Máster Universitario en Inteligencia Artificial

MÉTODOS DE SIMULACIÓN

Capítulo 5. Simulación y optimización. Introducción

ÍNDICE

1. Programación matemática

La **programación lineal** (PL) es una clase especial de modelos de programación matemática que se desarrolló a partir de la Segunda Guerra Mundial para resolver cierto tipo de problemas de asignación de recursos entre distintas actividades.

Las aplicaciones posteriores a una amplia variedad de problemas han sido numerosas y esto ha llevado a que los modelos de optimización lineal constituyan una de las herramientas básicas más utilizadas de la IO.

Se considera un conjunto de variables denominadas **variables de decisión** que representan entradas al modelo, controlables por el modelizador o decisor, y una **función objetivo**, que será una función lineal en las variables de decisión, que representa algún criterio o meta importante.

Además, existen ciertas limitaciones prácticas que constituyen las **restricciones** del modelo y que se representan como ecuaciones o inecuaciones lineales en las variables de decisión.

Proceso de formulación de un modelo lineal:

1. Determinar las variables de decisión y representarlas algebraicamente

Las variables de decisión son los factores sujetos a cambios cuyos valores pueden variar, o al menos hacerlo dentro de unos ciertos límites, bajo el control del decisor o modelizador.

En general, las representaremos mediante x_j, $j = 1,...,n$, y en notación vectorial como $\mathbf{x} = (x_1,...,x_n)$. La solución óptima se indica con $\mathbf{x}^* = (x_1^*,...,x_n^*)$.

Variables Continuas vs. Discretas (Programación Lineal Entera)

2. Formular las restricciones

Las restricciones, expresadas como ecuaciones o inecuaciones lineales en las variables de decisión, generalmente representan la limitación en la disponibilidad de algún recurso.

Matemáticamente, cada restricción puede representarse por una de las formas siguientes :

$$g_i(\mathbf{x}) \leq b_i \quad o \quad g_i(\mathbf{x}) = b_i \quad o \quad g_i(\mathbf{x}) \geq b_i$$

donde g_i es una función lineal en $\mathbf{x}$ y b_i son números reales.

3. Formular la función objetivo

La función objetivo a optimizar será una función lineal en las variables de decisión. De manera genérica podemos decir que representa los deseos del decisor de maximizar un beneficio o minimizar un coste. Su forma matemática es

$$max\ z = f(\mathbf{x}) \quad \text{o} \quad min\ z = f(\mathbf{x})$$

Más de un objetivo simultáneo → Programación Lineal Multiobjetivo

Formulación algebraica de un modelo lineal:

En general, la formulación algebraica problema o modelo de programación lineal, también denominado simplemente *programa lineal*, es la siguiente:

$$max\ o\ min\ \ z = f(\mathbf{x})$$

s.a

$$g_1(\mathbf{x})\ (\leq , = , \geq)\ b_1$$

$$\cdots$$

$$g_m(\mathbf{x})\ (\leq , = , \geq)\ b_m$$

$$\mathbf{x} \geq 0 \text{ o no restringidas}$$

EJEMPLO 1.

Supongamos una fábrica de cervezas en la que se producen tres tipos distintos denominados negra (N), rubia (R) y sin alcohol (A).

Para su elaboración son necesarios, además de agua y lúpulo, para los cuales no hay limitación de disponibilidad, los recursos malta y levadura, que por su disponibilidad limitada restringen la capacidad diaria de producción.

La siguiente tabla proporciona la cantidad necesaria de cada uno de estos recursos para la producción de un litro de cada una de las respectivas cervezas, la materia prima en kilogramos disponible de cada recurso y, finalmente, el beneficio por litro de cada tipo de cerveza producido.

Disponibilidades y recursos para la fabricación de cervezas

	Tipo de cerveza			
	Negra	Rubia	Sin alcohol	Disponibilidad
Malta	2	1	2	30
Levadura	1	2	2	45
Beneficio	4	7	3	

Se le plantea ahora al fabricante el problema que consiste en decidir los litros que debe producir de cada cerveza para que el beneficio total por día sea máximo.

EJEMPLO 2.

Un alumno decide distribuir el tiempo de que dispone (120 horas) para preparar los exámenes de febrero. El alumno se encuentra matriculado en cuatro asignaturas y ha estimado que el tiempo de estudio necesario para aprobarlas con 5.0 son, respectivamente, 12, 20, 25 y 18 horas, mientras que las necesarias para sacar un 10 son, respectivamente, 20, 45, 60 y 30 horas (aunque la realidad puede ser muy distinta de sus estimaciones).

