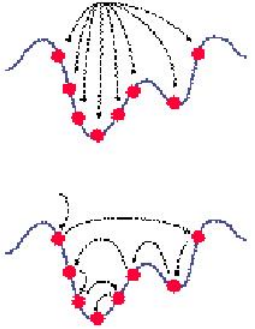


# ALGORÍTMICA
## 2012 - 2013

- Parte I. Introducción a las Metaheurísticas
  - Tema 1. Metaheurísticas: Introducción y Clasificación
- Parte II. Métodos Basados en Trayectorias y Entornos
  - Tema 2. Algoritmos de Búsqueda Local Básicos
  - Tema 3. Algoritmos de Enfriamiento Simulado
  - Tema 4. Algoritmos de Búsqueda Tabú
  - Tema 5. Métodos Basados en Trayectorias Múltiples I: Métodos Multiarranque Básicos y GRASP
  - Tema 6. Métodos Basados en Trayectorias Múltiples II: ILS y VNS
- Parte III. Métodos Basados en Poblaciones
  - Tema 7. Algoritmos Genéticos
- Parte IV. Intensificación y Diversificación
  - Tema 8. Estudio del Equilibrio entre Intensificación y Diversificación
- Parte V. Metaheurísticas Híbridas: Poblaciones y Trayectorias
  - Tema 9. Algoritmos Meméticos
  - Tema 10. Modelos Híbridos II: *Scatter Search*
- Parte VI. Paralelización de Metaheurísticas
  - Tema 11. Metaheurísticas en Sistemas Descentralizados. Metaheurísticas paralelas
- Parte VII. Conclusiones
  - Tema 12. Algunas Consideraciones sobre la Adaptación de Metaheurísticas a la Resolución de Problemas

# ALGORÍTMICA

## TEMA 2. Algoritmos de Búsqueda Local Básicos

1. Introducción
2. Búsqueda Aleatoria *versus* Búsqueda Local
3. Métodos de Búsqueda Local Básicos
4. Un Método de Búsqueda Local Avanzado: VND
5. Aplicaciones

- ***A. Díaz y otros. Optimización Heurística y Redes Neuronales. Paraninfo, 1996***
- ***J.M. Moreno, J.A. Moreno, Heurísticas en Optimización. Consejería de Cultura y Deporte, Gobierno de Canarias, 1999.***

# 1. INTRODUCCIÓN

**1.1. Término *"LOCAL"***
**1.2. Estructura de entorno**

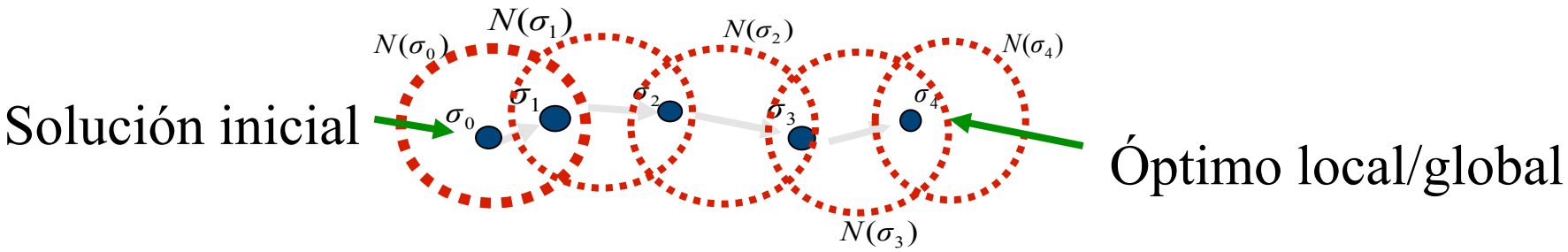
## 1.1. Término *"LOCAL"*

**El término "*local*" se utiliza frecuentemente en los estudios teóricos y prácticos de las metaheurísticas de búsqueda. Se asocia al uso de estructuras de entorno, reflejando el concepto de proximidad o vecindad entre las soluciones alternativas del problema.**

**Todas las soluciones incluidas en el entorno de la solución actual, que viene delimitado por un operador de generación de soluciones, se denominan soluciones vecinas.**

# 1. INTRODUCCIÓN

**Así, los algoritmos basados en esta estrategia efectúan un estudio local del espacio de búsqueda, puesto que analizan el entorno de la solución actual para decidir cómo continuar el recorrido de la búsqueda.**




**Una búsqueda local es un proceso que, dada la solución actual en la que se encuentra el recorrido, selecciona iterativamente una solución de su entorno para continuar la búsqueda.**

# 1. INTRODUCCIÓN

**Una búsqueda local es un proceso que, dada la solución actual en la que se encuentra el recorrido, selecciona iterativamente una solución de su entorno para continuar la búsqueda.**

## 1.2. Estructura de Entorno

**Basta con diseñar la estructura de entorno para obtener un modelo genérico de algoritmo de búsqueda.**

**DESCRIPCIÓN**

- **Se fija una codificación para las soluciones.**
- **Se define un operador de generación de vecino y, en consecuencia, se fija una estructura de entorno para las mismas.**
- **Se escoge una solución del entorno de la solución actual hasta que se satisfaga el criterio de parada.**

# 1. INTRODUCCIÓN

## ESTRUCTURA DE ENTORNO

### ELEMENTOS BÁSICOS

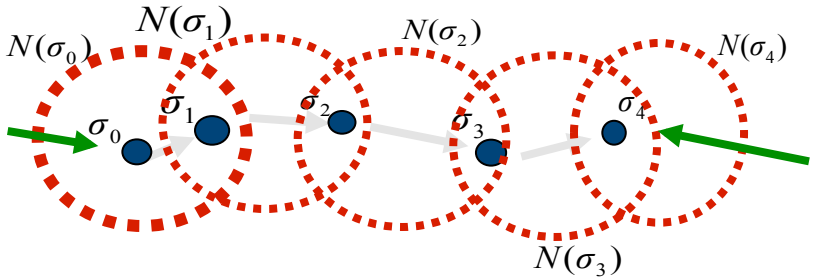
- **Proceso de elección de la solución inicial.**

- **Proceso de selección de solución/generación de una solución vecina:**

  **S → S', S' ∈ E(S) (también notado N(S)).**

- **Proceso de aceptación de solución vecina como solución actual.**

Solución inicial




Óptimo local/global

# 1. INTRODUCCIÓN

**Procedimiento Búsqueda por Entornos**
**Inicio**
GENERA(Solución Inicial)
Solución Actual ← Solución Inicial;
Mejor Solución ← Solución Actual;
**Repetir**
Solución Vecina ← GENERA_VECINO(Solución Actual);
**Si** Acepta(Solución Vecina)
**entonces** Solución Actual ← Solución Vecina;
**Si** Objetivo(Solución Actual) **es mejor que** Objetivo(Mejor Solución)
**entonces** Mejor Solución ← Solución Actual;
**Hasta** (Criterio de parada);
DEVOLVER (Mejor Solución);
**Fin**

