Optimización Y PSR

Dra. Ing. Selva S. Rivera

Profesora Titular

OPTIMIZACIÓN
algunas aplicaciones

- Crear un plan de mínimo costo para **repartir mercancías de clientes**
- Realizar una **asignación óptima de trabajadores a un conjunto de tareas**
- Encontrar una secuencia óptima de trabajos en una **cadena de producción**
- Encontrar una distribución de **tripulaciones de aviones con mínimo costo**
- Encontrar la configuración óptima en una red de **telecomnicaciones**

OPTIMIZACIÓN COMBINATORIA

- Existe un conjunto de objetos

(clientes, tareas, trabajos, etc.) que se deben colocar en distintas posiciones

- Existe un grupo de lugares en los cuales se deben colocar dichos objetos

- Cada colocación de objetos en un lugar determinado se denomina *"configuración"*

Se busca la mejor configuración, según sea el caso (maximizar o minimizar el valor requerido), para poder resolver un problema determinado.

Búsqueda local

No importa el camino al objetivo

- **se empieza de una configuración inicial**

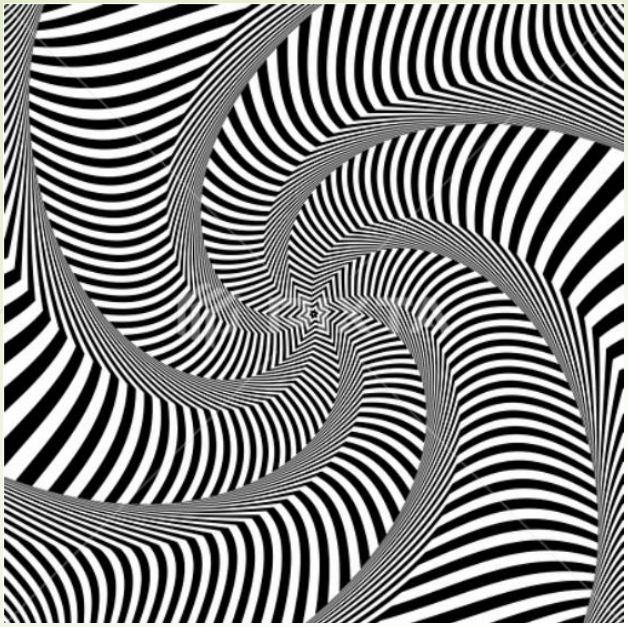
  (generalmente aleatoria) y se hacen pequeños cambios (a través de operadores)hasta alcanzar un estado desde el cual no se puede alcanzar ningún estado mejor.

- Las técnicas de BL son propensas a encontrar **óptimos locales** que no son la mejor solución posible. El óptimo global es generalmente imposible de alcanzar en un tiempo limitado, por el tamaño del espacio de soluciones.

- Estos algoritmos **no hacen una exploración sistemática**

Búsqueda local

- Hay una **función heurística** que evalúa la calidad de la solución, pero que no está necesariamente ligada al costo del camino.
- La función heurística se usará para **podar el espacio de búsqueda** (soluciones que no merecen la pena explorar).
- **No se suele** guardar historia del camino recorrido (el gasto de memoria es mínimo).



Búsqueda local

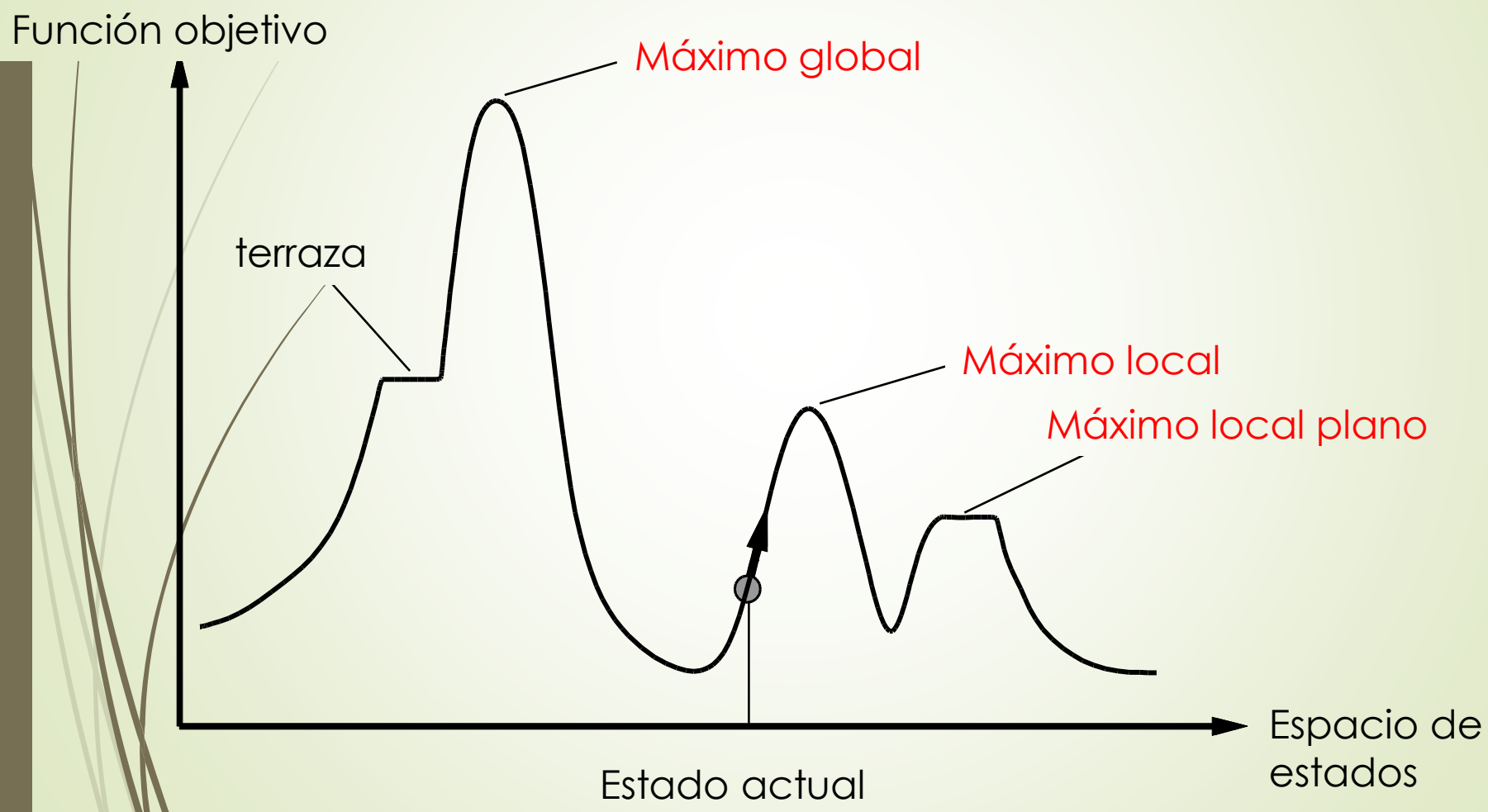
Dos ventajas claves:

1- usan muy poca memoria (por lo general una cantidad constante)

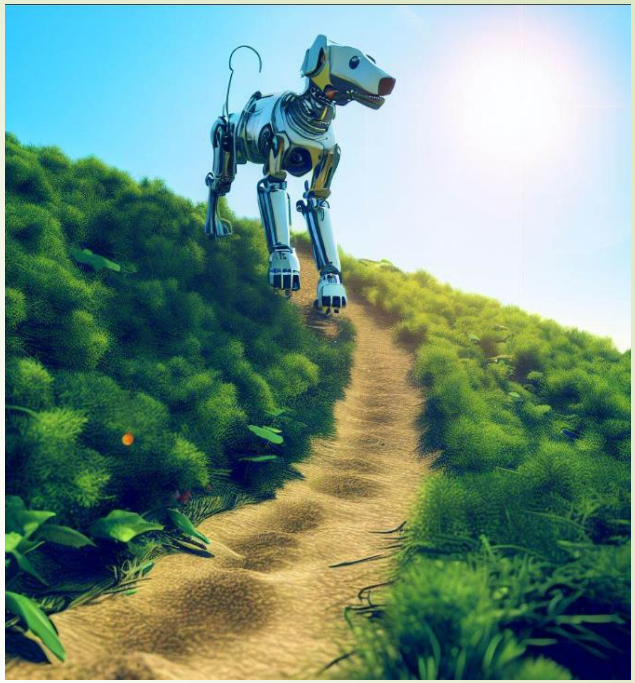
2- se pueden encontrar soluciones razonables en espacios de estados grandes o infinitos (continuos) para los cuales son inadecuados los algoritmos sistemáticos.

Paisaje del espacio de estados

Los métodos usados en BL son conocidos como **meta-heurísticas** u optimización local.



Ascensión de colinas



Utiliza una meta-heurística, que consiste en avanzar continuamente en dirección del valor creciente en el espacio de estados.

Es simplemente un bucle que constantemente se desplaza en la dirección de un valor ascendente. Como no mantiene un árbol de búsqueda, **la estructura de datos del nodo sólo tiene que registrar el estado actual y su valor de función objetivo**.

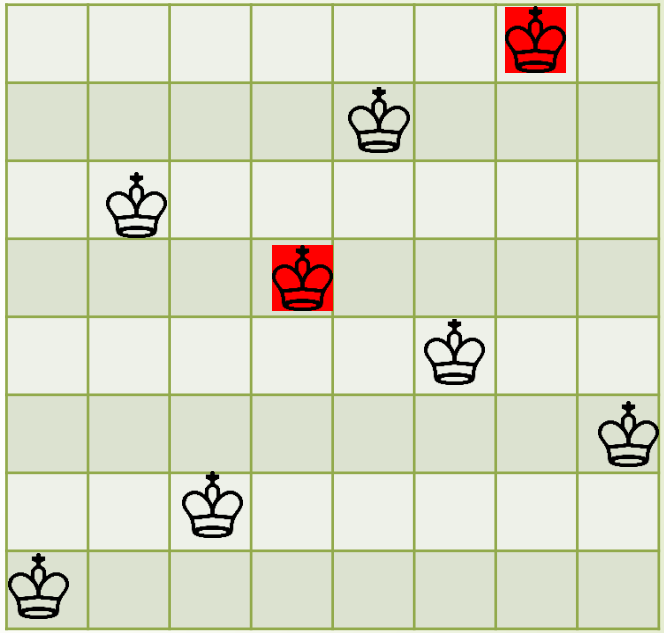
Suele llamarse a esta búsqueda **algoritmo voraz local**, porque toma un estado vecino "bueno" sin pensar en la próxima acción.

