Introducción a las Metaheurísticas

José Manuel Galán
Luis R. Izquierdo
José I. Santos

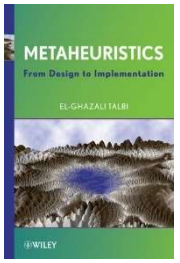
Universidad de Burgos

Estructura de la presentación

Bibliografía
Problemas de optimización combinatoria
Metaheurísticas
Clasificación de metaheurísticas
Mecanismos
¿cuándo utilizar metaheurísticas?
Búsqueda aleatoria
Búsqueda local
Un ejemplo. El TSP
Problemas de búsqueda local
Resumen

Bibliografía

- Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons
- Hillier, F.S., Lieberman, G.S. (2008) Introducción a la Investigación de Operaciones. McGraw-Hill. México D.F.
- B. Melián, J.A. Moreno Pérez, J.M. Moreno Vega. Metaheurísticas: un visión global. *Revista Iberoamericana de Inteligencia Artificial* 19 (2003) 7-28
- A. Duarte Muñoz et al (2007) Metaheurísticas. Dykinson. Madrid.
- S. Luke (2013) Essentials of Metaheuristics. Lulu. Available for free at http://cs.gmu.edu/$\sim$sean/book/metaheuristics/
- Zäpfel, G., & Braune, R. (2010). Metaheuristic search concepts: A tutorial with applications to production and logistics. Springer Science & Business Media.
- Apuntes sobre algoritmia y metaheurísticas de la Universidad de Granada http://sci2s.ugr.es/docencia/metah/



Optimización combinatoria

- Existe un conjunto de problemas reales que consiste en encontrar una solución óptima dentro de un conjunto finito de opciones. Estos problemas son considerados con frecuencia como muy difíciles.
  - el camino más corto entre varios puntos,
  - un plan de mínimo coste para repartir mercancías a clientes,
  - una asignación óptima de trabajadores a tareas a realizar,
  - una secuencia óptima de proceso de trabajos en una cadena de producción,
  - una distribución de tripulaciones de aviones con mínimo coste,
  - el mejor enrutamiento de un paquete de datos en Internet,
  - ...

Source: http://sci2s.ugr.es/docencia/metah/

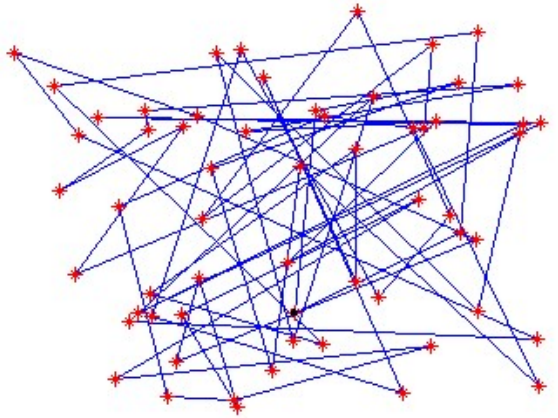
Optimización combinatoria. Algoritmos de búsqueda

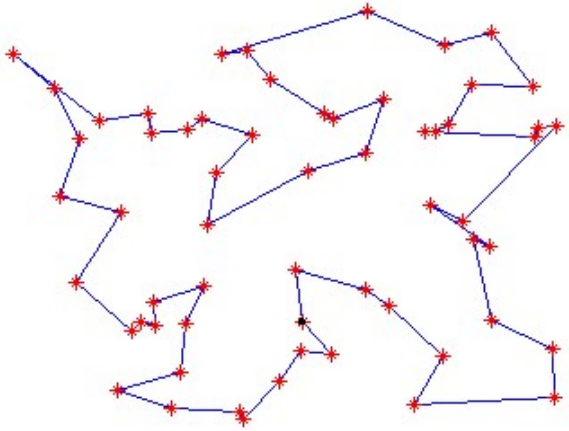
- Estos problemas se caracterizan por:
  - Suelen requerir agrupamientos, ordenaciones o asignaciones de un conjunto discreto de objetos que satisfagan ciertas restricciones
  - Es relativamente fácil evaluar una solución individual (*I know it when I see it* problems)
  - Se pueden aplicar a muchos dominios diferentes
  - Presentan una gran complejidad computacional (son NP duros), así, los algoritmos exactos (Programación Dinámica, Backtracking, Branch and Bound, ...) son ineficientes o simplemente imposibles de aplicar
  - Encontrar una solución aproximada, no necesariamente el óptimo, en un tiempo razonable es una opción viable

Source: http://sci2s.ugr.es/docencia/metah/

Un problema clásico. El problema del viajante. *Travelling salesman problem* (TSP)

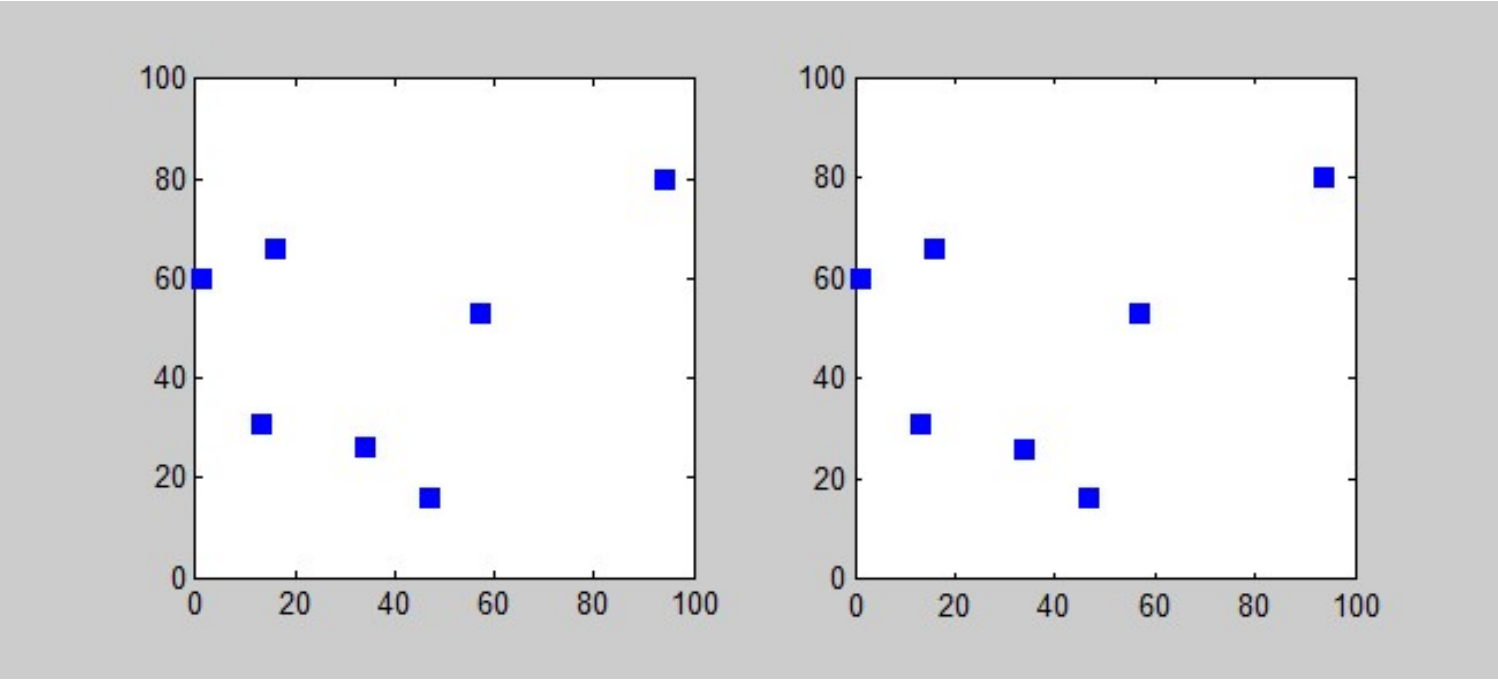
- Dada una red de nodos representando, por ejemplo, ciudades o lugares. Un individuo, un viajante, debe visitar cada ciudad o lugar exactamente una vez y volver al lugar de origen
- Cada enlace ij de la red representa el coste o distancia entre un par de lugares (C_{ij})
- Se trata de un problema muy popular con aplicaciones a logística, problemas de entrega o fabricación de circuitos integrados





Source: https://en.wikipedia.org/wiki/Travelling_salesman_problem
http://sci2s.ugr.es/docencia/metah/

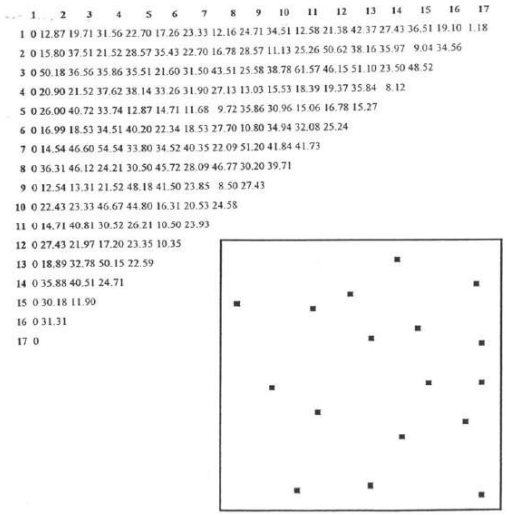
Un problema clásico. El problema del viajante. *Travelling salesman problem* (TSP)



Source: https://en.wikipedia.org/wiki/Travelling_salesman_problem

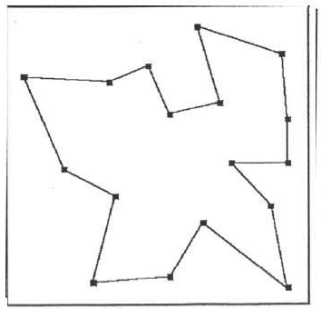
Un problema clásico. El problema del viajante. *Travelling salesman problem* (TSP)

- El problema reside en el número de posibles combinaciones que viene dado por el factorial del número de ciudades (N!) y esto hace que la solución por fuerza bruta sea impracticable para valores de N incluso moderados con los medios computacionales actualmente a nuestro alcance. Por ejemplo, si un ordenador fuese capaz de calcular la longitud de cada combinación en un microsegundo, tardaría algo más 3 segundos en resolver el problema para 10 ciudades, algo más de medio minuto en resolver el problema para 11 ciudades y **77.146 años en resolver el problema para sólo 20 ciudades.**