# 1. INTRODUCCIÓN

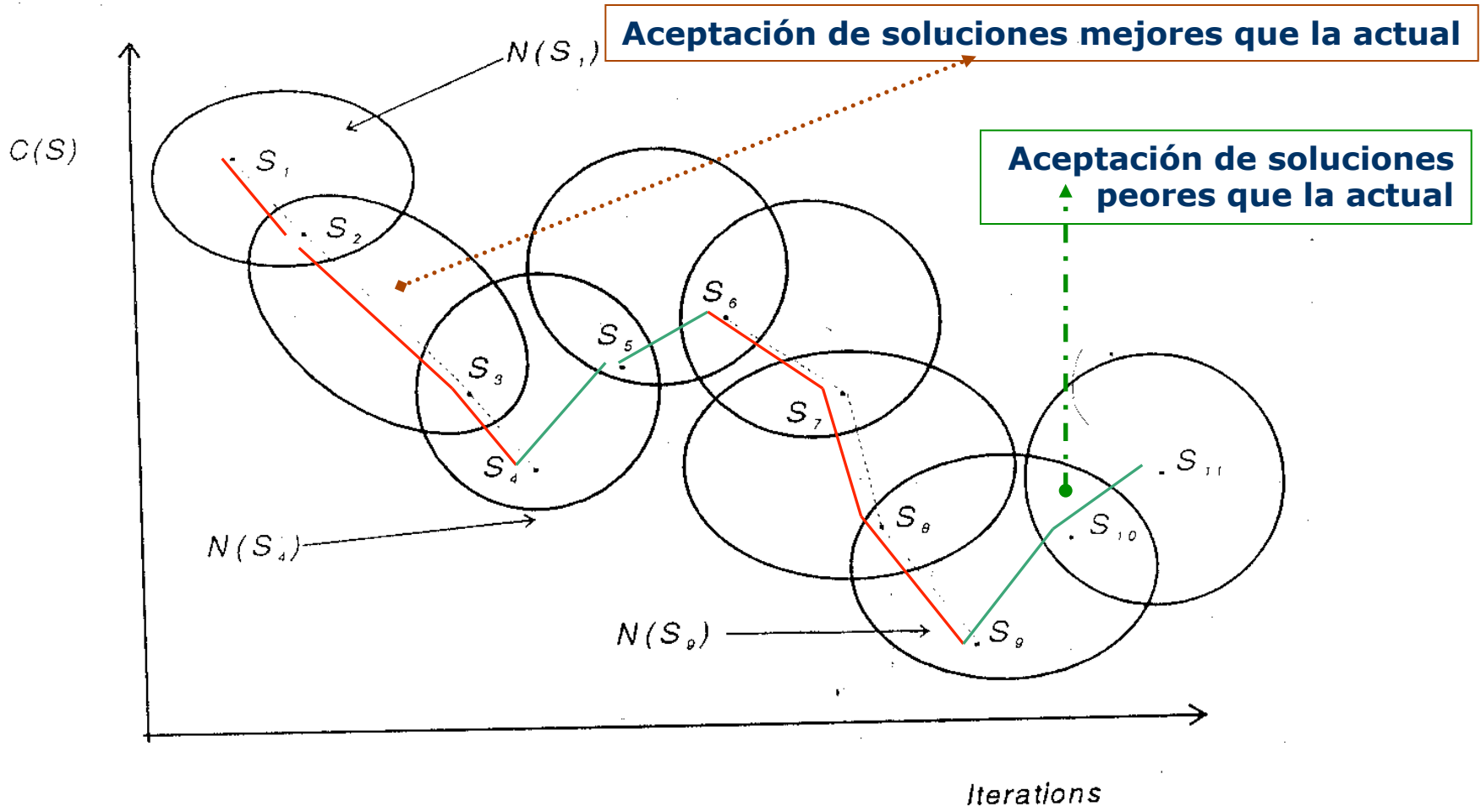



**Figura que muestra una trayectoria de búsqueda basada en entornos**

# 2. BÚSQUEDA ALEATORIA VERSUS BÚSQUEDA LOCAL

**2.1. Introducción**

**2.2. Búsqueda Aleatoria Pura**

**2.3. Búsqueda Aleatoria por Recorrido al Azar**

## 2.1. Introducción

**En esta sección pretendemos estudiar el comportamiento de la búsqueda aleatoria, realizando un estudio de su eficacia y eficiencia, el cual justificará el uso de los procedimientos de búsqueda local.**

# 2. BÚSQUEDA ALEATORIA VERSUS BÚSQUEDA LOCAL

## 2.2. Búsqueda Aleatoria Pura

- **Se elige aleatoriamente una muestra de soluciones del espacio de búsqueda y se devuelve la mejor.**

- **Se diría que el entorno de una solución es todo el espacio de búsqueda:**

**E(s) = "todo el espacio de búsqueda".**

# 2. BÚSQUEDA ALEATORIA VERSUS BÚSQUEDA LOCAL

## Procedimiento Búsqueda Aleatoria Pura

**Inicio**
GENERA(Solución Inicial)
Solución Actual ← Solución Inicial;
Mejor Solución ← Solución Actual;
**Repetir**
GENERA(Solución Actual); **%Generación Aleatoria**
**Si** Objetivo(Solución Actual) **es mejor que** Objetivo(Mejor Solución)
**entonces** Mejor Solución ← Solución Actual;
**Hasta** (Criterio de parada);
DEVOLVER (Mejor Solución);
**Fin**

# 2. BÚSQUEDA ALEATORIA VERSUS BÚSQUEDA LOCAL

***EJEMPLO:***

***Búsqueda Aleatoria Pura para el Viajante de Comercio***

| Iteración | Solución |
|---|---|
| 1 | (1 2 4 3 8 5 9 6 7) |
| 2 | (9 6 4 7 8 5 1 2 3) |
| 3 | (2 4 1 5 8 3 9 7 6) |
| 4 | (4 7 5 1 8 3 2 6 9) |
| 5 | (7 6 9 5 8 3 1 2 4) |
| 6 | (8 3 7 2 1 5 7 6 9) |
| 7 | (2 5 1 3 9 8 4 6 7) |
| 8 | (1 4 2 3 8 5 6 9 7) |
| 9 | (3 4 2 1 5 8 7 9 6) |
| 10 | (7 4 9 3 8 5 6 2 1) |

Una ejecución de la Búsqueda Aleatoria Pura

# 2. BÚSQUEDA ALEATORIA VERSUS BÚSQUEDA LOCAL

## *ESTUDIO TEÓRICO DE LA EFICIENCIA DE LA BÚSQUEDA ALEATORIA PURA*

***Si el problema tiene m soluciones y el óptimo es único, la probabilidad de que al generar aleatoriamente una solución se obtenga la óptima es 1/m.***

***Sean $A_i$ los sucesos siguientes, que determinan la probabilidad absoluta de obtener el óptimo en la iteración i:***

$$P(A_i) = 1/m \ \forall i = 1, \ldots, n.$$

# 2. BÚSQUEDA ALEATORIA VERSUS BÚSQUEDA LOCAL

**La probabilidad de obtener el óptimo en n iteraciones sería:**

$$P(A_1 \cup A_2 \cup \ldots \cup A_n) =$$

$$\begin{aligned}
&= 1 - P((A_1 \cup A_2 \cup \cdots \cup A_n)^c) \\
&= 1 - P(A_1^c \cap A_2^c \cap \cdots \cap A_n^c) \\
&= 1 - P(A_1^c)P(A_2^c)\cdots P(A_n^c) \\
&= 1 - (1 - P(A_1))\,(1 - P(A_2))\cdots(1 - P(A_n)) \\
&= 1 - \left(1 - \frac{1}{m}\right)\left(1 - \frac{1}{m}\right)\cdots\left(1 - \frac{1}{m}\right) \\
&= 1 - \left(1 - \frac{1}{m}\right)^n
\end{aligned}$$

# 2. BÚSQUEDA ALEATORIA VERSUS BÚSQUEDA LOCAL

**La expresión anterior se puede emplear para derivar el valor de *n* que, con una probabilidad suficientemente grande, garantiza la eficacia del método.**

$$P(A_1 \cap A_2 \cap \ldots \cap A_n) = 1 - \left(1 - \frac{1}{m}\right)^n > 1 - \alpha$$

