UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Representación de Espacios

Alan Reyes-Figueroa

Métodos Numéricos II

(Aula 34) 07.noviembre.2024

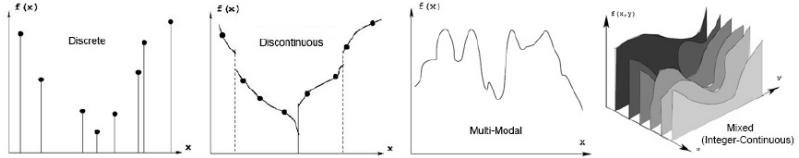
Optimización Estocástica

Introducimos ahora algunos métodos del área llamada optimización estocástica. La idea principal es que desarrollamos métodos de búsqueda en donde interviene algunas de las siguientes:

- aleatoriedad,
- distribuciones.

Esto métodos sirven principalmente para atacar problemas de tipo

- espacio factible discreto
- múltiples mínimos locales (multimodal)



Algunas funciones problemáticas: (a) discreta, (b) discontinua, (c) multimodal, (d) mixta.

Optimización Estocástica

Definiciones básicas:

Una **heurística**, o técnica heurística, es cualquier enfoque para la resolución de problemas o el autodescubrimiento que emplea un método práctico que no se garantiza que sea óptimo, perfecto o racional, pero que es suficiente para alcanzar una meta o aproximación inmediata a corto plazo.

Se utiliza cuando es imposible o poco práctico encontrar una solución óptima. En ese caso, se pueden utilizar métodos heurísticos para acelerar el proceso de búsqueda de una solución satisfactoria.

Ejemplos: prueba y error, regla empírica o una conjetura educada (*educated guess*).

En computación y optimización matemática, una **metaheurística** es un procedimiento de nivel superior o heurística diseñado para encontrar, generar o seleccionar una heurística (algoritmo de búsqueda parcial) que puede proporcionar una solución suficientemente buena para un problema de optimización.

Optimización Estocástica

Las metaheurísticas muestran un subconjunto de soluciones que, de otro modo, es demasiado grande para ser enumerado o explorado por completo. Hacen relativamente pocas suposiciones sobre el problema de optimización.

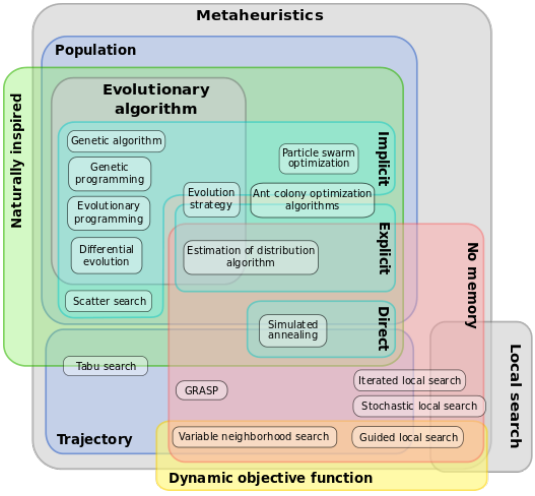
En comparación con los algoritmos de optimización y los métodos iterativos, las metaheurísticas no garantizan que se pueda encontrar una solución globalmente óptima. Muchas metaheurísticas implementan alguna forma de optimización estocástica $\Longrightarrow$ la solución encontrada depende de un conjunto de variables aleatorias generadas.

Ejemplos: Búsqueda local, algoritmos bio-inspirados, algoritmos evolutivos.