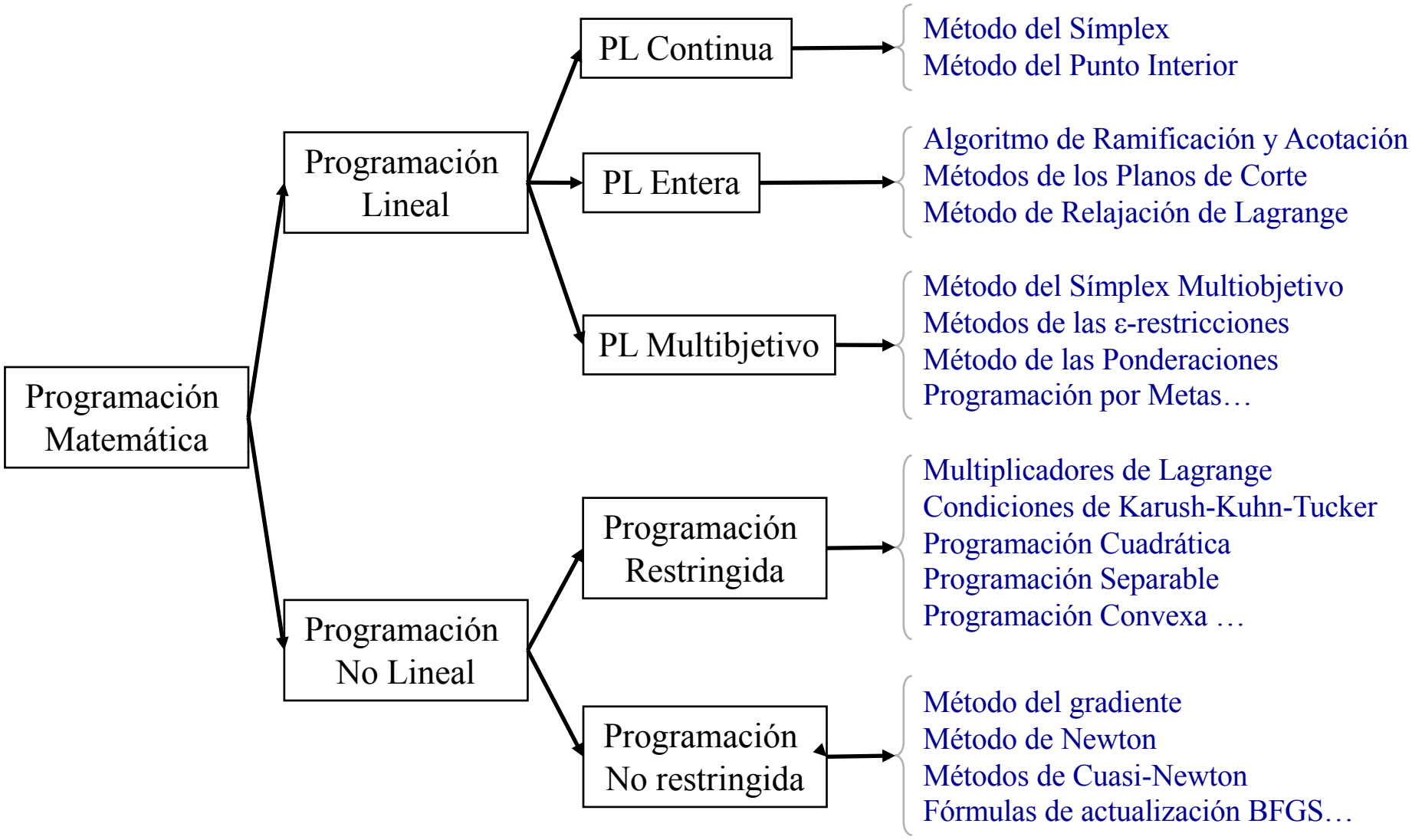
La nota que espera obtener será proporcional al tiempo dedicado a la asignatura y los tiempos necesarios para aprobar y sacar un diez, linealmente (Por ejemplo, si el tiempo para aprobar son 10 horas y para sacar un 10 son 20 horas, si dedica 17 horas su nota será 8.5).

Se impone como condición que el tiempo que dedica a las distintas asignaturas no difiera en más de 5 horas.

Modelizar el programa lineal que maximice la nota media obtenida en los exámenes de febrero, aprobando además las cuatro asignaturas, de acuerdo a sus estimaciones optimistas.

ÍNDICE

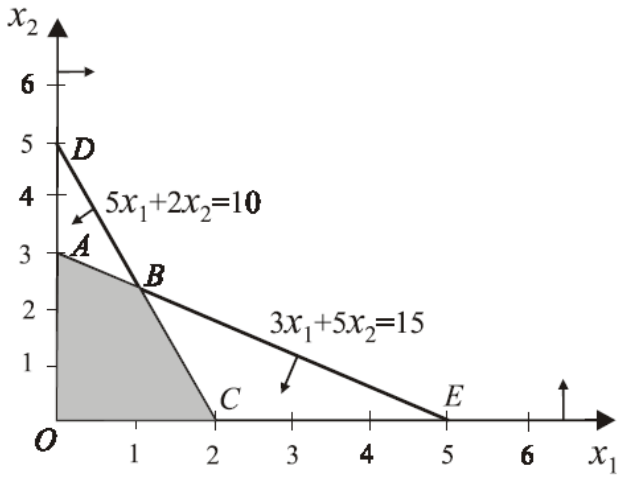
2. Métodos de solución exactos



Programación Lineal Continua

$$\begin{aligned} \max\ z &= 2x_1 + x_2 \\ \text{s. a} & \\ & 5x_1 + 2x_2 \leq 10 \\ & 3x_1 + 5x_2 \leq 15 \\ & x_1, x_2 \geq 0 \end{aligned}$$

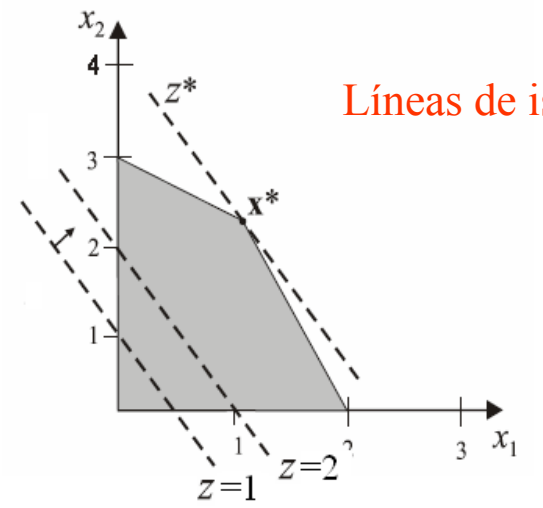
Obtención de la región factible



Obtención de la solución óptima

Evaluación de los puntos extremos

Punto extremo	Coordenadas (x_1, x_2)	Valor del objetivo z
O	$(0, 0)$	0
A	$(0, 3)$	3
B	$(20/19, 45/19)$	$85/19$
C	$(2, 0)$	4