**Si $\alpha$ es tal que $0 < \alpha < 1$ (probabilidad *a priori* de error en la búsqueda, es decir, probabilidad de que la BA no encuentre el óptimo), entonces:**

$$\Longleftrightarrow \quad \alpha > \left(1 - \frac{1}{m}\right)^n$$

$$\Longleftrightarrow \quad log(\alpha) > log\left(\left(1 - \frac{1}{m}\right)^n\right)$$

$$\Longleftrightarrow \quad log(\alpha) > nlog\left(\left(1 - \frac{1}{m}\right)\right)$$

$$\Longleftrightarrow \quad n > \frac{log(\alpha)}{log\,(1 - 1/m)}$$

**Es decir, n sería el nº de iteraciones necesario para garantizar la obtención del óptimo con probabilidad 1-$\alpha$**

# 2. BÚSQUEDA ALEATORIA VERSUS BÚSQUEDA LOCAL

***EJEMPLO:***

***Valores de n (número de iteraciones necesario) para distintos valores de la probabilidad de fallo $\alpha$ y del tamaño del espacio de búsqueda m***

| $P_{fallo}$ $\alpha$ | $P_{éxito}$ $1-\alpha$ | nºsol m | nºIt. n |
|---|---|---|---|
| 0.1 | 0.9 | 1000 | 2302 |
| 0.2 | 0.8 | 1000 | 1609 |
| 0.3 | 0.7 | 1000 | 1203 |
| 0.4 | 0.6 | 1000 | 916 |
| 0.1 | 0.9 | 2000 | 4605 |
| 0.2 | 0.8 | 2000 | 3219 |
| 0.3 | 0.7 | 2000 | 2408 |
| 0.4 | 0.6 | 2000 | 1833 |

| $P_{fallo}$ $\alpha$ | $P_{éxito}$ $1-\alpha$ | nºsol m | nºIt. n |
|---|---|---|---|
| 0.1 | 0.9 | 3000 | 6907 |
| 0.2 | 0.8 | 3000 | 4828 |
| 0.3 | 0.7 | 3000 | 3612 |
| 0.4 | 0.6 | 3000 | 2748 |
| 0.1 | 0.9 | 4000 | 9210 |
| 0.2 | 0.8 | 4000 | 6437 |
| 0.3 | 0.7 | 4000 | 4816 |
| 0.4 | 0.6 | 4000 | 3664 |

**Número de iteraciones para la búsqueda aleatoria pura**

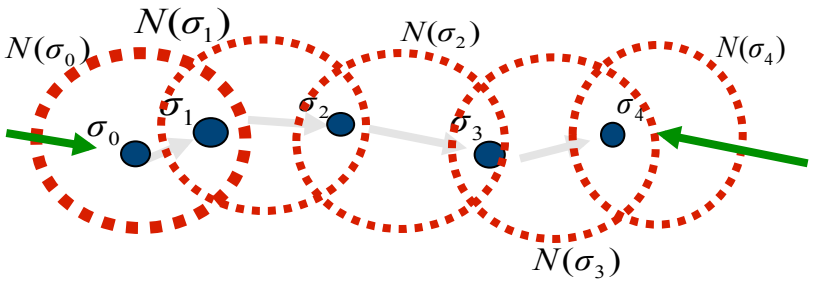
# 2. BÚSQUEDA ALEATORIA VERSUS BÚSQUEDA LOCAL

## 2.3. Búsqueda Aleatoria Por Recorrido al Azar

- **Se obtiene desde la descripción de una Búsqueda por Entornos.**
- **La solución inicial se genera aleatoriamente.**
- **El entorno de cualquier solución es propio (no consta de todas las soluciones del espacio de búsqueda).**
- **La solución vecina a la solución actual se escoge aleatoriamente dentro del entorno y se acepta automáticamente.**
- **Se almacena la mejor solución obtenida hasta el momento. Ésta es la solución que se devuelve finalmente.**

- **En definitiva, puede considerarse como una búsqueda local en la que se acepta el primer vecino generado, independientemente de que sea mejor o peor que la solución actual.**

Solución inicial




Óptimo local/global

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS



## 3.1. Introducción. Procedimiento Base

- **Consiste en el muestreo de soluciones vecinas mejores que la actual en el entorno de ésta.**
- **Hay dos versiones: del Mejor y del Primer Mejor.**
- **En ambos casos, el algoritmo devuelve la última solución visitada (no es necesario ir almacenando la mejor solución).**

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS

**Procedimiento General Búsqueda Local**

**Inicio**

GENERA(Solución Inicial);
Solución Actual ← Solución Inicial;

**Repetir**

GENERA_SOLUCIÓN_ENTORNO(Solución Vecina *tal que*
Objetivo(Solución Vecina) **mejor que** Objetivo(Solución Actual));

Si Objetivo(Solución Vecina) **mejor que** Objetivo(Solución Actual)
entonces Solución Actual ← Solución Vecina;

**Hasta** (Objetivo(Solución Vecina) **peor o igual que**
Objetivo(Solución Actual), ∀ S ∈ E(Solución Actual));

DEVOLVER(Solución Actual);

**Fin**

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS

## 3.2. Búsqueda Local del Mejor

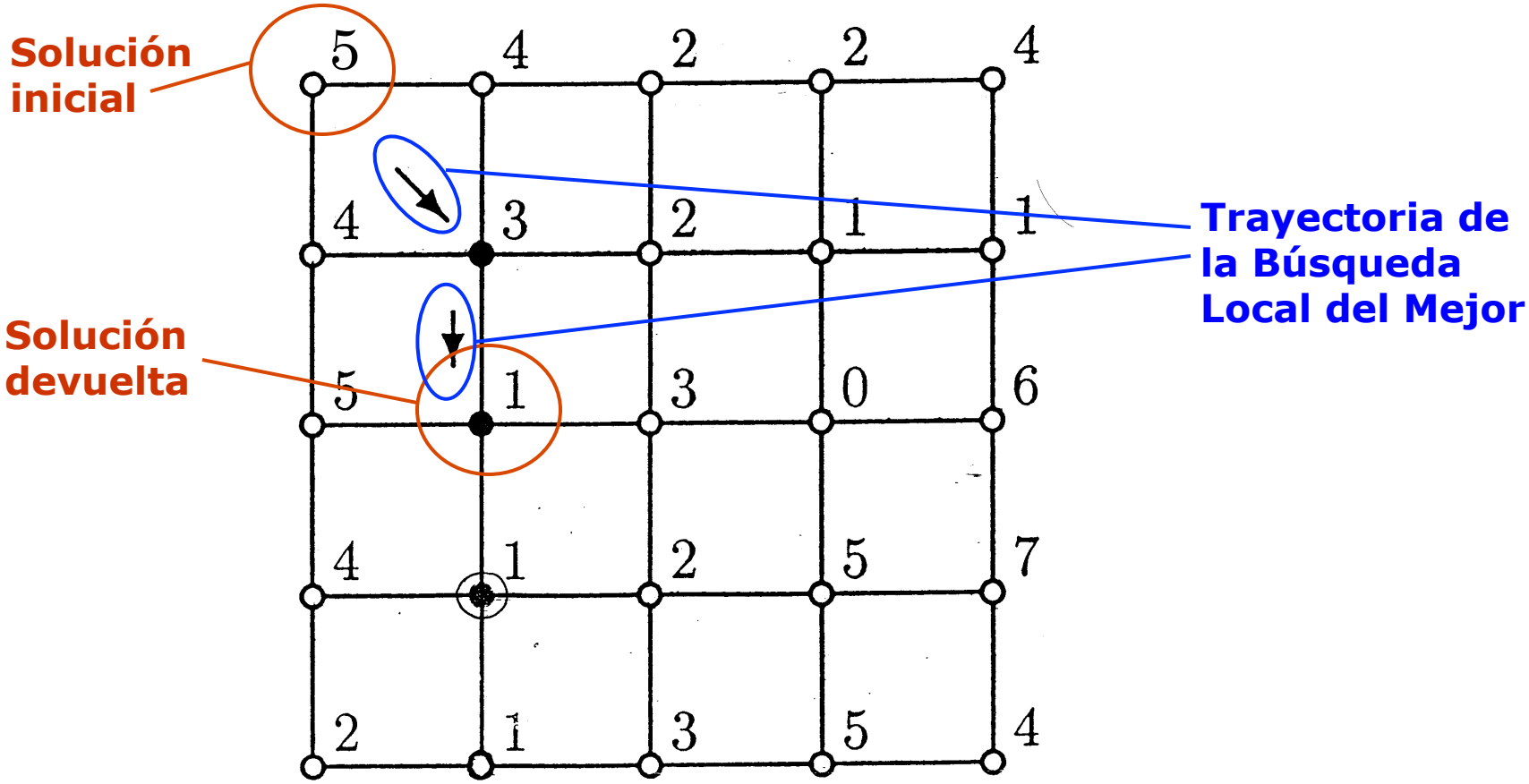
- **Genera el entorno completo de la solución actual y selecciona la mejor solución vecina.**
- **Si ésta es mejor que la solución actual, la sustituye y se continúa la iteración.**
- **En otro caso, el algoritmo finaliza.**