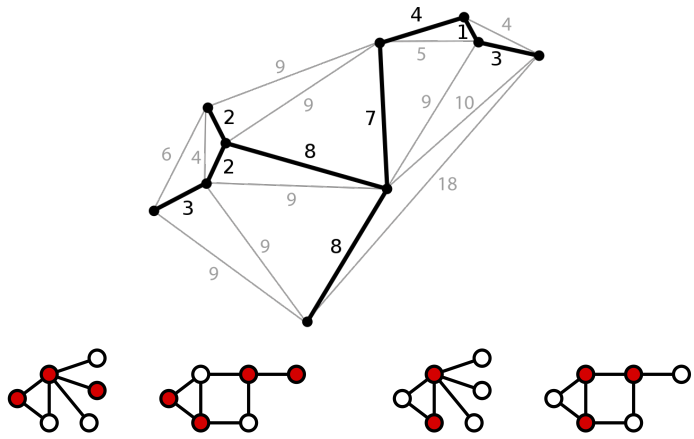
Características de las metaheurísticas:

- son estrategias que guían el proceso de búsqueda.
- exploran de manera eficiente el espacio de búsqueda, soluciones sub-óptimas.
- son algoritmos aproximados y generalmente no deterministas.
- no son específicas de un problema.

Optimización Estocástica



Ejemplos



(a) *Minimum spanning tree*, (b) *Vertex covering*.

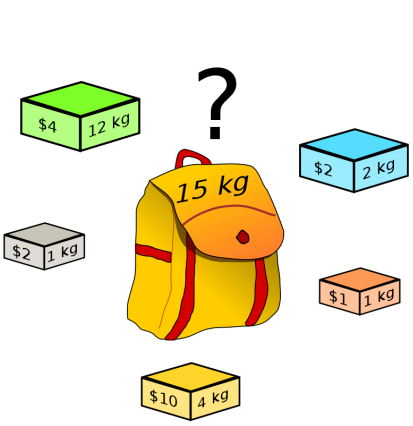
Ejemplos

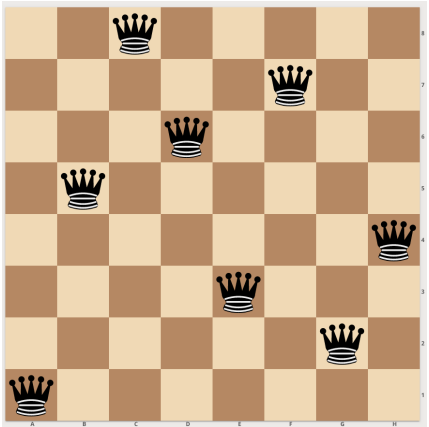
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Sudoku.

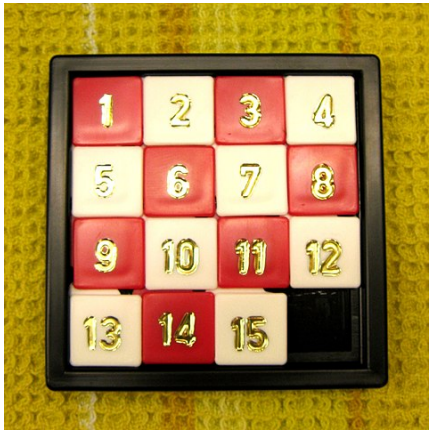
Ejemplos

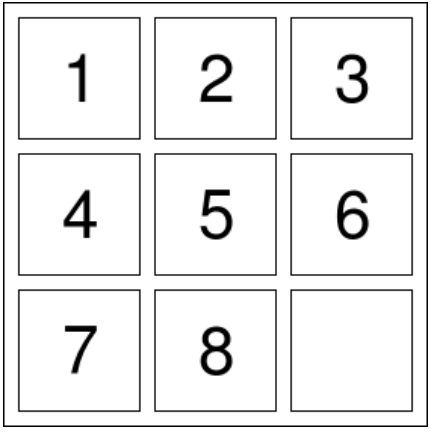




(a) *Knapsack problem*, (b) *8-queens Problem*.

Ejemplos





(a) 15-puzzle, (b) 8-puzzle.

Ejemplo

- Representación:
  Definimos un espacio de estados o configuraciones

$$S = \{\mathbf{x} = (a_0, a_1, \ldots, a_8) : \ \mathbf{x} \text{ es permutación de } (0, 1, 2, 3, 4, 5, 6, 7, 8)\}.$$

- Estado objetivo:

$$\mathbf{x}^* = (1, 2, 3, 8, 0, 4, 7, 6, 5).$$

- Función objetivo: Definimos como función objetivo alguna métrica que nos indique cuánto falta para llegar al estado objetivo.