Ascensión de colinas

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	♕	13	16	13	16
♕	14	17	15	♕	14	16	16
17	♕	16	18	15	♕	15	♕
18	14	♕	15	15	14	♕	16
14	14	13	17	12	14	12	18

Estado a

h= nº de pares de reinas que se atacan directa o indirectamente



Estado b

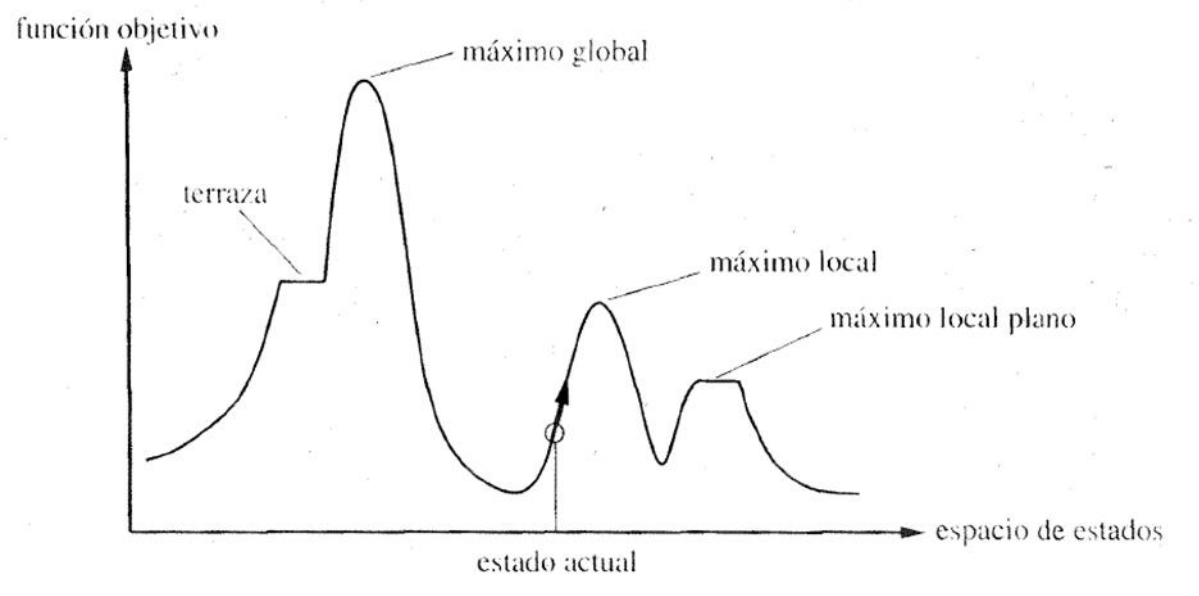
Un estado de 8-reinas con una heurística de estimación de costos h = 17, mostrando al valor de h para cada sucesor posible obtenido al mover una reina dentro de su columna. Los mejores movimientos están marcados.

Un mínimo local en el espacio de estados de las 8- reinas; el estado tiene h = 1

Ascensión de colinas

Generalmente se atasca sin obtener una solución al problema.

Máximo local: Es una cima cuya altura es superior a la de sus estados vecinos, pero que es inferior a la cima más alta en todo el espacio de estados. Una vez que ha alcanzado un máximo local, el algoritmo se detiene, a pesar de que no se haya llegado a una solución, o al menos a una solución óptima.

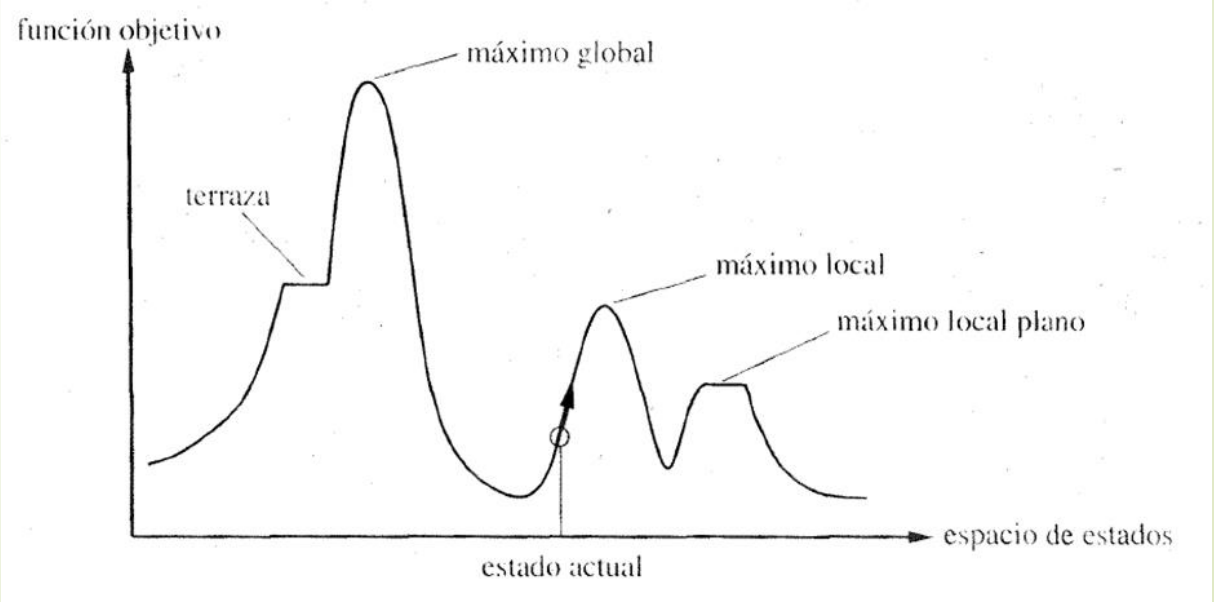


Ascensión de colinas

Meseta: Son áreas del espacio de estados en donde la función de evaluación es plana. Puede ser un **máximo local plano** o una **terraza**. La búsqueda podría ser incapaz de encontrar su camino.

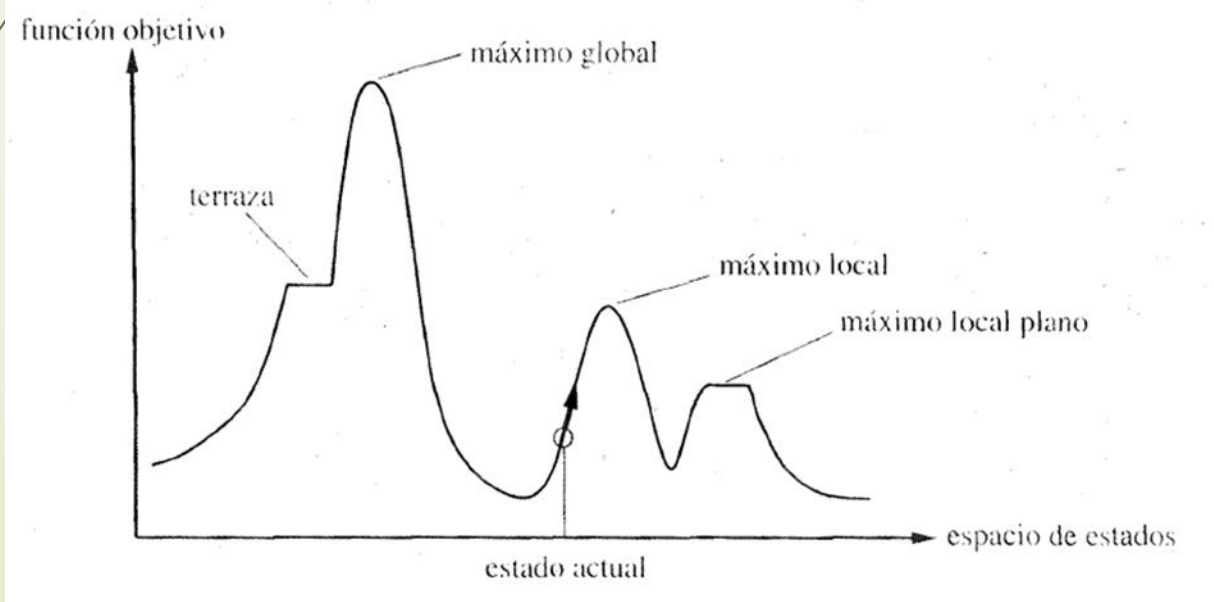
Terraza: Es un tipo de meseta, en la que luego de la planicie hay vecinos con valores ascendentes.

Máximo local plano: Es un tipo de meseta en la que a ambos lados de la planicie hay vecinos hay vecinos con valores Descendentes.

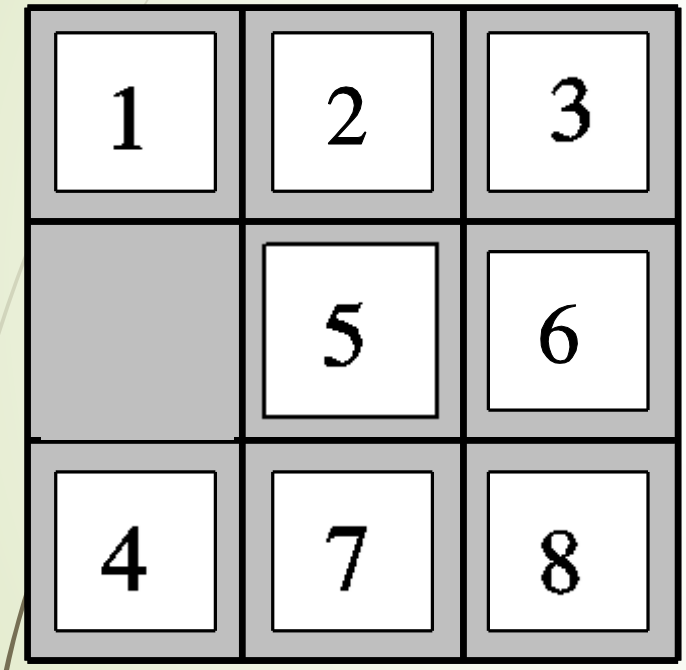


Ascensión de colinas

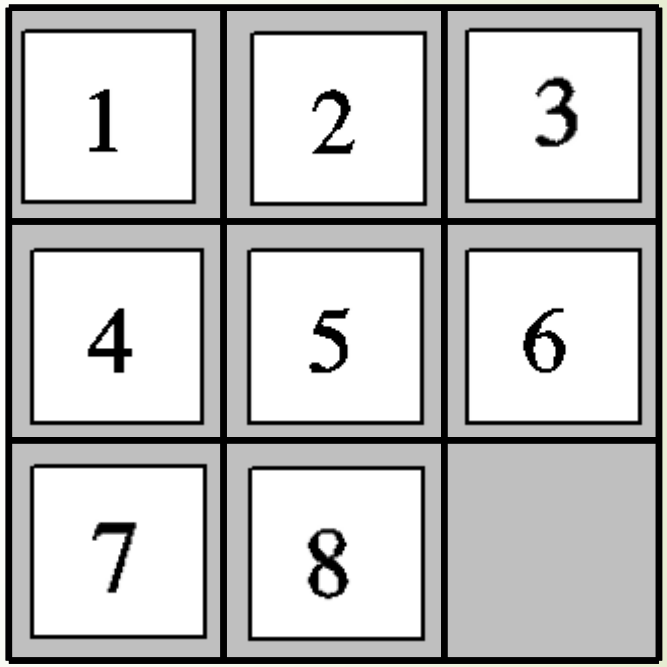
Crestas: Son laderas con pendientes muy pronunciadas, por lo que es fácil para una búsqueda llegar a la cima. Sin embargo, puede suceder que la pendiente se aproxime demasiado gradualmente a un pico y la búsqueda oscilará de un lado al otro, obteniendo muy poco o ningún avance.



