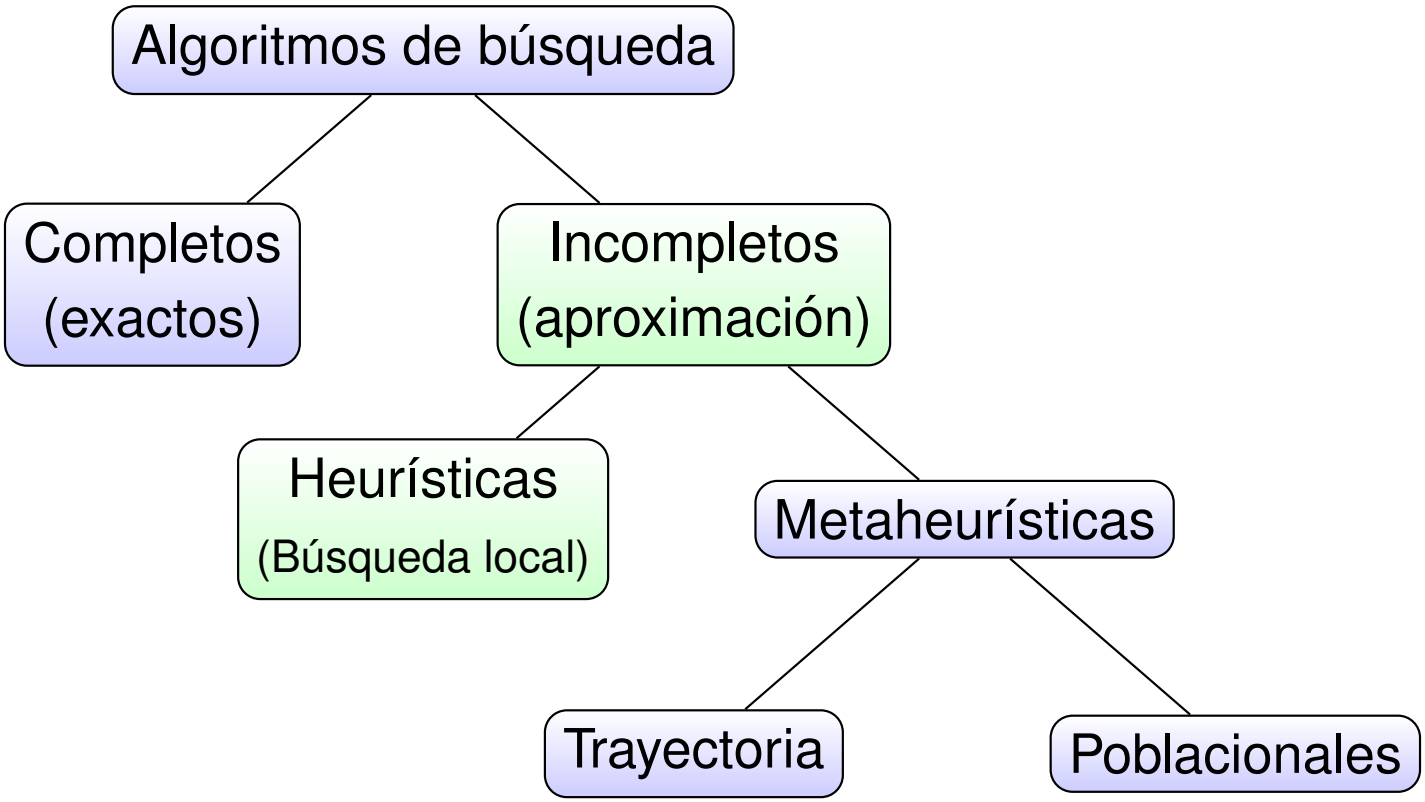
Investigación de Operaciones
Búsqueda local

Leslie Pérez Cáceres
leslie.perez@pucv.cl

Escuela de Informática
Pontificia Universidad Católica de Valparaíso

Técnicas de resolución



Heurísticas

Heurísticas:

- Derivado del griego `heuriskein`: encontrar; descubrir
- Técnicas específicas: diseñadas para un problema particular
  $\rightarrow$ utilizan información sobre el problema a resolver
- Su objetivo es resolver un problema rápidamente

La búsqueda local es un procedimiento de búsqueda heurística simple que aprenderemos a continuación.

Búsqueda local

Las búsquedas locales utilizan *información local* con el objetivo de encontrar un **óptimo local**.

```
s ← solucion_inicial();
while |V(s)| != ∅ do
    s* ← siguiente_vecino(V(s));
    if f(s*) < f(s) then
        s ← s*;
    end
end
return s;
```

Notar que asumimos un problema de minimización.

Revisemos que es un vecindario $V(s)$

Vecindario

¿Qué es un vecindario?

Conjunto de todas las soluciones que se pueden obtener aplicando un **movimiento** en particular a una solución inicial

Movimiento (**heurística perturbativa**): regla para cambiar los valores de las variables de decisión de una solución para obtener una solución diferente

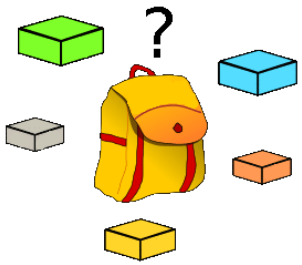
$\rightarrow$ usan información del problema para definir una regla de perturbación

Se puede aplicar esta regla sistemáticamente para **generar todas las soluciones posibles** a partir de una solución!

$\rightarrow$ esto genera una **noción de distancia** entre soluciones

Ejemplo mochila

Volvamos a nuestro ejemplo de los exploradores en al caverna:



	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
valor	32	47	18	26	85	33	45	59
peso	26	48	21	22	95	43	55	52

Seleccione los objetos que deben rescatar considerando:

- el peso total de los objetos seleccionados no debe ser mas de **101 kg**
- el valor de los objetos seleccionados debe ser el máximo posible

Ejemplo mochila

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
valor	32	47	18	26	85	33	45	59
peso	26	48	21	22	95	43	55	52

Las variables de decisión son:

$x_i = 1$ si el objeto i es seleccionado, 0 en otro caso

Representamos una solución como un arreglo:

$$s = \boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}$$

$$p(s) = 26 \qquad f(s) = 32$$

Utilizaremos la heurística perturbativa (movimiento) **bit flip**.

Ejemplo mochila: movimiento bit flip

bit flip: cambiar una variable de la solución de 0 a 1 o viceversa.

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
valor	32	47	18	26	85	33	45	59
peso	26	48	21	22	95	43	55	52

Solución inicial:

s = | 1 | **0** | 0 | 0 | 0 | 0 | **0** | 0 | $p(s) = 26, f(s) = 32$

Generemos el vecino bit flip que cambia la variable x_2

1	**1**	0	0	0	0	0	0	$p(s_2) = 74, \quad f(s_2) = 79$

Generemos el vecino bit flip que cambia la variable x_7

1	0	0	0	0	0	**1**	0	$p(s_7) = 81, \quad f(s_7) = 77$

Ejemplo mochila: vecindario bit flip

bit flip: cambiar una variable de la solución de 0 a 1 o viceversa.