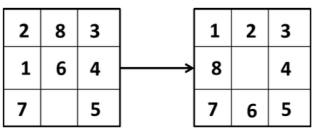
$$f(\mathbf{x}) = (\mathbf{x} \neq \mathbf{x}^*).sum()$$

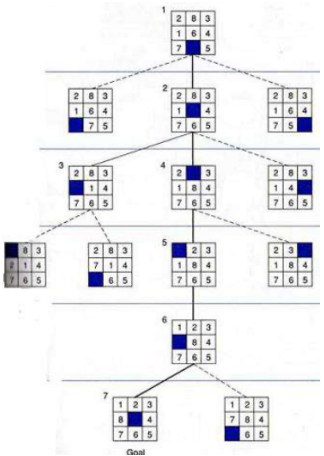
ó

$$f(\mathbf{x}) = \sum_{i=0}^{8} || \text{pos}_{x,y}(\text{argwhere}(\mathbf{x}, i)) - \text{pos}_{x,y}(\text{argwhere}(\mathbf{x}^*, i))||_1.$$

Ejemplo

- Árbol de configuraciones:





Esquemas de Búsqueda

En optimización discreta o combinatoria, es común tratar de buscar soluciones en espacios altamente grandes. Una forma de ordenar o sistematizar la búsqueda de soluciones de un problema de optimización es la siguiente:

1. configurar el conjunto de soluciones dentro de una estructura de grafo G,
2. utilizar estrategias de búsqueda dentro de G.

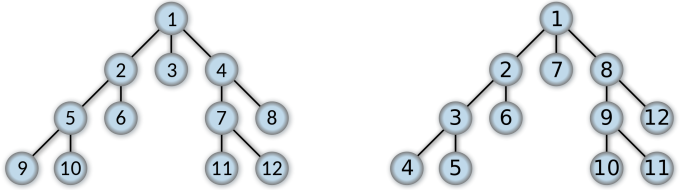
Existen diferentes esquemas de búsqueda en grafos. Mencionamos las más importantes:

- BFS (*breadth-first search*)
- DFS (*depth-first search*)
- *Best-first search* o *Greedy search*
- *Beam search*
- *Backtracking*.

Esquemas de Búsqueda

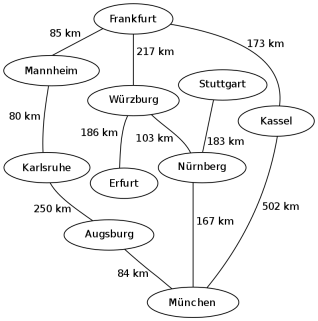
BFS: La **búsqueda en amplitud** (BFS) comienza en la raíz del árbol y explora todos los nodos en la profundidad actual antes de pasar a los nodos del siguiente nivel de profundidad. Se necesita memoria adicional, generalmente una cola, para realizar un seguimiento de los nodos secundarios que se encontraron pero que aún no se exploraron.

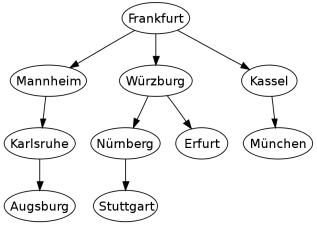
DFS: La **búsqueda en profundidad** (DFS) comienza en el nodo raíz (seleccionando algún nodo arbitrario como nodo raíz en el caso de un grafo) y explora en la medida de lo posible a lo largo de cada rama antes de retroceder.

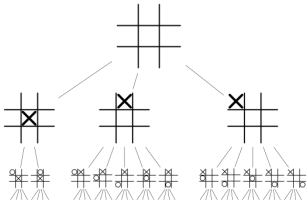


Estrategias de búsqueda: (a) BFS, (b) DFS.