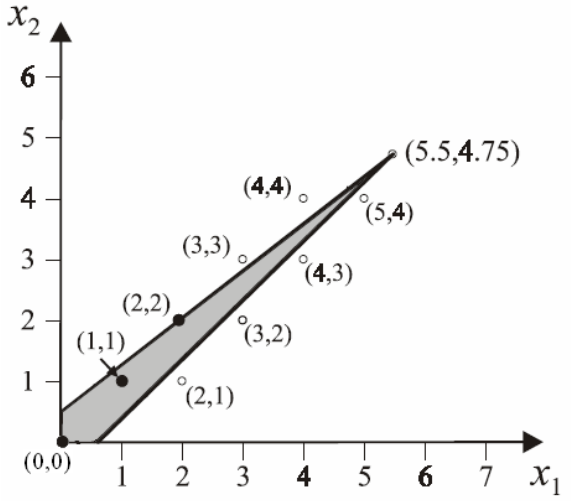


Líneas de isobeneficio

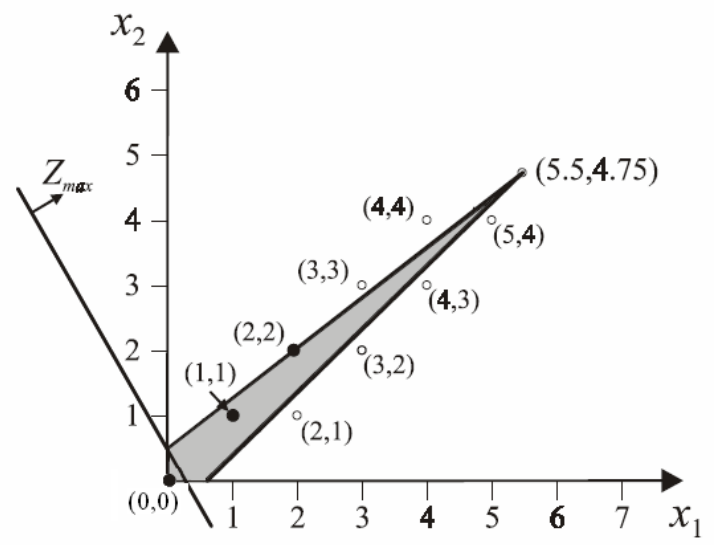
Programación Lineal Entera

$$\begin{aligned} \max z &= 2x_1 + x_2 \\ \text{s. a} & \\ -17x_1 + 22x_2 &\leq 11 \\ 4x_1 - 4x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \quad \text{y enteras} \end{aligned}$$

Obtención de la región factible



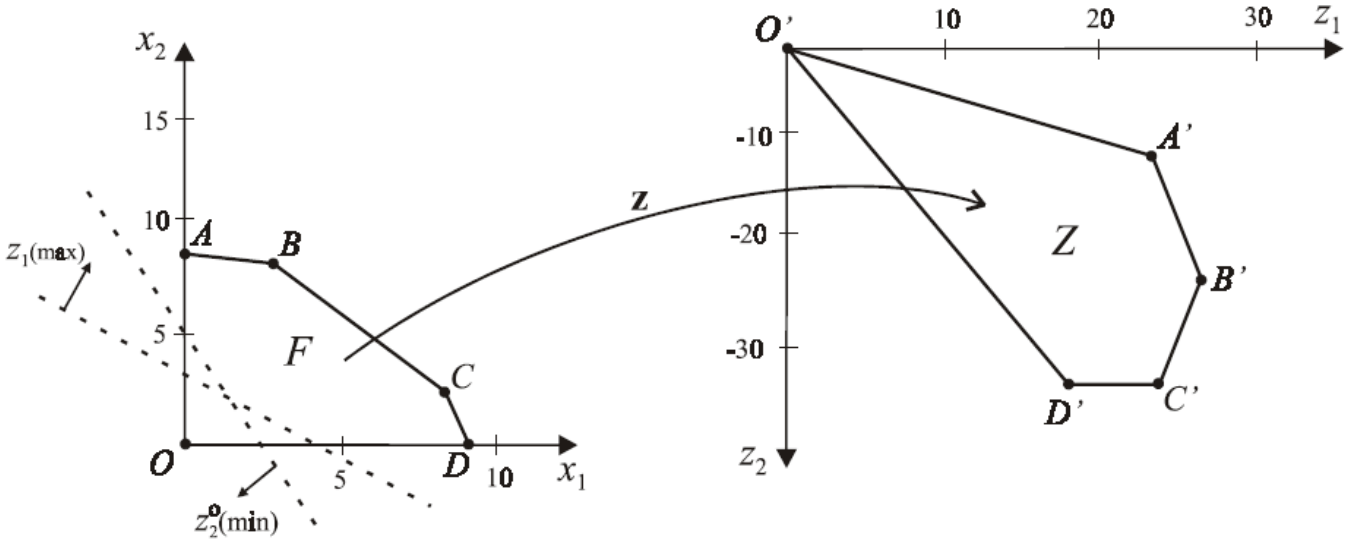
Obtención de la solución óptima



Programación Lineal Multiobjetivo

$$\max \mathbf{z} = (z_1(\mathbf{x}), z_2(\mathbf{x})) \quad \text{con} \quad \max z_1(\mathbf{x}) = 2x_1 + 3x_2$$
$$\text{s. a} \qquad\qquad\qquad\qquad \min z_2(\mathbf{x}) = 4x_1 + 2x_2$$
$$x_1 + 3x_2 \leq 24$$
$$2x_1 + x_2 \leq 18$$
$$x_1 + x_2 \leq 10$$
$$x_1, x_2 \geq 0$$

Región factible y obtención de la solución óptima



ÍNDICE

3. Heurísticas

Problemas de optimización combinatorios

El nº de soluciones factibles crece exponencialmente con el tamaño del problema

Los métodos exactos suelen ser inviables en instancias grandes, ya sea por limitación de tiempo o por falta de memoria.