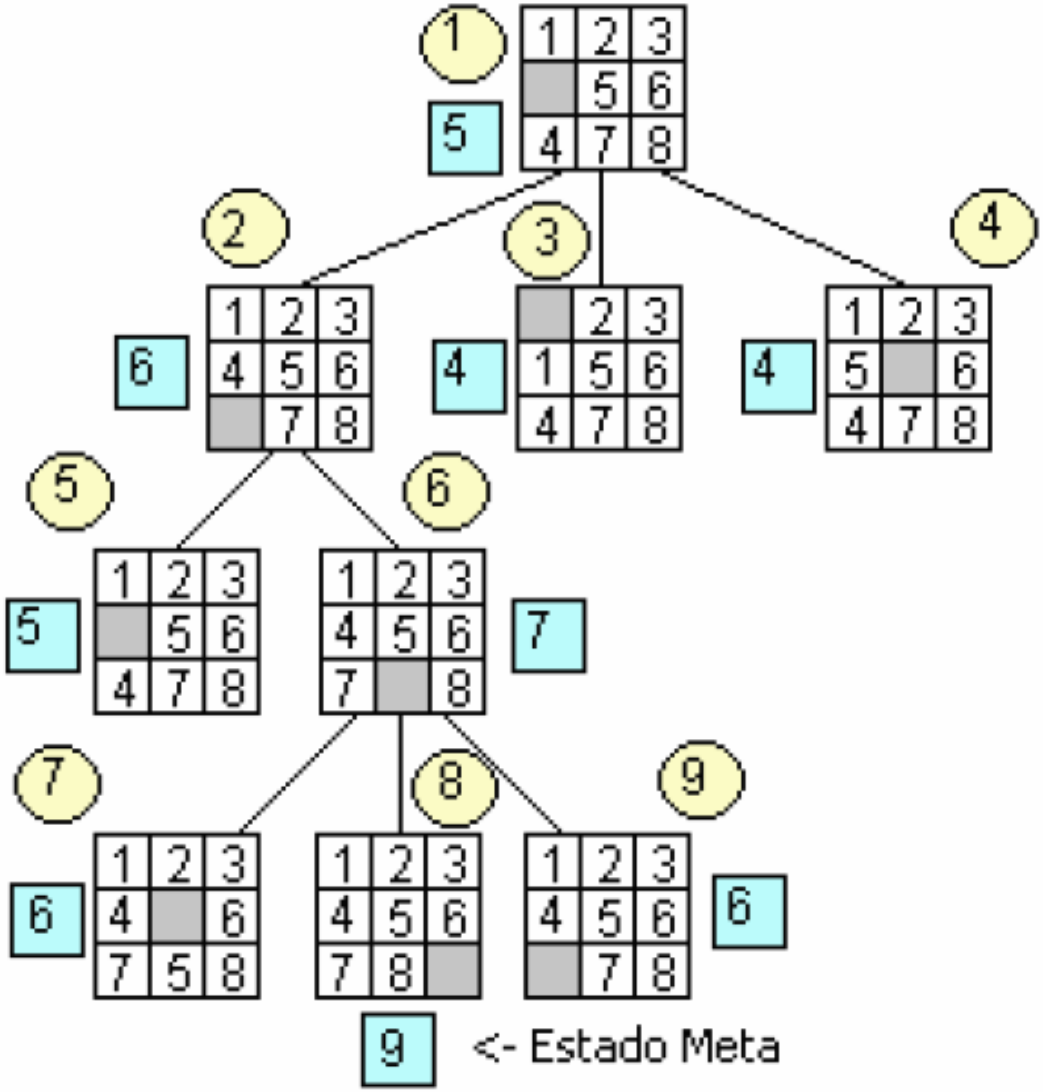
Juego 8-puzzle



Estado Inicial



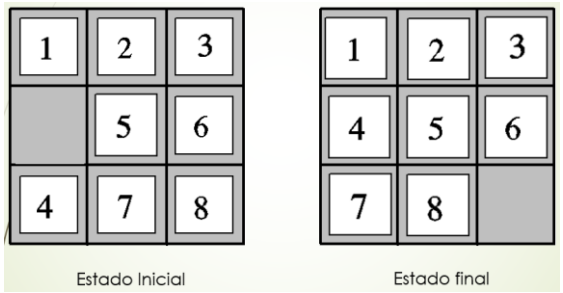
Estado final

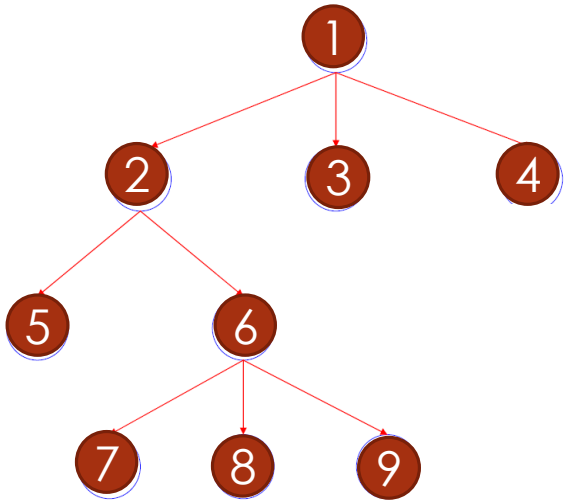


f(nodo)=número de casillas bien colocadas (maximizo)

= Secuencia de estados generados

= Valor que devuelve función f(nodo)





Árbol simplificado

RECOCIDO SIMULADO

Se caracteriza por un criterio de aceptación de soluciones vecinas que se adapta a lo largo de su ejecución.

Se creó para minimizar funciones de costo pero también se utiliza ampliamente en problemas de maximización.

RECOCIDO SIMULADO
características

- Es **estocástico**, ya que se elige un sucesor de entre todos los disponibles según una distribución de probabilidad.
- Se inspira en el proceso físico de enfriamiento controlado.
- A menor temperatura menor probabilidad de elegir sucesores peores.
- Estrategia de enfriamiento (depende del número de iteraciones)

RECOCIDO SIMULADO
aplicaciones

- **Problemas de optimización combinatoria** (configuración óptima de elementos) y continua (punto óptimo en un espacio N-dimensional)
- **Problemas grandes** en los que el óptimo está rodeado de **muchos óptimos locales**.
- Problemas en los que encontrar una heurística discriminante es difícil
- Aplicaciones típicas: el problema del viajante, **diseño de circuitos VLSI**, programación de una fábrica, etc.

RECOCIDO SIMULADO metodología

Se identifican los elementos del problema de búsqueda con los del problema físico:

Temperatura: parámetro de control principal

Energía: función heurística sobre la calidad de la solución f(n)

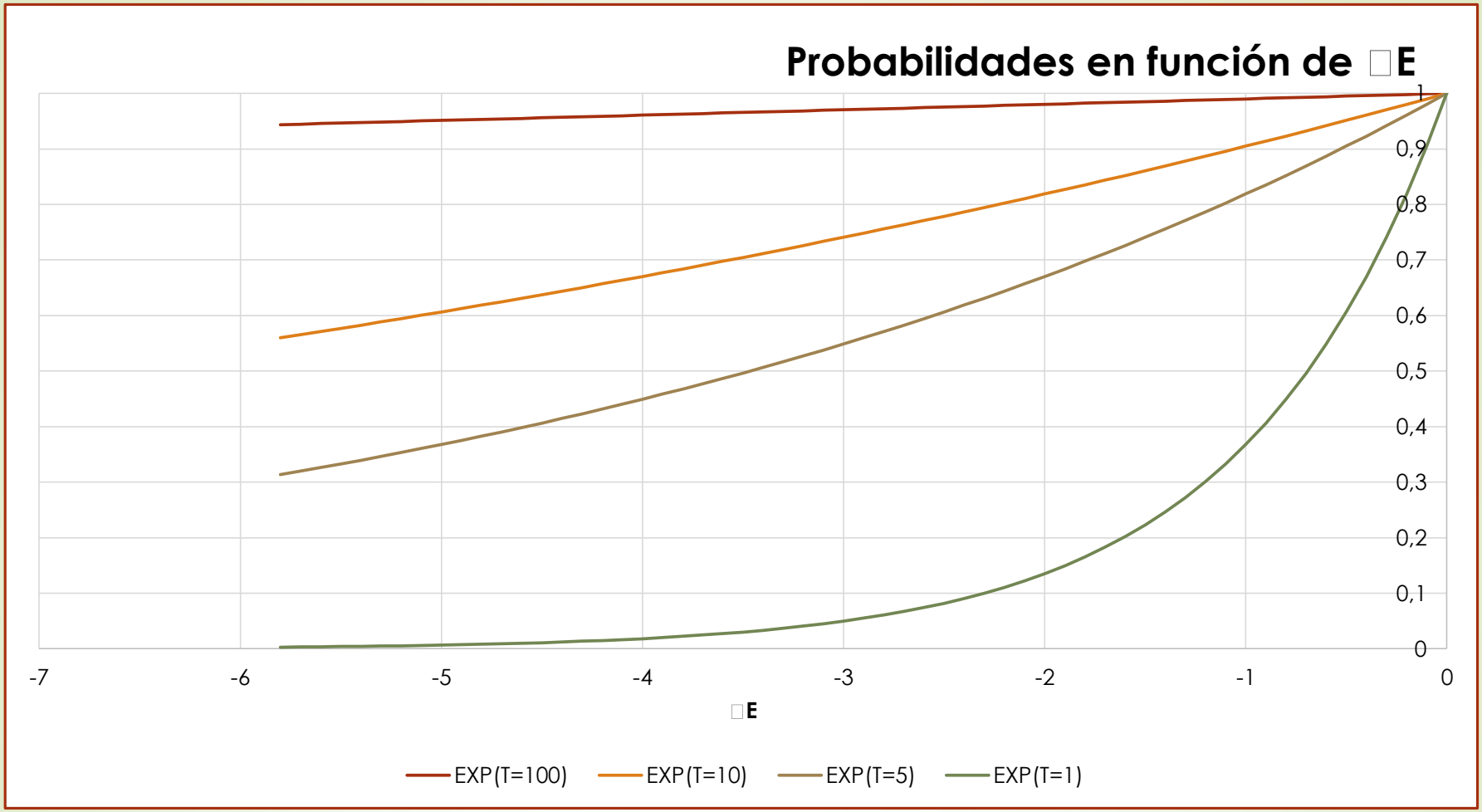
Función que determina la elección de un estado sucesor: F(Δf ,T), depende de la temperatura y la diferencia entre la calidad de los nodos. A menor temperatura menor probabilidad de elegir sucesores peores

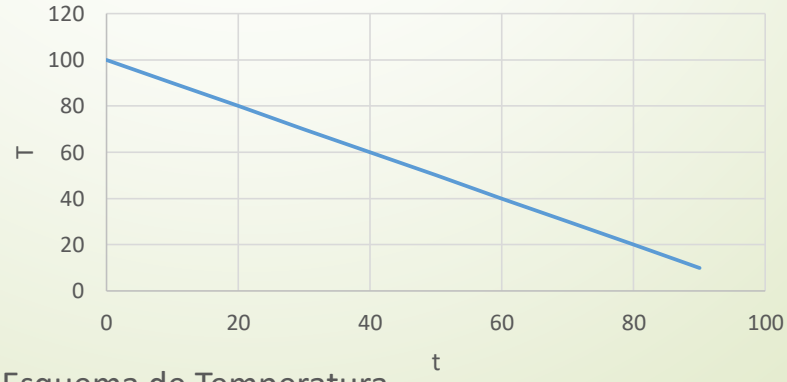
Estrategia de enfriamiento o esquema: parámetros que determinan el número de iteraciones de la búsqueda, disminución de la temperatura y número de pasos para cada temperatura.