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS

**Procedimiento Búsqueda Local del Mejor**

**Inicio**
  GENERA($S_{act}$);
  **Repetir**
    Mejor Vecino ← $S_{act}$
    **Repetir para** toda S' ∈ E($S_{act}$)
      S' ← GENERA_VECINO($S_{act}$);
      Si Objetivo(S') **mejor que** Objetivo(Mejor Vecino) entonces
        Mejor Vecino ← S';
    **Fin-Repetir-para**
    Si Objetivo(Mejor Vecino) **mejor que** Objetivo($S_{act}$) entonces
      $S_{act}$ ← Mejor Vecino;
  **Hasta** (Objetivo(Mejor Vecino) **peor o igual que** Objetivo($S_{act}$));
  DEVOLVER($S_{act}$);
**Fin**

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS




EJEMPLO: E(s) = {si / si = (xi ± {0,1}, yi ± {0,1}) ∧ si ≠ s}

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS

## 3.3. Búsqueda Local del Primer Mejor

- **Se va generando paso a paso el entorno de la solución actual hasta que se obtiene una solución vecina que mejora a la actual o se construye el entorno completo.**
- **En el primer caso, la solución vecina sustituye a la actual y se continúa iterando.**
- **En el segundo, se finaliza la ejecución del algoritmo.**

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS

**Procedimiento Búsqueda Local del Primer Mejor**

**Inicio**
  GENERA($S_{act}$);

  **Repetir**

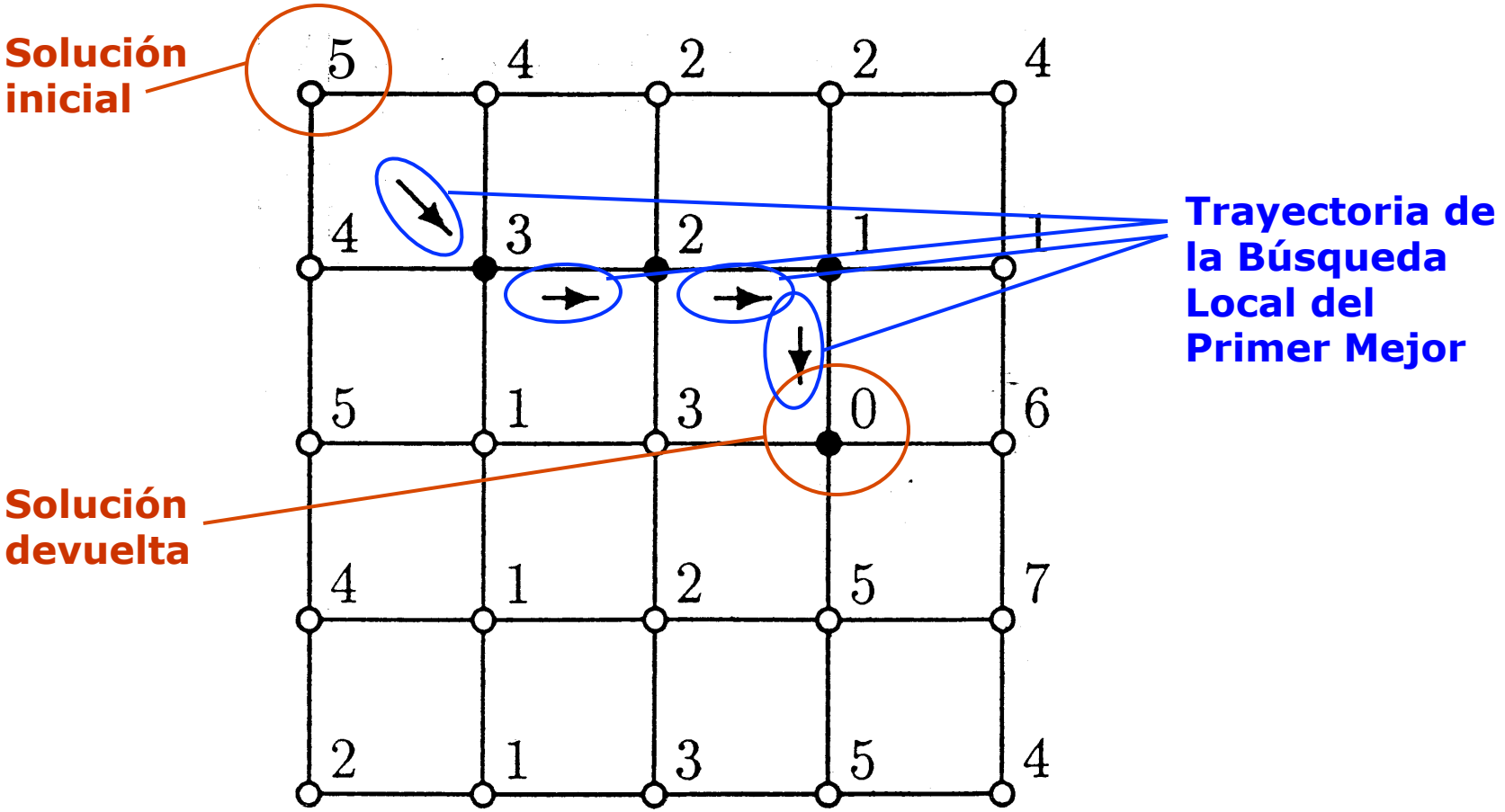
    **Repetir**
      S' ← GENERA_VECINO($S_{act}$);
    **Hasta** (Objetivo(S') **mejor que** Objetivo(Mejor Vecino)) **O**
        (se ha generado E($S_{act}$) al completo)

    Si Objetivo(S') **mejor que** Objetivo($S_{act}$) entonces
      $S_{act}$ ← S';

