

# Metaheurísticas

## Seminario 1. Ejemplos de resolución de problemas con metaheurísticas: problemas clásicos y reales

---

1. Introducción: Optimización
2. Problemas de Optimización Combinatoria Clásicos
3. Ejemplos de Problemas de Optimización Reales
4. Software de Metaheurísticas

# 1. Introducción: Optimización

---



*¿Optimizar?*

*- Mejorar*

*- Buscar la mejor manera de realizar una actividad*

- **Contexto científico:** *La optimización es el proceso de tratar de encontrar la mejor solución posible para un determinado problema.*
- **Nivel Empresarial:**
  - Reducir los costes.
  - Mejorar la Experiencia del cliente.

# 1. Introducción: Optimización

---

- Problema de optimización:
- ✓ Diferentes **soluciones**, un criterio para **discriminar** entre ellas y el **objetivo** es encontrar la mejor

*Encontrar el valor de unas variables de decisión (sujeto a restricciones) para los que una determinada función objetivo alcanza su valor máximo o mínimo*

# 1. Introducción: Optimización

---

- Problema de optimización (maximización):

- ✓ Dado un dominio  $X$  y una función objetivo

$$f(x): X \in X \rightarrow R$$

- ✓ El objetivo es encontrar  $x^*$  que verifique

$$x^* \in X: f(x^*) \geq f(x), \forall x \in X$$

- Optimización combinatoria  $\rightarrow$  Variable discreta

*“Un Problema de Optimización Combinatoria consiste en encontrar un objeto entre un conjunto finito (o al menos contable) de posibilidades”*

# 1. Introducción: Optimización

---

- Tipos de problemas de optimización (representación de una solución):
  - ✓ Permutaciones (Problemas de ordenación)
  - ✓ Binarios (Problemas de pertenencia)
  - ✓ Enteros (Problemas de cardinalidad, asignación, selección)
  - ✓ De optimización numérica (Optimización de funciones no lineales)

# 1. Introducción: Optimización

---

- Problemas de optimización fáciles de resolver:
  - ✓ *Lineales*: función objetivo y restricciones lineales (método *Simplex*)
- Problemas de optimización difíciles de resolver (NP-duros):
  - ✓ No se puede garantizar el encontrar la **mejor** solución posible en un **tiempo** razonable
  - ✓ Mayoría de los problemas con **aplicación** práctica, científica o industrial
  - ✓ Desarrollo de procedimientos **eficientes** para encontrar buenas soluciones aunque sean **no óptimas**

## 2. Problemas de Optimización Combinatoria Clásicos

---



- **Problemas de explosión combinatorios.**
- **Pocos modelos teóricos**

## 2. Problemas de Optimización Combinatoria Clásicos

---

- Viajante
- Mochila
- Coloreado de Grafos
- Partición de Conjuntos
- Asignación Cuadrática
- Asignación Generalizada
- Ordenación Lineal
- Diversidad
- Enrutamiento de vehículos
- ...

# Viajante de Comercio

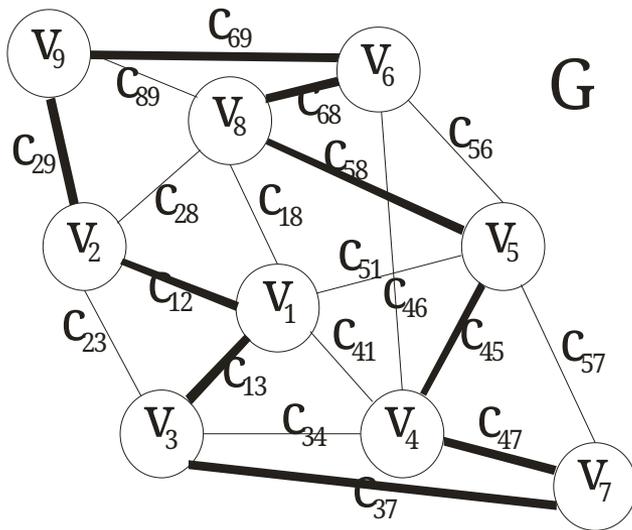
- Problema del Viajante de Comercio:
  - Travelling Salesman Problem.
- Problema:
  - Encontrar la ruta más rápida entre  $N$  ciudades.
  - Por cada ciudad se pasa una única vez.
  - Se debe de volver a la ciudad origen.



# Viajante de Comercio

## ■ Definición Problema del Viajante de comercio, TSP:

*Un viajante de comercio ha de visitar  $n$  ciudades, comenzando y finalizando en su propia ciudad. Conociendo el coste de ir de cada ciudad a otra, determinar el recorrido de coste mínimo*



$$TSP = \left\{ \begin{array}{l} \min : \\ c_{\pi(1)\pi(n)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} \end{array} \right.$$

# ¿Para qué sirve?

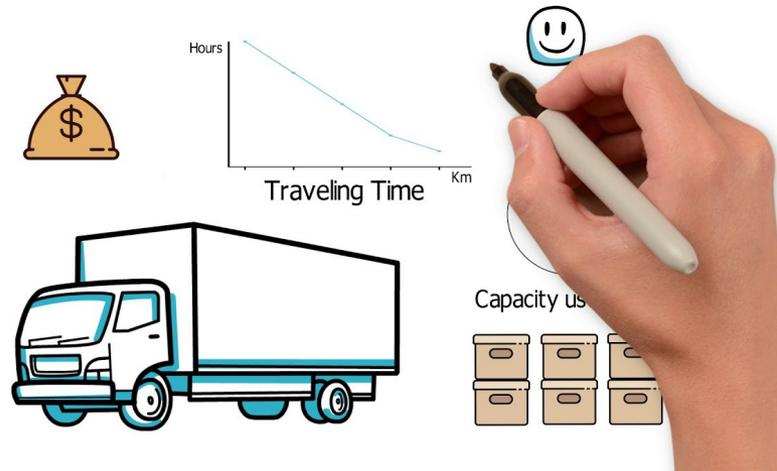
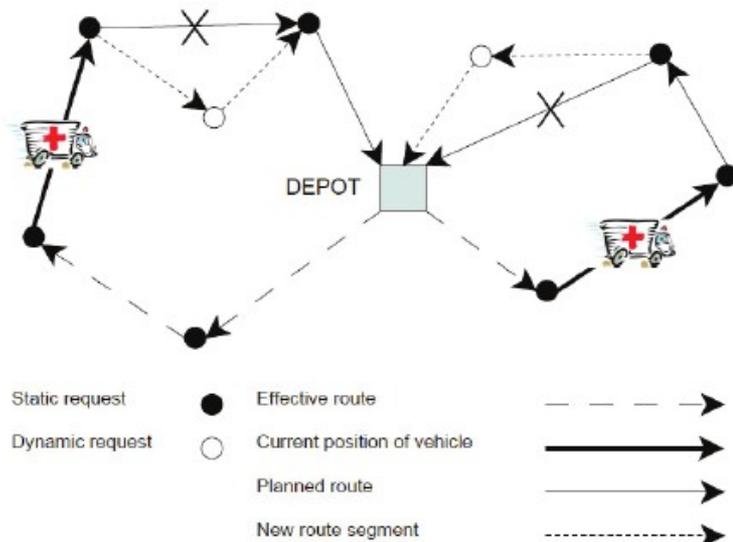
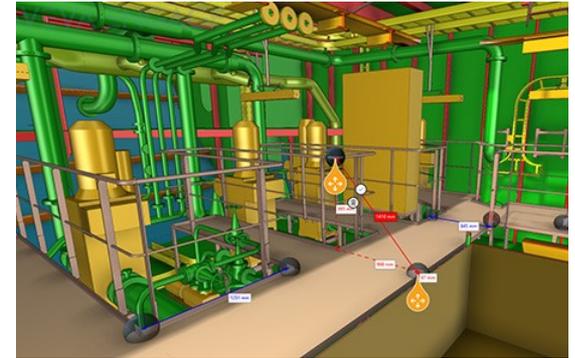
## ■ Muchas Aplicaciones

Diseño chips: Rutas más cortas.