Pseudocódigo

INPUT (T_0, α, L, T_f)
$T \leftarrow T_0$
$S_{act} \leftarrow$ Genera_Solución_Inicial
WHILE $T \geq T_f$ DO
    BEGIN
        FOR count ←1 TO L(T) DO
        BEGIN
        $S_{cand} \leftarrow$Genera_Vecino_Aleatorio(S_{act})
        $\delta \leftarrow$Cost(S_{cand})-Cost(S_{act})
        IF (U(0,1)<$e^{(-\delta/T)}$) OR (δ<0)
        THEN $S_{act} \leftarrow S_{cand}$
        END
    $T \leftarrow \alpha(T)$
    END

{Devuelve el mejor S_{act} visitado}





Esquema de Temperatura

TEMPERATURA INICIAL (propuesto por medina en 2001)

- Se propone una temperatura inicial arbitraria
- Se realizan algunas iteraciones y
  - si el porcentaje de aceptaciones es inferior al 20% se duplica la temp. inicial,
  - si el porcentaje de aceptaciones supera el 40% la temp. inicial se divide por 2

Temperatura de enfriamiento

- Se utiliza una velocidad geométrica de decrecimiento

$$t_{i+1} = \propto \ . \ \ t_i \qquad i = 0, 1, \dots, n$$

se aconseja $\propto \in [0{,}8 \ ; \ \ 0{,}99]$

Temperatura final

- Teóricamente $T_f = 0$
  - Inviable para decrecimientos geométricos
  - Probabilidad de aceptar soluciones peores es casi nula antes de llegar a $T_f = 0$
- Opciones para detener el algoritmo
  - Cuando no existan mejoras tras un determinado nº de iteraciones
  - Al alcanzar una fracción de la temperatura inicial

  (ej. $T_f = 0{,}01 \ \ T_0$)

Búsqueda por Haz local

- Comienza con *k* estados generados aleatoriamente
- En cada paso se generan **todos** los sucesores de cada uno de los *k* estados.
- Si alguno de estos sucesores es objetivo, el algoritmo para.
- Se seleccionan los ***k* mejores** sucesores de la lista completa y se repite.

Búsqueda de has estocástica

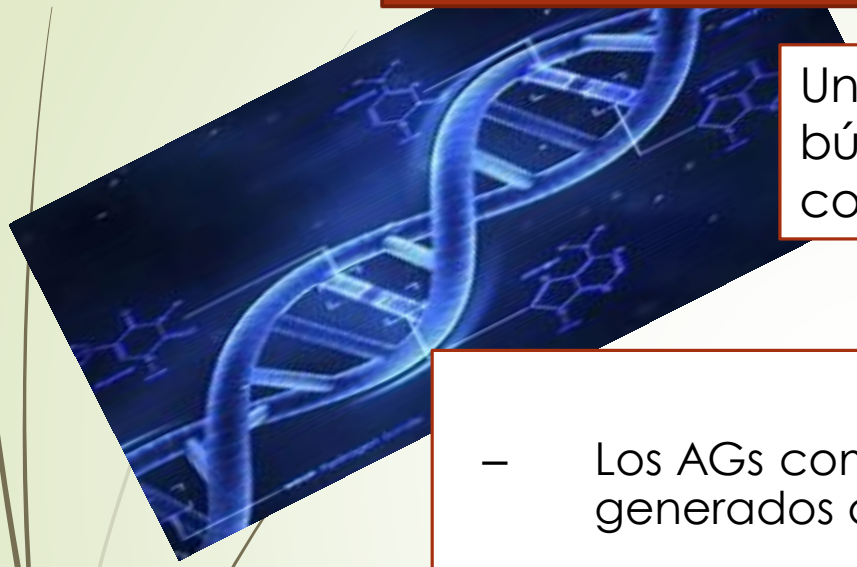
En vez de elegir los k mejores sucesores, escoge a k sucesores aleatoriamente (probabilidad de elegir a un sucesor como una función creciente del valor de la función de idoneidad)

Parecido con el proceso de selección natural: los sucesores (descendientes) de un estado (organismo) pueblan la siguiente generación según su valor (idoneidad, salud, adaptabilidad).

Algoritmos genéticos

- Modelan el proceso de evolución como una sucesión de frecuentes cambios en los genes, con soluciones análogas a cromosomas.
- Trabajan con una población de cadenas binarias para la representación del problema
- El espacio de soluciones posibles es explorado aplicando transformaciones a éstas soluciones candidatas (cruce, mutación)
- Como método de selección emplean el mecanismo de la ruleta.
- Son muy flexibles y se pueden hibridar fácilmente con otros paradigmas y enfoques.

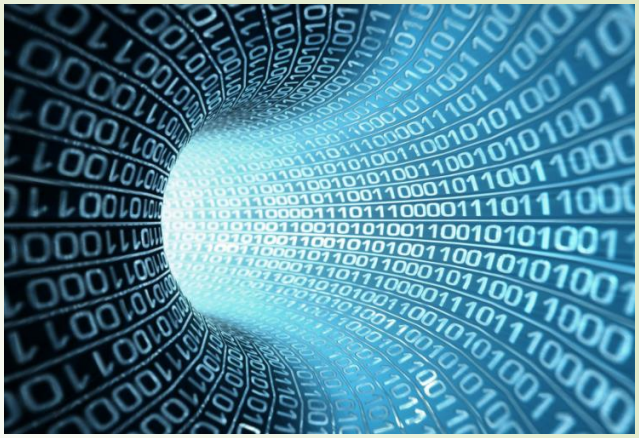
Algoritmos genéticos



Un algoritmo genético (AG) es una búsqueda estocástica en que se combinan dos estados padres.

- Los AGs comienzan con un conjunto de *k* estados generados aleatoriamente, llamados **población.**
- Los estados corresponden a **individuos**.
- Una **función de idoneidad/calidad/evaluación** indica/mide la bondad/calidad de los estados.
- Combinando buenos estados se obtienen estados mejores.

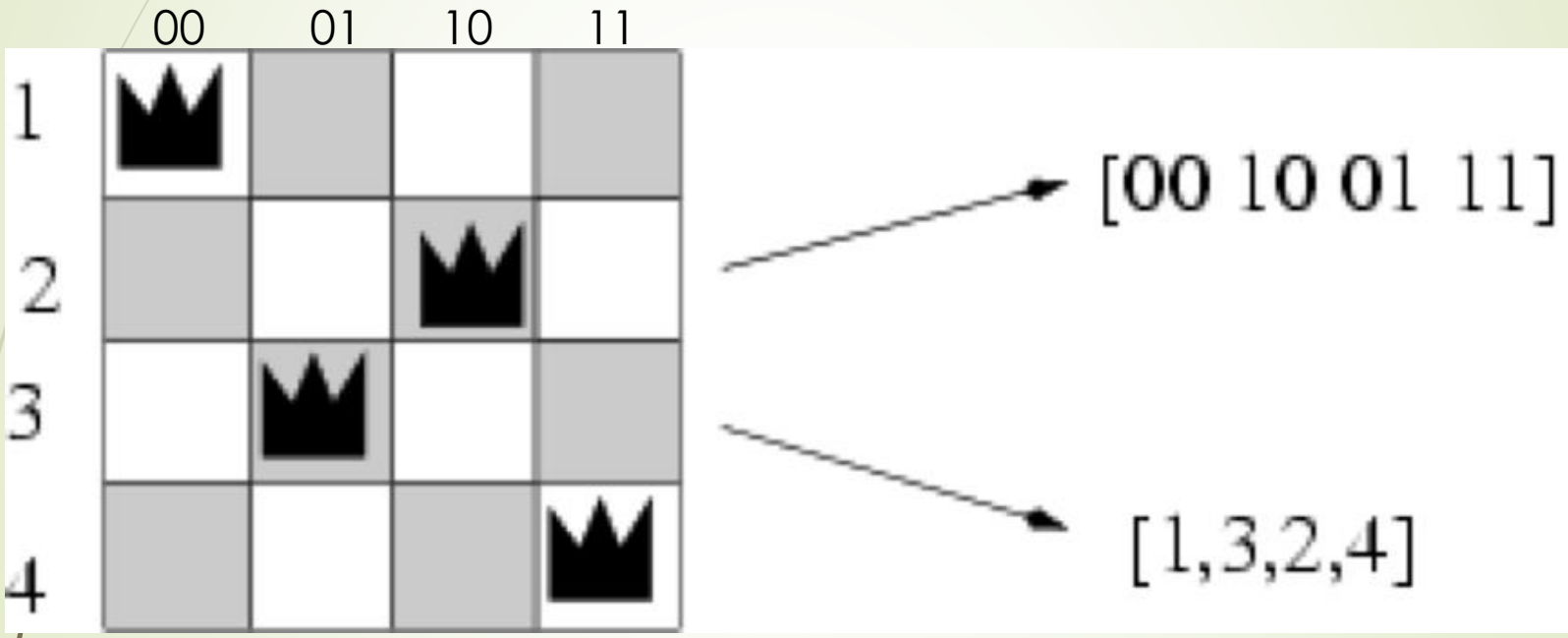
codificación



- Cada individuo se representa como **una cadena** sobre un alfabeto finito (comúnmente, una cadena de 0s y 1s).

- La codificación define el **tamaño del espacio de búsqueda** y el tipo de **operadores** de combinación necesarios.

codificación

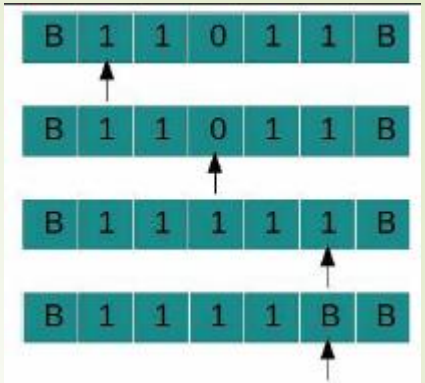


Codificado por columna

Función de idoneidad

- En la producción de la siguiente generación de estados, para cada estado se calcula la **función de idoneidad**.
- Una función de idoneidad debería devolver valores más altos para estados mejores.
  - Para el problema de las *n* reinas se puede utilizar el número de pares de reinas que no se atacan.
  - En el caso de 4 reinas, la función de idoneidad tiene un valor de 6 para la mejor solución.