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
valor	32	47	18	26	85	33	45	59
peso	26	48	21	22	95	43	55	52

Solución inicial:

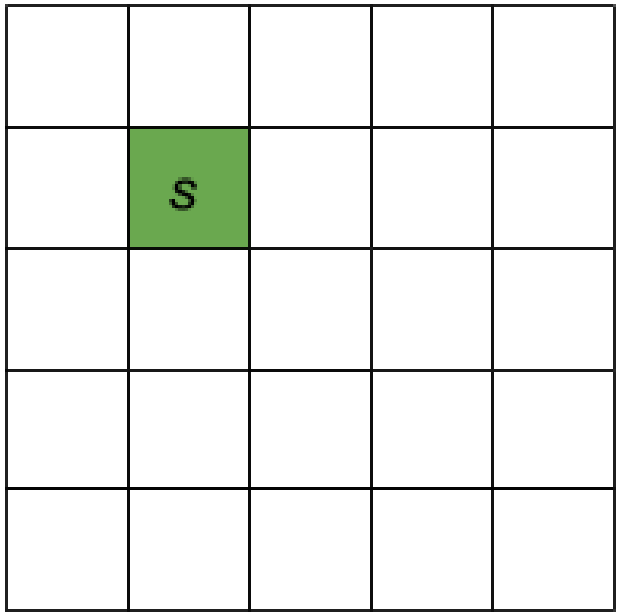
s = | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $p(s) = 26, f(s) = 32$

Construyamos el **vecindario bit flip** de s: le aplicamos sistemáticamente bit flip

0	0	0	0	0	0	0	0	$p(s_1) = 0, \quad f(s_1) = 0$
1	1	0	0	0	0	0	0	$p(s_2) = 74, \quad f(s_2) = 79$
1	0	1	0	0	0	0	0	$p(s_3) = 47, \quad f(s_3) = 50$
1	0	0	1	0	0	0	0	$p(s_4) = 48, \quad f(s_4) = 58$
1	0	0	0	1	0	0	0	$p(s_5) = 121, \quad f(s_5) = 117$
1	0	0	0	0	1	0	0	$p(s_6) = 89, \quad f(s_6) = 65$
1	0	0	0	0	0	1	0	$p(s_7) = 81, \quad f(s_7) = 77$
1	0	0	0	0	0	0	1	$p(s_8) = 78, \quad f(s_8) = 91$

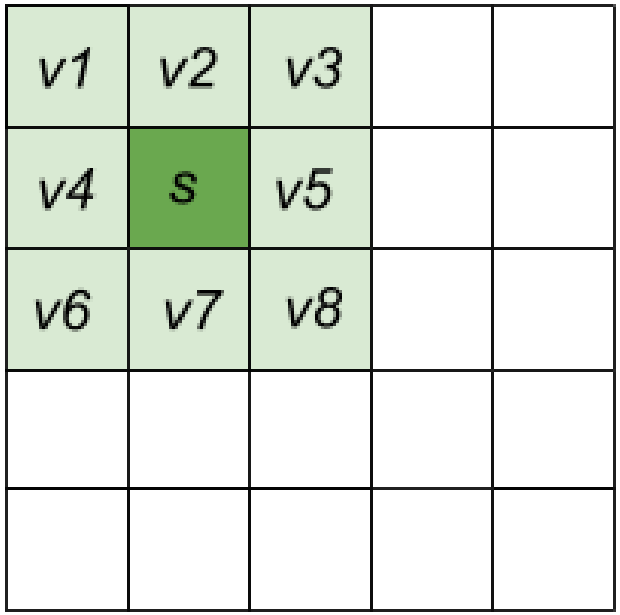
Vecindario y criterio de aceptación

El **movimiento** (**heurística perturbativa**) permite definir como moverse a través del espacio de búsqueda



Vecindario y criterio de aceptación

El **movimiento** (**heurística perturbativa**) permite definir como moverse a través del espacio de búsqueda

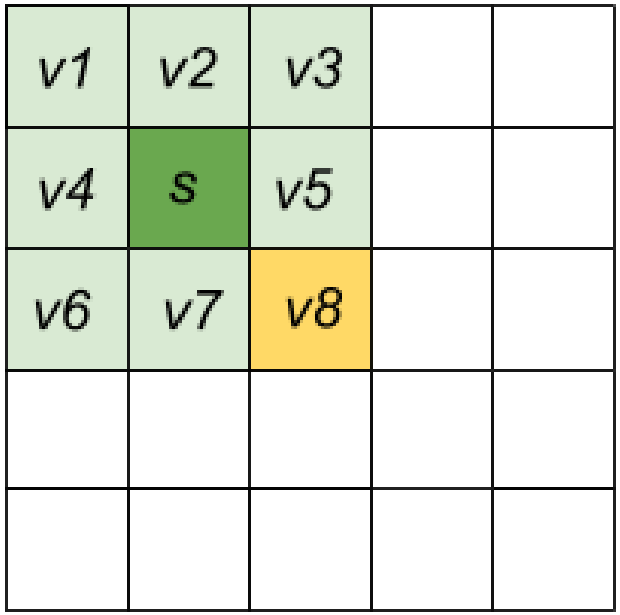


El **criterio de aceptación** define que solución elegir del vecindario:

- primera mejora: elige la primera solución del vecindario que sea mejor que la solución inicial *s*
- mejor mejora: elige la mejor solución de todo el vecindario

Vecindario

El **movimiento** (**heurística perturbativa**) define como moverse a través del espacio de búsqueda

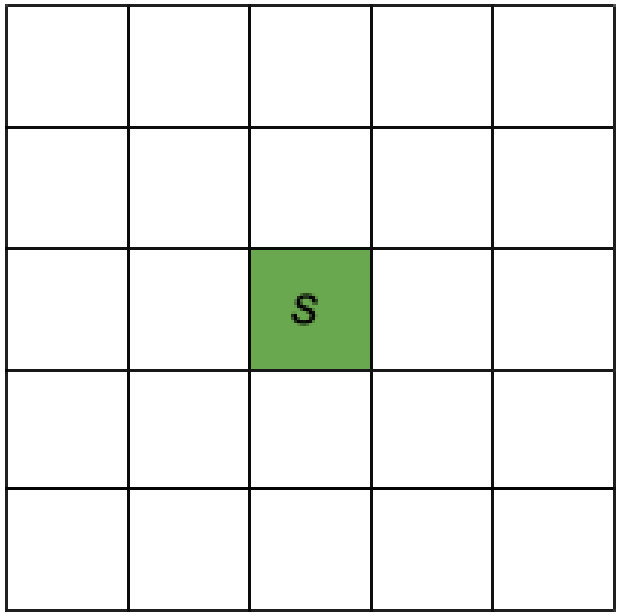


El **criterio de aceptación** define que solución elegir del vecindario:

- primera mejora: elige la primera solución del vecindario que sea mejor que la solución inicial *s*
- mejor mejora: elige la mejor solución de todo el vecindario

Vecindario

El **movimiento** (**heurística perturbativa**) define como moverse a través del espacio de búsqueda (paso)

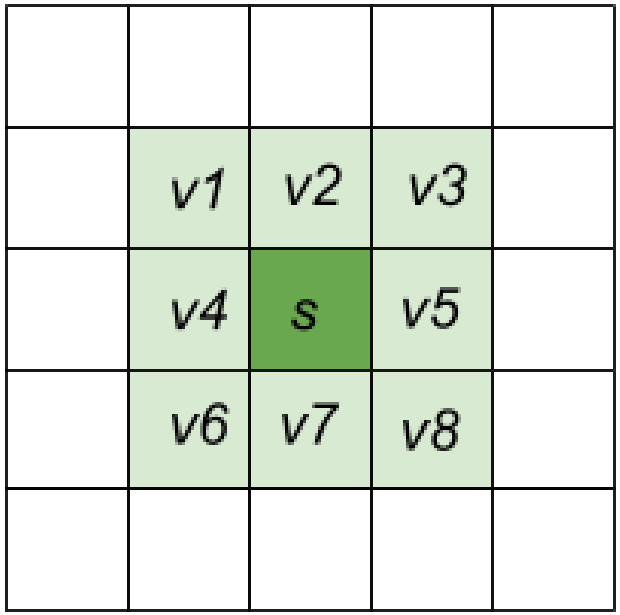


El **criterio de aceptación** define que solución elegir del vecindario:

- primera mejora: elige la primera solución del vecindario que sea mejor que la solución inicial *s*
- mejor mejora: elige la mejor solución de todo el vecindario

Vecindario

El **movimiento** (**heurística perturbativa**) define como moverse a través del espacio de búsqueda (paso)

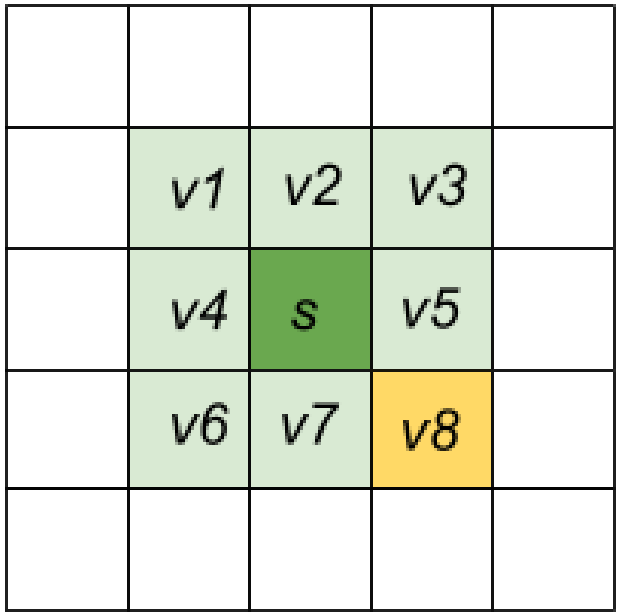


El **criterio de aceptación** define que solución elegir del vecindario:

- primera mejora: elige la primera solución del vecindario que sea mejor que la solución inicial *s*
- mejor mejora: elige la mejor solución de todo el vecindario

Vecindario

El **movimiento** (**heurística perturbativa**) define como moverse a través del espacio de búsqueda (paso)

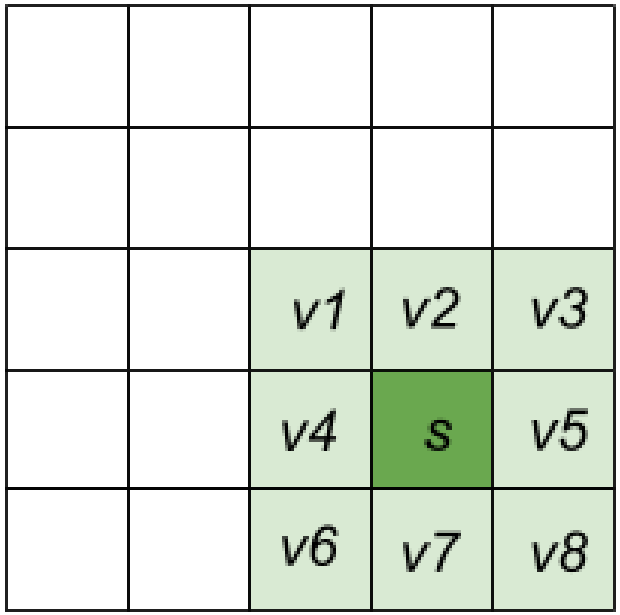


El **criterio de aceptación** define que solución elegir del vecindario:

- primera mejora: elige la primera solución del vecindario que sea mejor que la solución inicial *s*
- mejor mejora: elige la mejor solución de todo el vecindario

Vecindario

El **movimiento** (**heurística perturbativa**) define como moverse a través del espacio de búsqueda (paso)

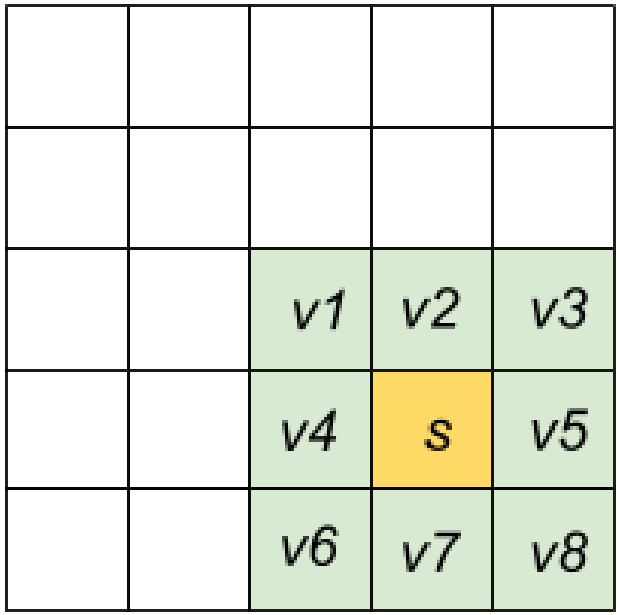


El **criterio de aceptación** define que solución elegir del vecindario:

- primera mejora: elige la primera solución del vecindario que sea mejor que la solución inicial *s*
- mejor mejora: elige la mejor solución de todo el vecindario

Vecindario

El **movimiento** (**heurística perturbativa**) define como moverse a través del espacio de búsqueda (paso)



El **criterio de aceptación** define que solución elegir del vecindario:

- primera mejora: elige la primera solución del vecindario que sea mejor que la solución inicial *s*
- mejor mejora: elige la mejor solución de todo el vecindario

Ejemplo mochila: vecindario bit flip

bit flip: cambiar una variable de la solución de 0 a 1 o viceversa.

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
valor	32	47	18	26	85	33	45	59
peso	26	48	21	22	95	43	55	52

Solución inicial:

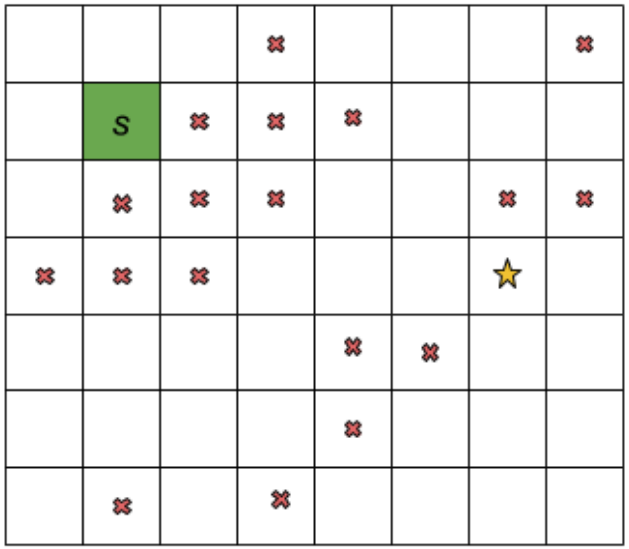
s = | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $p(s) = 26, f(s) = 32$

Vecindario:

$s_1 =$	0	0	0	0	0	0	0	0	$p(s_1) = 0,$	$f(s_1) = 0$
$s_2 =$	1	1	0	0	0	0	0	0	$p(s_2) = 74,$	$f(s_2) = 79$
$s_3 =$	1	0	1	0	0	0	0	0	$p(s_3) = 47,$	$f(s_3) = 50$
$s_4 =$	1	0	0	1	0	0	0	0	$p(s_4) = 48,$	$f(s_4) = 58$
$s_5 =$	1	0	0	0	1	0	0	0	$p(s_5) = 121,$	$f(s_5) = 117$
$s_6 =$	1	0	0	0	0	1	0	0	$p(s_6) = 89,$	$f(s_6) = 65$
$s_7 =$	1	0	0	0	0	0	1	0	$p(s_7) = 81,$	$f(s_7) = 77$
$s_8 =$	1	0	0	0	0	0	0	1	$p(s_8) = 78,$	$f(s_8) = 91$

Vecindario y restricciones

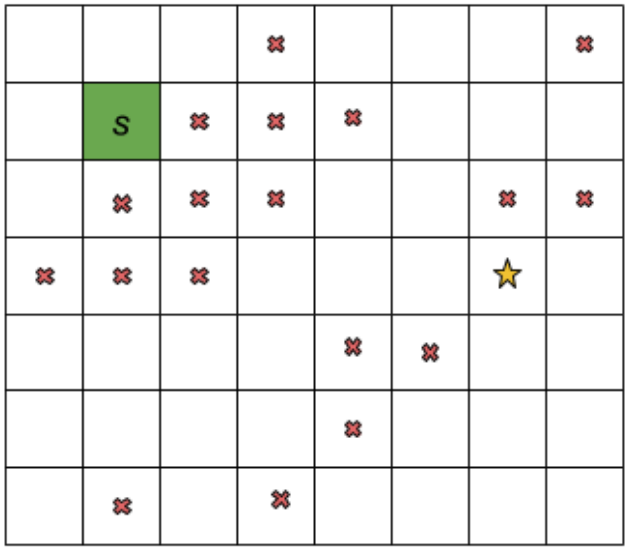
¿Qué sucede con las soluciones infactibles?



En general, en las técnicas incompletas **todas** las soluciones, factibles o infactibles, pueden ser parte del espacio de búsqueda.

Manejo de restricciones

¿Qué sucede con las soluciones infactibles?



Opciones para manejar restricciones:

- Descartar soluciones infactibles
- Penalización $\rightarrow$ función de evaluación (Ej. Mochila)
- Manejar restricciones en la representación y movimiento (Ej. TSP)

Función de evaluación (recordatorio

La **función de evaluación** añade una **penalización** a la función objetivo

$$z(s) = f(s) + \Phi(s)$$

Permite que la búsqueda tienda a satisfacer restricciones a medida que optimiza la función de objetivo.

Ejemplo mochila: función de evaluación

Para el problema de la mochila podemos definir:

$$\Phi(s) = \begin{cases} 0, & \text{si } s \text{ es factible} \\ -100, & \text{si } s \text{ es infactible} \end{cases}$$

$s_5 =$ | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | $\quad p(s_5) = 121 \quad f(s_5) = 117$

$$\begin{aligned} z(s_5) &= f(s_5) + \Phi(s_5) \\ &= 117 - 100 \\ &= 17 \end{aligned}$$

Ejemplo mochila: vecindario

bit flip: cambiar una variable de la solución de 0 a 1 o viceversa.

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
valor	32	47	18	26	85	33	45	59
peso	26	48	21	22	95	43	55	52

Solución inicial:

1	0	0	0	0	0	0	0

$p(s) = 26, f(s) = 32$

Vecindario:

0	0	0	0	0	0	0	0	$p(s_1) = 0, \quad f(s_1) = 0$
1	1	0	0	0	0	0	0	$p(s_2) = 74, \quad f(s_2) = 79$
1	0	1	0	0	0	0	0	$p(s_3) = 47, \quad f(s_3) = 50$
1	0	0	1	0	0	0	0	$p(s_4) = 48, \quad f(s_4) = 58$
1	0	0	0	1	0	0	0	$p(s_5) = 121, \quad f(s_5) = 117$
1	0	0	0	0	1	0	0	$p(s_6) = 89, \quad f(s_6) = 65$
1	0	0	0	0	0	1	0	$p(s_7) = 81, \quad f(s_7) = 77$
1	0	0	0	0	0	0	1	$p(s_8) = 78, \quad f(s_8) = 91$

Ejemplo mochila: vecindario

bit flip: cambiar una variable de la solución de 0 a 1 o viceversa.

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
valor	32	47	18	26	85	33	45	59
peso	26	48	21	22	95	43	55	52

Solución inicial:

1	0	0	0	0	0	0	0

$p(s) = 26, z(s) = 32$

Vecindario:

0	0	0	0	0	0	0	0	$p(s_1) = 0, \quad z(s_1) = 0$
1	1	0	0	0	0	0	0	$p(s_2) = 74, \quad z(s_2) = 79$
1	0	1	0	0	0	0	0	$p(s_3) = 47, \quad z(s_3) = 50$
1	0	0	1	0	0	0	0	$p(s_4) = 48, \quad z(s_4) = 58$
1	0	0	0	1	0	0	0	$p(s_5) = 121, \quad z(s_5) = 17$
1	0	0	0	0	1	0	0	$p(s_6) = 89, \quad z(s_6) = 65$
1	0	0	0	0	0	1	0	$p(s_7) = 81, \quad z(s_7) = 77$
1	0	0	0	0	0	0	1	$p(s_8) = 78, \quad z(s_8) = \mathbf{91}$

Búsqueda local

Las búsquedas locales utilizan *información local* con el objetivo de encontrar un **óptimo local**.

```
s ← solucion_inicial();
while |V(s)| != ∅ do
    s* ← siguiente_vecino(V(s));
    if f(s*) < f(s) then
        s ← s*;
    end
end
return s;
```

Notar que asumimos un problema de minimización.

Revisemos como generamos soluciones iniciales

Soluciones iniciales

Pueden obtenerse:

- de soluciones conocidas (si las hay),
- generarse de manera aleatoria o,
- generarse utilizando un procedimiento heurístico

Las heurísticas que se utilizan para generar soluciones se llaman **heurísticas constructivas**.

- Son reglas sencillas para **construir** paso a paso una solución
- Utilizan información del problema para definir estas reglas

Son comúnmente llamadas técnicas **Greedy** o **Voraces** debido a que explotan la información del problema enfocándose siempre en lo mejor localmente

Ejemplo mochila 1

Nos preparamos para comenzar a jugar un juego y estamos equipando a nuestro personaje:

Seleccione el equipamiento entre los siguientes objetos maximizando su valor:

Restricción: nuestro personaje **no puede** llevar más de 6 *kg*

Ejemplo mochila 1: heurísticas constructivas

$W = 6\ kg.$

- Heurística del objeto más valioso: en cada paso seleccionar el objeto más valioso
- Heurística del objeto más liviano: en cada paso seleccionar el objeto más liviano
- Heurística del mejor valor/peso: en cada paso seleccionar el objeto con mejor razón valor/peso

Ejemplo mochila 1: heurísticas constructivas

$W = 6\ kg.$

Heurística del objeto más valioso: en cada paso seleccionar el objeto más valioso

$\rightarrow$ Objetos ordenados por valor:

$\rightarrow$ Solución generada:

$$f(x) = 2800 \qquad p(x) = 6$$

Ejemplo mochila 1: heurísticas constructivas

$W = 6\ kg.$

Heurística del objeto más liviano: en cada paso seleccionar el objeto más liviano

$\rightarrow$ Objetos ordenados por peso:

$\rightarrow$ Solución generada:

$f(x) = 4200 \qquad p(x) = 5.5$

Ejemplo mochila 1: heurísticas constructivas

$W = 6\ kg.$

Heurística del mejor valor/peso:: en cada paso seleccionar el objeto con mejor razón valor/peso

$\rightarrow$ Objetos ordenados por razón valor/peso:

$\rightarrow$ Solución generada:

$f(x) = 4200 \qquad p(x) = 5.5$

Ejemplo mochila 2: heurísticas constructivas [trabajo autónomo]

Supongamos que nuestro personaje ha subido de nivel y ahora hay mas objetos disponibles:

Ademas, su fuerza y resistencia fué aumentada.

