vamos a estudiar la llamada búsqueda adversaria, una forma de primero por lo mejor pero aplicada a la resolución de problemas de juego o competencia entre dos aversarios. Específicamente **estudiamos una forma de**

búsqueda llamada MinMax. Entre los tópicos aquí estudiados están los siguientes:

1. El espacio estado de un juego
2. Los adversarios Min y Max
3. El criterio de utilidad de Max
4. Niveles Min y niveles Max
5. La búsqueda MinMax
6. La poda alfa y beta

El espacio estado de un juego

El espacio estado de un juego es esencialmente el conjunto de todas las posibles configuraciones que puede tomar un juego en cualquier punto dado. Por ejemplo, en el juego del ajedrez, el espacio estado incluye todas las posibles disposiciones de las piezas en el tablero, así como información sobre los turnos de los jugadores y la situación del juego en general. Este espacio estado puede ser vasto y complejo, ya que cada posición de una pieza y cada posible jugada de cada jugador cuenta como un estado distinto.

Los adversarios Min y Max

En el contexto del algoritmo Minimax, Min y Max representan a los dos jugadores en un juego competitivo. Por ejemplo, en el ajedrez, Max sería el jugador que intenta maximizar su puntaje (como maximizar sus piezas en el tablero o maximizar su posición para un posible jaque mate), mientras que Min sería el jugador que intenta minimizar el puntaje de Max (como minimizar sus propias pérdidas o bloquear las jugadas potenciales de Max).

El criterio de utilidad de Max

El criterio de utilidad de Max es una función que evalúa un estado del juego desde la perspectiva del jugador Max. Por ejemplo, en el ajedrez, esta función podría asignar valores numéricos a los diferentes estados del tablero, donde un valor más alto representa una posición más favorable para Max (como tener más piezas en el tablero o estar en una posición más dominante).

Niveles Min y niveles Max

En el algoritmo Minimax, los niveles Min representan los niveles del árbol de búsqueda donde el jugador Min está tomando decisiones, y los niveles Max representan los niveles donde el jugador Max está tomando decisiones. Por ejemplo, en un árbol de búsqueda para el ajedrez, los niveles Min representarían los movimientos posibles del oponente de Max, mientras que los niveles Max representarían las posibles respuestas de Max a esos movimientos.

La búsqueda MinMax

La búsqueda MinMax es un algoritmo utilizado para determinar la mejor jugada posible en un juego de dos jugadores con información perfecta. Por ejemplo, en el ajedrez, la búsqueda MinMax evaluaría todas las posibles secuencias de movimientos a partir de una posición dada, asumiendo que tanto Max como Min juegan de manera óptima, y determinaría la mejor jugada posible para Max en esa posición.

La poda alfa y beta

La poda alfa-beta es una técnica utilizada para mejorar la eficiencia de la búsqueda MinMax al reducir el número de nodos evaluados. Esta técnica se basa en mantener un rango de valores conocidos (llamados alfa y beta) para cada nodo del árbol de búsqueda. Si se descubre que una rama no afectará la elección de la mejor jugada, se puede detener la exploración de esa rama, lo que ahorra tiempo y recursos computacionales. La poda alfa-beta es especialmente útil en árboles

de búsqueda grandes y profundos, como los encontrados en juegos complejos como el ajedrez o el Go.

BÚSQUEDA LOCAL

Búsqueda Local.pdf

En esta unidad vamos a estudiar aún otra forma de búsqueda que a diferencia de las anteriores no asume que cuenta con todo el espacio estado del problema, y por lo tanto, es capáz de llevar a cabo un proceso de búsqueda en ese subespacio que si poseé. Además y como consecuencia de lo anterior, la solución alcanzada no es necesariamente la mejor pero es lo suficientemente satisfactoria. Entre los tópicos que en esta unidad estudiamos tenemos los siguientes:

1. El concepto de vecindad y de localidad
2. El algoritmo general de búsqueda local (AGBL).
3. Búsqueda por ascenso (o descenso) de colina.
    a. Acoplando en AGBL a ascenso de colina
4. Búsqueda concentrada.
    a. Acoplando en AGBL a la búsqueda concentrada
5. Búsqueda tabú.
    a. Acoplando en AGBL a la búsqueda tabú
6. Búsqueda por templado simulado.
    a. Acoplando en AGBL a la búsqueda por templado simulado.
7. Búsqueda con algoritmos genéticos (AGs).
    a. Acoplando en AGBL a un AG.

El concepto de vecindad y de localidad

- **Vecindad:** En el contexto de la optimización y la búsqueda local, la vecindad se refiere al conjunto de soluciones que son "cercanas" o adyacentes a una solución dada en el espacio de búsqueda. Por ejemplo, en un problema de optimización de ruta, la vecindad de una solución podría ser el conjunto de todas las soluciones que se pueden obtener cambiando una única ciudad en la ruta.
- **Localidad:** La localidad se refiere al principio de que las soluciones óptimas tienden a estar agrupadas en regiones del espacio de búsqueda. Esto implica que, al explorar las soluciones cercanas en la vecindad de una solución actual, es más probable encontrar soluciones de alta calidad.