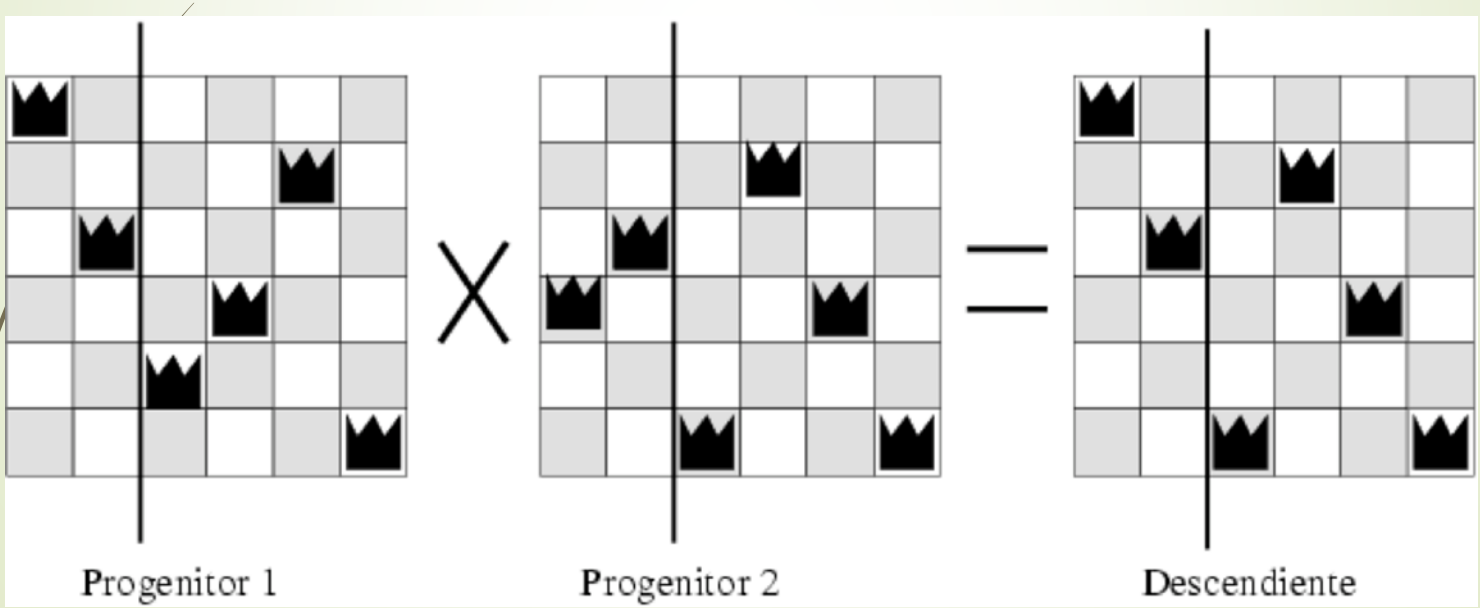
cruce



- La combinación de individuos se realiza mediante operadores de **cruce**.
- El operador básico es el cruce por un punto:
  - Se elige aleatoriamente un punto de la codificación.
  - Los descendientes se crean cruzando las cadenas paternales en el punto de cruce.

cruce

- En el ejemplo, el hijo consigue las dos primeras columnas del primer padre y las columnas restantes del segundo padre.



cruce

- Cada paso es una generación de individuos.
  - El tamaño de la **población** se mantiene constante (N).
- Existen otras posibilidades:
  - Cruce en dos puntos
  - Intercambio aleatorio de bits
  - Operadores *ad hoc* según la representación

mutación

- Analogía con la combinación de genes:
  - A veces la información de parte de ellos cambia aleatoriamente.
- La **mutación** consiste en cambiar el signo de cada bit (si se trata con una cadena binaria) con cierta probabilidad.

combinación

- Los AGs comienzan con una población de *k* estados generados **aleatoriamente**.

- Para pasar a la siguiente población debemos elegir qué individuos se han de combinar (*población intermedia*), por ejemplo:
  - Los individuos se eligen con probabilidad proporcional a su función de idoneidad.
  - Se establecen N *torneos* aleatorios entre parejas de individuos y se eligen los que ganan en cada torneo.
  - Se suelen eligir dos veces individuos con las mejores funciones de idoneidad.

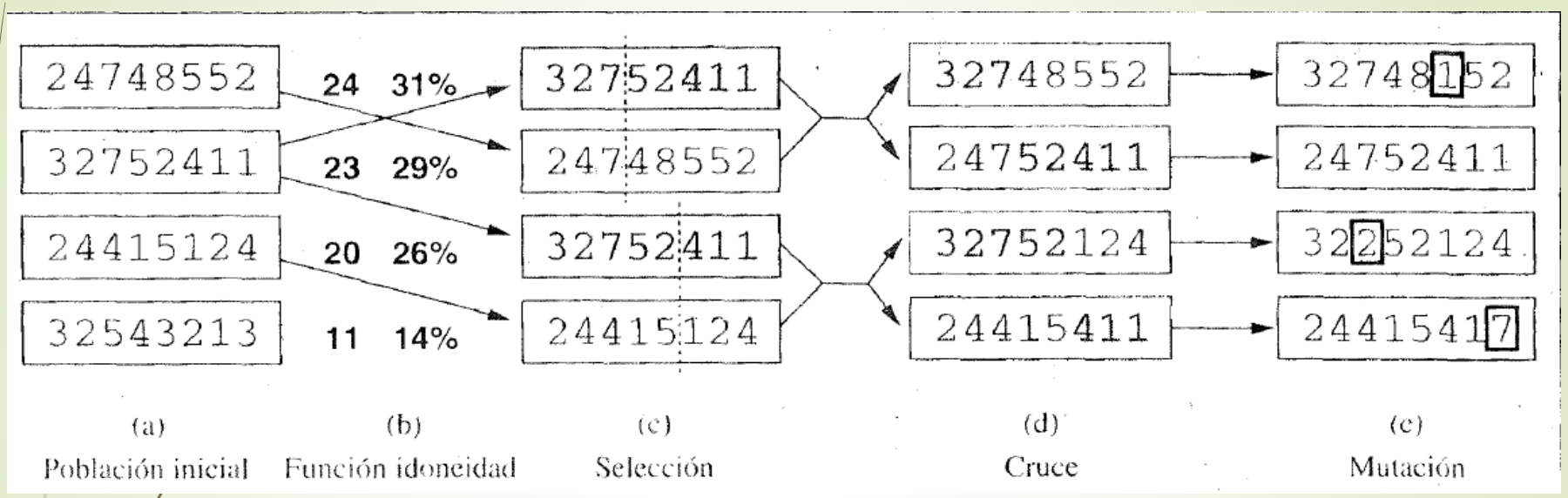
- En la *población intermedia,* siempre habrá individuos que aparezcan más de una vez e individuos que no aparezcan.

Pasos de ejecución

1. Se escogen N individuos de la población actual para la población intermedia.
2. Se emparejan los individuos y para cada pareja:
   - con una probabilidad (P_cruce) se aplica el operador de cruce a los individuos y se obtienen dos nuevos individuos;
   - con una probabilidad (P_mutación) se mutan los nuevos individuos.
3. Estos individuos forman la nueva población.

- El procedimiento se itera hasta que la población converge o pasa un número específico de iteraciones.

Ejemplo: 8 reinas



Función idoneidad: número de pares de reinas que no se atacan (min. = 0, máx. = 28)

24/(24+23+20+11) = 31%

23/(24+23+20+11) = 29%

Etc.

Ejemplo: 8 reinas

- En (a): se muestra una población de 4 cadenas de 8 dígitos que representan estados de 8 reinas.
- En (b)-(e): se muestra la producción de la siguiente generación de estados.

Ejemplo: 8 reinas

- En (b): para cada estado se calcula la **función idoneidad**.
- En esta variante particular del AG, la probabilidad de ser elegido para la reproducción es directamente proporcional a la idoneidad.

Ejemplo: 8 reinas

- En (c): se seleccionan 2 pares, de manera aleatoria, para la reproducción, de acuerdo con las probabilidades en (b).
- Se puede notar que un individuo se selecciona 2 veces y uno ninguna.

Ejemplo: 8 reinas

- En (c): para que cada par se aparee, se elige aleatoriamente un punto de **cruce** de las posiciones en la cadena.
- En (d): los descendientes se crean cruzando las cadenas paternales en el punto de cruce.

Ejemplo: 8 reinas

- En (e): cada posición está sujeta a la mutación aleatoria con una pequeña probabilidad independiente.

- Un dígito fue transformado en 3 casos.

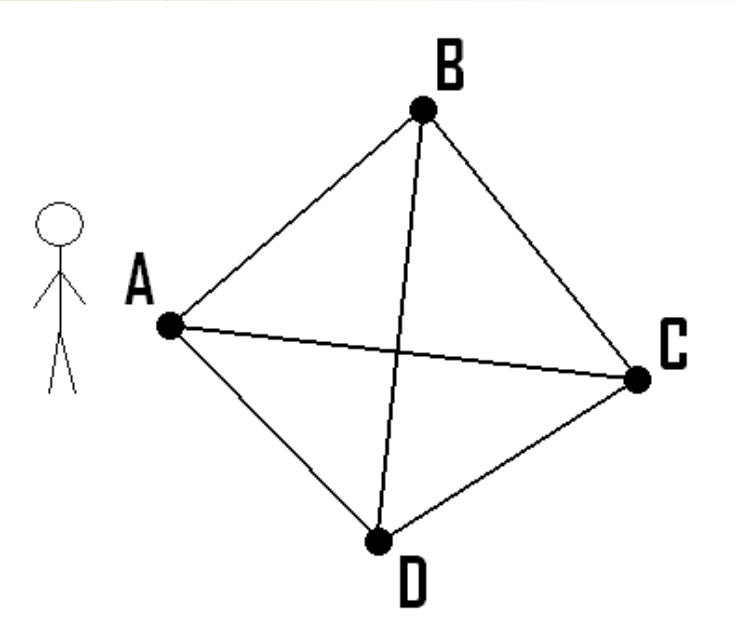
ventajas y desventajas

- Son aplicables casi a cualquier tipo de problema.
- Permiten abordar problemas para los que no se dispone de una función heurística adecuada.
- Por lo general serán peores que un algoritmo clásico con una buena heurística.
- Dificultades:
  - codificación de los estados
  - determinación de los parámetros del algoritmo:
    - tamaño de la población
    - iteraciones
    - probabilidad de cruce y mutación

ventajas y desventajas

- Dificultades:
  - Si la función a optimizar tiene muchos máximos/mínimos locales se requerirán más iteraciones del algoritmo para "asegurar" el máximo/mínimo global.
  - Si la función a optimizar contiene varios puntos muy cercanos en valor al óptimo, solamente podemos asegurar que encontraremos uno de ellos (no necesariamente el óptimo).

Problema del viajante



- Si un viajante parte de la ciudad A y las distancias a todas las demás ciudades son conocidas:

¿cuál es la ruta óptima que debe elegir para visitar todas las ciudades (una sóla vez) y volver a la ciudad de partida?.