Restricción nuestro personaje **no puede** llevar más de 30 *kg*

Ejemplo mochila 2: heurísticas constructivas [trabajo autónomo]

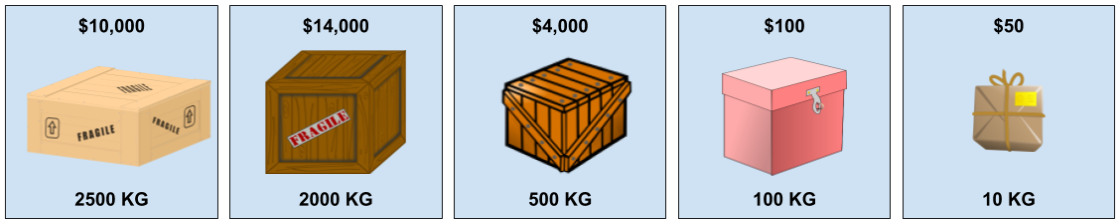
$W = 30\ kg.$

¿Qué heurística encuentra la mejor solución?

- Heurística del objeto más valioso
- Heurística del objeto más liviano
- Heurística del mejor valor/peso

Ejemplo mochila 3: heurísticas constructivas [trabajo autónomo]

Supongamos que queremos seleccionar la carga de un camión que puede llevar máximo 5000 *kg*.



$W = 5000\ kg$.

¿Qué heurística encuentra la mejor solución?

- Heurística del objeto más valioso
- Heurística del objeto más liviano
- Heurística del mejor valor/peso

Búsqueda local

Las búsquedas locales utilizan *información local* con el objetivo de encontrar un **óptimo local**.

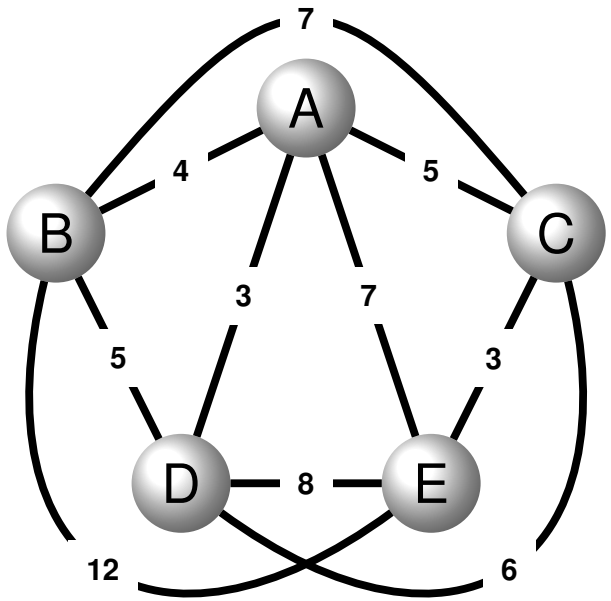
```
s ← solucion_inicial();
while |V(s)| ! = ∅ do
    s* ← siguiente_vecino(V(s));
    if f(s*) < f(s) then
        s ← s*;
    end
end
return s;
```

Notar que asumimos un problema de minimización.

¿Como aplicar esto a otros problemas?

Búsqueda local: Ejemplo TSP

¿Cómo aplicar una búsqueda local para el TSP?



- vecindario
- solución inicial

Búsqueda local TSP: vecindario

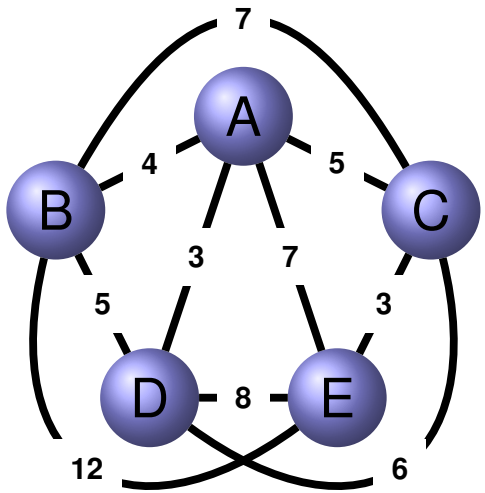
Para el problema del vendedor viajero vimos que podemos definir las variables de decisión como:

$x_i =$ ciudad (ubicación) visitada en la posición i

Podemos representar una solución como un vector:

$$s = \boxed{\text{A}}\boxed{\text{B}}\boxed{\text{C}}\boxed{\text{D}}\boxed{\text{E}}$$

$$f(s) = 4 + 7 + 6 + 8 + 7 = 32$$

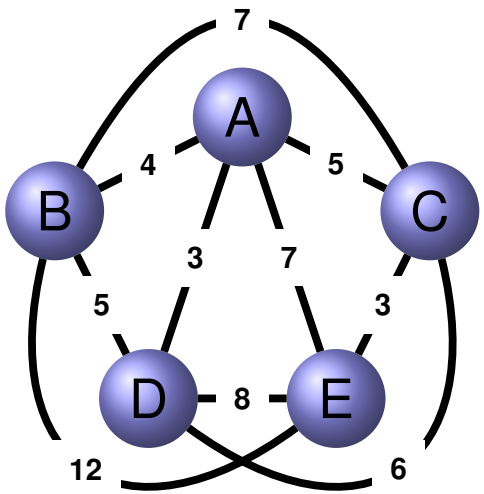


Búsqueda local TSP: vecindario

swap: Intercambia la posición de dos ciudades de una solución.

Solución inicial:

$s =$ | B | A | D | C | E | $f(s) = 4 + 3 + 6 + 3 + 12 = 28$



$s_1 =$ | A | B | D | C | E | $f(s_1) = 25$

$s_2 =$ | D | A | B | C | E | $f(s_2) = 25$

$s_3 =$ | C | A | D | B | E | $f(s_3) = 28$

$s_4 =$ | E | A | D | C | B | $f(s_4) = 35$

$s_5 =$ | B | D | A | C | E | $f(s_5) = 28$

$s_6 =$ | B | C | D | A | E | $f(s_6) = 35$

$s_7 =$ | B | E | D | C | A | $f(s_7) = 35$

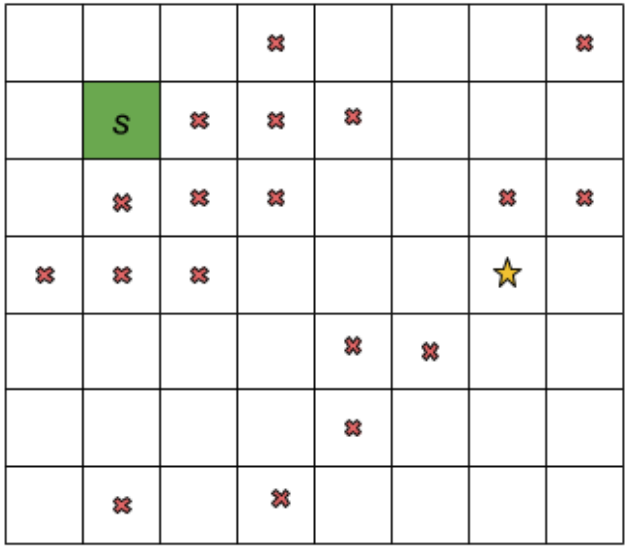
$s_8 =$ | B | A | C | D | E | $f(s_8) = 35$

$s_9 =$ | B | A | E | C | D | $f(s_9) = 25$

$s_{10} =$ | B | A | D | E | C | $f(s_{10}) = 25$

Vecindario y restricciones

¿Qué sucede con las soluciones infactibles?