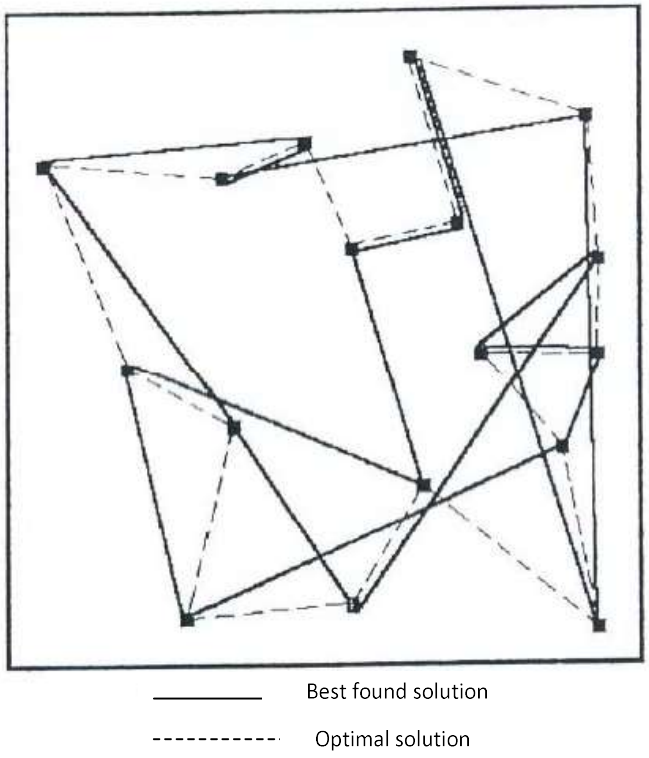
17! (3.55* 10^{14} soluciones posibles)

Solución óptima. Coste: 226.64

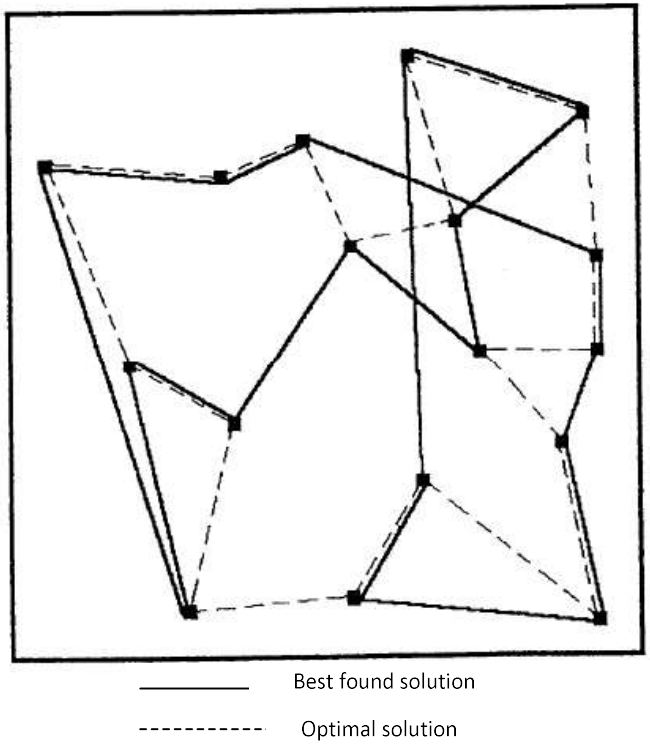


Source: http://sci2s.ugr.es/docencia/metah/

Un problema clásico. El problema del viajante. *Travelling salesman problem* (TSP)



Iteración: 0 Coste: 403.7

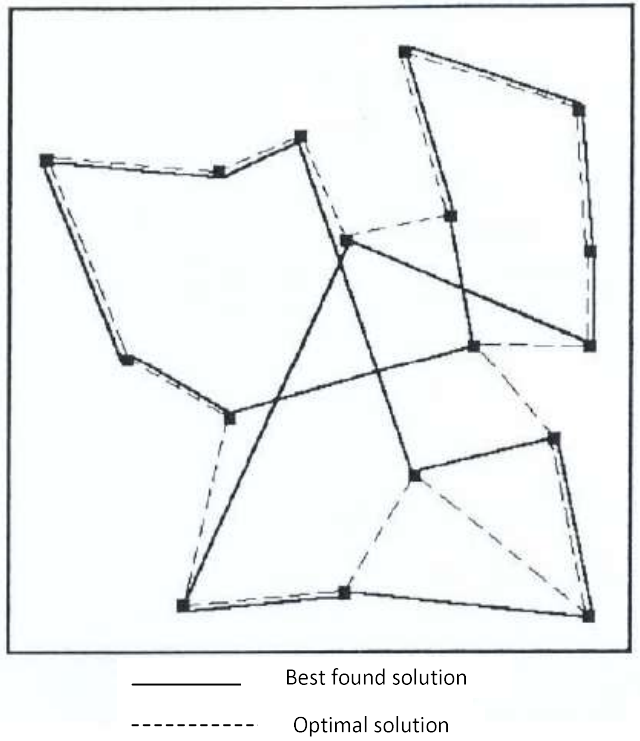


Iteración: 25 Coste: 303.86

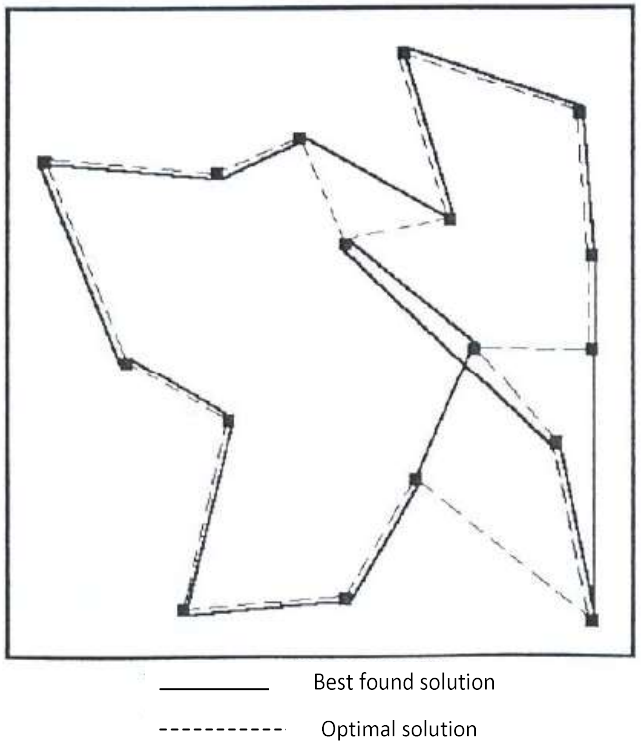
Solución óptima: 226.64

Source: http://sci2s.ugr.es/docencia/metah/

Un problema clásico. El problema del viajante. *Travelling salesman problem* (TSP)



Iteración: 50 Coste: 293.6

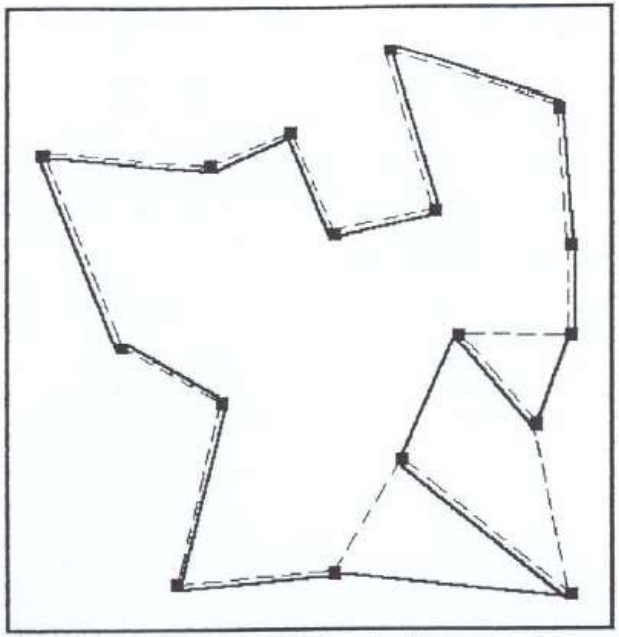


Iteración: 100 Coste: 256.55

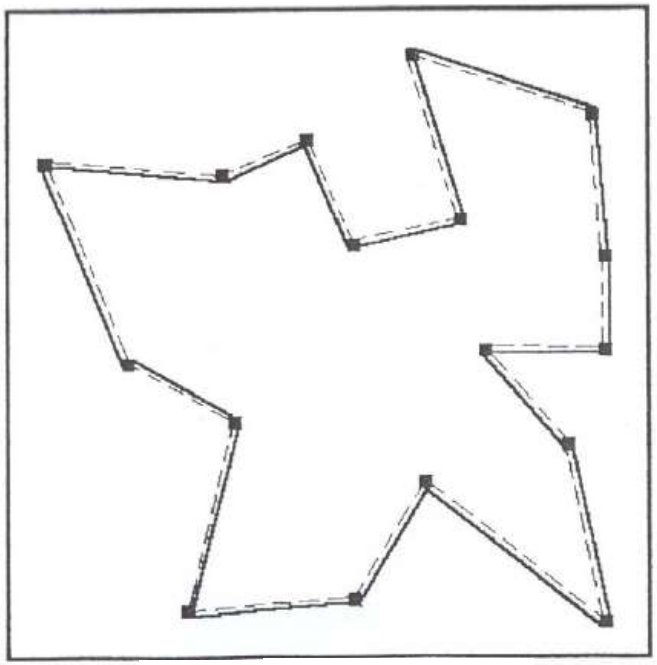
Solución óptima: 226.64

Source: http://sci2s.ugr.es/docencia/metah/

Un problema clásico. El problema del viajante. *Travelling salesman problem* (TSP)