Esquemas de Búsqueda



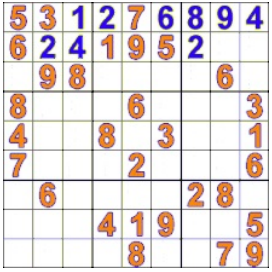




Estrategias de búsqueda: (a) BFS, (b) DFS.

Esquemas de Búsqueda

Backtracking: El **retroceso** o *backtracking* es un algoritmo general para encontrar soluciones a algunos problemas computacionales, en particular problemas de satisfacción de restricciones, que construye gradualmente candidatos a las soluciones y abandona a un candidato ("retrocesos") tan pronto como determina que el candidato no puede ser completado a un valor válido. solución.

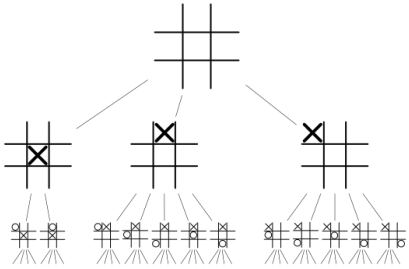


Ejemplo de uso de *backtracking*.

Búsqueda Local

La **búsqueda local** es un método heurístico para resolver problemas de optimización.

La búsqueda local se puede utilizar en problemas que se pueden formular como encontrar una solución que maximice un criterio entre varias soluciones candidatas. Los algoritmos de búsqueda local se mueven de una solución a otra en el espacio de las soluciones candidatas (el espacio de búsqueda) mediante la aplicación de cambios locales, hasta que se encuentra una solución considerada óptima o transcurre un límite de tiempo.



Búsqueda Local

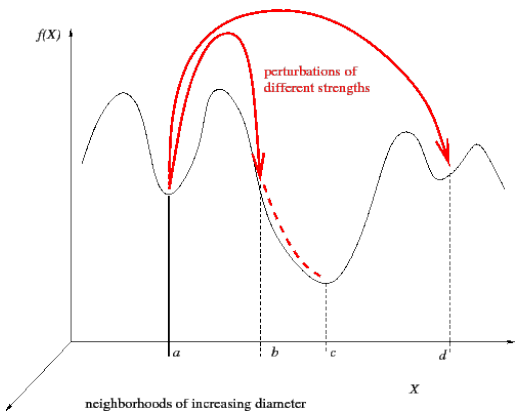
En el análisis numérico, la *escalada* o *hill-climbing* es una técnica de optimización matemática que pertenece a la familia de la búsqueda local. Es un algoritmo iterativo que comienza con una solución arbitraria a un problema, luego intenta encontrar una mejor solución haciendo un cambio incremental en la solución. Si el cambio produce una mejor solución, se realiza otro cambio incremental en la nueva solución y así sucesivamente hasta que no se puedan encontrar más mejoras.

En este tipo de técnicas se suele aplicar una estrategia *greedy*. Dentro de las soluciones candidatas (una muestra de todas las soluciones posibles), se elige aquella o aquellas que maximizan la función objetivo, dentro de este conjunto de soluciones cantidades.

Aunque esta es una estrategia que reduce de manera considerable el espacio de búsqueda, suele conducir a soluciones sub-óptimas.

- Repetir experimentos con diferentes tipos puntos iniciales.
- Aplicar estrategias de salto.

Búsqueda Local



Existen muchas variantes y mejoras para la búsqueda local:

- *local beam search*,
- *hill-climbing* y *local search* estocásticos,
- búsqueda tabú o *tabu-search*.
- algoritmos de estimación de distribuciones,
- algoritmos y métodos evolutivos.

Búsqueda Local

```
function HILL-CLIMBING( problem) return a state that is a local maximum
    input: problem, a problem
    local variables: current, a node.
                     neighbor, a node.

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest valued successor of current
        if VALUE [neighbor] ≤ VALUE[current]
            then return STATE[current]
        current ← neighbor
```

Algoritmo de *Hill-climbing* o búsqueda local.