En general, en las técnicas incompletas **todas** las soluciones, factibles o infactibles, pueden ser parte del espacio de búsqueda.

Opciones para manejar restricciones:

- Descartar soluciones infactibles
- Penalización $\rightarrow$ función de evaluación (Ej. Mochila)
- Manejar restricciones en la representación y movimiento (Ej. TSP)

Búsqueda local

Las búsquedas locales utilizan *información local* con el objetivo de encontrar un **óptimo local**.

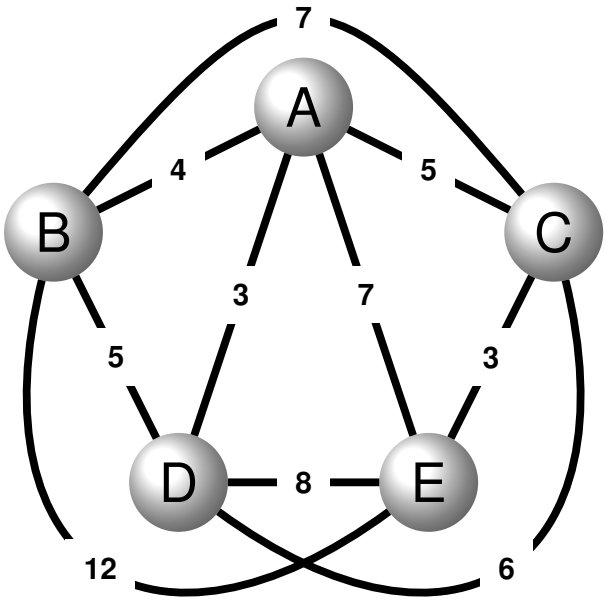
```
s ← solucion_inicial();
while |V(s)| ! = ∅ do
    s* ← siguiente_vecino(V(s));
    if f(s*) < f(s) then
        s ← s*;
    end
end
return s;
```

Notar que asumimos un problema de minimización.

¿Como aplicar esto a otros problemas?

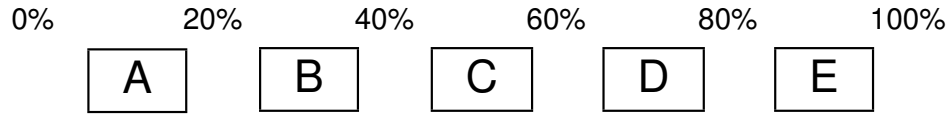
Ejemplo TSP: heurística constructiva

Heurística del vecino mas cercano: En cada paso seleccionar la ciudad más cercana a la actual.

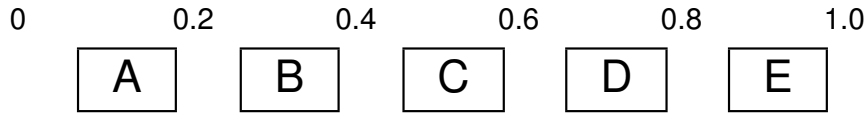


¿Cómo empezar? $\rightarrow$ elegir un nodo al azar!

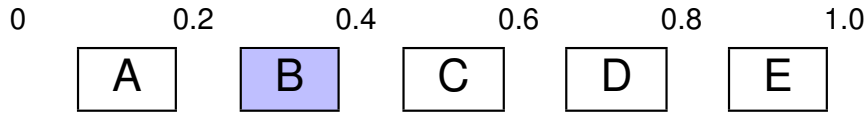
1. Asignar la misma probabilidad de selección a todas las ciudades



2. Generar un número aleatorio **rand()** $= 0.34$

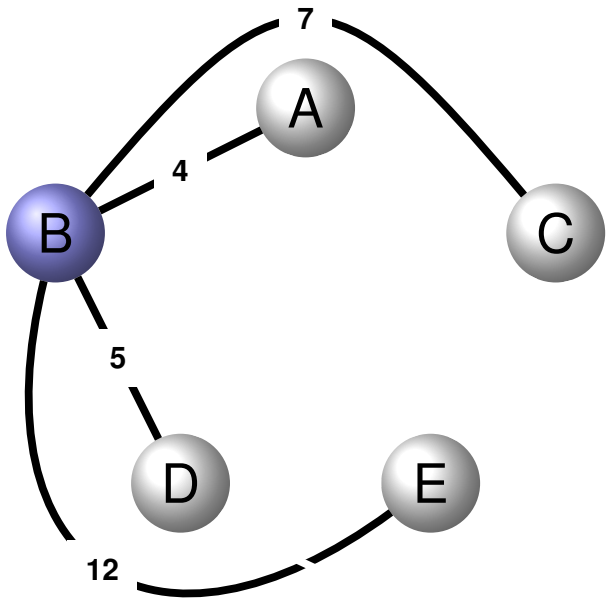


3. Seleccionar la ciudad en base al número obtenido



Ejemplo TSP: heurística constructiva

Heurística del vecino mas cercano: En cada paso seleccionar la ciudad más cercana a la actual.

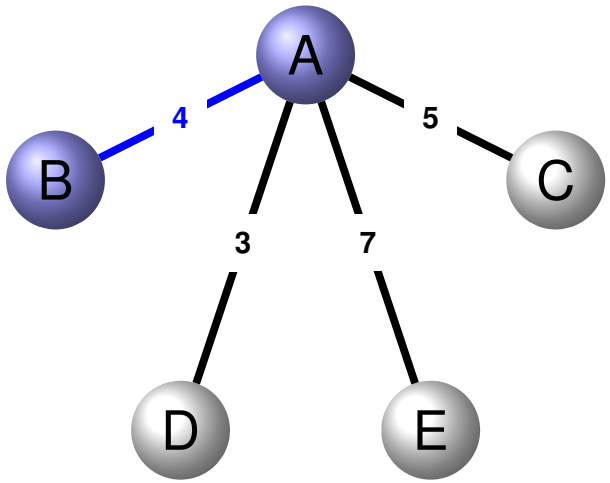


Ruta: B $\rightarrow$

$f(s)$:

Ejemplo TSP: heurística constructiva

Heurística del vecino mas cercano: En cada paso seleccionar la ciudad más cercana a la actual.

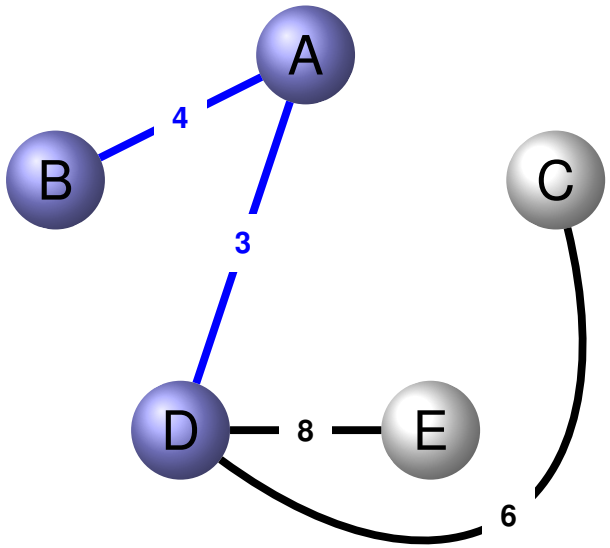


Ruta: B $\rightarrow$ A $\rightarrow$

$f(s)$: 4

Ejemplo TSP: heurística constructiva

Heurística del vecino mas cercano: En cada paso seleccionar la ciudad más cercana a la actual.