Iteración: 200 Coste: 231.4



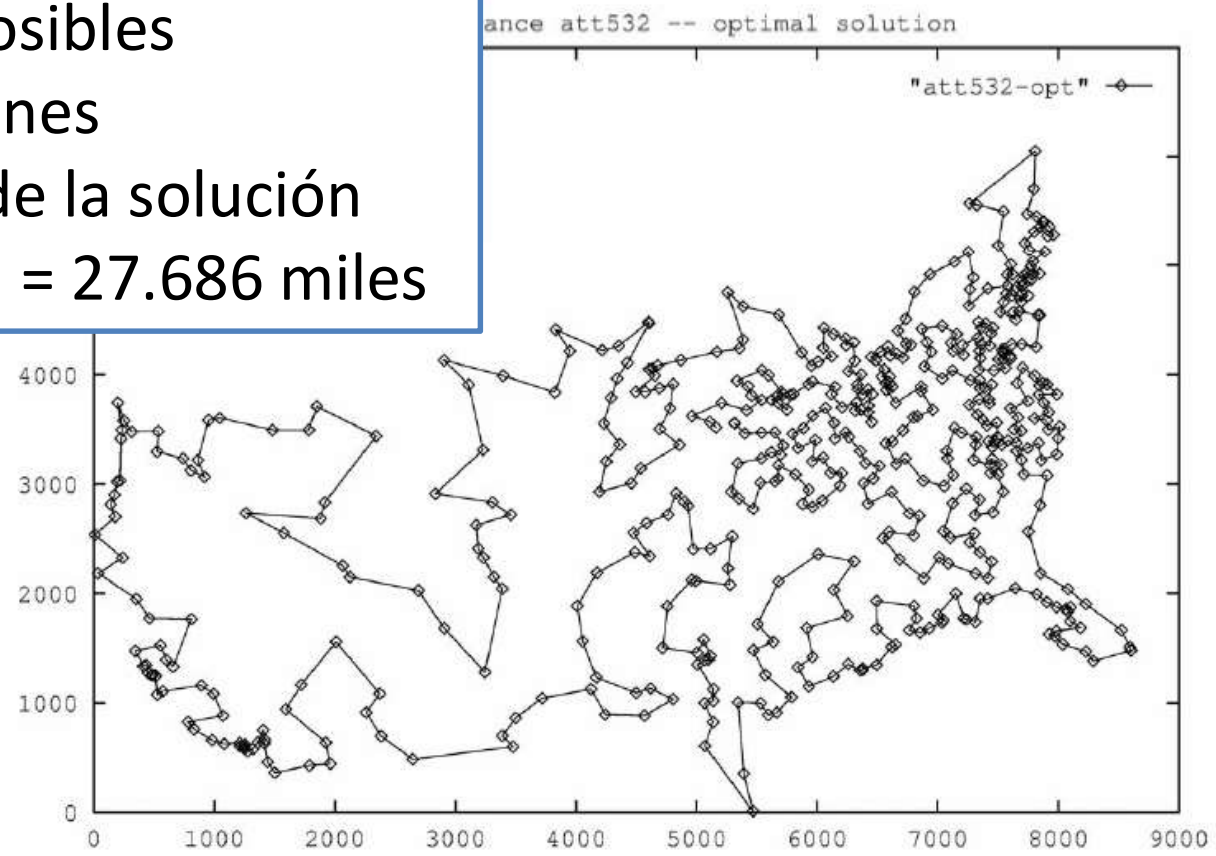
Iteración: 250
Coste: 226.64

Solución óptima: 226.64

Source: http://sci2s.ugr.es/docencia/metah/

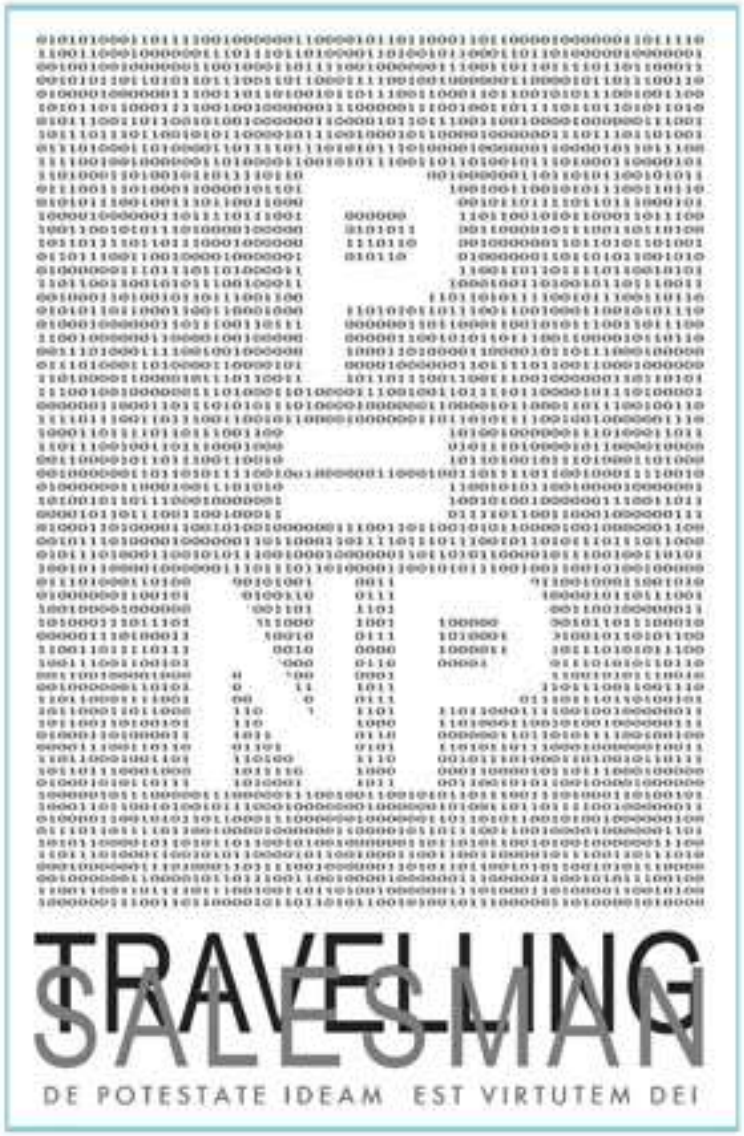
Un problema clásico. El problema del viajante. *Travelling salesman problem* (TSP)

532! Posibles soluciones
Coste de la solución óptima = 27.686 miles



Source: http://sci2s.ugr.es/docencia/metah/

Problemas de optimización combinatoria



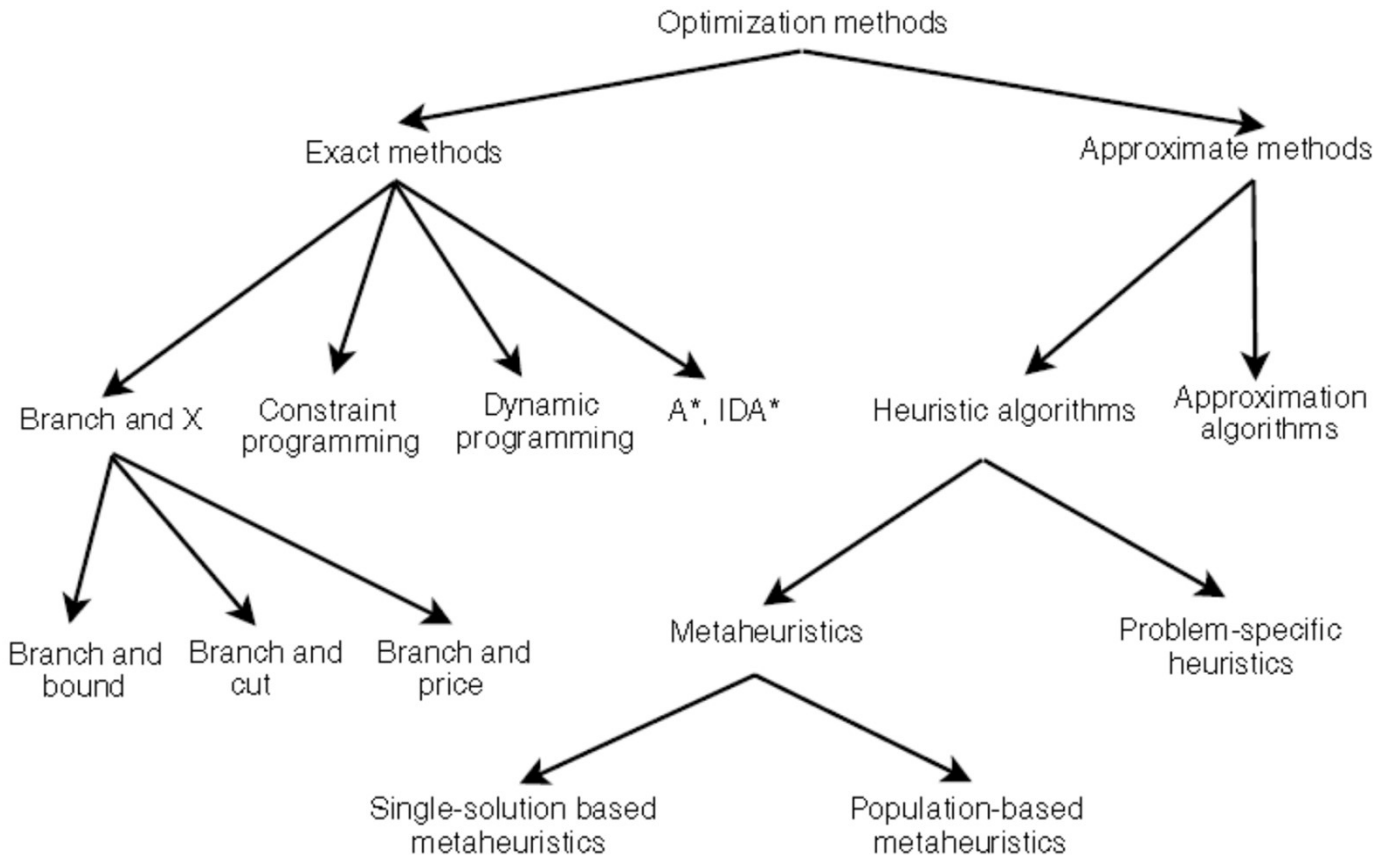
Youtube trailer: https://www.youtube.com/watch?v=6ybd5rbQ5rU

Algoritmos aproximados

- Se utilizan para encontrar soluciones aproximadas a problemas de optimización. Están frecuentemente asociados a problemas NP-duros. Al contrario que las metaheurísticas, que normalmente buscan soluciones buenas en tiempos cortos, se buscan cotas en la calidad de las soluciones y e tiempo de ejecución.
- Son apropiados cuando:
  - Existe un método exacto que es ineficiente (requiere demasiado tiempo de ejecución y/o memoria)
  - Cuando una solución óptima no es necesaria y una buena solución en un tiempo razonable es aceptable

Source: http://sci2s.ugr.es/docencia/metah/

Métodos de optimización clásicos



Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Estado del arte de los problemas de optimización

TABLE 1.4 Order of Magnitude of the Maximal Size of Instances that State-of-the-Art Exact Methods can Solve to Optimality

Optimization Problems	Quadratic Assignment	Flow-Shop Scheduling (FSP)	Graph Coloring	Capacitated Vehicle Routing
Size of the instances	30 objects	100 jobs 20 machines	100 nodes	60 clients

For some practical problems, this maximum size may be negligible. For the TSP problem, an instance of size 13,509 has been solved to optimality [32].