  **Hasta** (Objetivo(S') **peor o igual que** Objetivo($S_{act}$));

  DEVOLVER($S_{act}$);
**Fin**

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS




EJEMPLO: E(s) = {si / si = (xi ± {0,1}, yi ± {0,1}) ∧ si ≠ s}

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS

## 3.4. Ejemplo: Viajante de Comercio

1. Esquema de representación:
   Permutación de {1, ..., n}.

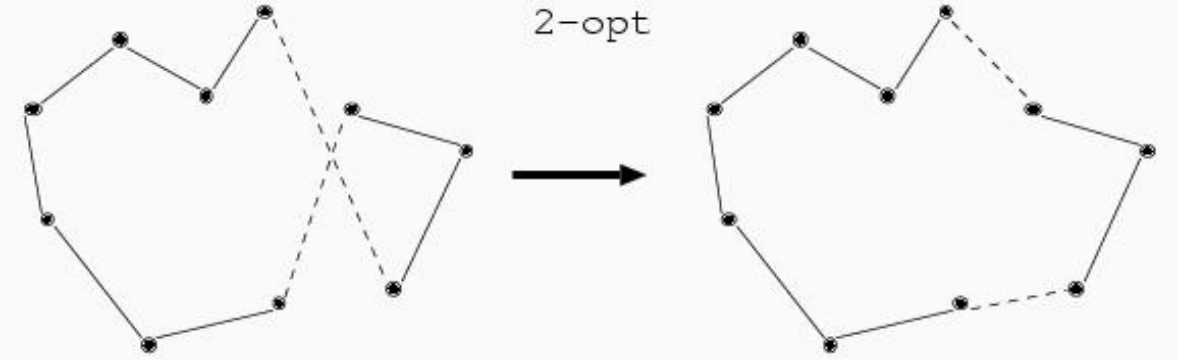
2. Función objetivo:

$$Min\ C(S) = \sum_{i=1}^{n-1} \left( D[S[i], S[i+1]] \right) + D[S[n], S[1]]$$

3. Mecanismo de generación de la solución inicial:
   Permutación *aleatoria.*

# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS

4. Operador de generación de nuevas soluciones: escoger dos posiciones e intercambiar sus valores (2-opt):

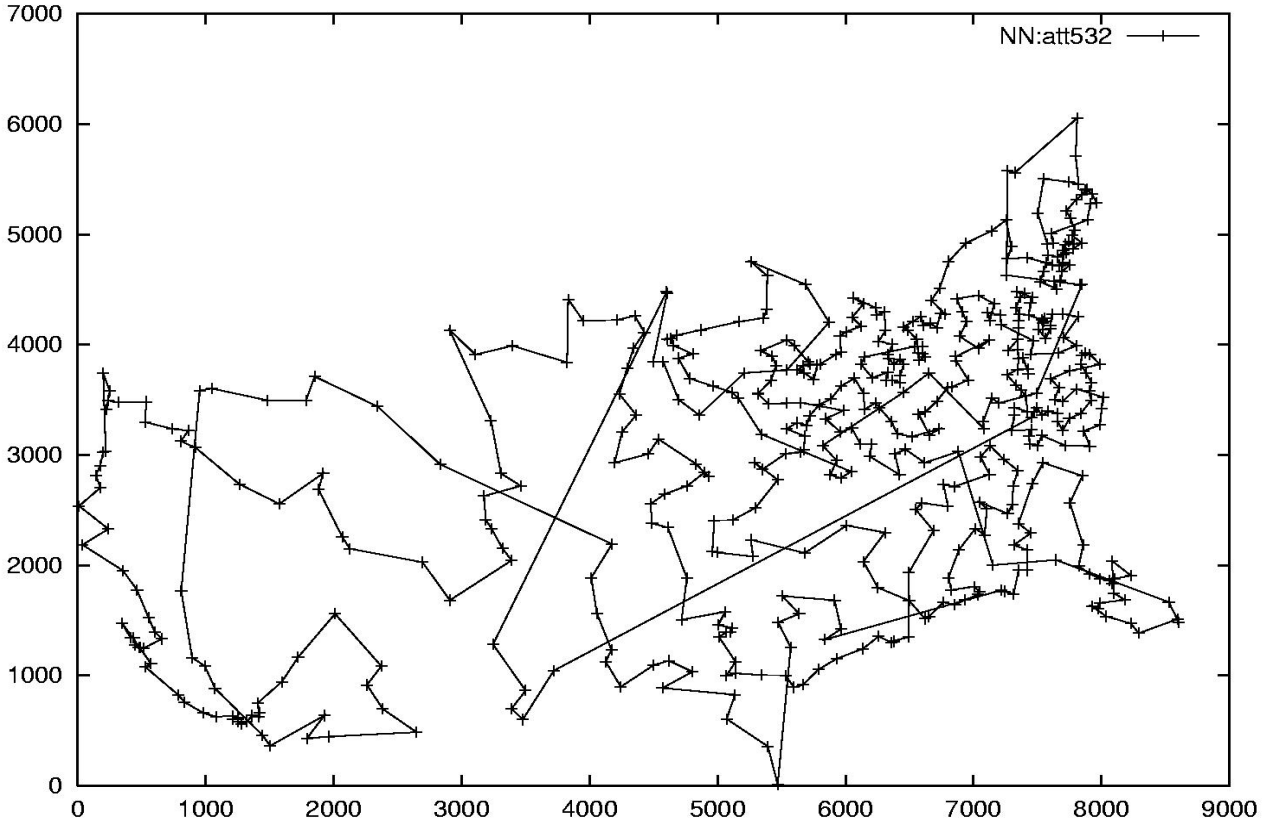



5. Mecanismo de selección: Selección del mejor o el primer mejor.

6. Criterio de parada: Cuando el vecino generado no mejore a la solución actual.

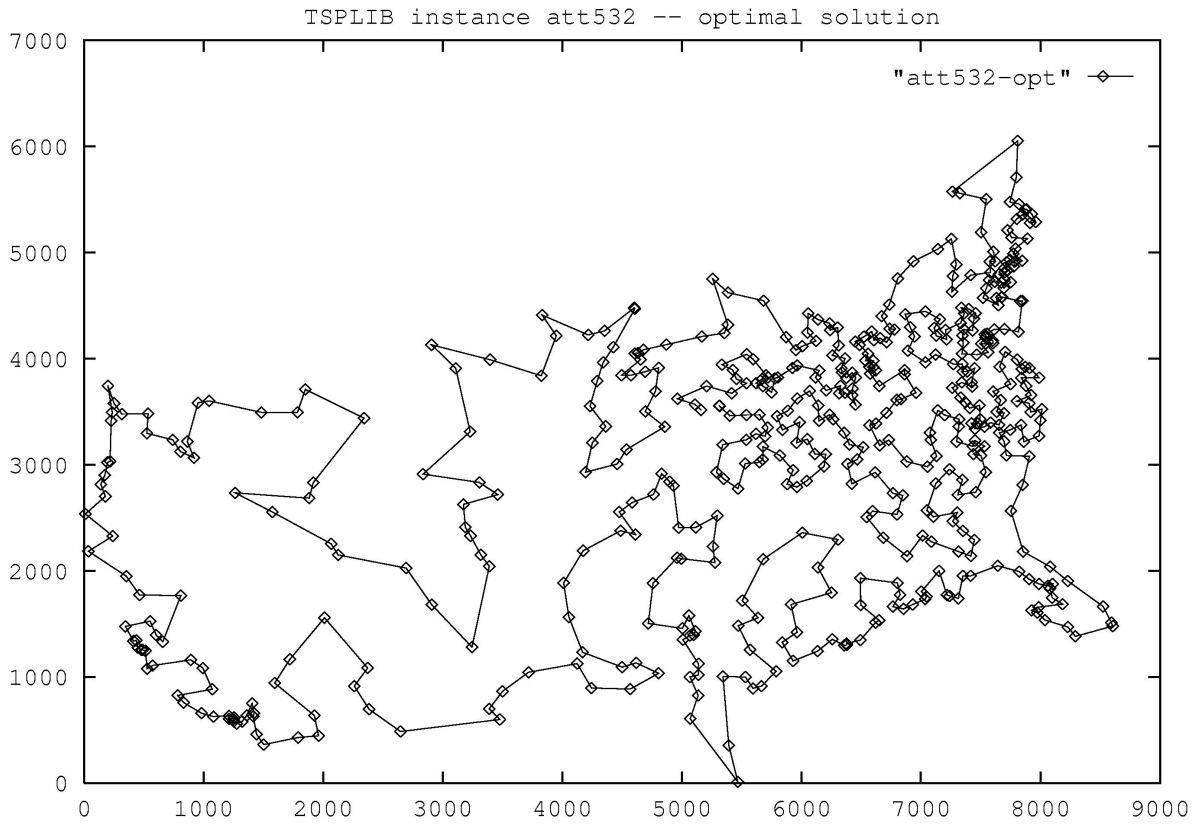
# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS

**Solución obtenida en una ejecución del algoritmo de Búsqueda Local del Mejor**



# 3. MÉTODOS DE BÚSQUEDA LOCAL BÁSICOS

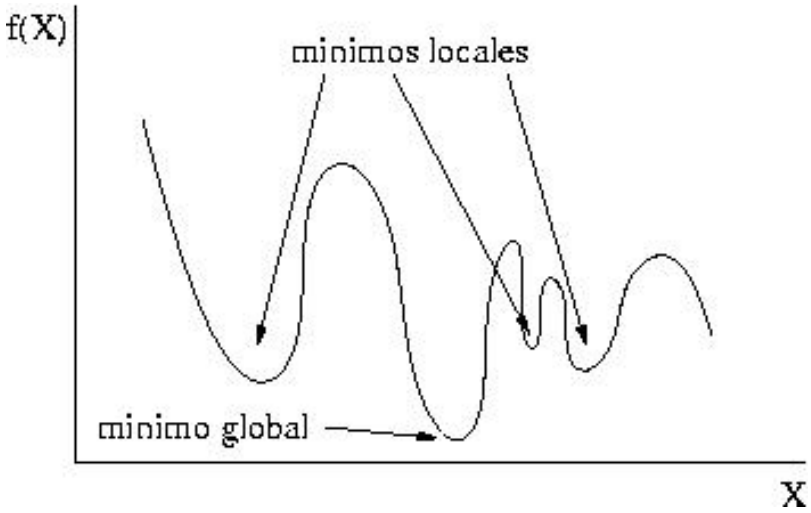
## Solución óptima

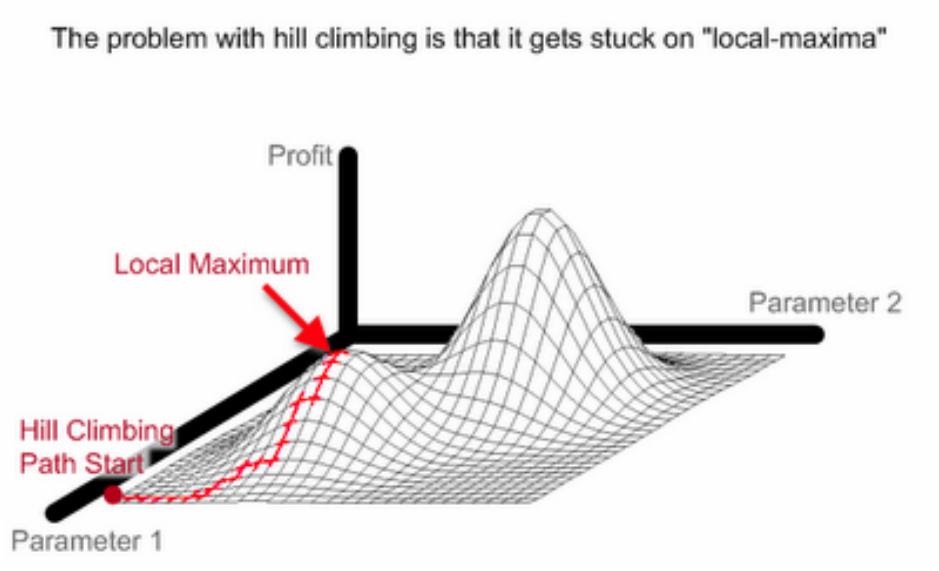


# 3. MÉTODOS DE BÚSQUEDA LOCAL BASICOS

## 3.5. Problemas de la Búsqueda Local

- Suele caer en óptimos locales, que a veces están bastante alejados del óptimo global del problema



# 3. MÉTODOS DE BÚSQUEDA LOCAL BASICOS

## 3.5. Problemas de la Búsqueda Local

**SOLUCIONES:** 3 opciones para salir de los óptimos locales

- Permitir movimientos de empeoramiento de la solución actual

  Ejemplo: Enfriamiento Simulado, Búsqueda Tabú (T3 y 4).

- Modificar la estructura de entornos

  Ejemplo: Búsqueda Tabú, Búsqueda Descendente Basada en Entornos Variables: VND, Búsqueda en Entornos Variables: VNS, … (T4 y 5)

- Volver a comenzar la búsqueda desde otra solución inicial

  Ejemplo: Búsquedas Multiarranque, ILS, VNS, … (T5 y 6).

# 4. UN MÉTODO AVANZADO DE BÚSQUEDA LOCAL: VND

*N. Mladenovic, P. Hansen, Variable Neighborhood Search. Computers Oper. Res. 24:11 (1997) pp. 1097-1100*

P. Hansen, N. Mladenovic, J.A. Moreno Pérez. Búsqueda de Entorno Variable. Revista Iberoamericana de Inteligencia Artificial 19 (2003) 77-92.

**VNS: Búsqueda de Entorno Variable**

**Caso particular de VNS: VND**

**VND: Búsqueda Descendente de Entorno Variable**

La VNS está basada en tres hechos simples:

1. Un mínimo local con una estructura de entornos no lo es necesariamente con otra.
2. Un mínimo global es mínimo local con todas las posibles estructuras de entornos.
3. Para muchos problemas, los mínimos locales con la misma o distinta estructura de entorno están relativamente cerca.

# 4. UN MÉTODO AVANZADO DE BÚSQUEDA LOCAL: VND

- Los hechos 1 a 3 sugieren el empleo de varias estructuras de entornos en las búsquedas locales para abordar un problema de optimización.

- El cambio de estructura de entornos se puede realizar de forma determinística (técnica VND), estocástica (VNS reducida), o determinística y estocástica a la vez (VNS básica).

- La búsqueda VND modifica la estructura de entornos como mecanismo para evitar la obtención de óptimos locales.

- Se basa en el algoritmo de búsqueda local del mejor, permitiendo que el operador de vecino cambie de entorno (ampliándolo) cuando el mejor vecino generado no mejora a la solución actual.

# 4. UN MÉTODO AVANZADO DE BÚSQUEDA LOCAL: VND

Los pasos de la VND se muestran a continuación:

**Inicialización**

- Seleccionar el conjunto de estructura de entornos $E_k$, k = 1, …, $k_{max}$, que se usarán en el descenso.
- Generar una solución inicial s.