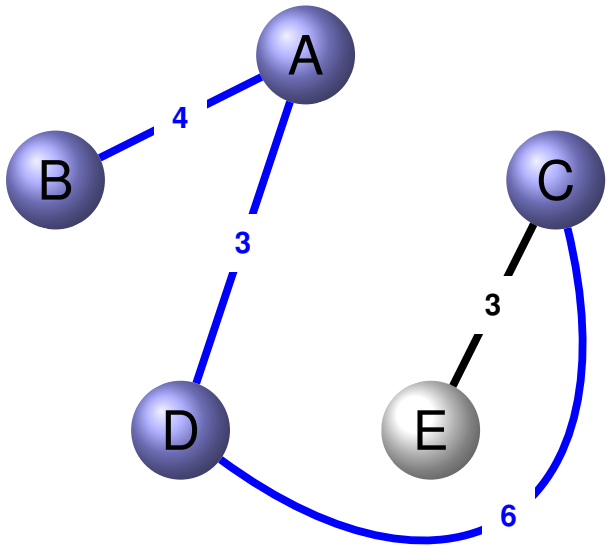


Ruta: B $\rightarrow$ A $\rightarrow$ D $\rightarrow$

$f(s)$: 4 + 3

Ejemplo TSP: heurística constructiva

Heurística del vecino mas cercano: En cada paso seleccionar la ciudad más cercana a la actual.

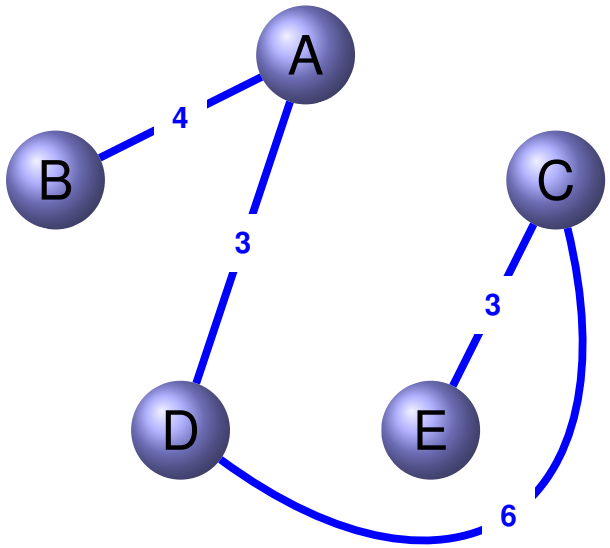


Ruta: B $\rightarrow$ A $\rightarrow$ D $\rightarrow$ C $\rightarrow$

$f(s)$: 4 + 3 + 6

Ejemplo TSP: heurística constructiva

Heurística del vecino mas cercano: En cada paso seleccionar la ciudad más cercana a la actual.



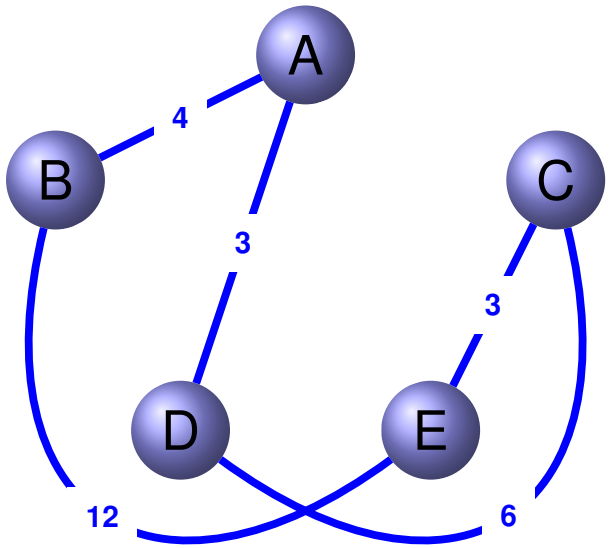
Ruta: B $\rightarrow$ A $\rightarrow$ D $\rightarrow$ C $\rightarrow$ E $\rightarrow$

$f(s)$: 4 + 3 + 6 + 3

¿Qué falta para que sea una solución factible del TSP?

Ejemplo TSP: heurística constructiva

Heurística del vecino mas cercano: En cada paso seleccionar la ciudad más cercana a la actual.



Ruta: B $\rightarrow$ A $\rightarrow$ D $\rightarrow$ C $\rightarrow$ E $\rightarrow$ (B)

$f(s)$: 4 + 3 + 6 + 3 + 12 = 28

Búsqueda local

Las búsquedas locales utilizan *información local* con el objetivo de encontrar un **óptimo local**.

```
s ← solucion_inicial();
while |V(s)| ! = ∅ do
    s* ← siguiente_vecino(V(s));
    if f(s*) < f(s) then
        s ← s*;
    end
end
return s;
```

Notar que asumimos un problema de minimización.

Ejercicio búsqueda local: asignación de tareas

En una fábrica se debe realizar la planificación de la jornada:

- hay 4 estaciones de trabajo cada una con un operador
- 4 tareas para completar la labor diaria
- el trabajo t_i tarda d_{ij} horas en la estación e_j (ver matriz)

Se busca una asignación que minimice el tiempo total empleado.

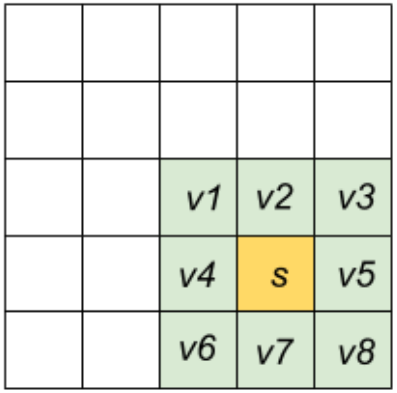
d_{ij}	t_1	t_2	t_3	t_4
e_1	9	6	8	7
e_2	6	5	7	2
e_3	8	6	3	6
e_4	5	5	6	3

Dado este problema:

1. Defina un modelo matemático para modelarlo
2. Diseñe una búsqueda local para resolverlo
3. A partir de la solución que asigna t_1 a e_1, t_2 a e_2, t_3 a e_3 y t_4 a e_4 (solución inicial), aplique una búsqueda local hasta encontrar un óptimo local.