Problema del viajante

- Representación

  si consideramos 9 ciudades una posible representación es por orden de visita:

  3 – 2 – 5 – 4 – 7 – 1 – 6 – 9 – 8

- Cruce

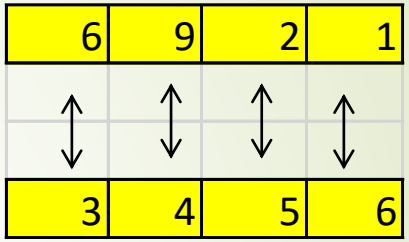
  1) seleccionar una subcadena en forma aleatoria

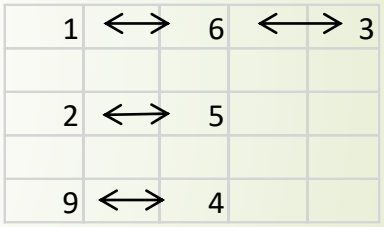
1	2	3	4	5	6	7	8	9	padre 1
5	4	6	9	2	1	7	8	3	padre 2

Problema del viajante

2) Intercambio y mapeo

1	2	6	9	2	1	7	8	9	proto-hijo 1
5	4	3	4	5	6	7	8	3	proto-hijo 2





Descendencia

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

Problema del viajante

Mutación

Ejemplo 1

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

Ejemplo 2

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

Problemas CON satisfacción de restricciones (PSR)

- Son problemas de búsqueda cuyos estados y test objetivo forman una representación muy simple, estándar y estructurada.
- Los algoritmos de búsqueda se pueden definir aprovechándose de la estructura de los estados y utilizan las heurísticas de propósito general más que heurísticas específicas del problema para así permitir la solución de problemas grandes.
- La representación estándar del test objetivo revela la estructura del problema.

PROBLEMAS CON SATISFACCIÓN DE RESTRICCIONES

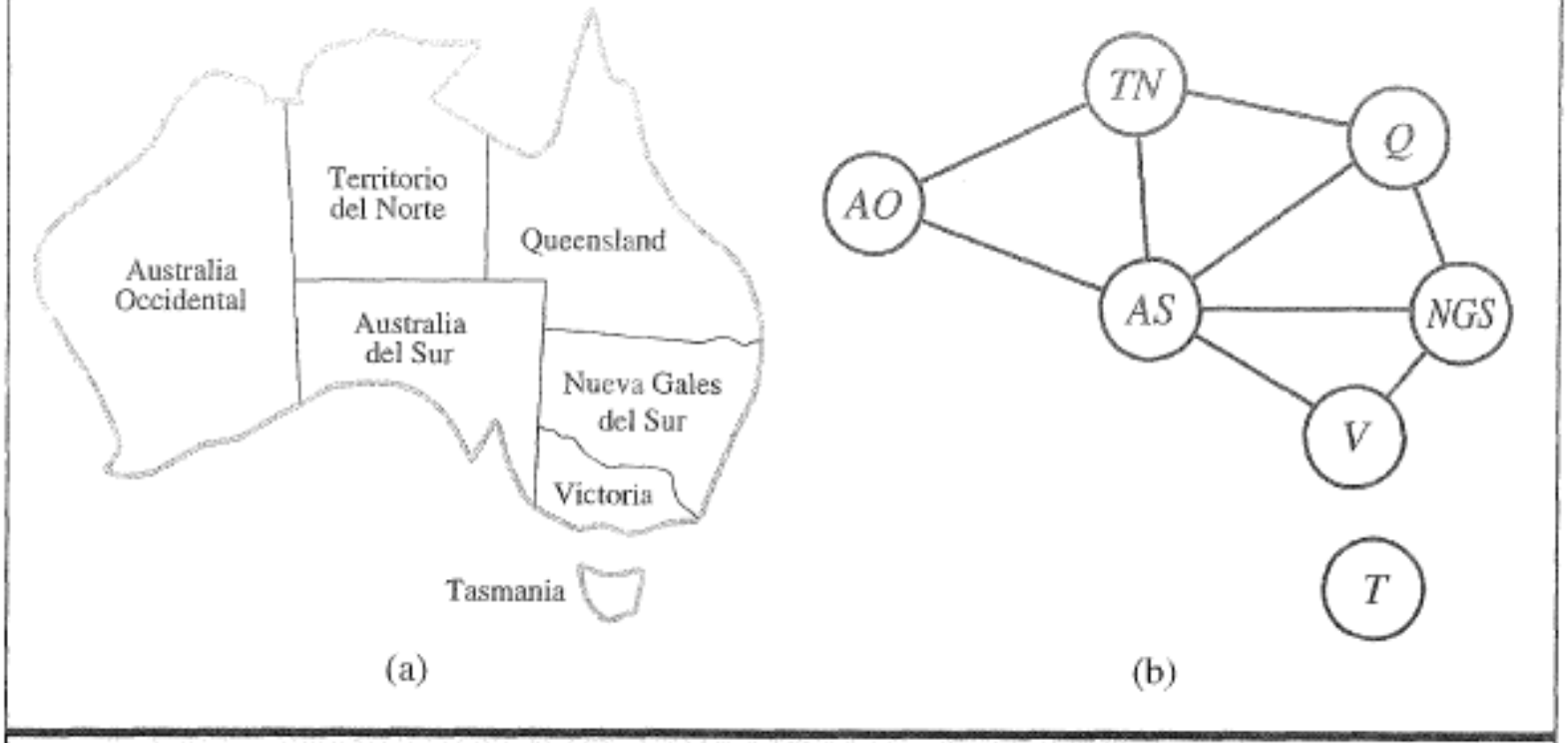
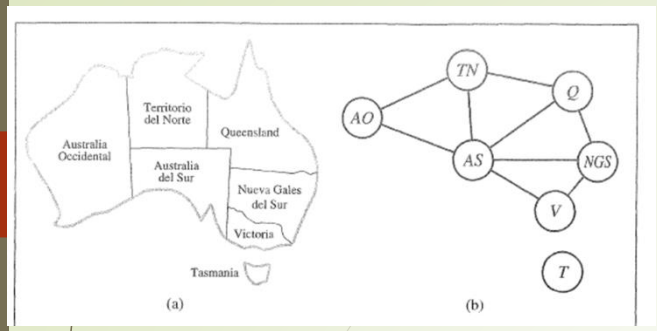


Figura 5.1 (a) Los estados y territorios principales de Australia. Colorear este mapa puede verse como un problema de satisfacción de restricciones. El objetivo es asignar colores a cada región de modo que ninguna de las regiones vecinas tengan el mismo color. (b) El problema del coloreo del mapa representado como un grafo de restricciones.



PROBLEMAS CON SATISFACCIÓN DE RESTRICCIONES

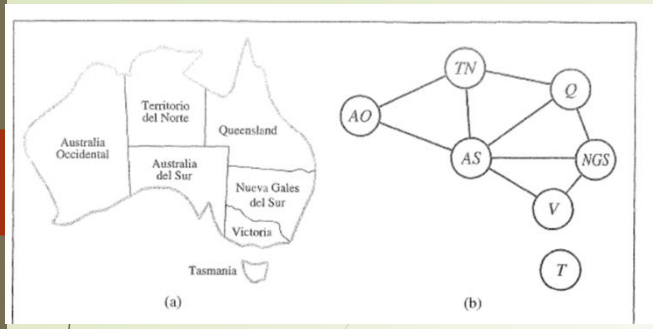
- Un PSR está definido por
  - **Conjunto de variables** $x_1, x_2, \ldots, x_n$

    (AO, TN, Q, NGS, V, AS y T)
  - **Dominio de valores posibles** $D_1, D_2, \ldots, D_n$

    {rojo, verde, azul}
  - **Conjunto de restricciones** $C_1, C_2, \ldots, C_n$

    (AO ≠ TN)

Un PSR se puede visualizar como un **grafo de restricciones**. Los nodos corresponden a variables del problema y los arcos corresponden a restricciones.



PROBLEMAS DE SATISFACCIÓN DE RESTRICCIONES

Un estado del problema está definido por una **asignación** de valores a unas o todas las variables.

- **Asignación consistente o legal**: aquella que no viola ninguna restricción.
- **Asignación completa:** aquella en la que se menciona cada variable
- **Solución:** asignación completa que satisface todas las restricciones.

Ventajas

- La representación del estado en un PSR se ajusta a un modelo estándar
- La función sucesor y el test objetivo pueden escribirse de un modo genérico para que se aplique a todo PSR.
- Se pueden desarrollar heurísticas eficaces y genéricas que no requieran ninguna información adicional ni experta del dominio específico.
- La estructura del grafo de restricciones puede usarse para simplificar el proceso de solución.

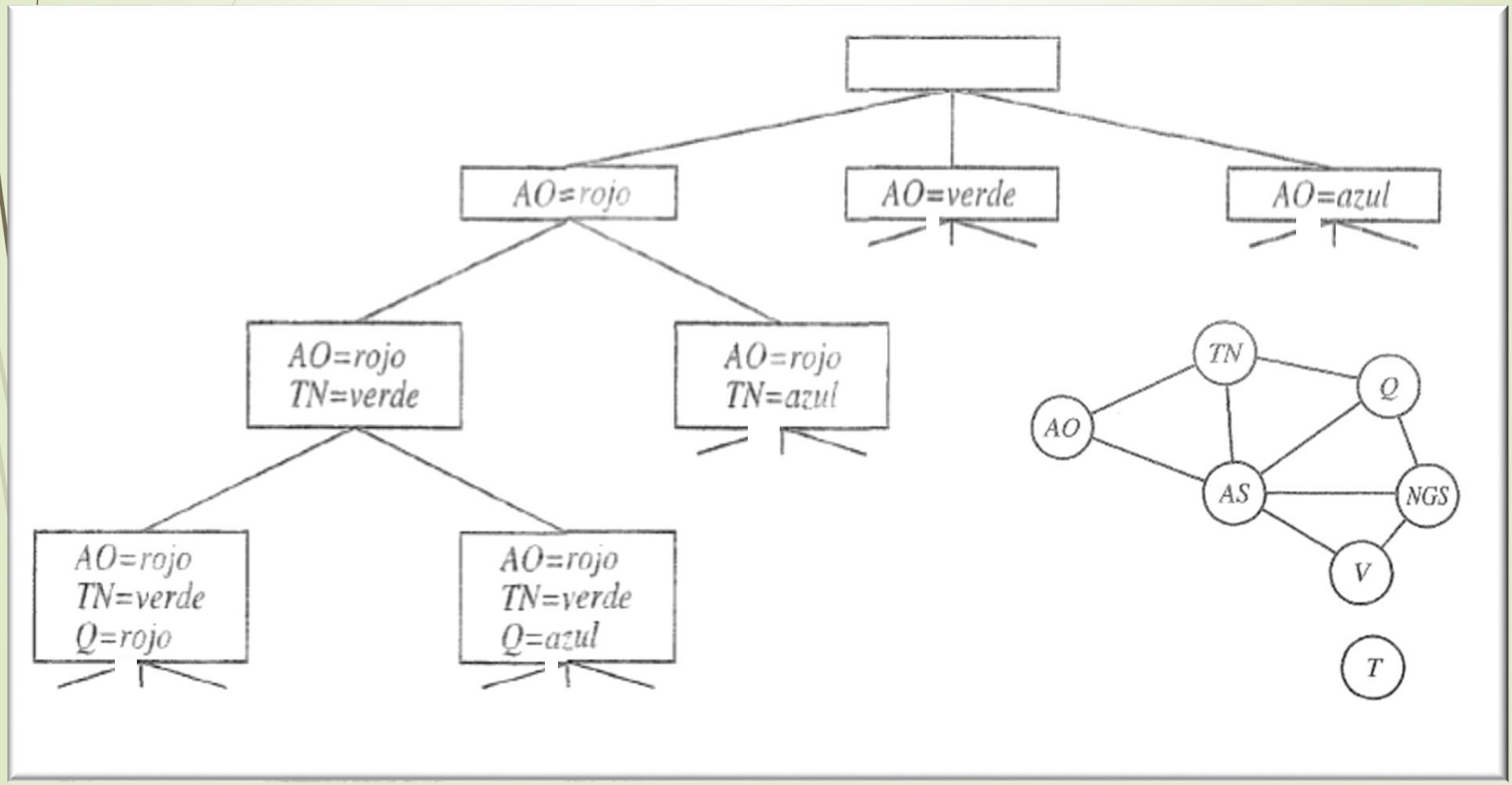
Tipos de restricción

- **Unitaria :** restringe los valores de una sola variable
- **Binaria:** relaciona dos variables
- **Orden alto:** implican tres o más variables