HEURÍSTICAS

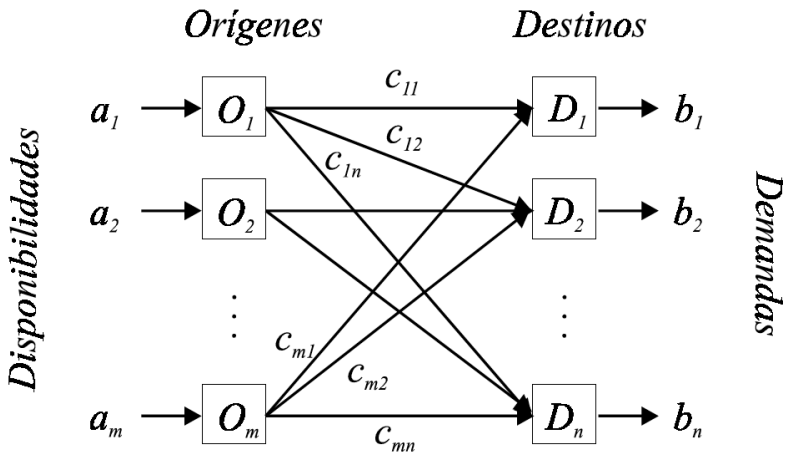
Procedimiento (aproximado) que trata de aportar una solución a un problema de manera eficiente, es decir, intenta encontrar una solución de calidad o próxima a la optima con un coste computacional razonable

Alcanzar soluciones buenas (no óptimas) mediante el examen de un subconjunto de soluciones del total

Ejemplos de heurísticas

Problema de transporte

Algoritmo de transporte (MEN, MAV, MODI)



Problema de asignación

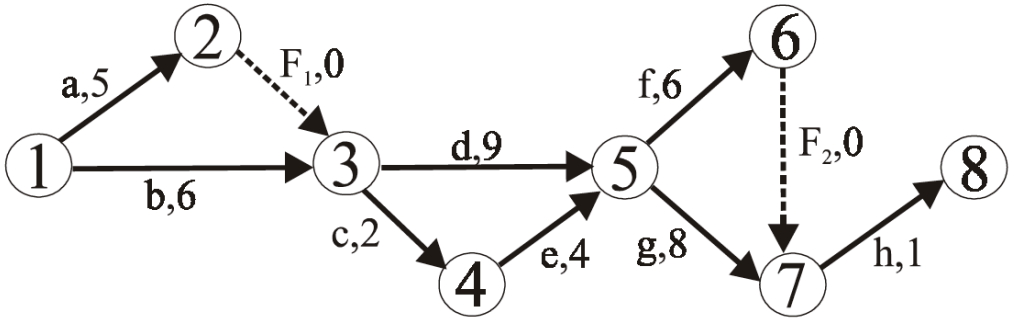
Método Húngaro

	D_1	D_2	⋯	D_m
O_1	c_{11}	c_{12}	⋯	c_{1m}
O_2	c_{21}	c_{22}	⋯	c_{2m}
⋮	⋮	⋮		⋮
O_m	c_{m1}	c_{m2}	⋯	c_{mm}

Ejemplos de heurísticas

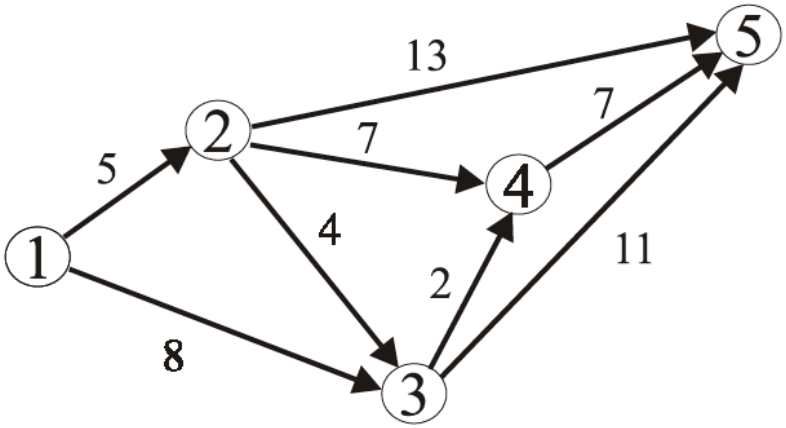
Problema secuenciación y control en redes