El algoritmo general de búsqueda local (AGBL)

El algoritmo general de búsqueda local es un enfoque general para resolver problemas de optimización al buscar iterativamente soluciones mejores dentro de una vecindad de la solución actual. El algoritmo comienza con una solución inicial y luego busca de manera iterativa soluciones vecinas que mejoren el valor de la función objetivo. El proceso continúa hasta que se alcanza un criterio de parada, como un máximo de iteraciones o una mejora mínima en la solución.

Búsqueda por ascenso (o descenso) de colina

La búsqueda por ascenso de colina es un tipo de búsqueda local que sigue la estrategia de moverse siempre hacia la solución vecina que mejora más el valor de la función objetivo. Esto significa que el algoritmo "asciende" la colina de la función objetivo, buscando un máximo local. Si la función objetivo es de minimización, se puede aplicar una búsqueda por descenso de colina.

Búsqueda concentrada

La búsqueda concentrada es una variante de la búsqueda local que se enfoca en explorar intensamente regiones específicas del espacio de búsqueda donde es más probable encontrar soluciones óptimas. Esto se logra mediante la adaptación de la definición de vecindad para enfocarse en áreas prometedoras del espacio de búsqueda.

Búsqueda tabú

La búsqueda tabú es una técnica de búsqueda local que evita quedarse atrapada en óptimos locales mediante el uso de una lista tabú que registra movimientos prohibidos o "tabúes". Estos movimientos prohibidos evitan que el algoritmo vuelva a visitar soluciones previamente exploradas, lo que ayuda a explorar más exhaustivamente el espacio de búsqueda.

Búsqueda por templado simulado

La búsqueda por templado simulado es un algoritmo de búsqueda local que simula el proceso físico de templado de metales. Comienza con una solución inicial y permite movimientos que empeoran el valor de la función objetivo con cierta probabilidad. Esta probabilidad disminuye a medida que avanza la búsqueda, lo que permite escapar de óptimos locales.

Búsqueda con algoritmos genéticos (AGs)

Los algoritmos genéticos son técnicas de optimización inspiradas en la evolución biológica. Utilizan conceptos como la selección natural, la reproducción y la mutación para explorar el espacio de soluciones en busca de soluciones óptimas. En la búsqueda local, los algoritmos genéticos pueden utilizarse para generar nuevas soluciones y explorar vecindades más amplias del espacio de búsqueda.

Para cada punto de acoplamiento con el AGBL, la idea básica es utilizar la capacidad del algoritmo genético para generar nuevas soluciones y explorar diferentes regiones del espacio de búsqueda, mejorando así la capacidad del AGBL para encontrar soluciones óptimas o de alta calidad.

PROBLEMAS CON RESTRICCIONES

Problemas con Restricciones.pdf

En esta última unidad vamos a estudiar la manera de resolver un estilo de problema muy especial conocido como problemas con restricciones usando para ello un proceso de búsqueda, uno muy parecido a la búsqueda primero en profundidad. Entre los tópicos que en esta uidad vamos a tratar están los siguientes:

1. El concepto de problema con restricciones.
2. Variables, dominios y restricciones.
3. Satisfacción de restricciones usando búsqueda exahustiva.

El concepto de problema con restricciones

Un problema con restricciones es un tipo de problema en el que se deben encontrar soluciones que satisfagan un conjunto de restricciones específicas. Estas restricciones limitan las posibles combinaciones de valores que pueden tomar las variables en la solución. Por ejemplo, en un problema de asignación de horarios, las restricciones podrían incluir que ciertas clases no pueden tener lugar al mismo tiempo o que ciertos recursos deben estar disponibles en momentos específicos.

Variables, dominios y restricciones

- **Variables:** Son las entidades que representan las incógnitas o elementos de interés en el problema. Por ejemplo, en un problema de asignación de horarios, las variables podrían ser los horarios de las clases o las asignaciones de profesores a cursos.
- **Dominios:** Son los conjuntos de posibles valores que pueden tomar las variables. Por ejemplo, para el problema de asignación de horarios, el dominio de las variables de tiempo podría ser un conjunto de intervalos de tiempo disponibles para programar las clases.
- **Restricciones:** Son las reglas que limitan las combinaciones válidas de valores para las variables. Por ejemplo, una restricción en el problema de asignación de horarios podría ser que dos clases no pueden tener lugar al mismo tiempo o que ciertos profesores solo pueden enseñar ciertos cursos.

Satisfacción de restricciones usando búsqueda exhaustiva

La satisfacción de restricciones mediante búsqueda exhaustiva implica explorar de manera sistemática todas las posibles combinaciones de valores para las variables, verificando si cada combinación satisface todas las restricciones del problema. Esto a menudo se realiza utilizando algoritmos de búsqueda, como la búsqueda primero en profundidad, que exploran recursivamente todas las ramas del árbol de búsqueda hasta encontrar una solución viable o hasta que se agoten todas las posibilidades.

Por ejemplo, en un problema de asignación de horarios, se podría usar búsqueda exhaustiva para probar todas las combinaciones posibles de horarios para las clases, verificando que cada combinación cumpla con las restricciones, como la disponibilidad de profesores y aulas, y la ausencia de conflictos de horario. Este enfoque garantiza encontrar una solución si existe, pero puede ser costoso computacionalmente en problemas con un gran espacio de búsqueda.