PSR: BÚSQUEDA ESTÁNDAR

- Estado inicial: la asignación vacía {}
- Función sucesor: asignar un valor a cualquier variable no asignada (cumpliendo con las restricciones)
- Test objetivo: la asignación actual completa y cumple las restricciones
- Costo del camino: un costo cte. para cada paso

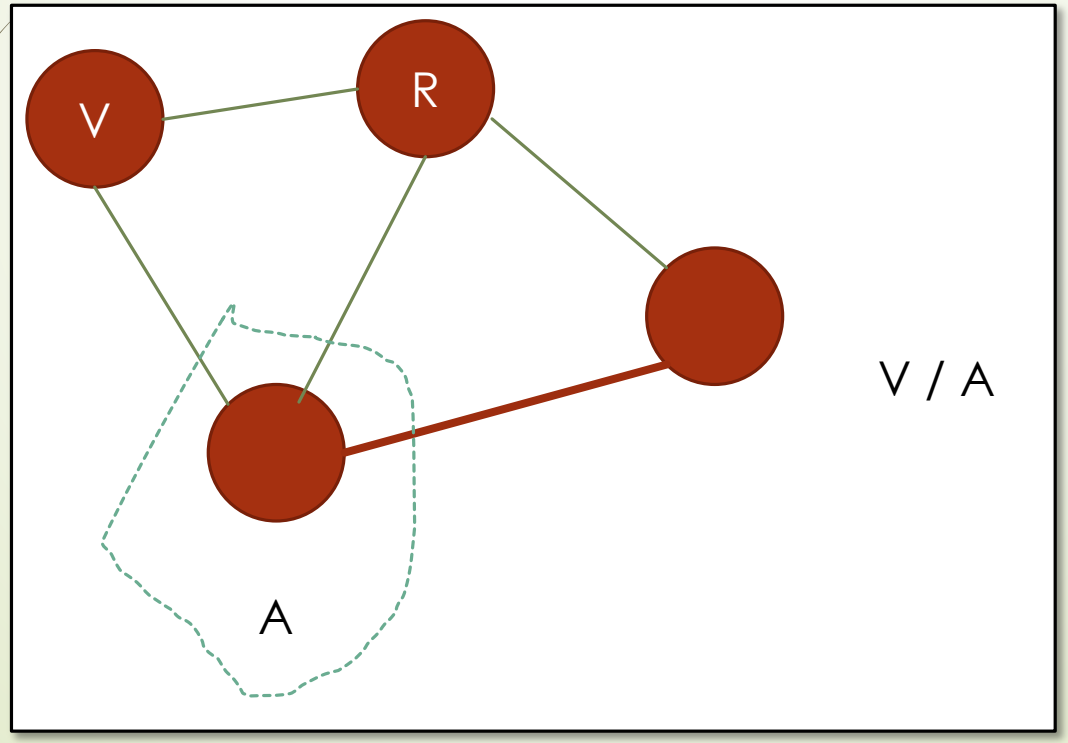
Parte del árbol de búsqueda para el problema de colorear el mapa de Oceanía



VARIABLE Y ORDENAMIENTO DE VALOR

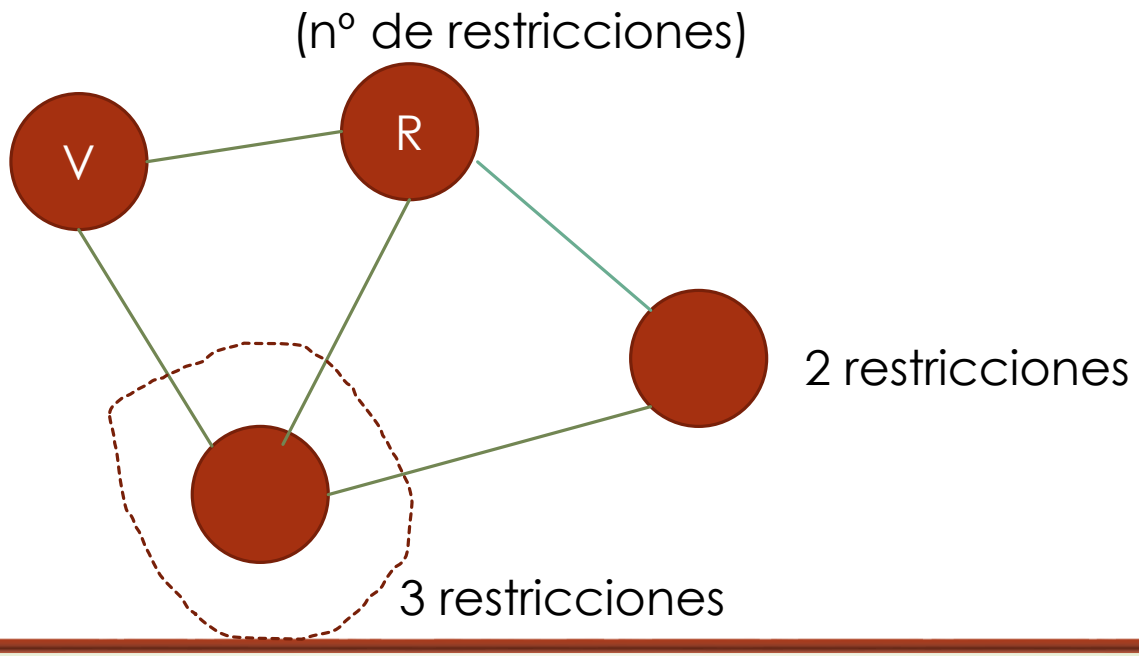
- heurística de “mínimos valores restantes”

  (escoge la variable con menos valores legales)



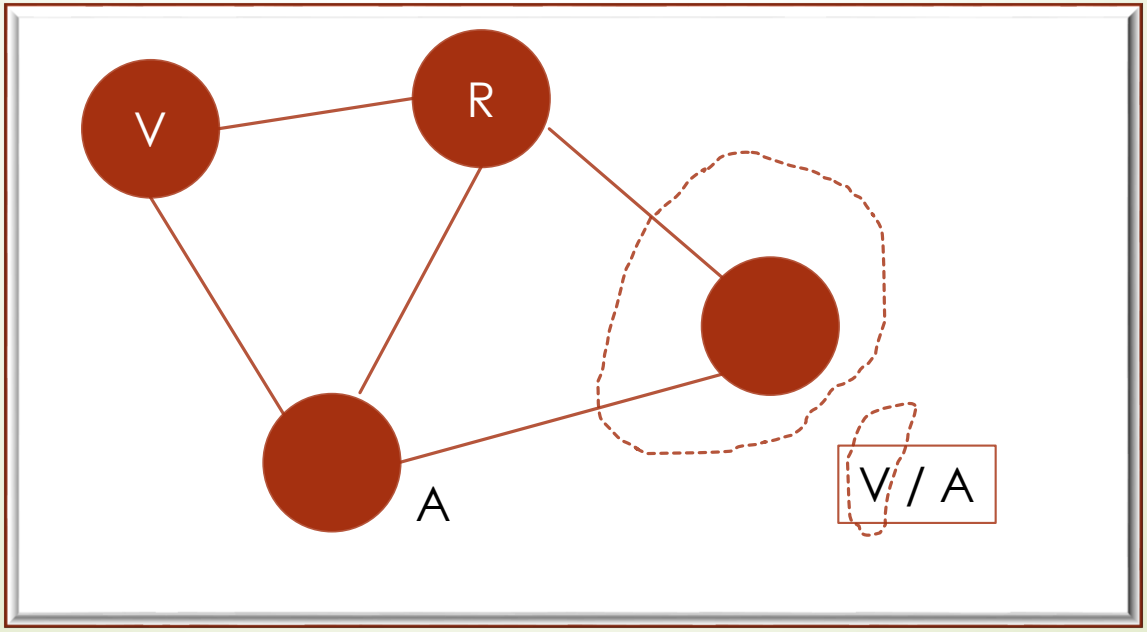
VARIABLE Y ORDENAMIENTO DE VALOR

- "grado heurístico"



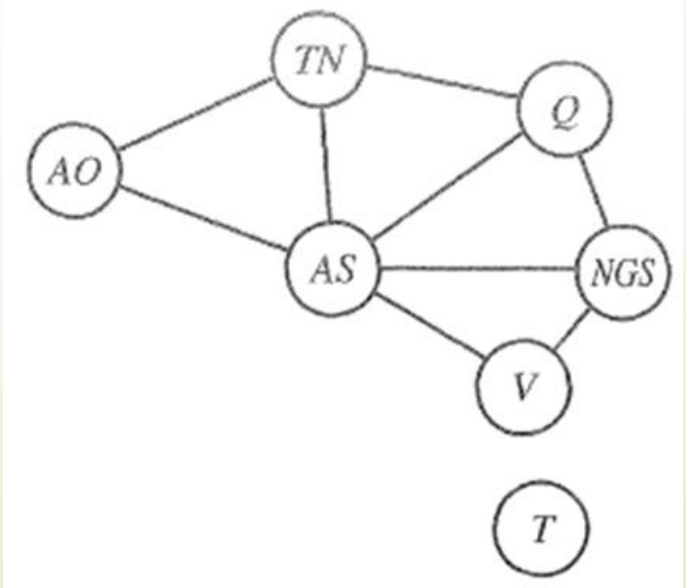
VARIABLE Y ORDENAMIENTO DE VALOR

- heurística “valor menos restringido”



Comprobación hacia adelante

	AO	*TN*	*Q*	*NGS*	*V*	*AS*	*T*
Dominios iniciales	R V A	R V A	R V A	R V A	R V A	R V A	R V A
Después de *AO=rojo*	(R)	V A	R V A	R V A	R V A	V A	R V A
Después de *Q=verde*	(R)	A	(V)	R A	R V A	A	R V A
Después de *V=azul*	(R)	A	(V)	R	(A)		R V A