(grid computing platforms) composed of more than 2000 processors with more than 2 months of computing time [546]!

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Metaheurísticas

- El término fue acuñado por Glover (1986)
- *"A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms"* (Sörensen and Glover, 2013)
- *"a metaheuristic is a higher-level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity"* (Bianchi et al, 2009)
- *"Metaheuristics is a rather unfortunate term often used to describe a major subfield, indeed the primary subfield, of stochastic optimization.* ***Stochastic optimization*** *(also called randomized optimization) is the general class of algorithms and techniques which employ some degree of randomness to find optimal (or as optimal as possible) solutions to hard problems."* (Luke 2013)

Source: Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5), 533-549.
Bianchi, L.; Dorigo, M.; Gambardella, L.M.; Gutjahr, W.J. (2009). "A survey on metaheuristics for stochastic combinatorial optimization". Natural Computing: an international journal. 8 (2): 239–287. doi:10.1007/s11047-008-9098-4
Sörensen K. and Glover. F. Metaheurísticas. In S.I. Gass and M. Fu, editors, *Encyclopedia of Operations Research and Management Science*, pages 960–970. Springer, New York, 2013.
S. Luke (2013) Essentials of Metaheurísticas. Lulu.

Metaheurísticas

- Ventajas
  - De propósito general
  - Buenos resultados en la práctica
  - Fácil implementación
  - Fácil paralelización
- Desventajas
  - Son algoritmos aproximados no exactos
  - No son deterministas
  - En algunos casos no están muy fundamentados teóricamente

Source: http://sci2s.ugr.es/docencia/metah/

Taxonomía

- Existen múltiples criterios para clasificar las técnicas metaheurísticas. Los más habituales son:
  - Inspiradas en la naturaleza versus no inspiradas
  - Utilización de memoria versus métodos sin memoria
  - Deterministas versus estocásticos
  - Iterativos versus glotones (*greedy*)
  - Basados en poblaciones versus basados en trayectorias

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Inspiradas en la naturaleza versus no inspiradas en la naturaleza

- Muchas metaheurísticas están inspiradas en procesos naturales: los algoritmos evolutivos y los sistemas inmunes artificiales en la biología; las hormigas, las colonias de abejas o la optimización de partículas de enjambre (PSO) en la inteligencia enjambre de diferentes especies (ciencias sociales); y el recocido simulado de la termodinámica y la física.

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Utilización de memoria versus métodos sin memoria

- Algunos algoritmos metaheurísticos no tienen memoria, es decir, ninguna información obtenida dinámicamente de la búsqueda se utiliza posteriormente. Algunos algoritmos representativos de esta aproximación son la búsqueda local, GRASP o el recocido simulado. Otras técnicas metaheurísiticas utilizan la memoria para almacenar y utilizar información sobre la búsqueda. Por ejemplo, la memoria a corto y largo plazo de la búsqueda tabú.

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Deterministas versus estocásticos

- Existen algoritmos metaheurísticos que tratan de resolver un problema de optimización mediante exclusivamente decisiones deterministas (e.g., búsqueda local, búsqueda tabú). En las metaheurísticas estocásticas, algunas reglas aleatorias se utilizan durante la búsqueda (e.g. el recocido simulado, los algoritmos evolutivos). En un algoritmo determinista, partiendo de la misma solución inicial conducirá a la misma solución final, mientras que en las metaheurísticas estocásticas se pueden obtener soluciones diferentes partiendo de la misma solución inicial. Esta característica se debe tener en cuenta en la evaluación del performance de los algoritmos metaheurísticos.

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Iterativos versus glotones (*greedy*)

- En los algoritmos iterativos, se parte de una solución complete (o de una población de soluciones) que se transforma en cada iteración de acuerdo a algún operador de búsqueda. Los algoritmos *greedy* parten de una solución vacía, en cada paso una variable de decisión del problema es añadida hasta que se obtiene una solución completa. La gran mayoría de las metaheurísticas son algoritmos iterativos.

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

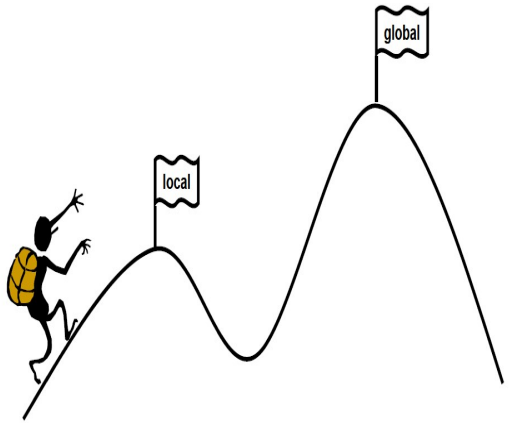
Basados en poblaciones versus basados en trayectorias

- Los algoritmos basado en trayectorias (o en soluciones únicas) (e.g. búsqueda local, el recocido simulado) manipulan y transforma una única solución durante la ejecución del algoritmo, mientras que en los algoritmos basados en poblaciones (e.g. algoritmos de enjambre, algoritmos evolutivos) se evoluciona una población completa. Ambas familias de algoritmos tienen características complementarias: las metaheurísticas basadas en poblaciones están más orientadas hacia la explotación e intensificación de la búsqueda en regiones locales. La metaheurísticas basadas en poblaciones están más orientadas hacia la exploración; en principio presentan buenas característica de cara a la diversificación en el espacio de búsqueda.

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Basadas en soluciones únicas o trayectorias

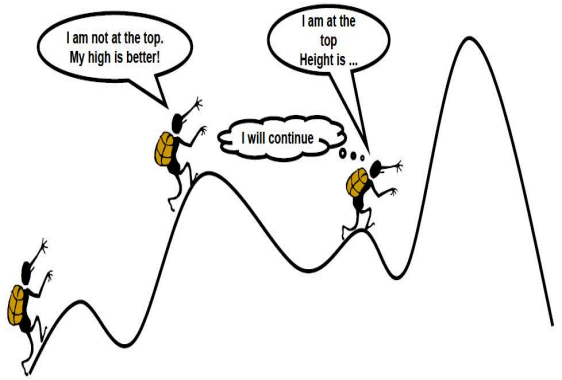
- Las soluciones basadas en trayectorias se basan en ir modificando y mejorando una única solución candidata iterativamente explotando las estructuras de entornos; ejemplos de este tipo de aproximación:
  - **Recocido simulado**
  - Búsqueda Iterativa Local
  - Búsqueda Tabú
  - VNS Variable neighborhood search
  - Búsqueda local guiada
  - …



Source: Vega, J. M. M., Batista, M. B. M., & Pérez, J. A. M. (2003). Metaheurísticas: Una visión global. Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial, 7(19), 7-28.

Algoritmos basados en poblaciones

- Los algoritmos basados en poblaciones mantienen y mejoran múltiples soluciones candidatas de forma simultánea, a menudo utilizando características de la población para guiar la búsqueda; algoritmos basados en poblaciones son:
  - La computación evolutiva
  - **Los algoritmos genéticos**
  - Particle swarm optimization PSO
  - Los algoritmos meméticos
  - …



Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Mecanismos

- Normalmente para obtener buenas soluciones es necesario establecer un balance adecuado entre dos objetivos contrapuestos del proceso:
  - Diversificación o exploración: generar soluciones diversas de forma que se explore el espacio de búsqueda a una escala global
  - Intensificación o explotación: focalizar la búsqueda en una región concreta potencialmente buena
- Se necesita un buen balance para:
  - Identificar rápidamente regiones del espacio con soluciones de buena calidad
  - No consumir mucho tiempo en regiones del espacio no prometedoras o ya exploradas
- Básicamente una técnica metaheurística utiliza diferentes estrategias para balancear estas dos fuerzas. De hecho, prácticamente todas las metaheurísticas podrían considerarse esencialmente como una combinación elaborada de búsqueda aleatoria y descenso de gradiente

Source: http://www.scholarpedia.org/article/Metaheuristic_Optimization
http://sci2s.ugr.es/docencia/metah/
S. Luke (2013) Essentials of Metaheurísticas. Lulu.

¿Cuándo utilizar técnicas metaheurísticas?

- La complejidad de un problema da indicación de su dificultad
  - Pero es importante tener en cuenta el tamaño. Instancias pequeñas de un problema complejo pueden ser resueltas de forma exacta, mientras que problemas sencillos de alto tamaño pueden requerir de técnicas metaheurísticas.
- El tiempo que se necesita para solucionar un problema es un aspecto clave en la selección de un algoritmo de optimización. Limitaciones de tiempo real pueden sugerir el uso de metaheurísticas.
- La optimización de problemas continuos no lineales (NLP) pueden requerir de la aplicación de metaheurísticas si los métodos basados en derivadas o gradientes, por ejemplo el método quasi-Newton o el gradiente conjugado, fallan como consecuencia de un espacio de soluciones muy escabroso o rugoso (e.g., *discontinuous, nonlinear, ill-conditioned, noisy, multimodal, nonsmooth, and nonseparable*). La función f debe ser de moderada dimensionalidad (considerablemente superior a tres variables). Muchos problemas reales tienen estas propiedades.
- Optimización por simulación. Modelos no analíticos con escenarios de caja negra