**Iteraciones**

- Repetir, hasta que k=$k_{max}$, la siguiente secuencia:

(a) Exploración del entorno: Encontrar la mejor solución s' del k-ésimo entorno de s (s' $\in E_k(s)$);

(b) Moverse o no: Si la solución obtenida s' es mejor que s, hacer s ← s' y k ← 1; en otro caso, hacer k ← k + 1

# 5. APLICACIONES: Representación de Orden. Ej. VIJANTE DE COMERCIO

## Viajante de Comercio

1. Esquema de representación:
   Permutación de {1, ..., n}.

2. Función objetivo:

$$Min\ C(S) = \sum_{i=1}^{n-1} (D[S[i], S[i+1]]) + D[S[n], S[1]]$$

3. Mecanismo de generación de la solución inicial:
   Permutación *aleatoria.*

# 5. APLICACIONES: Representación de Orden

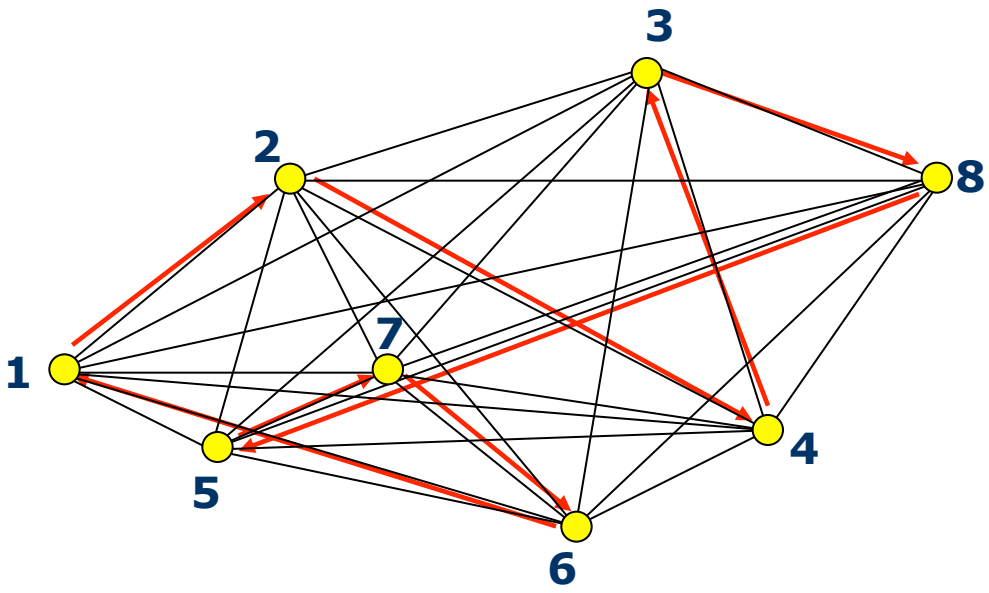
- Se utiliza para problemas donde la solución se representa como una permutación de 1, ..., N

$$X = (x_1, ....., x_n) \quad x_i \in \{1, ..., N\}$$

- Aplicaciones: Viajante de Comercio (TSP), Coloreo de Grafos, Secuenciación de tareas, QAP (asignación cuadrática), ....

# 5. APLICACIONES: Representación de Orden

- Ejemplo: Viajante de Comercio



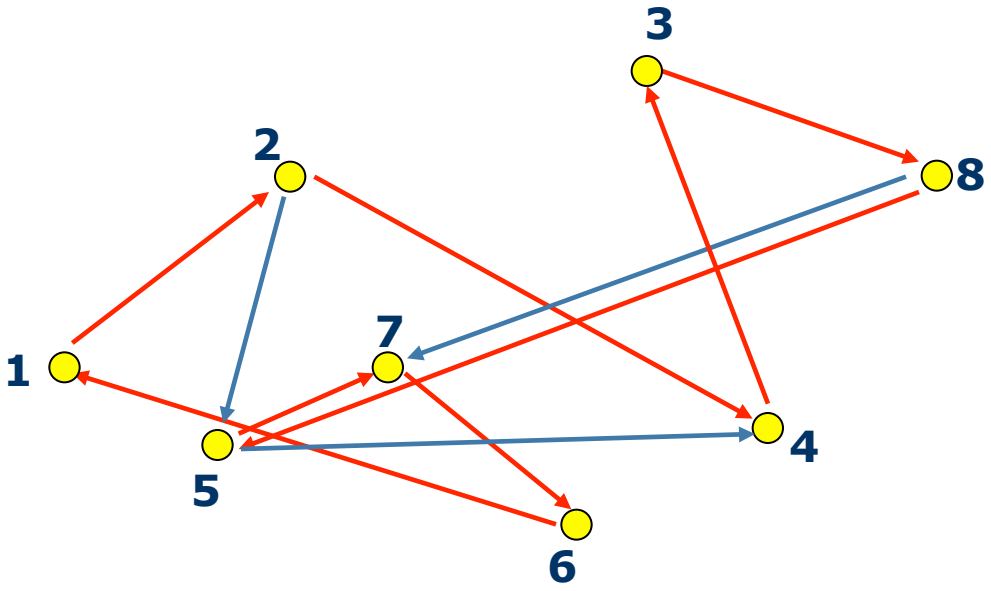

- Representación de una solución: Camino

(1 2 4 3 8 5 7 6)

# 5. APLICACIONES: Operadores de Vecino para Representación de orden

- Posición: se elige un elemento, se extrae, y se inserta en una posición determinada

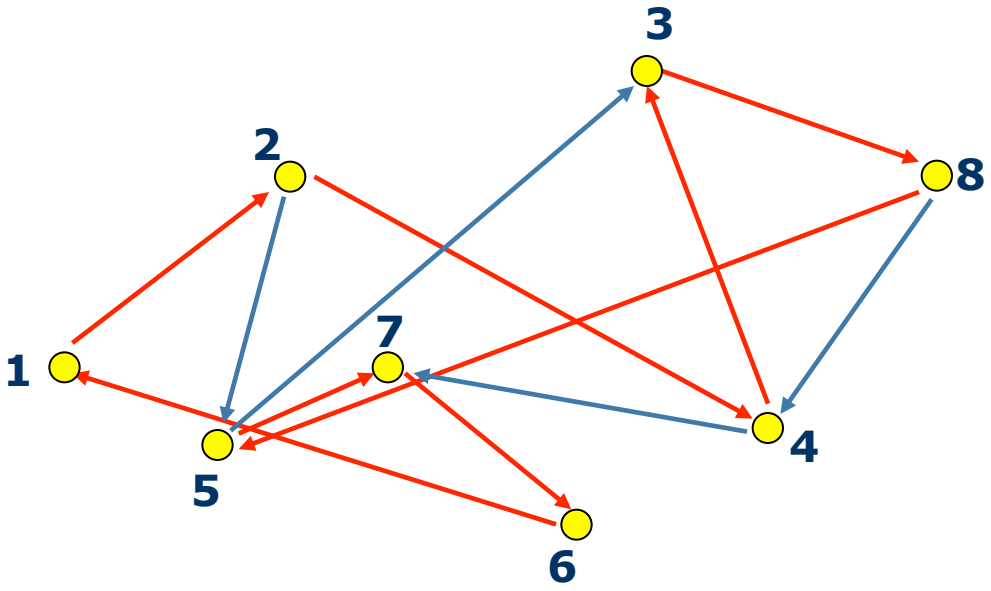
**(1 2 _ 4 3 8 5 7 6)**
**(1 2 5 4 3 8 7 6)**



# 5. APLICACIONES: Operadores de Vecino para Representación de orden

- Intercambio: se escogen dos elementos y se intercambian (2-opt)