Rutas aéreas: Entre aeropuertos.

Reparto almacén: Optimizar entregas.

Cableado (eléctrico): Recorrido mínimo.



# Tiempos

Ciudades (N)	Fuerza Bruta	Algoritmo Held-Karp
10	2 segundos	0.1 segundo
11	22 segundos	0.2 segundos
14	13 horas	3 segundos
16	200 días	11 segundos
25	270000 años	4 horas
50	$5 \cdot 10^{50}$ años	58000 años

<https://nbviewer.jupyter.org/url/norvig.com/ipython/TSP.ipynb>

No es abordable con ningún algoritmo tradicional.

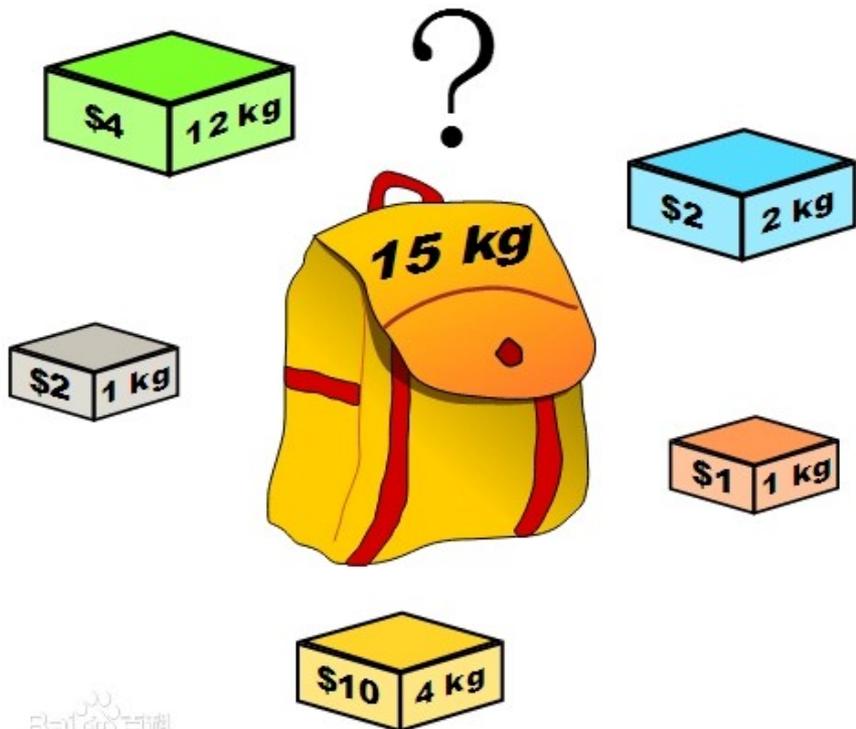
***¡Necesitamos buenos algoritmos y eficientes!***

***Algoritmos que proporcionen una buena solución en un tiempo razonable***

# Problema de la Mochila

## ■ Problema de la mochila, *Knapsack Problem*:

*Dados  $n$  objetos, cada uno con un peso  $w_j$  y un valor  $v_j$ , se debe seleccionar el conjunto de objetos cuyo valor total sea máximo, sin exceder un peso máximo  $W$*



$$KP = \begin{cases} \max : & \sum_{i=1}^n v_i x_i \\ \text{s.a.}, & \\ & \sum_{i=1}^n w_i x_i \leq W \end{cases}$$

# Versión cuadrática del problema

- **Problema cuadrático de la mochila, *Knapsack Quadratic knapsack problem (KQP)*:**  
Hay beneficio combinado de objetos



# Versión cuadrática del problema

---

- Problema cuadrático de la mochila, *Knapsack Quadratic knapsack problem (KQP)*:

Evaluación de soluciones

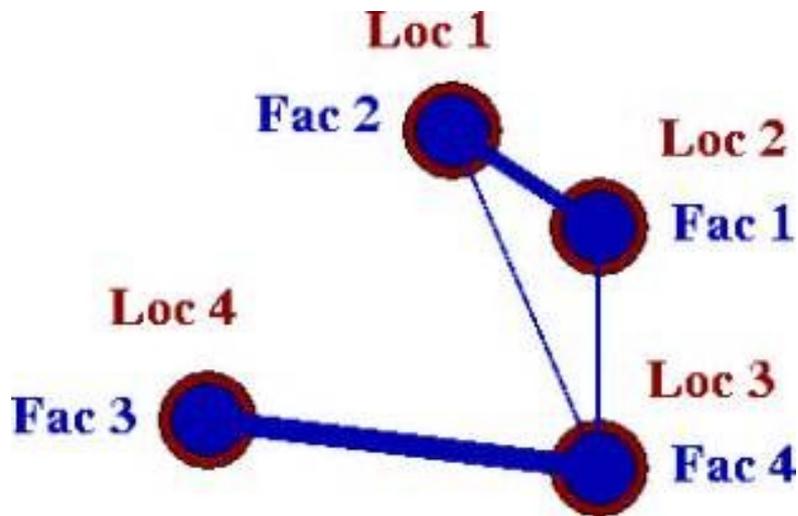
$$\text{maximizar} \left\{ \sum_{i=1}^n v_i \cdot x_i + \sum_{i=1}^n \sum_{j=1, j \neq i}^n v_{ij} \cdot x_i \cdot x_j : x_i \in \{0, 1\} \right\}$$

$$\text{X debe cumplir que } \left( x \in \{0, 1\}^n : \sum_{i=1}^n w_i \cdot x_i \leq W \right)$$

# Problema de la Asignación Cuadrática

## ■ Problema de la asignación cuadrática, QAP:

*Dadas  $n$  unidades y  $n$  localizaciones posibles, el problema consiste en determinar la asignación óptima de las unidades en las localizaciones conociendo el flujo existente entre las primeras y la distancia entre las segundas*

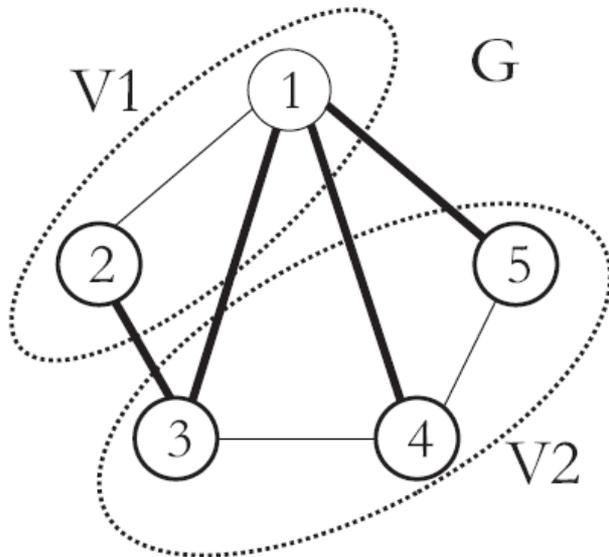


$$QAP = \min_{S \in \Pi_N} \left( \sum_{i=1}^n \sum_{j=1}^n f_{ij} \cdot d_{S(i)S(j)} \right)$$

# Problema del Corte Máximo

## ■ Problema del corte máximo, *Max-Cut Problem*:

*Dado un grafo ponderado y no dirigido y un conjunto de pesos asociados a cada arista, encontrar una partición del conjunto de vértices en dos subconjuntos disjuntos, tal que maximice la suma de los pesos en aristas cuyos extremos se encuentran en subconjuntos distintos*