$$f(\vec{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

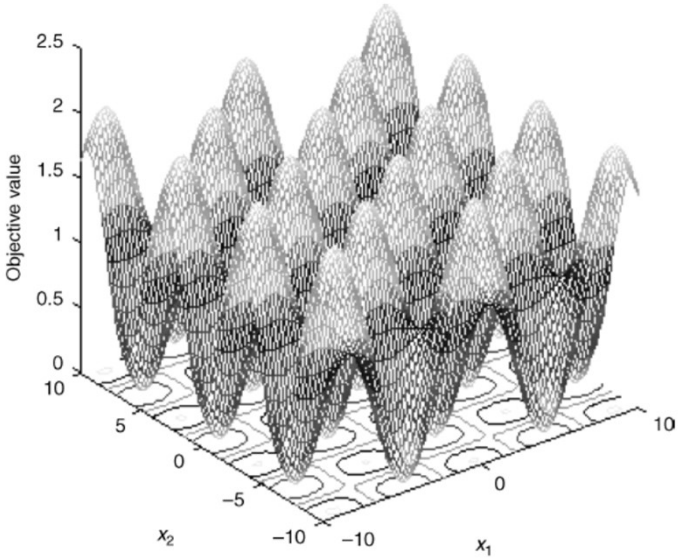
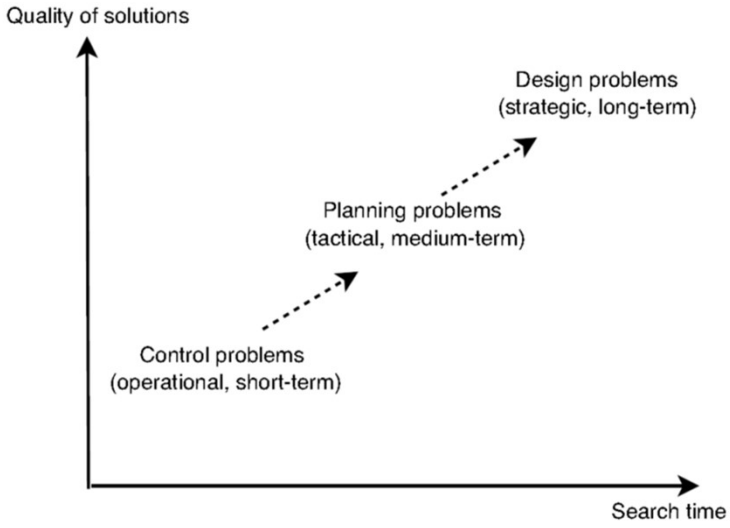


FIGURE 1.13 The Griewangk multimodal continuous function.

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Problemas de diseño frente a control

- La importancia relativa de las dos principales medidas de performance, la calidad de la solución y el tiempo de búsqueda, depende de las características del problema objetivo a optimizar. Se pueden considerar dos casos extremos:
  - **Problemas de diseño:** son problemas que generalmente se van a resolver una única vez. Necesitan una muy Buena calidad en la solución, mientras que el tiempo de búsqueda es un objetivo secundario (e.g. horas, días, meses). Dentro de esta clase de problemas se encuentran los problemas *estratégicos*, tales como el diseño de una red de telecomunicaciones o el diseño de un procesador. Estos problemas implican una gran inversión económica y cualquier imperfección puede tener un impacto a largo plazo. En estos casos si es posible se ha de encontrar un algoritmo de optimización exacta y si no, buscar calidad más que velocidad de ejecución.
  - **Problemas de control:** estos problemas representan el otro extremo, en el que nos enfrentamos a problemas que se han de resolver de forma frecuente en tiempo real. A esta clase de problemas pertenecen los problemas *operacionales* que implican decisiones a corto plazo (e.g. segundos o fracciones de Segundo), como mensajes de enrutamiento en una red de computadores o gestión de tráfico en una ciudad. En los problemas de decisión operacionales, se necesita una heurística rápida; la Calidad de la solución es menos crítica.



Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Búsqueda Aleatoria

- La búsqueda aleatoria es el caso extremo de exploración. Se elige aleatoriamente una muestra de soluciones del espacio de búsqueda y se devuelve la mejor.