Métodos CPM y PERT



Camino de longitud mínima y máxima

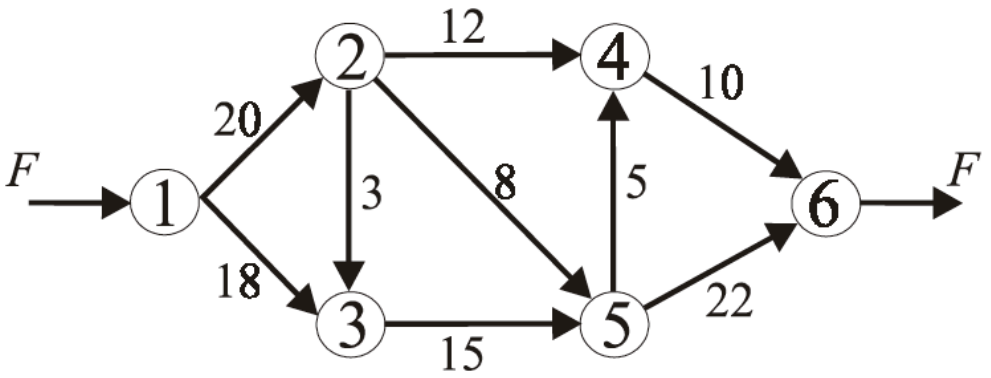
Algoritmos de Dijkstra y de Floyd



Ejemplos de heurísticas

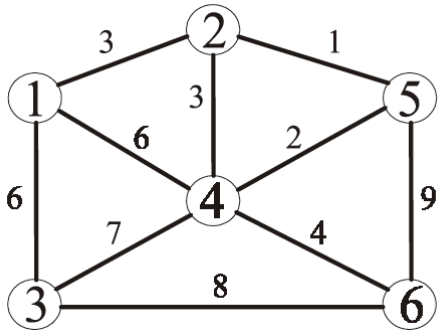
Flujo máximo en redes

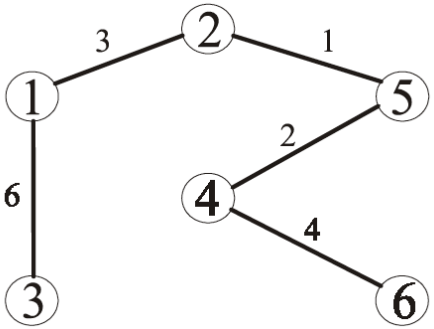
Algoritmo de etiquetación de Ford-Fulkerson



Árbol de máximo alcance

Algoritmo de Prim





Problemas de algunas heurísticas:

- no son capaces de garantizar la optimalidad de la solución ni de establecer cuan próxima de la solución óptima está la solución encontrada.
- Desarrolladas *ad hoc* para resolver problemas concretos.
- Se pueden quedar atrapadas en óptimos locales.

ÍNDICE

4. Heurísticas vs. Metaheurísticas

El término **metaheurística** apareció por primera vez en el artículo sobre búsqueda tabú de Glover en 1986 y etimológicamente deriva de la composición de dos palabras de origen griego, que son **meta** (más allá, en un nivel superior) y **heurística**.

- Están dirigidas hacia la obtención en un tiempo razonable de una solución eficiente y satisfaciente, es decir, procedimientos que tratan de alcanzar buenas soluciones (satisfacientes) con un coste computacional razonable, pero sin garantizar la optimalidad.

- **Aplicables a un gran número de problemas**.

- Tienen como objetivo **evitar quedarse atrapados en óptimos locales**.

- Referencia a mecanismos de optimización de la naturaleza.

Las metaheurísticas son una clase de métodos aproximados de optimización combinatoria que proporcionan un marco general para crear algoritmos híbridos combinando diferentes conceptos derivados de la IA, la evolución biológica y los mecanismos estadísticos.

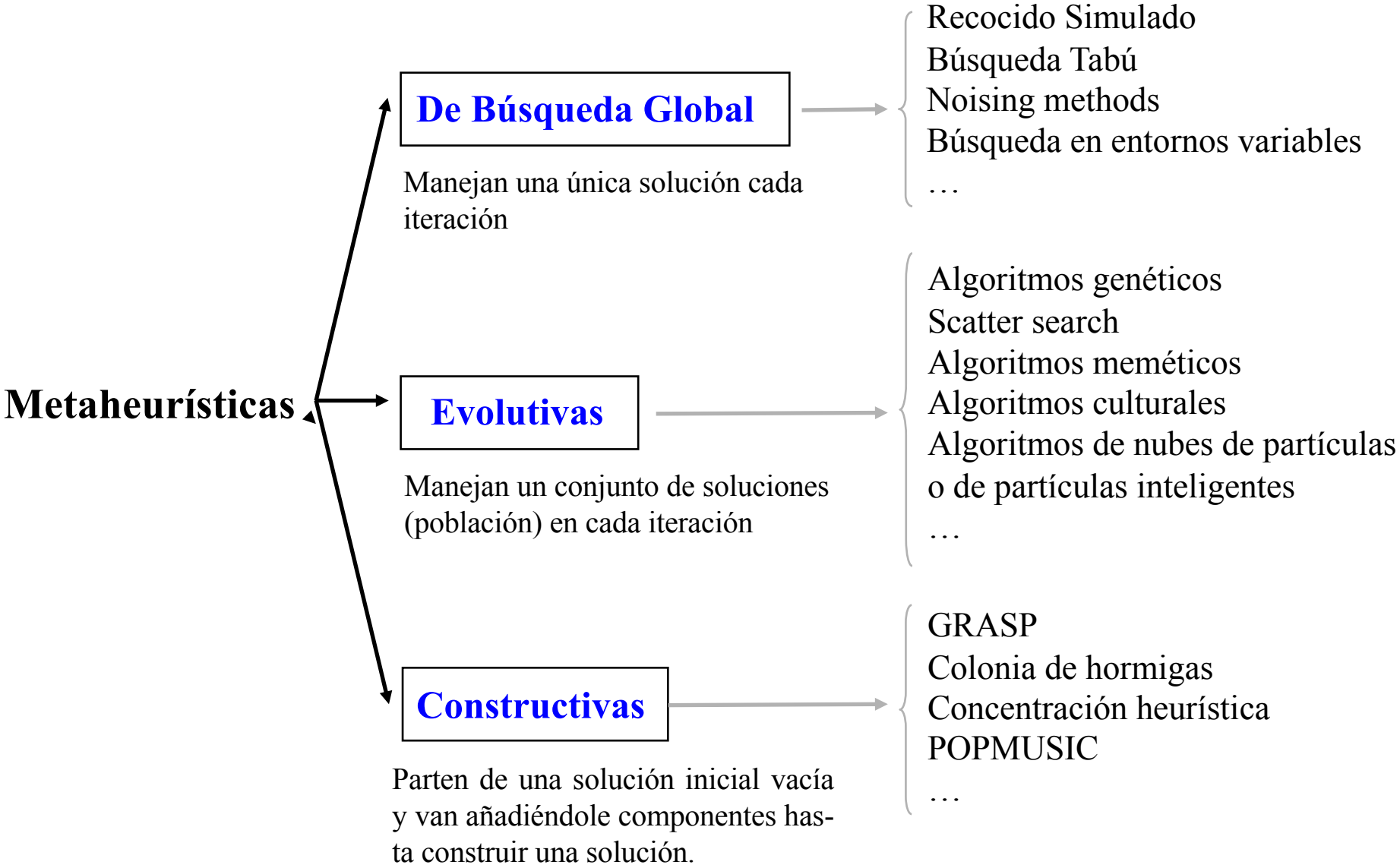
La evolución de las metaheurísticas ha tenido un comportamiento exponencial en los últimos 35 años, resolviéndose problemas que tiempo atrás parecían inabordables.