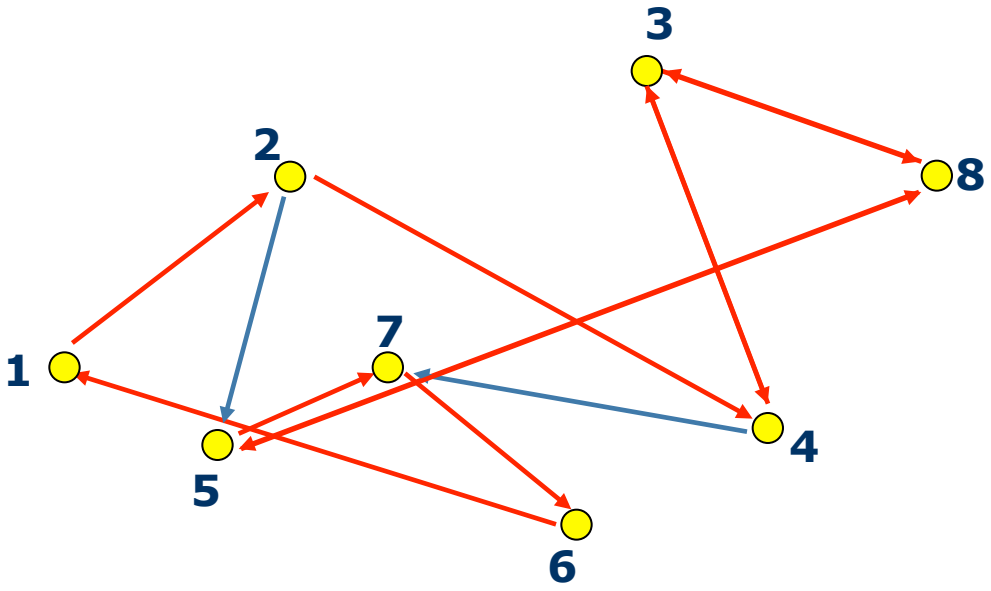
**(1 2 4 3 8 5 7 6)**
**(1 2 5 3 8 4 7 6)**



# 5. APLICACIONES: Operadores de Vecino para Representación de orden

- Inversión simple: se escoge una sublista y se invierte el orden

**(1 2 4 3 8 5 7 6)**
**(1 2 5 8 3 4 7 6)**

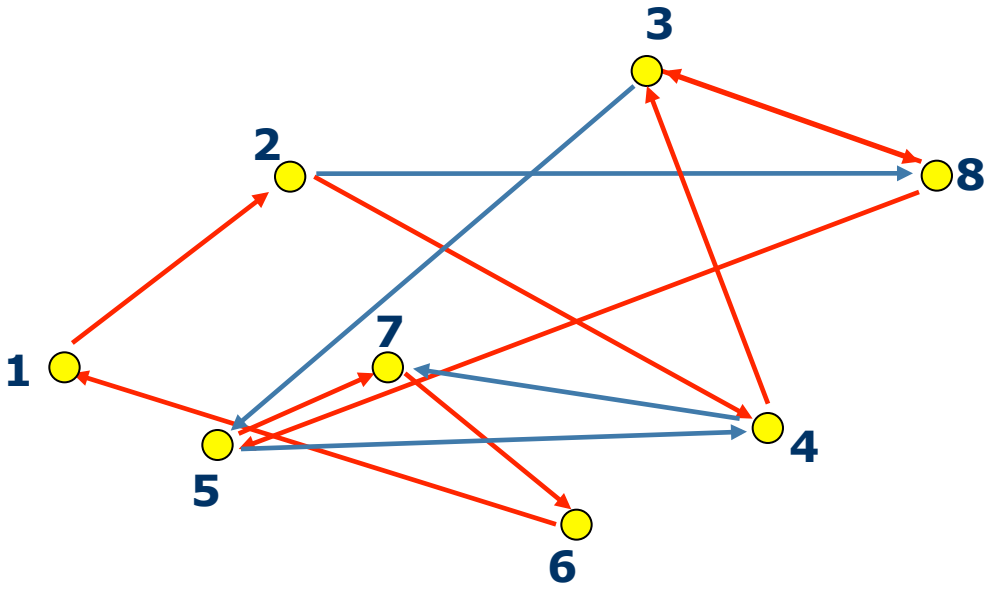


# 5. APLICACIONES: Operadores de Vecino para Representación de orden

- Sublista aleatoria: se escoge una sublista y se baraja. El tamaño de la sublista suele ser fijo y mucho menor que N

**(1 2 4 3 8 5 7 6)**

**(1 2 8 3 5 4 7 6)**



# 5. APLICACIONES: Operadores de Vecino para Representación de orden

- Preguntas:
  - ¿Cómo de diferente puede llegar a ser la solución vecina respecto de la solución actual según el operador empleado?
  - ¿Cuál es el tamaño del entorno generado por cada operador de vecino?
  - El efecto al aplicar un operador concreto, ¿es el mismo en todos los problemas? Por ejemplo, ¿es lo mismo generar un vecino por posición en TSP que en otro problema?

# 5. APLICACIONES: Representación Binaria

- Entre los algoritmos de búsqueda local aplicados a la Selección de Instancias, podemos destacar el *Random Mutation Hill Climbing.*

## Representación

- En este algoritmo de búsqueda local, las soluciones al problema se representan como vectores de dimensión *n* cuyos valores pertenecen al dominio binario {0, 1}, indicando la pertenencia o no de las instancias al subconjunto seleccionado.

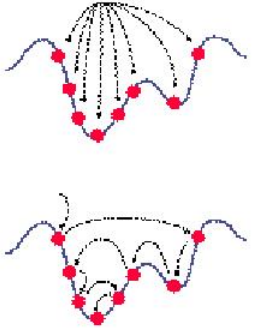
## Función objetivo (maximización):

$$F(S) = \alpha \cdot tasa\_clas + (1 - \alpha) \cdot porc\_red$$

# 5. APLICACIONES: Representación Binaria

## Estructura del algoritmo y operador de generación de vecino

- Se parte de una solución inicial que puede ser obtenida como una generación binaria aleatoria o bien a partir de un algoritmo *greedy.*
- Se itera un bucle en el que se genera una solución vecina de la actual y se sustituye por ella siempre que tenga mayor coste.
- El operador de vecino es el de *inversión* (*1-opt*), que consiste en cambiar o invertir el valor de una posición dentro del vector.
- Para generar el vecino, se sigue la técnica del *primer mejor intercambio*: En cada iteración se busca el intercambio que produce una solución con mayor coste que la solución actual.
- El algoritmo se detiene cuando no existe un intercambio que produzca un vecino con mayor coste que el actual.



# ALGORÍTMICA
## 2012 - 2013

- Parte I. Introducción a las Metaheurísticas
  - Tema 1. Metaheurísticas: Introducción y Clasificación
- Parte II. Métodos Basados en Trayectorias y Entornos
  - Tema 2. Algoritmos de Búsqueda Local Básicos
  - Tema 3. Algoritmos de Enfriamiento Simulado
  - Tema 4. Algoritmos de Búsqueda Tabú
  - Tema 5. Métodos Basados en Trayectorias Múltiples I: Métodos Multiarranque Básicos y GRASP
  - Tema 6. Métodos Basados en Trayectorias Múltiples II: ILS y VNS
- Parte III. Métodos Basados en Poblaciones
  - Tema 7. Algoritmos Genéticos
- Parte IV. Intensificación y Diversificación
  - Tema 8. Estudio del Equilibrio entre Intensificación y Diversificación
- Parte V. Metaheurísticas Híbridas: Poblaciones y Trayectorias
  - Tema 9. Algoritmos Meméticos
  - Tema 10. Modelos Híbridos II: *Scatter Search*
- Parte VI. Paralelización de Metaheurísticas
  - Tema 11. Metaheurísticas en Sistemas Descentralizados. Metaheurísticas paralelas
- Parte VII. Conclusiones
  - Tema 12. Algunas Consideraciones sobre la Adaptación de Metaheurísticas a la Resolución de Problemas