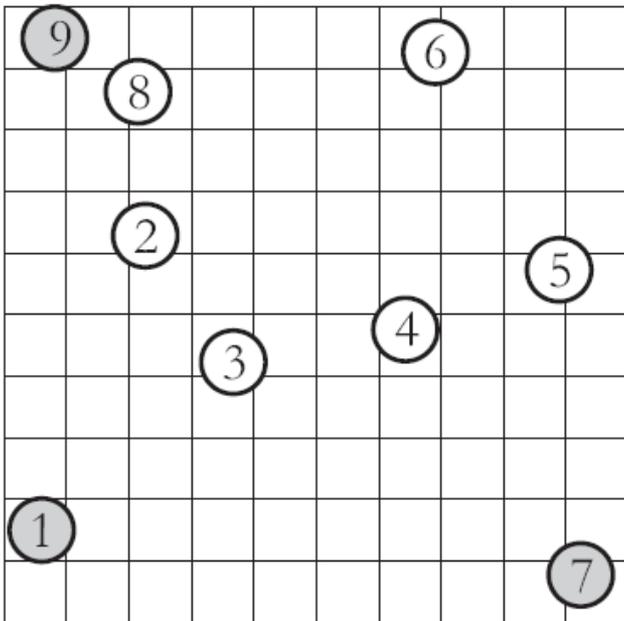


$$MCP = \begin{cases} \max : & \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij}(1 - y_i y_j) \\ \text{s.a.}, & \\ & y_i \in \{-1, 1\}, \quad \forall i \in V \end{cases}$$

# Problema de la Máxima Diversidad

## ■ Problema de la máxima diversidad, MDP:

*Seleccionar un conjunto de elementos  $m$  de una colección más grande  $n$  de tal forma que los elementos seleccionados tengan las características más variadas entre sí*

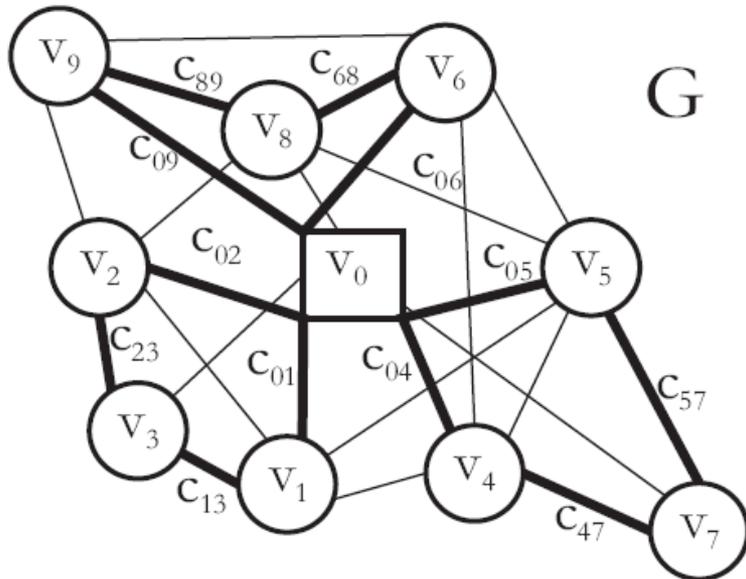


$$MDP = \left\{ \begin{array}{l} \max : \sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{ij} x_i x_j \\ \text{s.a.}, \\ \sum_{i=1}^n x_i = m \\ x_i = \{0, 1\} \mid 1 \leq i \leq n \end{array} \right.$$

# Enrutamiento de Vehículos

## ■ Problema del enrutamiento de vehículos, VRP:

*Obtener el conjunto de rutas más cortas posibles utilizando un conjunto de vehículos (con capacidad limitada) lo más pequeño posible tal que, partiendo de un almacén y regresando sucesivamente a él, abastezcan a una serie de clientes (con demanda diferente) distribuidos geográficamente*



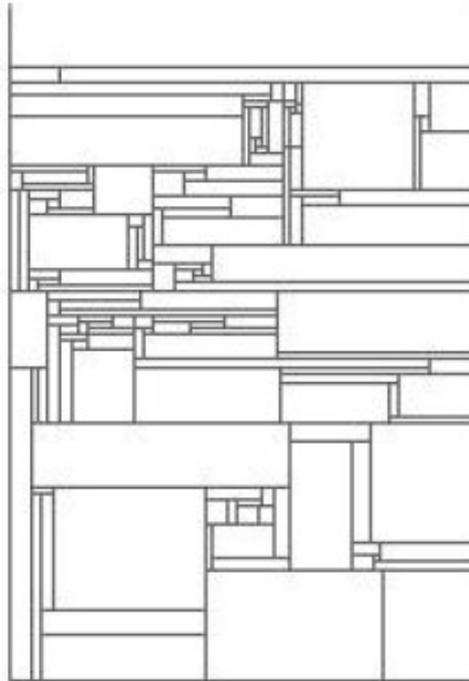
$$VRP = \left\{ \begin{array}{l} \min : \sum_{k=1}^m \sum_{i=1}^{n_k} C_{\pi^k(i)\pi^k(i+1)} \\ \text{s.a.}, \\ \sum_{i=1}^{n_k} q_{\pi^k(i)} \leq Q_k \quad \forall k \in [1, m] \\ \sum_{k=1}^m n_k \leq n \end{array} \right.$$

# Empaquetado en Cinta

---

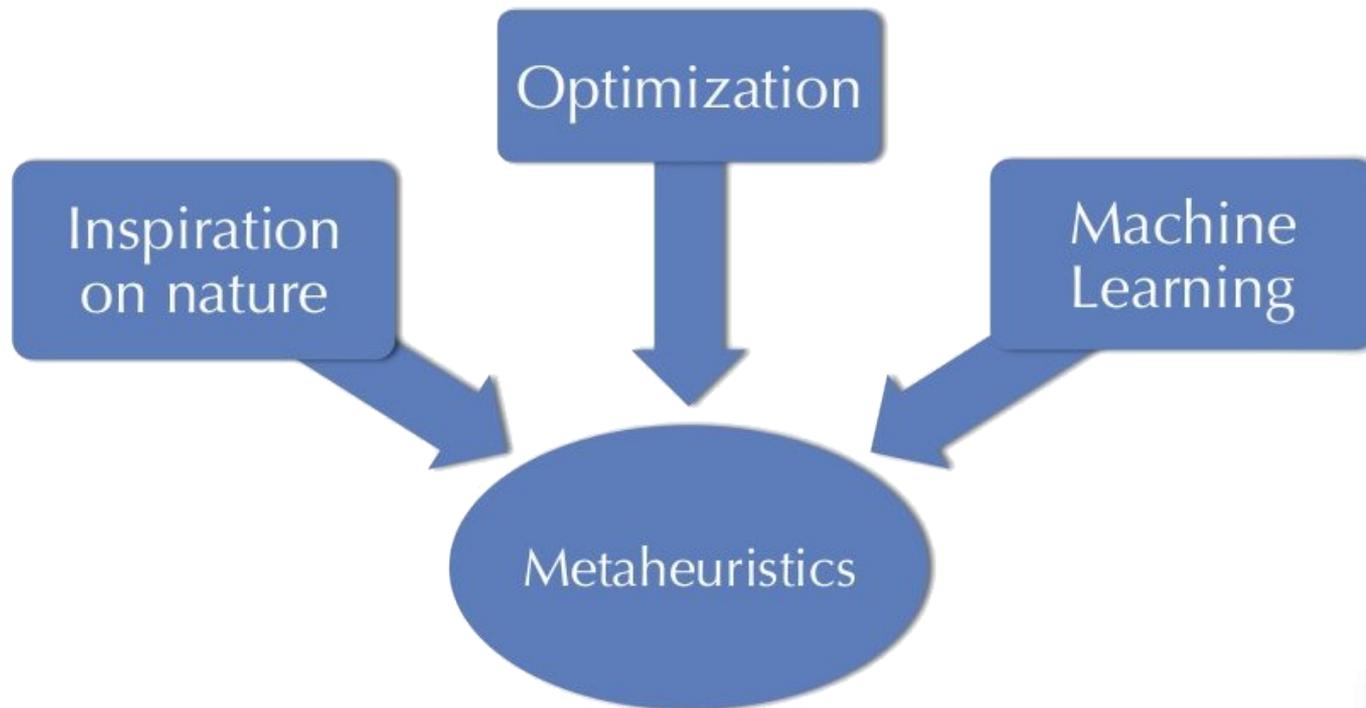
## ■ Problema del empaquetado en cinta, SPP:

*Dado un conjunto de rectángulos de diferentes dimensiones se colocan sobre una cinta de anchura fija y altura indefinida de tal forma que se minimice la altura alcanzada por dicha colocación*



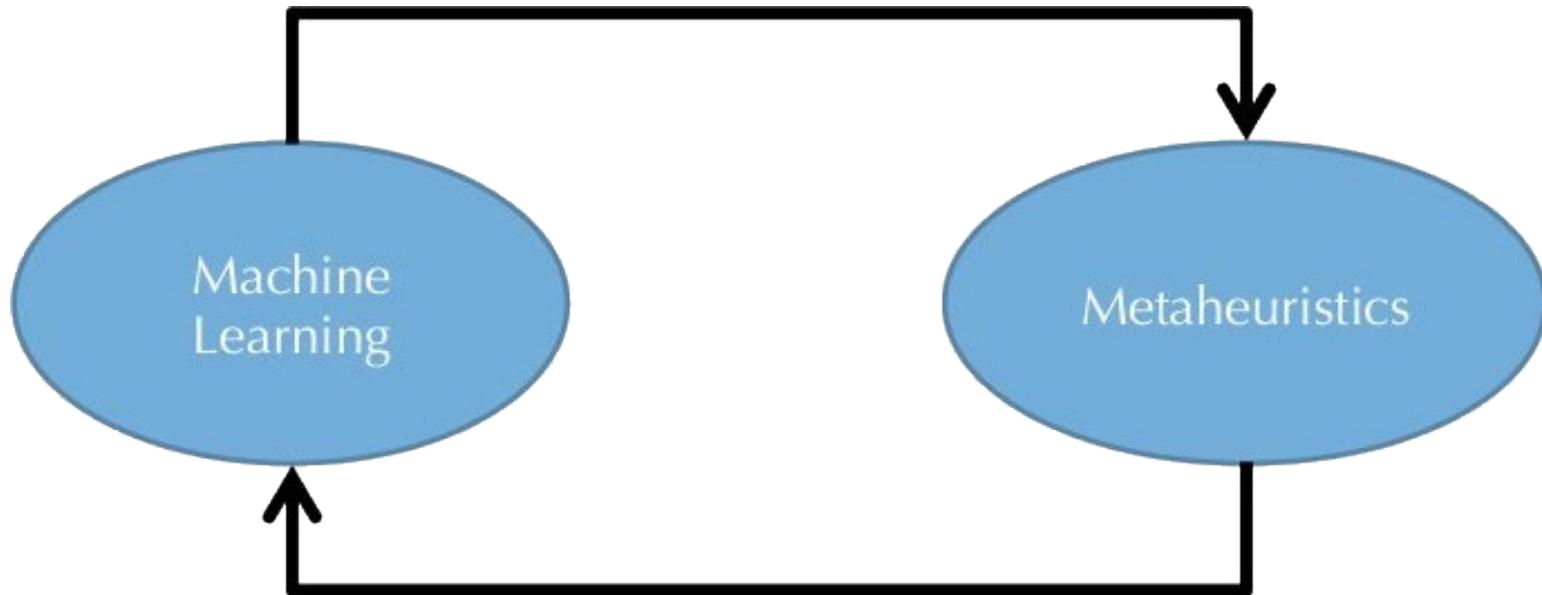
# Metaheurísticas

---



# Optimización y *Machine Learning*

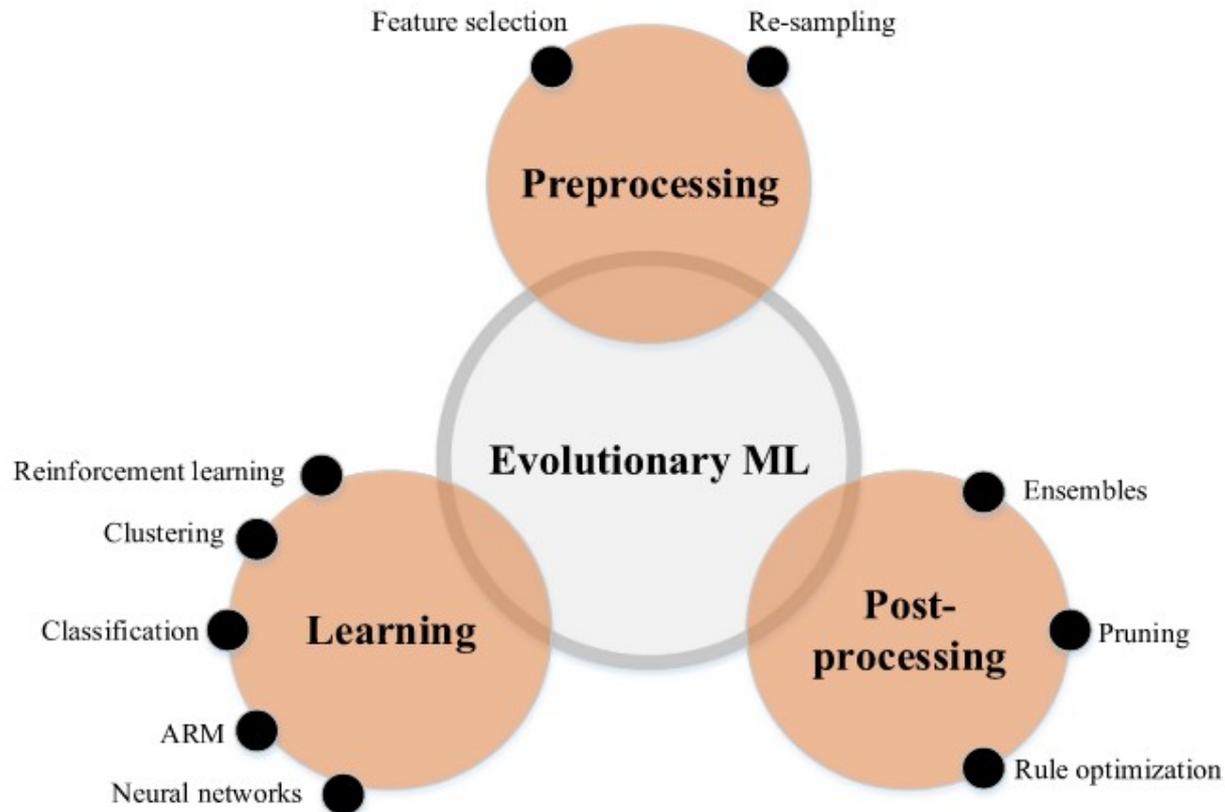
---



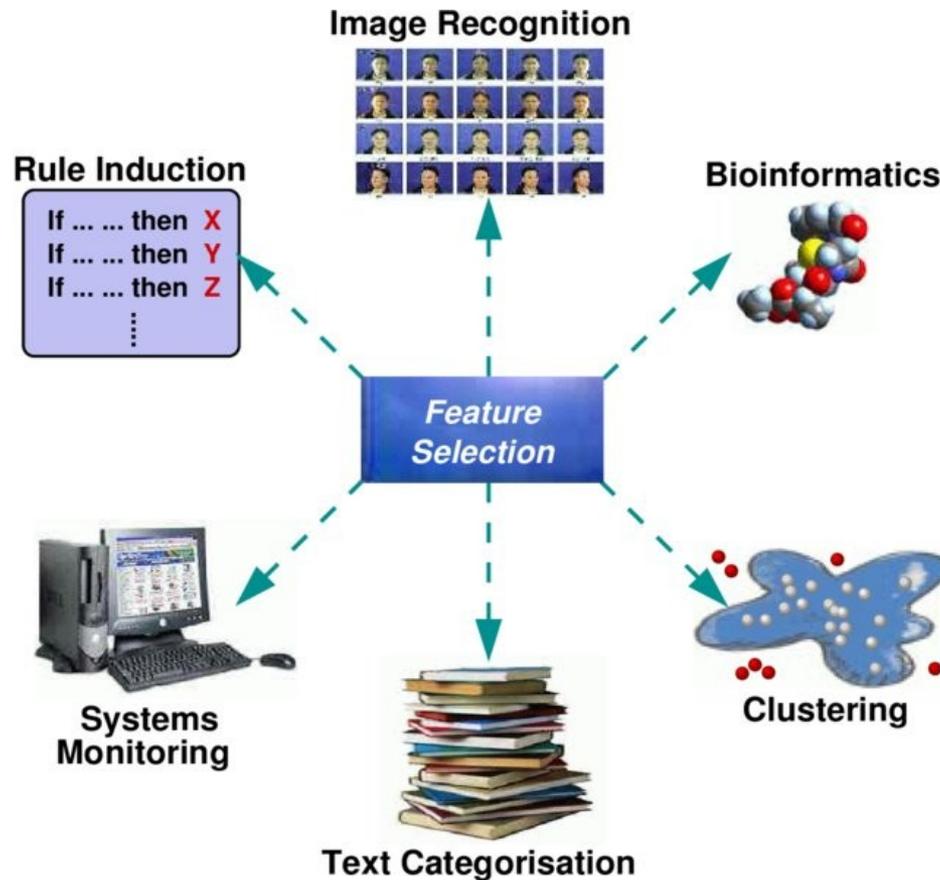
- El uso de metaheurísticas se pueden aplicar a técnicas de aprendizaje automático y viceversa.

# Optimización y *Machine Learning*

---



# Ejemplo: Feature Selection



# Ejemplo: Feature Selection

---

All Features



Feature Selection

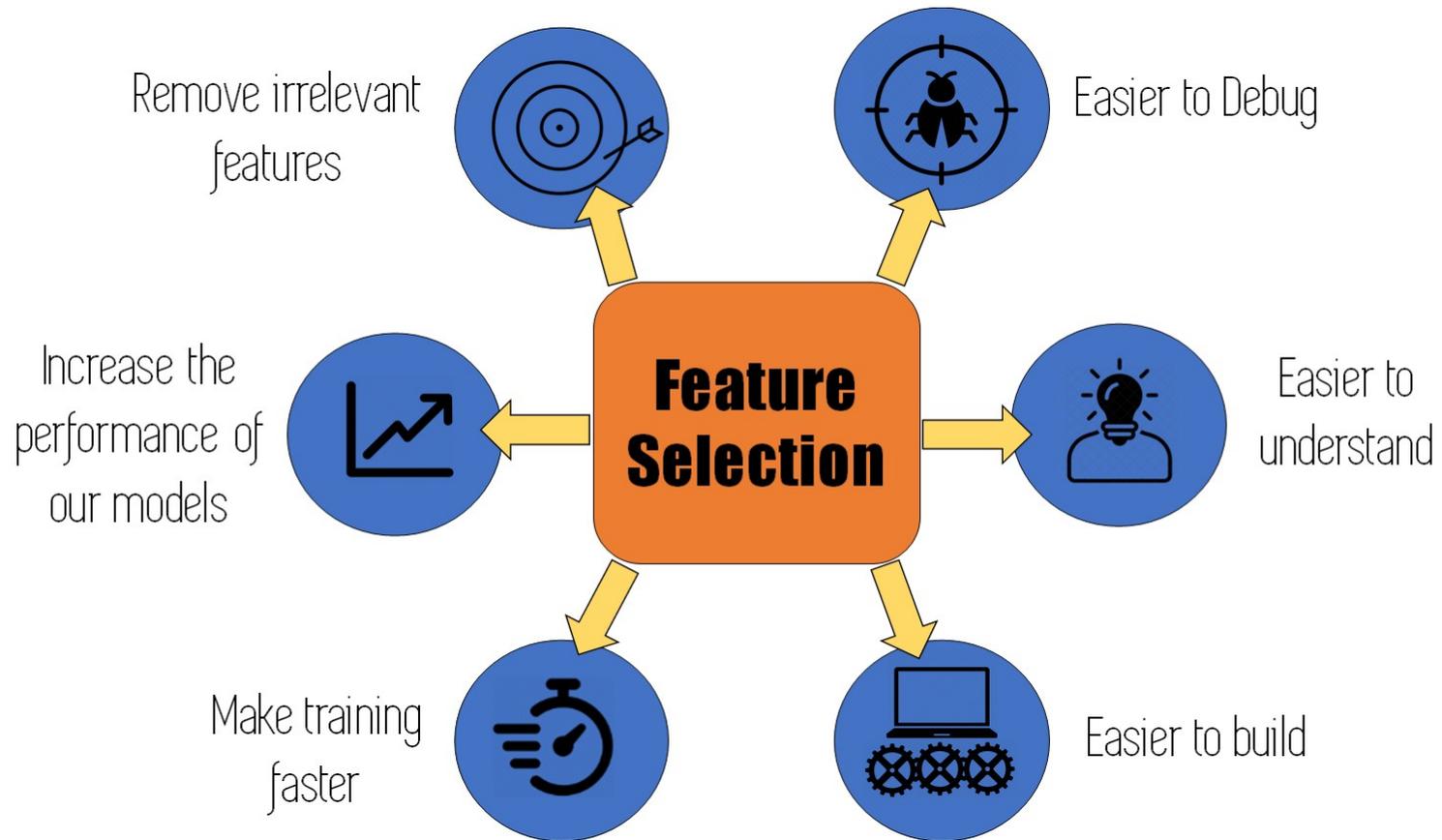


Final Features



# Ejemplo: Feature Selection

---



# Ejemplo: Feature Selection

---

- Ponderar características en un modelo de *Machine Learning* que use distancias (*KNN*).

En vez de:

$$Distancia(X, Y) = \sqrt{\left( \sum_{i=1}^N (x_i - y_i)^2 \right)}$$

**Todas se consideran igualmente importantes**

**En Realidad no suele ser así**

# Ejemplo: Feature Selection

---

- Ponderar características en un modelo de *Machine Learning* que use distancias (*KNN*).

Usamos:

$$Distancia(X, Y) = \sqrt{\left( \sum_{i=1}^N peso_i \cdot (x_i - y_i)^2 \right)}$$

**Algunas características son más importante que otras**

**¿Cómo calcular los mejores pesos?**

# 3. Algunos Ejemplos de Problemas de Optimización Reales

---

- Diseño Aerodinámico
- Planificación de Rutas para Transporte de Mercancías
- Canalización automática
- Juegos
- Equilibrado de Líneas de Montaje en Nissan
- Preparación de Envíos de Venta Online en Almacenes
- Identificación Forense de Personas Desaparecidas