En época reciente se han extendido los métodos de optimización como **hiperheurísticas** que son procedimientos de búsqueda de muy alto nivel que se sitúan por encima de las metaheurísticas.

Elevan la generalidad de las metaheurísticas y se encargan de seleccionar, en cada momento y para cada problema, el método heurístico o metaheurístico más adecuado. Esta selección se hace en función del tiempo que tarda cada algoritmo o en función de la calidad de la solución que alcanza cada uno de ellos.

ÍNDICE

5. Clasificación de metaheurísticas



Metaheurísticas de búsqueda global (o trayectoriales)

Pueden ser entendidas como un enfoque inteligente de los algoritmos de búsqueda local. Comienzan con una única **solución inicial** normalmente aleatoria que van **mejorando** iteración tras iteración durante el proceso de optimización.

Su nombre proviene del hecho de que la solución tratada va moviéndose por el espacio de soluciones como si dibujase una trayectoria.

Se apoyan en búsqueda probabilística y su filosofía consiste en no quedar atrapadas en un óptimo local y, para escapar de esa situación se considera:

a) volver a comenzar la búsqueda desde otra solución inicial;

b) permitir movimientos de empeoramiento de la solución actual.

c) modificar la estructura de entornos y;

Recocido Simulado (*simulated annealing*). Kirpatrick *et al.* (1983) y Cerny (1985). Inspirado en la **industria metalúrgica**. El recocido es una técnica que consiste en calentar el material a muy altas temperaturas y enfriar de manera muy lenta y progresiva para conseguir un estado de mínima entropía, que da lugar a sólidos con mejores propiedades.

Escapa de óptimos locales **permitiendo movimientos hacia soluciones peores que la actual**, en función del parámetro *temperatura*.

Al principio del proceso la temperatura se ha inicializado a un valor bastante alto, por tanto, la probabilidad de poder aceptar peores soluciones es alta → permite explorar el espacio de soluciones para no quedar atrapado en un optimo local.

Según pasan las iteraciones, la temperatura desciende, disminuyendo la probabilidad de que se acepte una solución peor a la actual → deja de explorar y comienza a intensificar la búsqueda para que se produzca la convergencia hacia la solución óptima.

Búsqueda Tabú (*tabu search*). Glover (1989, 1990). Se inspira en el concepto social "tabú", para proporcionar una técnica de búsqueda eficiente que evite el anclaje en un óptimo local.

Procedimiento heurístico que combina el proceso de búsqueda global con técnicas para evitar caer en óptimos locales o caer en ciclos durante el proceso de búsqueda.

La *lista tabú* (LT) es una estructura de memoria donde se almacenan los últimos movimientos de búsqueda realizados, para evitar la repetición de los mismos. Mediante la prohibición de repetir estos movimientos se consigue escapar de los ciclos.

Para garantizar la convergencia del algoritmo hacia la solución óptima, la LT debe complementarse con el *criterio de aspiración*. Este criterio es un conjunto de reglas que permiten al algoritmo, en caso de que se cumplan, obviar las restricciones tabú.

Búsqueda en entornos variables (*variable neighbourhood search*, *VNS*). Hansen et al. (2003). Está basada en el principio de cambiar sistemáticamente de estructura de entornos dentro de la búsqueda. Cuando la búsqueda queda atrapada, se cambia a un entorno más amplia.

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.

Noising methods (Charon y Hudry, 2001). Se basan en la introducción de una serie de fluctuaciones (ruido) decreciente sobre la función objetivo o los datos del problema, de tal forma que al final esas fluctuaciones nos hagan converger hacia los datos que representan al óptimo.

Metaheurísticas evolutivas

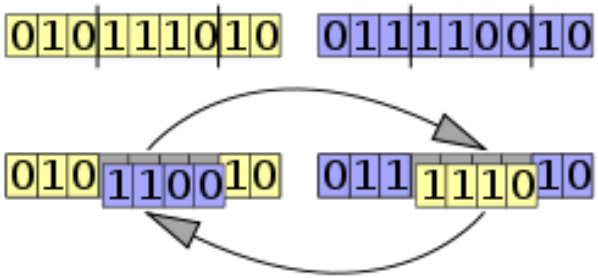
Algoritmos Evolutivos (*evolutionary algorithms*). Holland (1975) y Goldberg (1989). Son un conjunto de algoritmos poblacionales de optimización inspirados en la teoría de la evolución del NeoDarwinismo, que postula que las especies de seres vivos que consiguen sobrevivir son aquellas mas adaptadas al entorno.