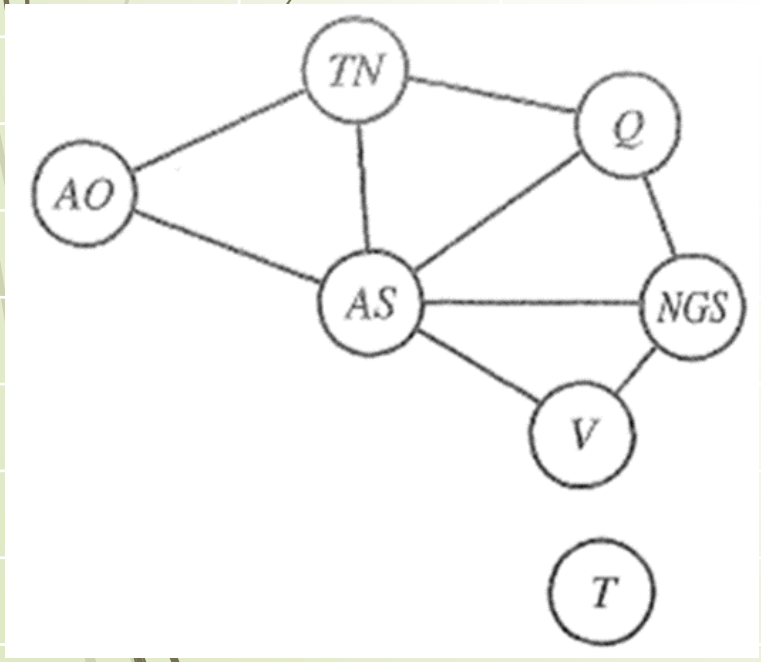


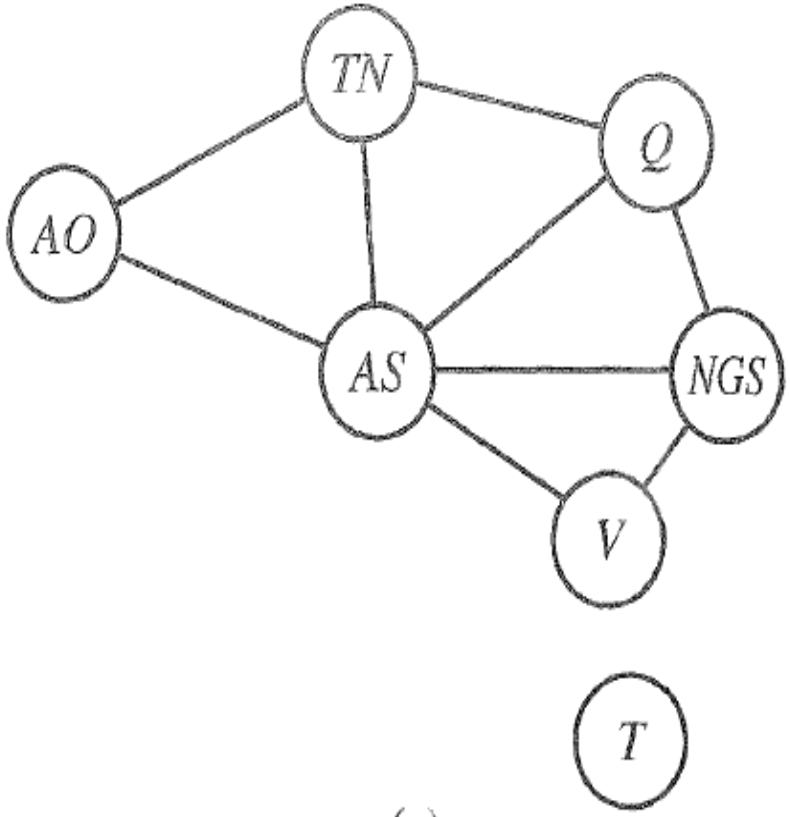
ARCO CONSISTENCIA

- Un arco es consistente si, para todo valor de “x” hay algún valor de “y” que es consistente con “x”.
- La consistencia del arco puede aplicarse como un paso de proceso antes de comenzar la búsqueda o como un paso de propagación después de cada asignación durante la búsqueda.

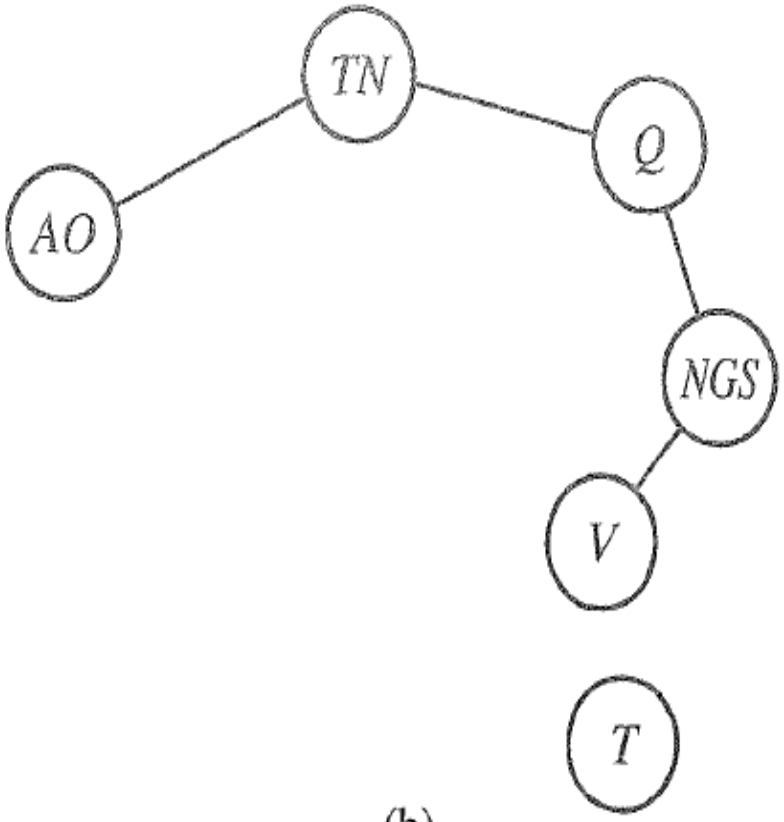
ALGORITMO AC-3

AO	TN	Q	NGS	V	AS	T
R V A	R V A	R V A	R V A	R V A	R V A	R V A
R	V A	R VA	R V A	R V A	V A	R V A
	V	R A	R V A	R V A	A	R V A
		R	V A	R V A	A	R V A
			V	R A	A	R V A
						R V A
				R	A	R V A
					A	R V A





(a)



(b)

Figura 5.11 (a) Grafo restricción original de la Figura 5.1. (b) Grafo restricción después de quitar *AS*.

Algoritmo general

- Elegir un subconjunto S de VARIABLES[PSR] tal que el grafo de restricciones se convierta en un árbol después de quitar S. Llamamos a S un "ciclo de corte".
- Para cada asignación posible a las variables en S que satisface todas las restricciones sobre S:
  - (a) quitar de los dominios de las variables restantes cualquier valor que sea inconsistente con la asignación para S, y
  - (b) si el PSR restante tiene una solución, devolverla junto con la asignación para S.

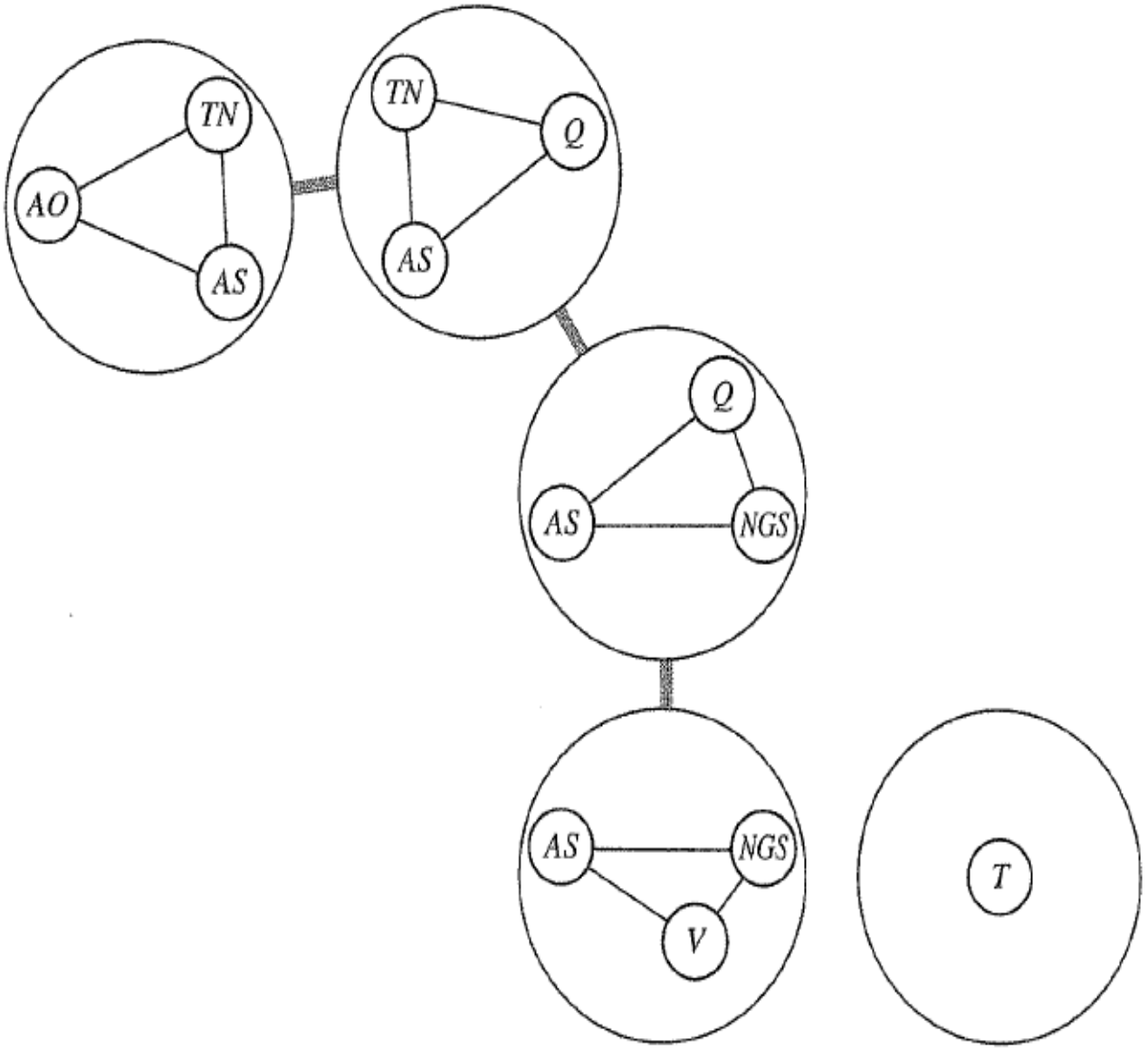


Figura 5.12 Una descomposición en árbol del grafo restricción de la Figura 5.11(a)

Descomposición en árbol

- Cada variable en el problema original aparece en al menos uno de los subproblemas.
- Si dos variables están relacionadas por una restricción en el problema original, deben aparecer juntas (junto con la restricción) en al menos uno de los subproblemas.
- Si una variable aparece en dos subproblemas en el árbol, debe aparecer en cada subproblema a lo largo del camino que une a esos subproblemas.