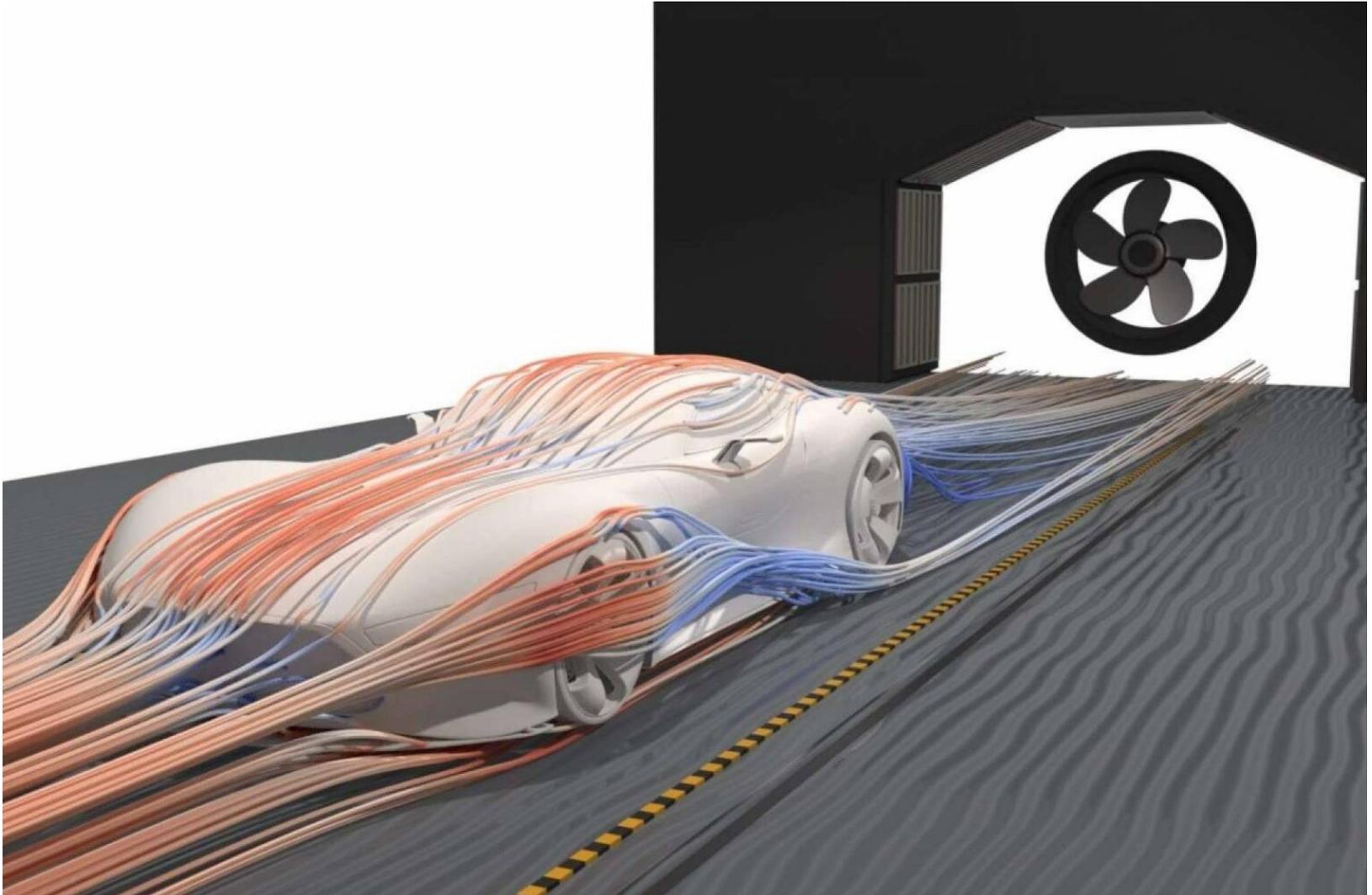
# Diseño Aerodinámico

---

- La **disminución de la resistencia al avance** es clave
- Se calcula resolviendo unas ecuaciones que simulan el comportamiento de un objeto sólido (el avión) en interacción con un fluido (el aire), según la **Dinámica Computacional de Fluidos**
- Después se usan **métodos de optimización** para obtener la forma óptima del avión que minimiza la resistencia verificando los requisitos geométricos y físicos
- Los diseños prometedores mediante la simulación computacional son validados en el **túnel de viento**

# Diseño Aerodinámico

---



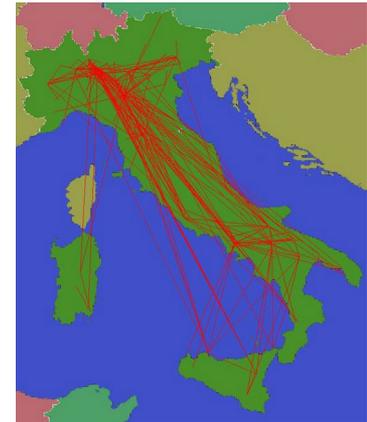
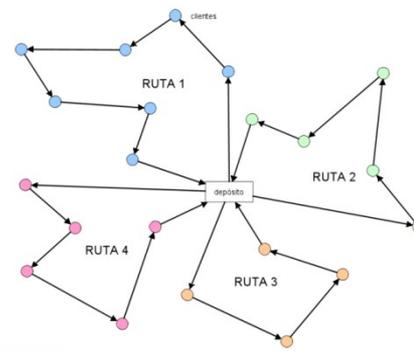
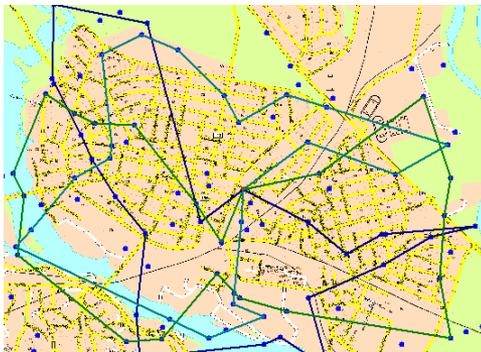
# Diseño Aerodinámico

---

- En casos reales, cada ejecución puede requerir meses de cómputo en máquinas de alto rendimiento al tener que realizar una gran cantidad de simulaciones
- Por ello, se han empleado los **algoritmos evolutivos** para esta tarea, que son capaces de proporcionar **diseños de buena calidad en un tiempo mucho más reducido**
- Además, como optimizadores multiobjetivo, **pueden optimizar varios criterios a la vez** (velocidad, estabilidad o gasto de combustible, por ejemplo)

# Planificación de Rutas para Transporte de Mercancías

- Hoy en día es difícil encontrar empresas que gestionen las **operaciones de logística** sin la ayuda del ordenador
- El problema típico es **diseñar las rutas más adecuadas de transporte/recogida de productos** entre un almacén central y unos destinos dispersos geográficamente



- Su resolución de forma adecuada puede suponer **ahorros muy significativos para la empresa**

# Planificación de Rutas para Transporte de Mercancías

- Esta tarea se lleva a cabo empleando una **flota de vehículos** pertenecientes o no a la empresa



- Un **sistema de planificación de vehículos** debe proporcionar un conjunto de **rutas de reparto** a los conductores
- Las mercancías deben ser entregadas cuándo y donde se requieran, con el mínimo coste posible y verificando todas las restricciones legales y políticas de la empresa
- Los **algoritmos de hormigas (AntRoute)** son una herramienta muy potente para la planificación de rutas

# Planificación de Rutas para Transporte de Mercancías



- **AntRoute** planifica diariamente las rutas de reparto desde el almacén central de **Migros**, una gran cadena suiza con 600 supermercados, localizado en **Suhr (AG)**, a **toda Suiza**
- Migros dispone de una **flota de entre 150 y 200 vehículos** con tres tamaños: camiones (capacidad de 17 palés), trailers (35 palés) y unidades tractoras (33 palés)
- Esto provoca restricciones de acceso a los almacenes de los supermercados, restricciones de uso de ciertas carreteras, ...
- Los repartos tienen de realizarse a horas específicas, todos ellos en un solo día (**productos perecederos**) y el último tiene que hacerse lo más lejos posible del almacén (**servicios extra**)