Los algoritmos evolutivos simulan la evolución, mediante procesos de selección y reproducción aplicados a la población de soluciones iniciales para producir mejores soluciones.

- Cada iteración del algoritmo simula una generación, donde a una *población* de soluciones formada por distintas soluciones denominadas *individuos* se les aplica un operador de reproducción. La reproducción combina los *genes* o características de los individuos actuales, los padres, para producir uno o mas nuevos individuos, los hijos o descendencia.

- Este operador, se complementa con el operador de mutación, que permite la aparición aleatoria de nuevos rasgos en la descendencia.

- La nueva descendencia es evaluada según su aptitud o fitness, es decir, se evalúa la calidad de la solución para seleccionar a aquellos individuos que pasaran a la siguiente generación.



Los algoritmos con mas renombre de la computación evolutiva son las *estrategias evolutivas*, los *algoritmos genéticos* y la *programación genética*.

Los **algoritmos genéticos** son, con diferencia, la técnica de computación evolutiva mas utilizada. Difieren unos de otros por la forma en la que representan o codifican la solución, su estrategia de selección, el tipo de operador de cruce y mutación o la estrategia de reemplazo.

La **programación genética** fue introducida por Koza a principios de la década de 1990. Este método no tiene la intención de buscar soluciones sino programas enteros que resuelvan el problema.

Optimización de partículas inteligentes o algoritmos de nubes de partículas (*particle swarm optimization*). Kennedy y Eberhart (1995). Se inspira en el comportamiento social de las bandadas de pájaros o bancos de peces que evolucionan teniendo en cuenta la mejor solución encontrada en su recorrido y al líder.

El algoritmo emplea una población de soluciones (partículas) que influyen unas a otras para recorrer el espacio de búsqueda → se basan en la *velocidad actual* de la partícula, su *mejor solución* encontrada hasta el momento y la mejor solución encontrada por el enjambre en su totalidad.

El algoritmo consta de dos partes:

- *Inicialización*: se genera una población y se le asignan los parámetros iniciales para el proceso posterior.

- *Proceso iterativo*: una vez inicializada la población, iterativamente, se va actualizando la velocidad y posición de cada partícula basándose en la mejor solución encontrada por cada partícula y la mejor solución global.

Este proceso que sigue el algoritmo intenta mezclar explotación (enjambre en su conjunto) y exploración (partículas individuales).

Búsqueda dispersa (*scatter search*). Glover (1977). Se trata de un algoritmo que opera sobre un cjto de soluciones denominado *conjunto de referencia* (*CR*).

A diferencia de otras metaheurísticas evolutivas, la BD no está fundamentada en la aleatorización sobre un conjunto relativamente grande de soluciones sino en elecciones sistemáticas y estrategias sobre un conjunto pequeño (10).

Se basa en integrar la combinación de soluciones con la búsqueda local. Consta básicamente de cinco elementos o métodos:

- *Método de diversificación y generación*: genera soluciones diversas.
- *Método de mejora*: método local para mejorar las soluciones, tanto del *CR* como las combinadas antes de estudiar su inclusión en el *CR*.
- *Método de actualización del CR*: a partir del conjunto de soluciones diversas se extrae el *CR* según el criterio calidad y diversidad.

- *Método de generación de subconjuntos*: un método para generar subconjuntos del *CR* a los que se aplicara el método de combinación.

- *Método de combinación de soluciones*: Las soluciones que se obtienen de esta combinación pueden ser inmediatamente introducidas en el CR o almacenadas temporalmente en una lista hasta terminar de realizar todas las combinaciones.

Algoritmos meméticos (*memetic algorithms*). Moscató (1999), Moscato y Cotta (2003). Surgen de la combinación de algoritmos genéticos, que se encarga de realizar la búsqueda aplicando el énfasis en la exploración, con búsquedas locales (convergencia más rápida hacia una solución).

Estimación de distribuciones (*Estimation of Distribution Algorithms*, *EDA*). Larrañaga y Lozano (2001). Utilizan una colección de soluciones candidatas para realizar trayectorias de búsqueda evitando mínimos locales (se apoyan en la estimación y simulación de distribuciones de probabilidad).

Metaheurísticas constructivas

Parten de una solución inicial vacía y van añadiéndole componentes hasta construir una solución. En general, las soluciones obtenidas con estos métodos suelen ser de una calidad moderada y es necesario aplicar un algoritmo de mejora a las soluciones construidas.

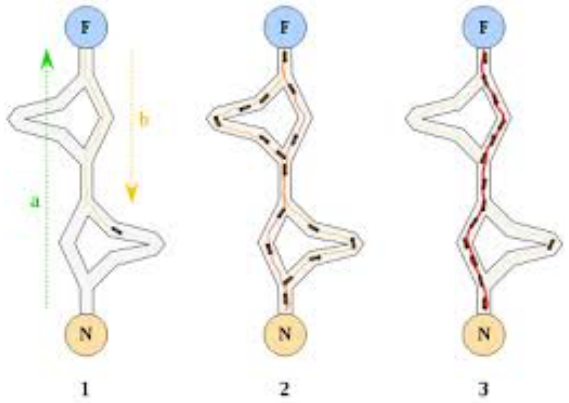
Colonias de hormigas (*ant colonies*). Dorigo et al (1996) y Dorigo y Stützle (2003, 2004). Se inspira en el comportamiento real de las hormigas para encontrar un camino desde el hormiguero a la comida.