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
```

Algorithm 9 *Random Search*

1: $Best \leftarrow$ some initial random candidate solution
2: **repeat**
3: $S \leftarrow$ a random candidate solution
4: **if** Quality(S) > Quality$(Best)$ **then**
5: $Best \leftarrow S$
6: **until** $Best$ is the ideal solution or we have run out of time
7: **return** $Best$

Source: S. Luke (2013) Essentials of Metaheurísticas. Lulu. http://sci2s.ugr.es/docencia/metah/

Búsqueda Aleatoria

- 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

Source: http://sci2s.ugr.es/docencia/metah/

Búsqueda Aleatoria

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

Búsqueda Aleatoria

La probabilidad de obtener el óptimo en n iteraciones sería:

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

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

Source: http://sci2s.ugr.es/docencia/metah/

Búsqueda Aleatoria

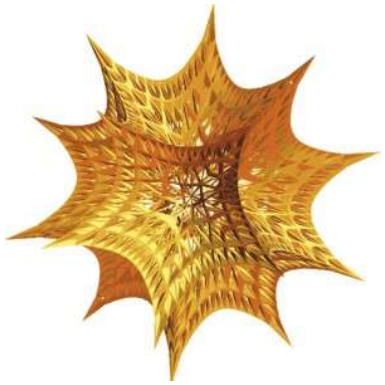
$$1 - \left(1 - \frac{1}{m}\right)^n > 1 - \alpha$$

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

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

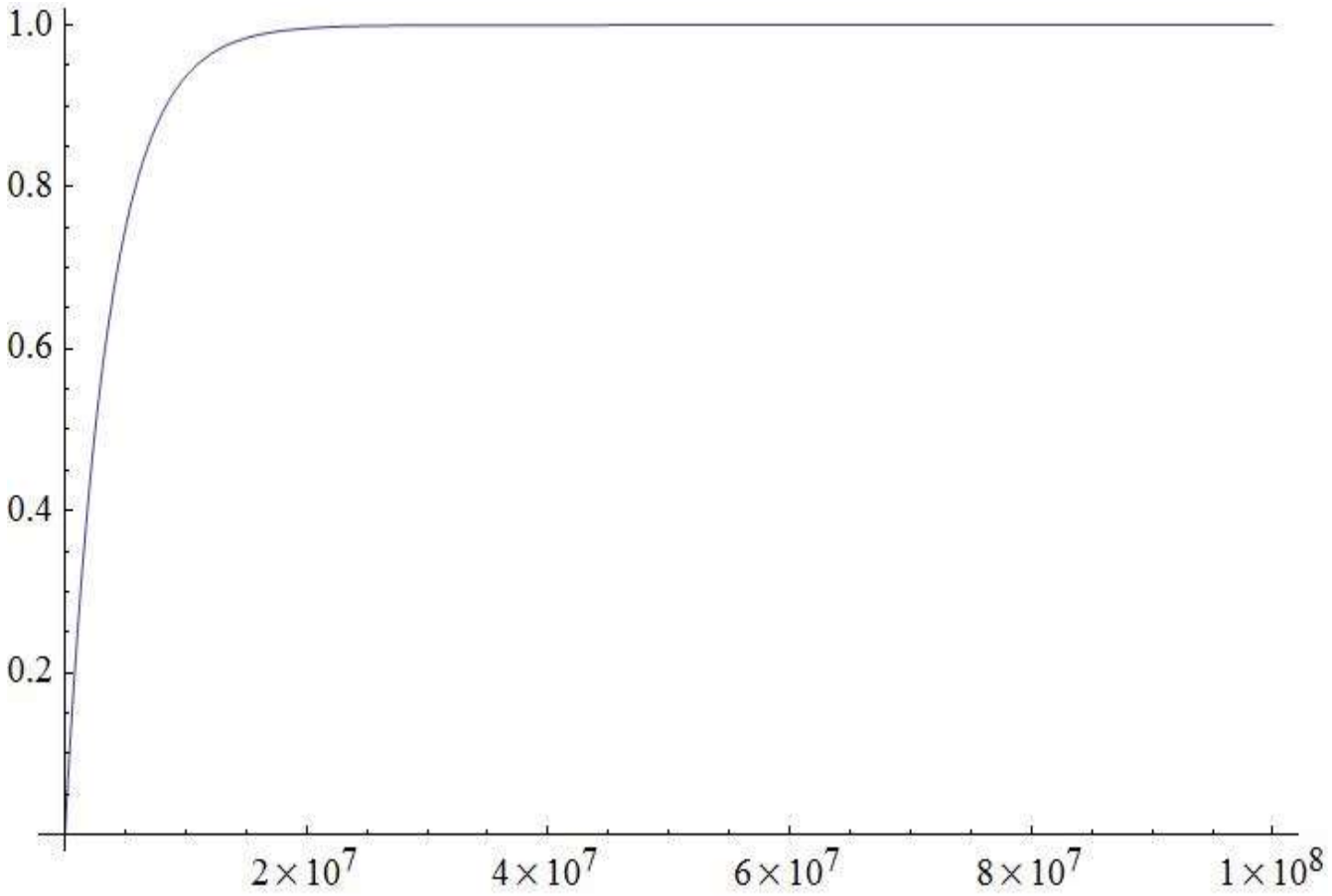
$$log(\alpha) > nlog\left(\left(1 - \frac{1}{m}\right)\right)$$

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



Source: http://sci2s.ugr.es/docencia/metah/

Búsqueda Aleatoria



Búsqueda local

- Búsqueda local es un ejemplo de búsqueda por trayectorias

Algorithm 2.1 High-level template of S-metaheuristics.

Input: Initial solution s_0.
$t = 0$;
Repeat
  /* Generate candidate solutions (partial or complete neighborhood) from s_t */
  Generate($C(s_t)$) ;
  /* Select a solution from $C(s)$ to replace the current solution s_t */
  s_{t+1} = Select($C(s_t)$) ;
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Entorno (vecindad)

Definition 2.1 Neighborhood. *A neighborhood function N is a mapping $N : S \rightarrow 2^S$ that assigns to each solution s of S a set of solutions $N(s) \subset S$.*

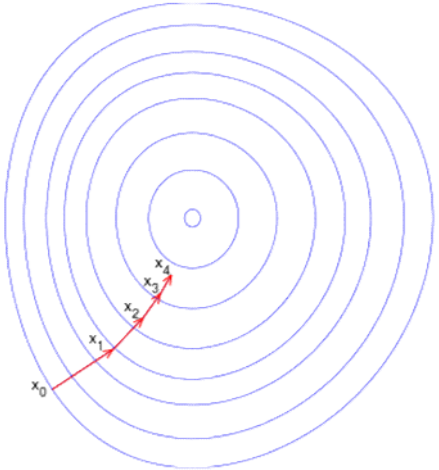
- La principal propiedad que debe caracterizar una vecindad es la localidad. Localidad es el efecto en la solución cuando se produce un movimiento (perturbación) en la representación. Cuando se hacen pequeños cambios en la representación, la solución debe implicar también pequeños cambios. Cuando esto ocurre se dice que el entorno o vecindad presenta una localidad fuerte. Por tanto, una metaheurística basada en trayectorias desarrollará una búsqueda con sentido en el espacio de soluciones.

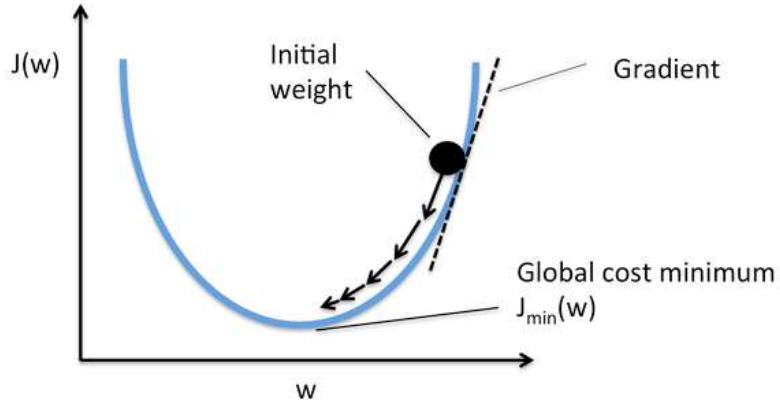
Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Entorno

Descenso del gradiente

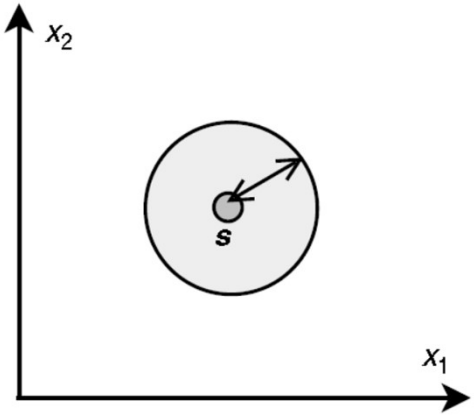
$$y = x - t * \nabla f(x)$$



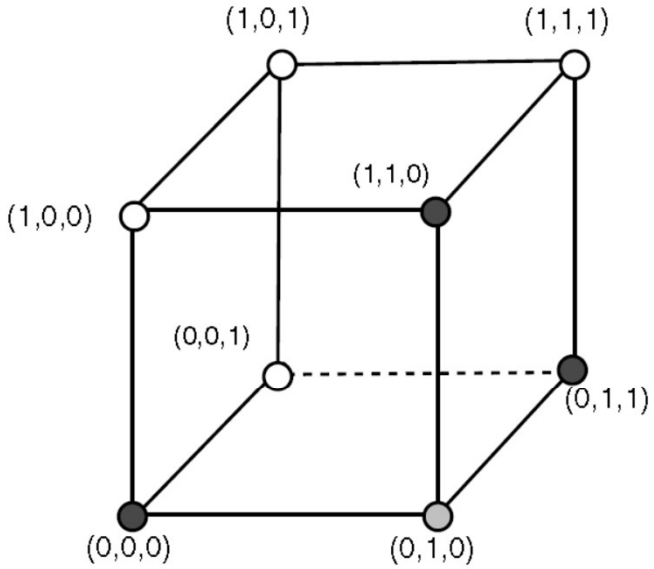


Source: https://www.quora.com/Whats-the-difference-between-gradient-descent-and-stochastic-gradient-descent

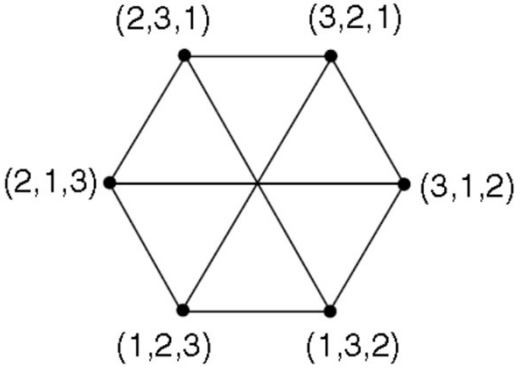
Entorno



The circle represents the neighborhood of *s* in a continous problem with two dimensions.



- Nodes of the hypercube represent solutions of the problem.
- The neighbors of a solution (e.g., (0,1,0)) are the adjacent nodes in the graph.



Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Búsqueda local

- Búsqueda local

Algorithm 2.2 Template of a local search algorithm.

$s = s_0$; /* Generate an initial solution s_0 */
While not Termination_Criterion **Do**
  Generate ($N(s)$) ; /* Generation of candidate neighbors */
  If there is no better neighbor **Then** Stop ;
  $s = s'$; /* Select a better neighbor $s' \in N(s)$ */
Endwhile
Output Final solution found (local optima).

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Búsqueda local del mejor

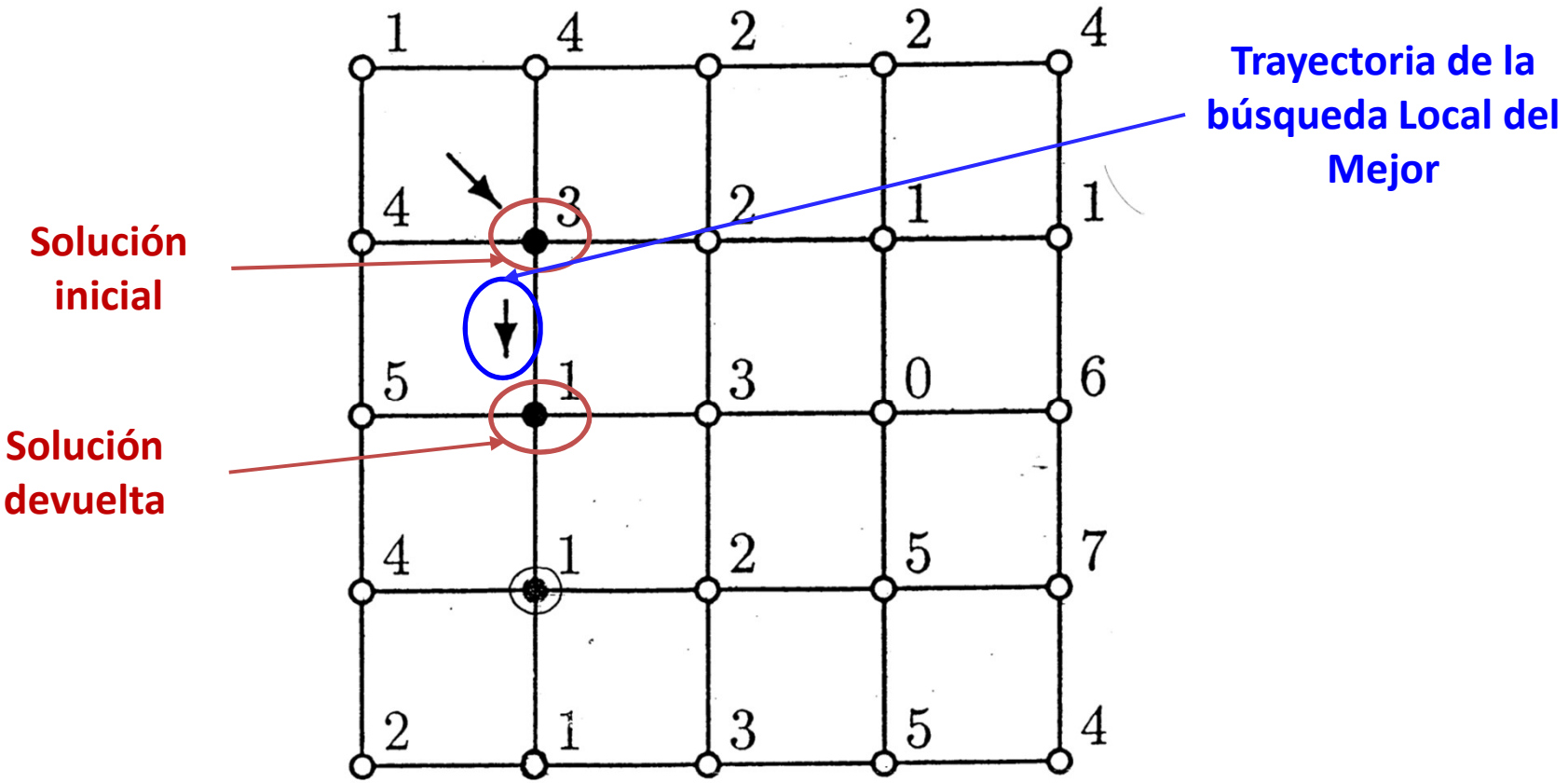
- Mediante esta estrategia, el mejor vecino (i.e. el vecino que produce una mayor mejora en la función objetivo) es seleccionado. El entorno es evaluado de forma complete y de manera determinista. Por tanto, la búsqueda en el vecindario es exhaustiva, y todos los posibles vecinos a la solución actual son evaluados para obtener 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.

Algorithm 6.1: Best Improvement Local Search