# Planificación de Rutas para Transporte de Mercancías

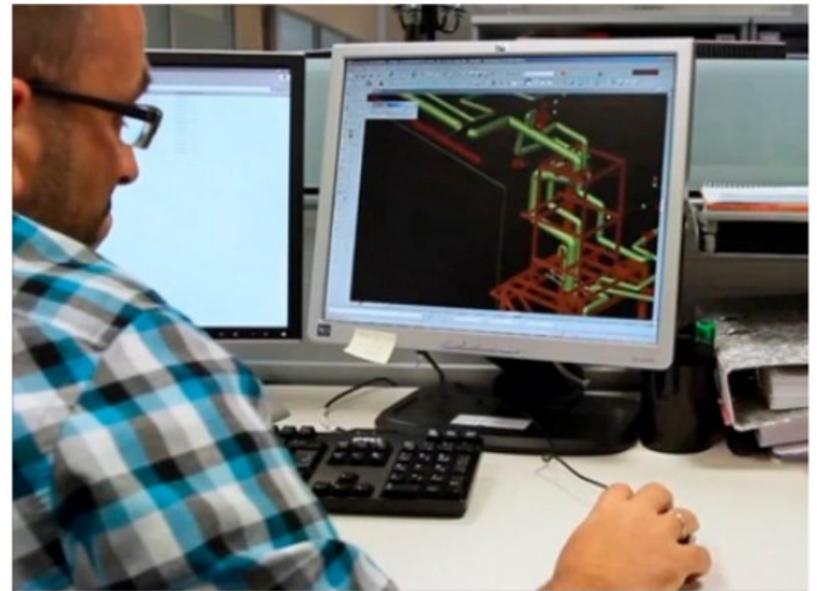
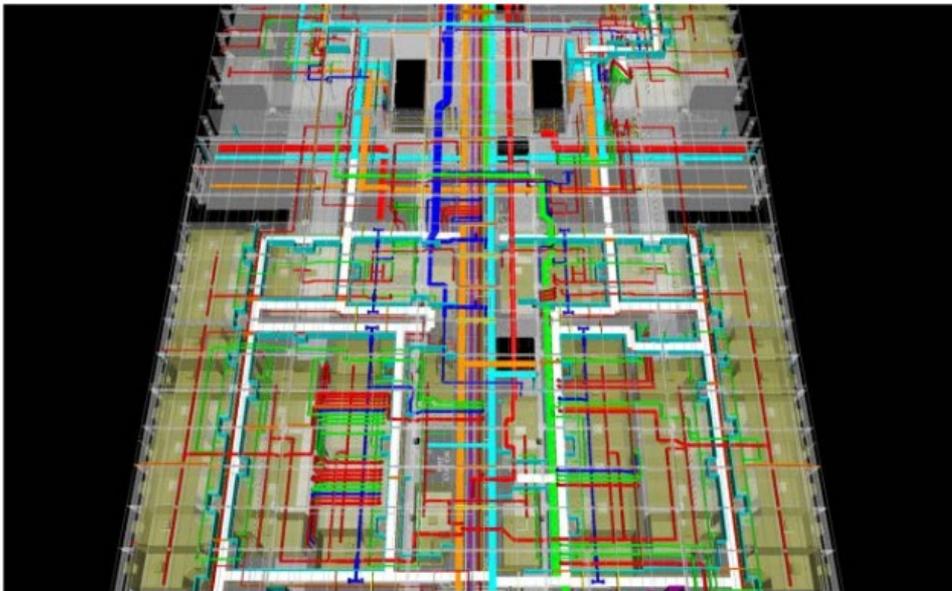


- Por ejemplo, en un reparto de 52000 palés a 6800 clientes en un periodo de 20 días, AntRoute obtuvo el diseño diario de rutas en menos de 5 minutos en un PC estándar
- Los expertos de la empresa necesitaron tres horas...
- Las soluciones de AntRoute fueron de mucha mejor calidad en cuanto al número de rutas necesario, la distancia total recorrida y al aprovechamiento de los vehículos:

	Human Planner	AR-RegTW	AR-Free
Total number of tours	2056	1807	1614
Total km	147271	143983	126258
Average truck loading	76.91%	87.35%	97.81%

# Optimizando canalización en buques

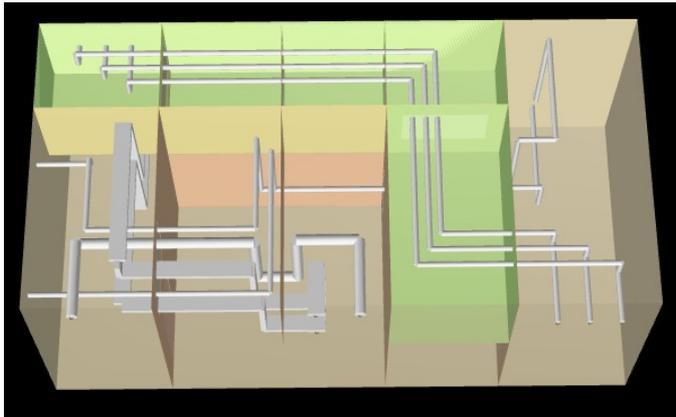
- Trabajamos con empresa de ingeniería para optimizar canalización del diseño de buques.
- Se diseñó un modelo automático con criterios que había que optimizar.



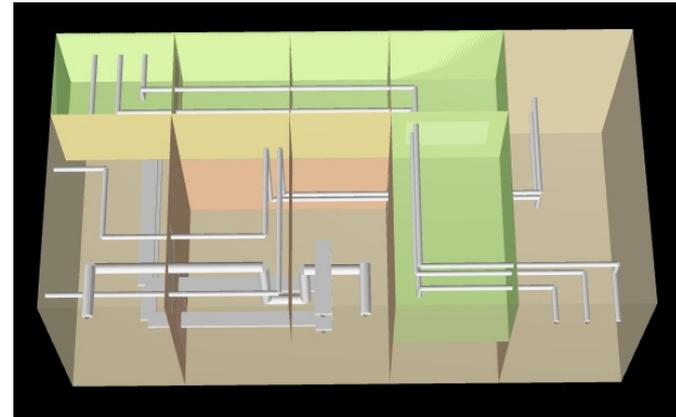
# Optimizando canalización en buques

- Problema real pero que se discretizó (ángulos fijos).
- Hablando con expertos, múltiples restricciones.
- Múltiples Objetivos:
  - Reducir longitud total de canales.
  - Reducir número de codos (cambios).