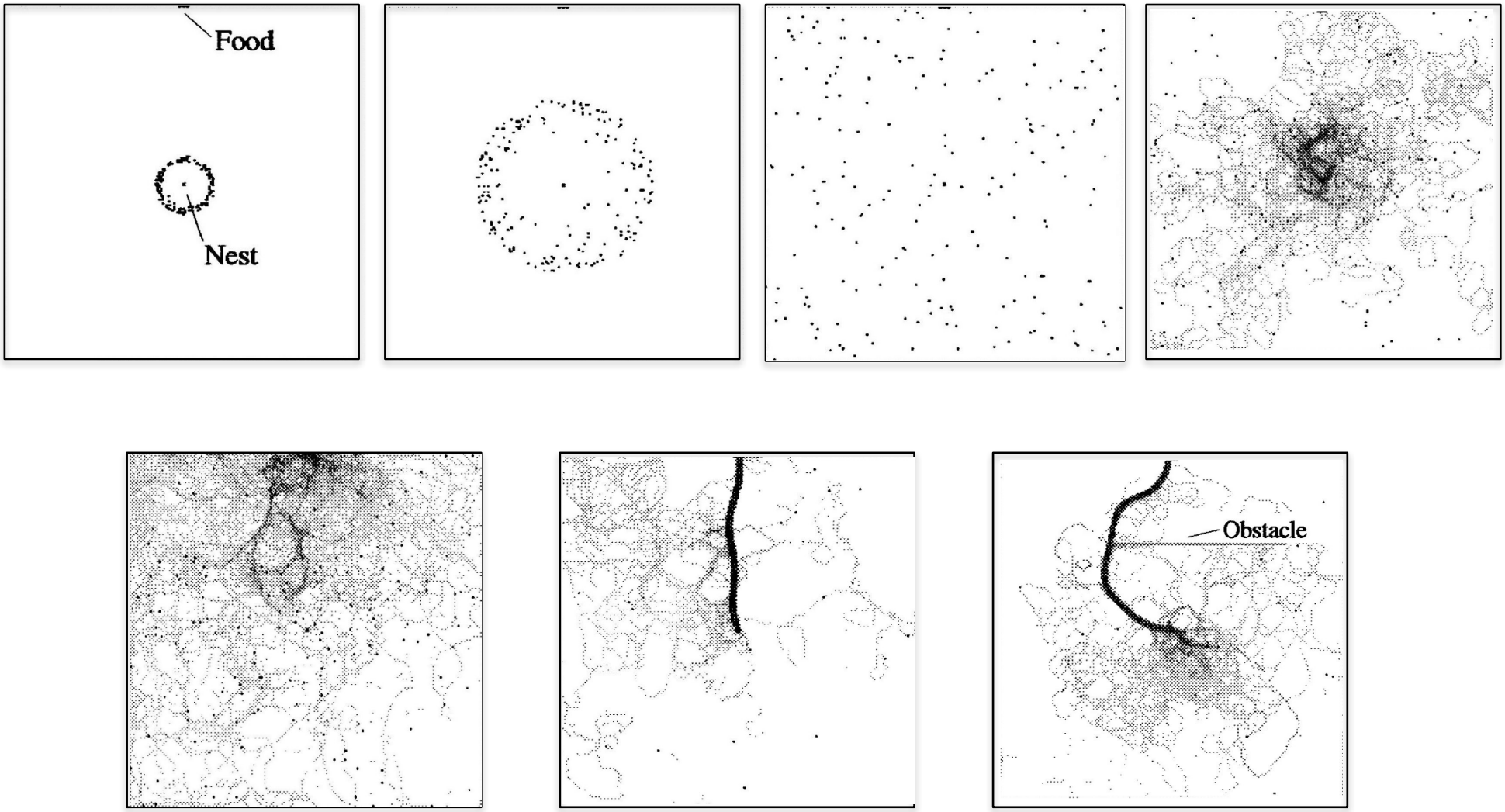
Cada hormiga artificial es un mecanismo probabilístico de construcción de soluciones al problema que usa:

- *Rastros de feromona*. Cambian con el tiempo para reflejar la experiencia adquirida por los agentes en la resolución del problema. Los caminos más prometedores irán aumentando su nivel de feromonas mientras que los menos prometedores irán disminuyéndolo.

- *Información heurística*. Al ser una metaheurística constructiva, el procedimiento necesita información heurística extra para determinar el modo en el que se van añadiendo componentes a la solución.

El procedimiento, por tanto, consiste en una serie de hormigas explorando el espacio de búsqueda. Estas van eligiendo los arcos entre nodos del grafo mediante una función probabilística basada en la cantidad de feromona que haya en el arco e información heurística del problema. El algoritmo tiende pues a aumentar el nivel de feromona en los arcos que forman el camino optimo.




Food
Nest
Obstacle

GRASP (*Greedy Randomized Adaptive Search Procedure*) o **Procedimiento voraz de búsqueda aleatorizado y adaptativo.** Feo y Resende (1995), Resende y Ribeiro (2003) .

Metaheurística de múltiple comienzo iterativa. Cada iteración consiste en dos fases:

- **fase de construcción** se construye de forma iterativa una solución factible inicial, añadiendo un elemento en cada paso cuya elección viene determinada por una función voraz, que mide el beneficio asociado a su incorporación.

- Seguidamente, tras haber obtenido esta solución inicial, se aplica sobre ella un **proceso de búsqueda local**.

Ventaja características de GRASP → facilidad de combinarlo con otras estrategias de búsqueda o metaheurísticas, reemplazando el algoritmo de búsqueda local por otro tipo de metaheurística.