```
Create initial solution s;
while local optimum not reached do
    Determine complete neighborhood 𝒩 of current solution s;
    if 𝒩 contains at least one improving solution then
        Choose best solution s' from 𝒩;
        Switch over to solution s' (current solution s is replaced by s');
    end
end
return s';
```

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.
Zäpfel, G., & Braune, R. (2010). Metaheuristic search concepts: A tutorial with applications to production and logistics. Springer Science & Business Media.

Búsqueda local de mejor



Source: Exampled modified from http://sci2s.ugr.es/docencia/metah/

Búsqueda Local del Primer Mejor

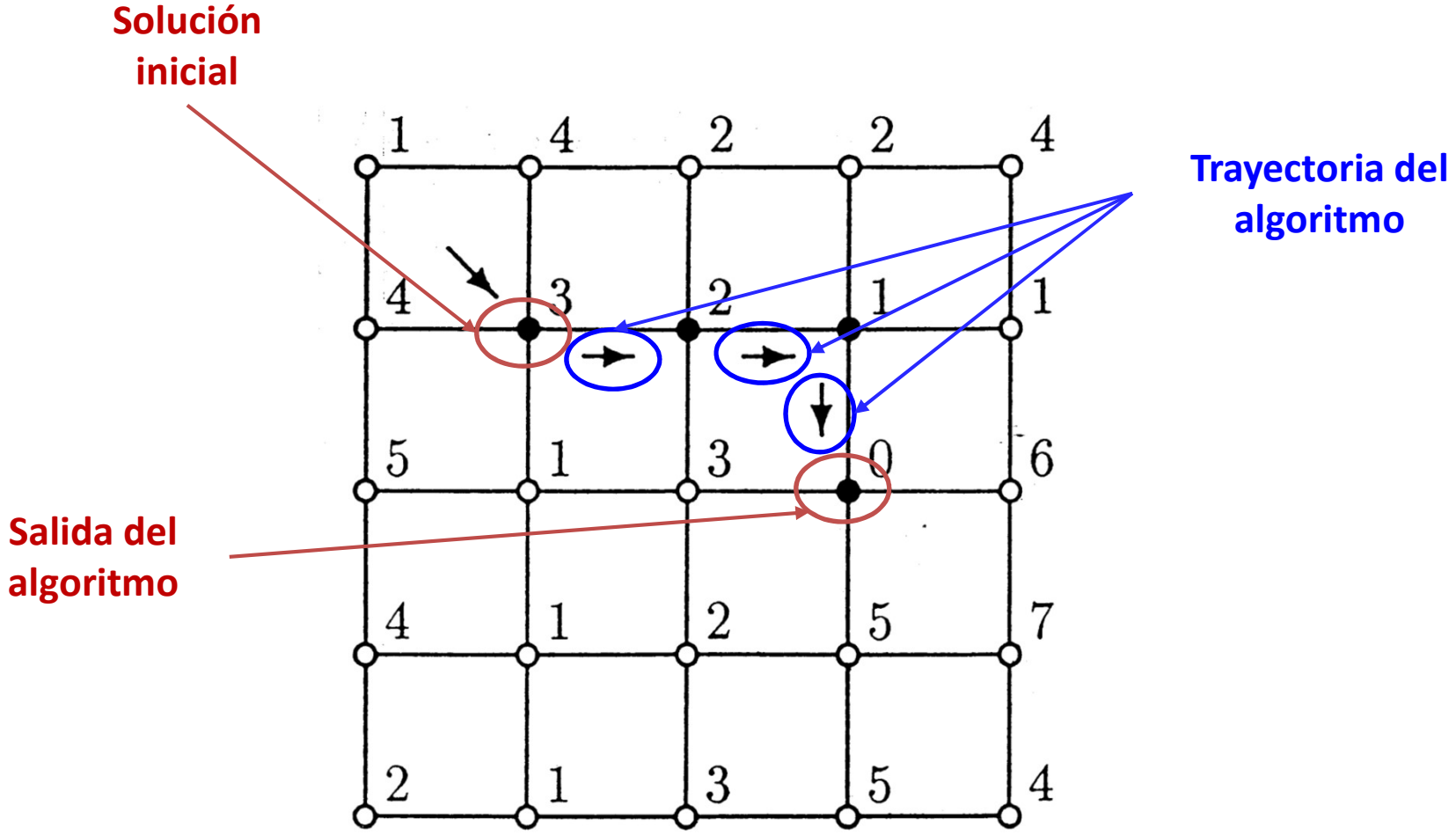
- Esta estrategia consiste en elegir el primer vecino que mejora la solución actual. En cuanto se encuentra uno, el vecino es inmediatamente seleccionado para reemplazar a la solución actual. El vecindario se explora de forma determinista siguiendo un determinado orden de formación de los vecinos. Si se completa el vecindario sin mejora el algoritmo termina.

Algorithm 6.2: First Improvement Local Search