Óptimos

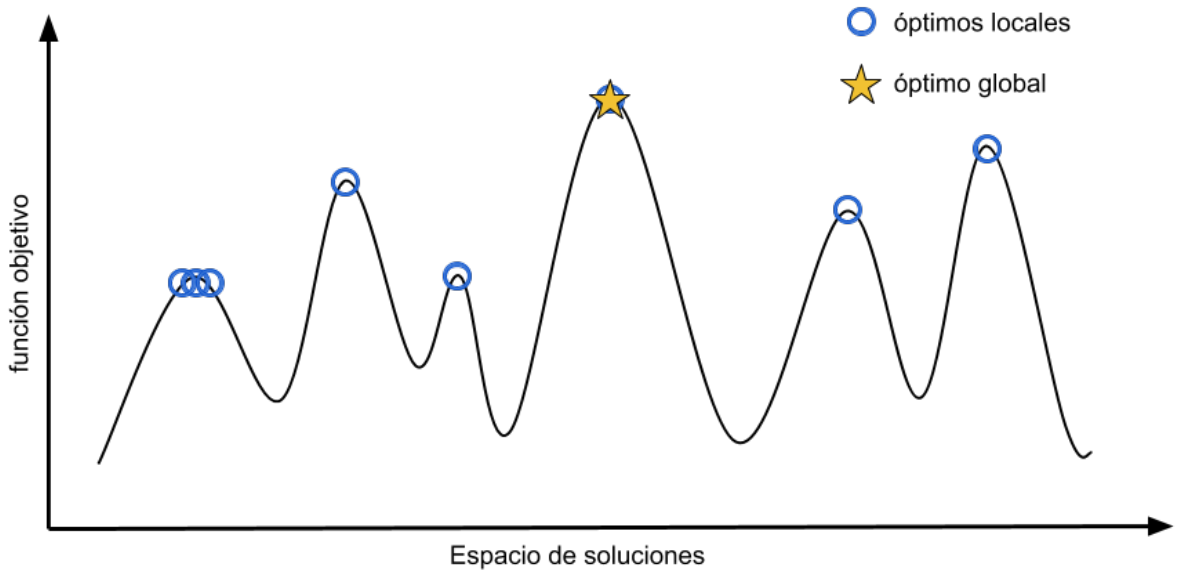
- Un **óptimo local** es una solución que esta rodeada de vecinos que no son mejores que ella misma, por lo tanto es la mejor solución del vecindario



- El **óptimo global** es el o los mejor(es) óptimos locales de todo el espacio de búsqueda

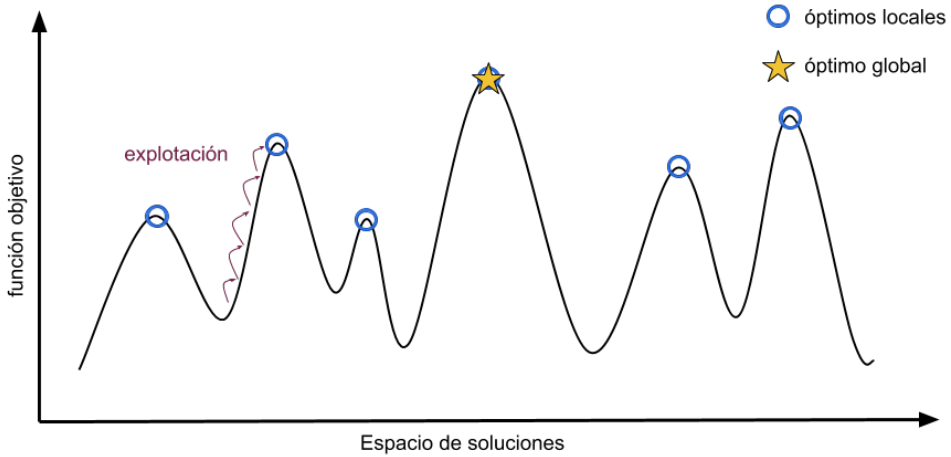
Espacio de búsqueda y movimiento

Los óptimos locales pueden ser **estrictos** o **no estrictos** ya que puede haber más de uno en el vecindario



Explotación y exploración

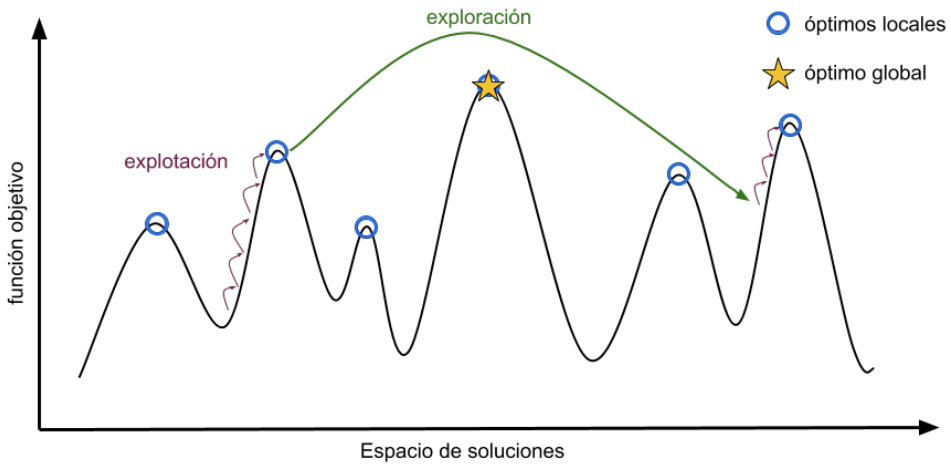
Explotación (intensificación): consiste en revisar extensamente un área acotada del espacio búsqueda en donde se sospecha pueden existir mejores soluciones



Las búsquedas realizan principalmente explotación del espacio de búsqueda

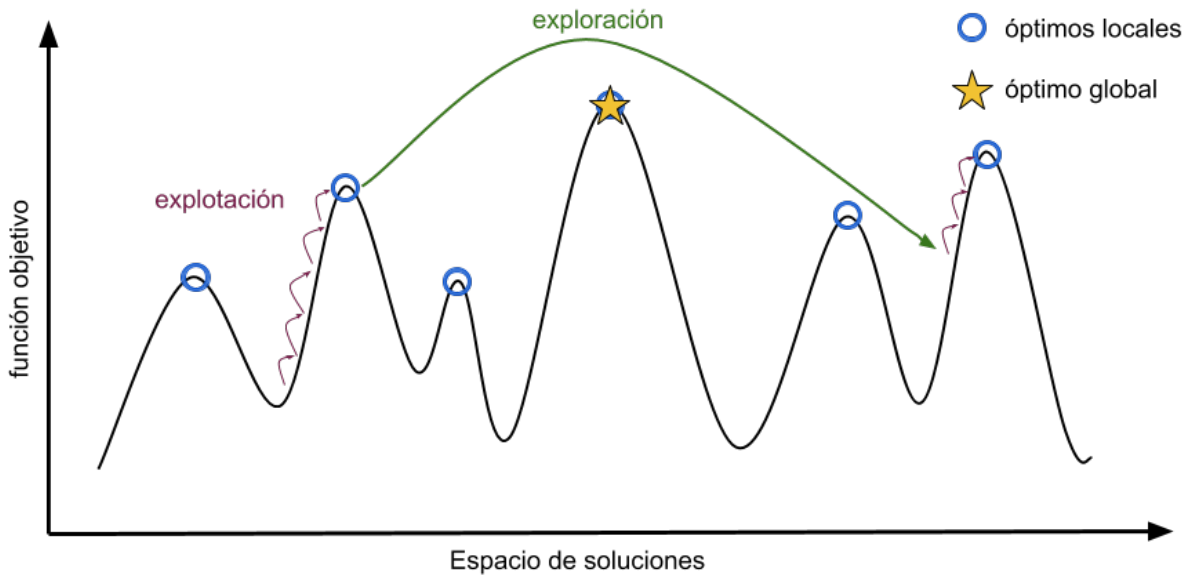
Explotación y exploración

Exploración (diversificación): consiste en revisar diferentes áreas del espacio de búsqueda de manera de identificar áreas donde pueden existir buenas soluciones



Permite que la búsqueda no se quede estancada en un óptimo local

Explotación y exploración



- Los conceptos de explotación y exploración son esenciales para las metaheurísticas
- Una buena metaheurística logra un buen **balance** de estas dos estrategias de búsqueda

Técnicas de resolución