```
Create initial solution s;
while local optimum not reached do
    repeat
        Create new neighbor s' by applying a move
    until until s' is better than s or no more moves available ;
    if s' is better than s then
        Switch over to solution s' (current solution s is replaced by s');
    end
end
return s';
```

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.
Zäpfel, G., & Braune, R. (2010). Metaheuristic search concepts: A tutorial with applications to production and logistics. Springer Science & Business Media.

Búsqueda Local del Primer Mejor



Source: Exampled modified from http://sci2s.ugr.es/docencia/metah/

Búsqueda local

- **Búsqueda Local del Mejor**
- **Búsqueda Local del Primer Mejor**
- **Selección aleatoria:** Mediante esta estrategia se selecciona aleatoriamente un vecino entre aquellos que mejoran la solución actual.

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Un ejemplo. TSP

1. Esquema de representación:
   – Permutación de {1, …,n}
2. Función objetivo

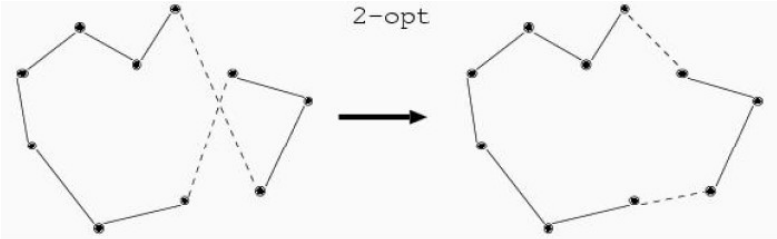
$$Min\ C(S) = \sum_{i=1}^{n-1} (D[S[i], S[i+1]]) + D[S[n], S[1]]$$

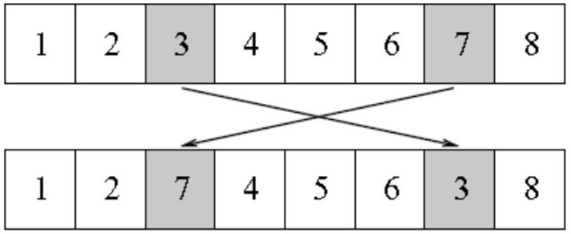
3. Mecanismo de generación de solución inicial
   – Permutación aleatoria

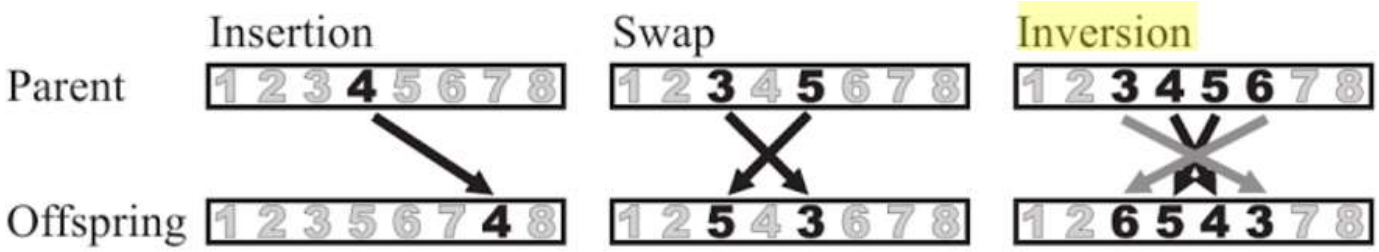
Source: http://sci2s.ugr.es/docencia/metah/

Un ejemplo. TSP

4. Función para generar nuevas soluciones: Operador intercambio (swap) o 2-opt (inversión)
   - Permutaciones {1, …,n}







Source: http://sci2s.ugr.es/docencia/metah/
Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.
Alba, E., & Dorronsoro, B. (2009). Cellular genetic algorithms (Vol. 42). Springer Science & Business Media.

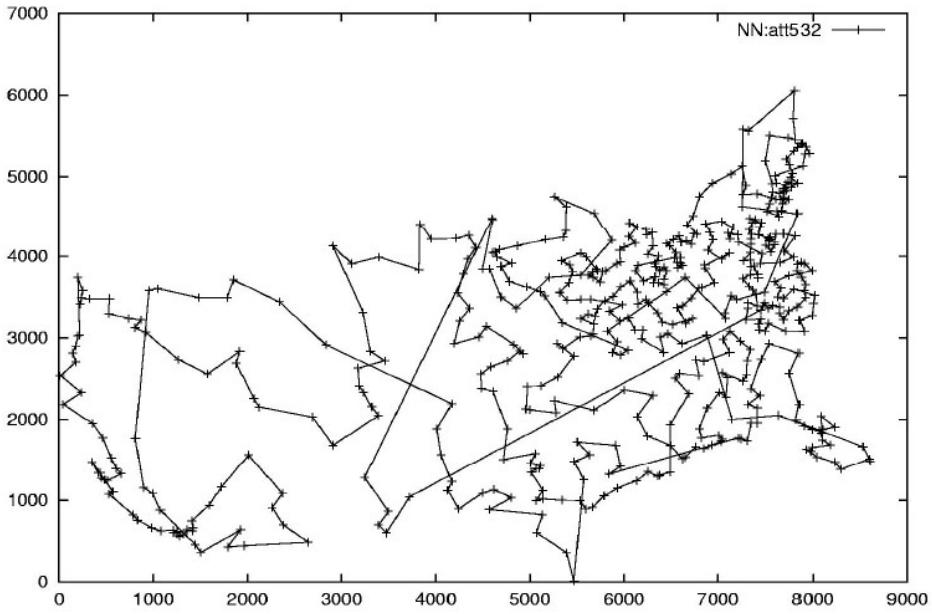
Un ejemplo. TSP

5. Mecanismo de selección: primer mejor o mejor
6. Criterio de parada
   - Encontrar un óptimo local

Source: http://sci2s.ugr.es/docencia/metah/
Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Un ejemplo. TSP

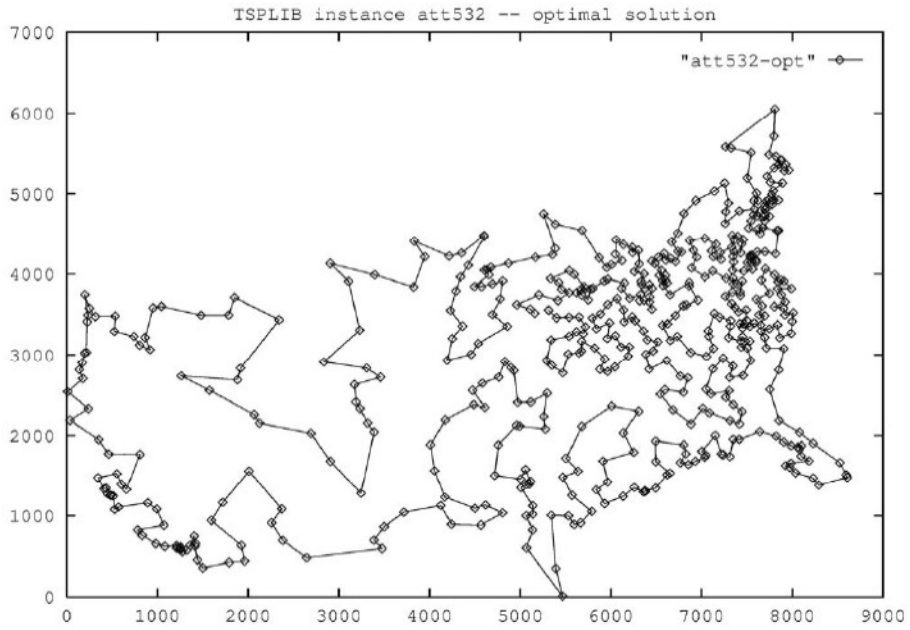
- Solución obtenida en una ejecución del algoritmo de búsqueda Local del Mejor



Source: http://sci2s.ugr.es/docencia/metah/

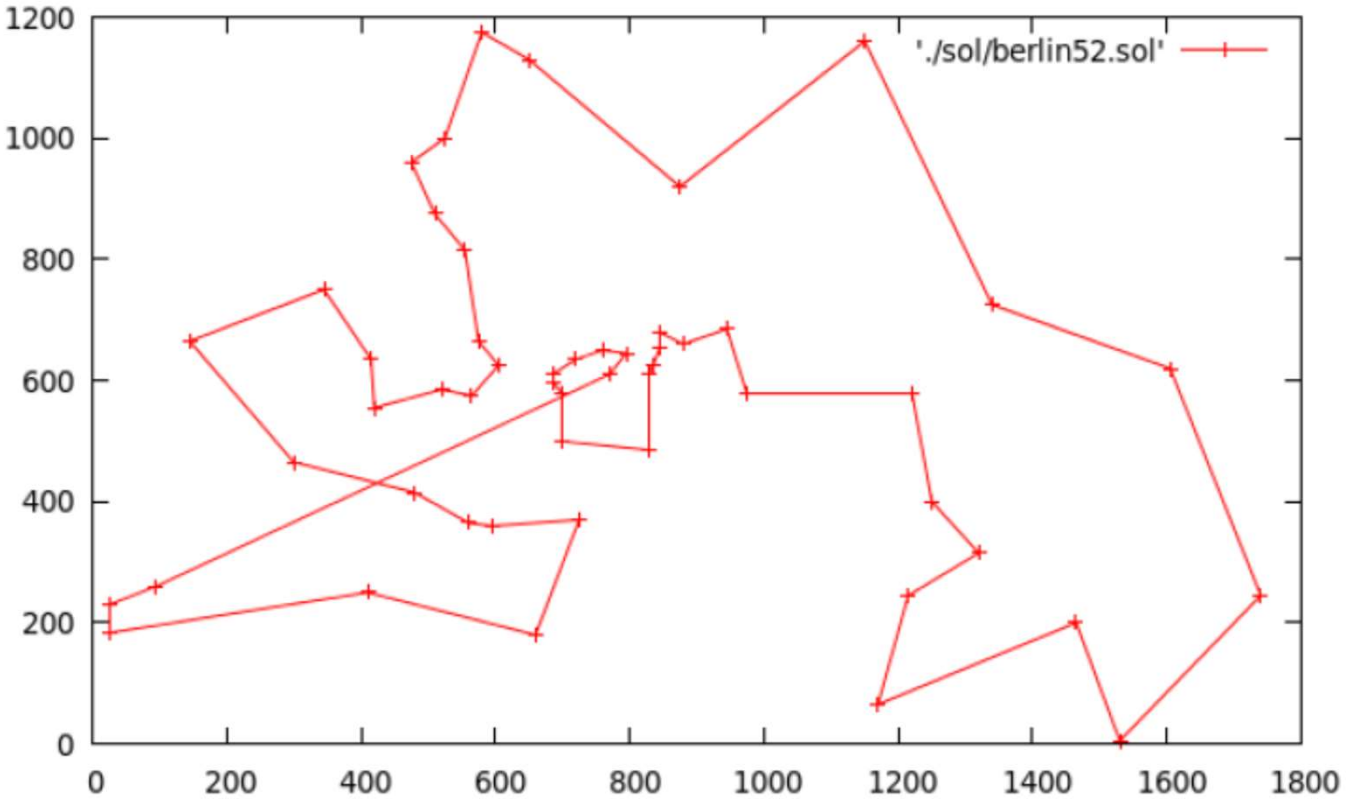
Un ejemplo. TSP

- Solución óptima



Source: http://sci2s.ugr.es/docencia/metah/

Un ejemplo. TSP



Source: sci2s.ugr.es/sites/default/files/files/Teaching/.../Tema02-BusquedaLocal-15-16.pdf
http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/berlin52.tsp

Problemas de la búsqueda local

- Con frecuencia cae en óptimos locales, a veces alejados del óptimo global

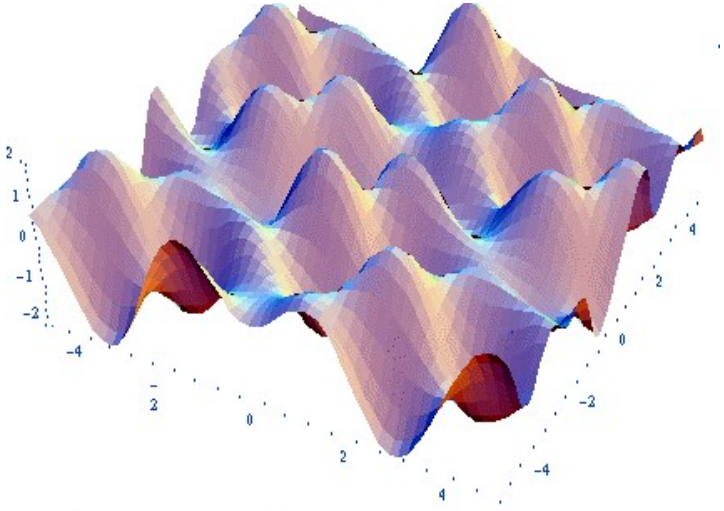


Figure 1. Generated using the Mathematica statement:

Plot3D[-0.2*(Sin[x + 4*y] - 2*Cos[2*x + 3*y] - 3*Sin[2*x - y] + 4*Cos[x - 2*y]), {x, -5, 5}, {y, -5, 5}, PlotPoints->50]

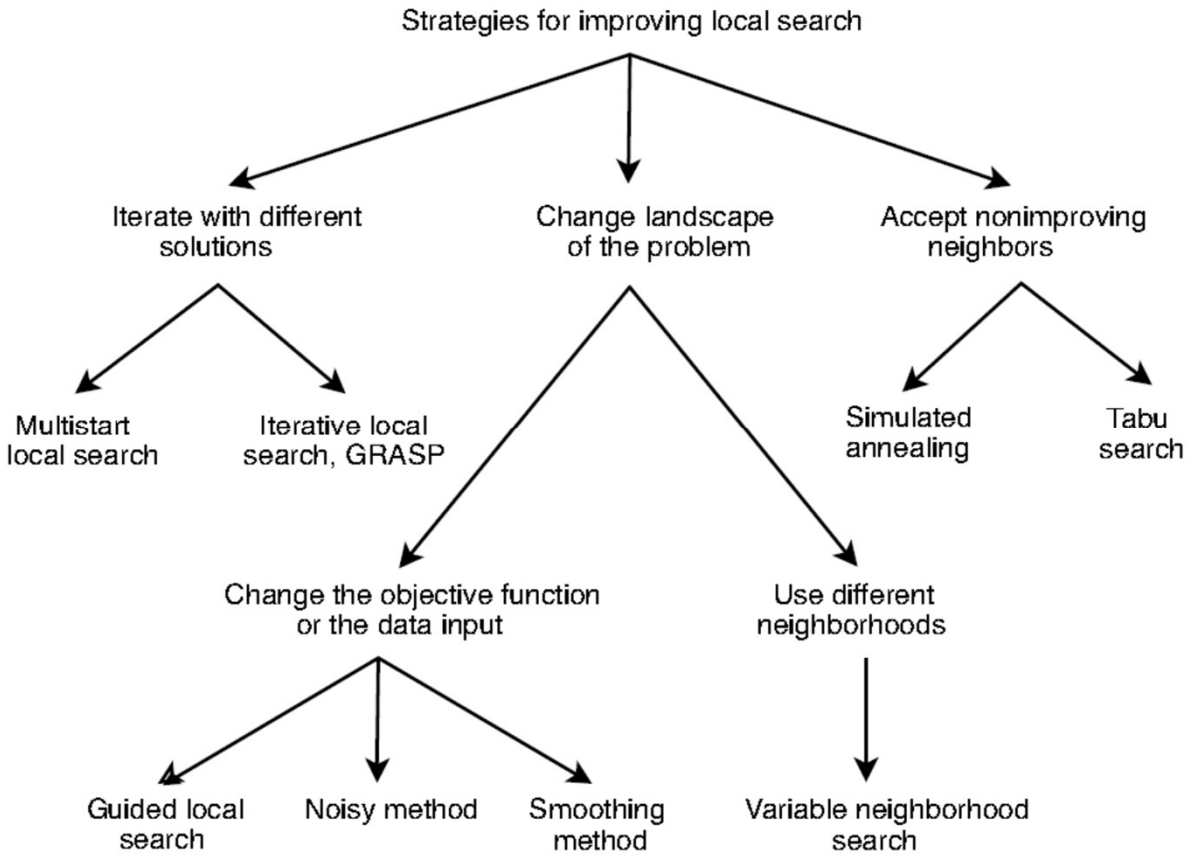
Source: http://plato.asu.edu/gom.html

Problemas de la búsqueda local

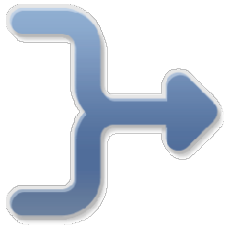
- La principal desventaja de los algoritmos de búsqueda local es la convergencia hacia óptimos locales. Para evitar esto se han propuesto algoritmos que consideran diferentes alternativas:
  - Aceptar movimientos hacia soluciones peores
  - Cambiar el vecindario
  - Cambiar la función objetivo o los datos de entrada del problema

Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

Problemas de la búsqueda local



Source: Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.



Resumen

- Problemas de optimización combinatoria
- Metaheurísticas
- Clasificación de metaheurísticas
  - Soluciones basadas en trayectorias vs basadas en poblaciones
- Mecanismos
  - Exploración vs explotación
- ¿cuándo utilizar metaheurísticas?
- Algunas aproximaciones ingenuas
  - Búsqueda aleatoria
  - Búsqueda local
    - Mejor vecino
    - Primer mejor
- Un ejemplo. TSP
- Problemas de la búsqueda local
  - Óptimos locales
  - Diferentes estrategias por diferentes algoritmos