**Peor Solución**



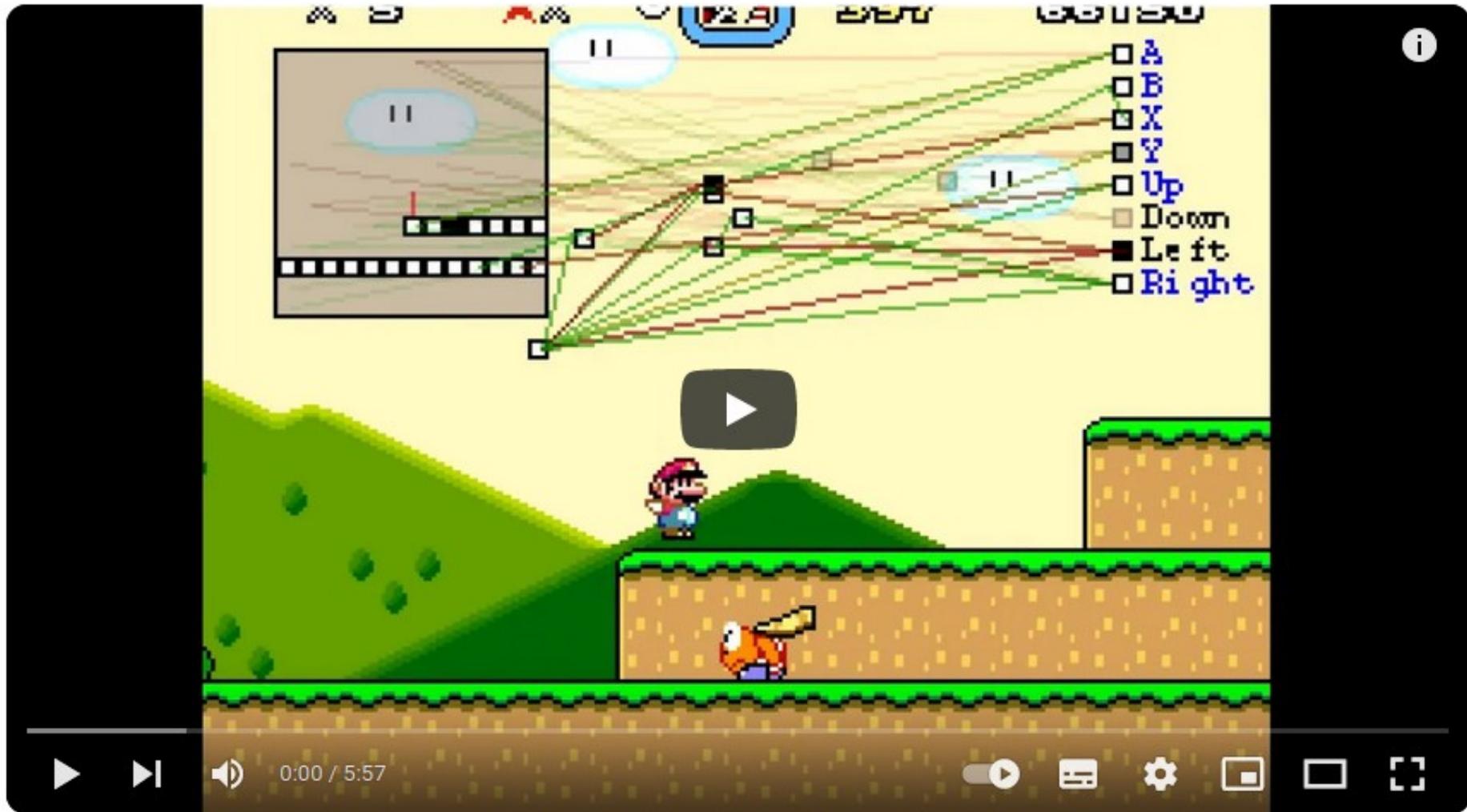
**Mejor Solución**



# Aproximando cuadro



# Optimizar una red para jugar



# Aprendiendo a jugar

---



- Lanzado en 2013.
- Más de 40 millones de jugadores.
- 9 héroes.
- Distintas cartas.



# Aprendiendo a jugar

---

- Objetivo: Crear mazos para Juego de Cartas intercambiables utilizando EAs.
- Cada mazo es una solución.
- El fitness se mide contra otros mazos.
- Aprendiendo jugando contra sí mismo.

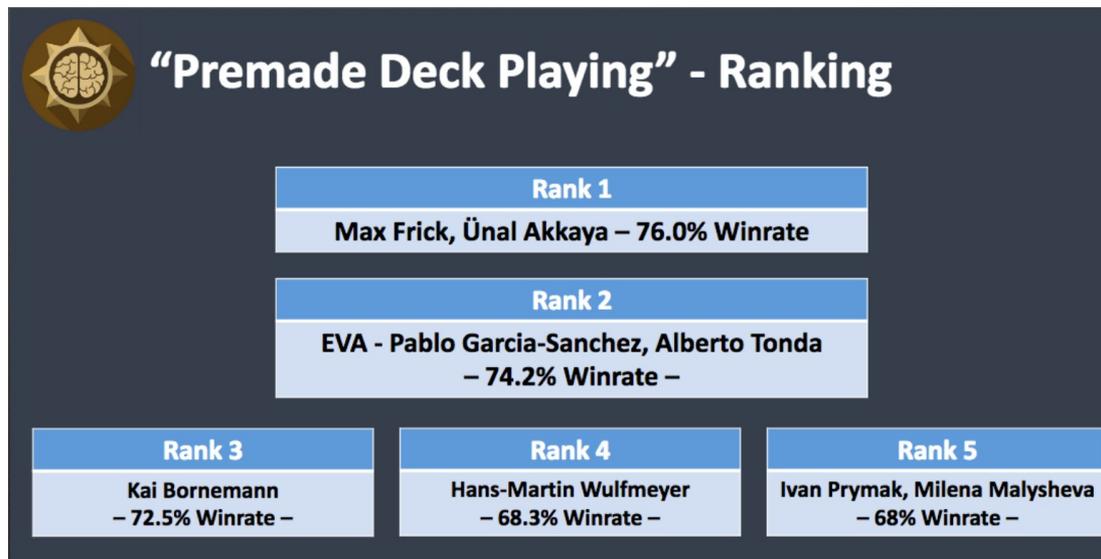


# Aprendiendo a jugar



# Aprendiendo a jugar

- Usando MetaStone, simulador del juego.
- 82% de victorias mejor solución.
- 9 turnos para ganar de media.
- Segundo mejor sistema:
  - 74% victorias competición.



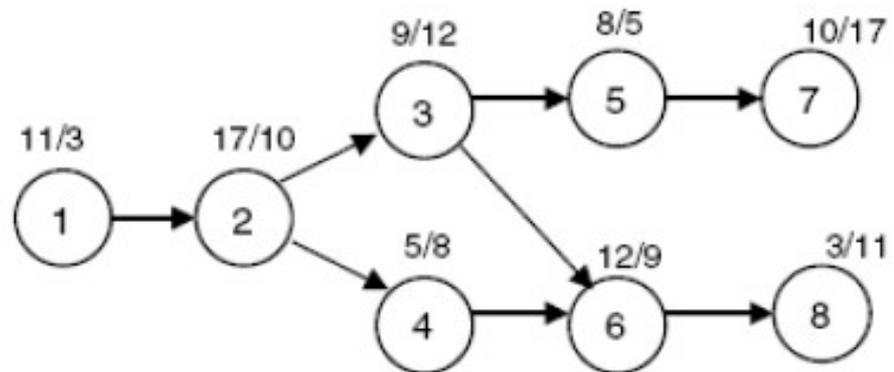
# Equilibrado de Líneas de Montaje en Nissan

- La mayoría de los sistemas productivos actuales se basan en líneas de montaje
- La producción de un ítem se divide en un **conjunto de tareas** que tienen que llevarse a cabo según un **orden concreto** y respetando una serie de **precedencias**
- Cada tarea necesita un tiempo dado (más un área de trabajo) y tiene asociada un conjunto de predecesores directos
- El diseño (**equilibrado**) de la línea requiere **agrupar de forma eficiente las tareas necesarias en estaciones de trabajo** para maximizar la producción y reducir tiempos muertos



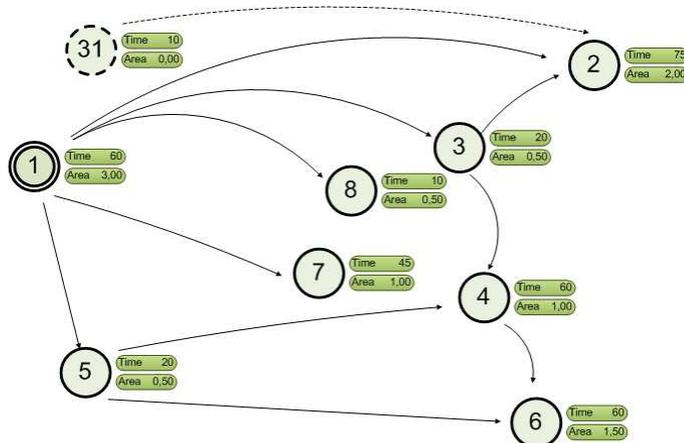
# Equilibrado de Líneas de Montaje en Nissan

- El parámetro clave es el **tiempo de ciclo** que indica el máximo tiempo permitido para que una estación procese sus tareas. **A menor tiempo de ciclo, mayor capacidad productiva de la línea**
- Los objetivos del equilibrado son:
  - agrupar las tareas en el menor número posible de estaciones de trabajo satisfaciendo un tiempo de ciclo, u
  - obtener la agrupación que minimiza el tiempo de ciclo



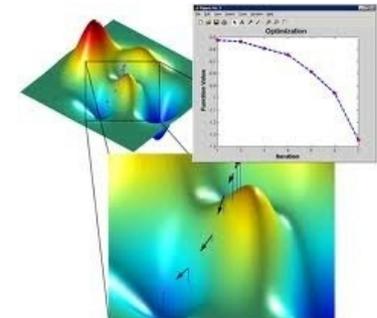
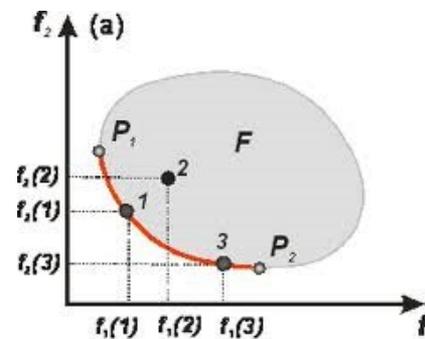
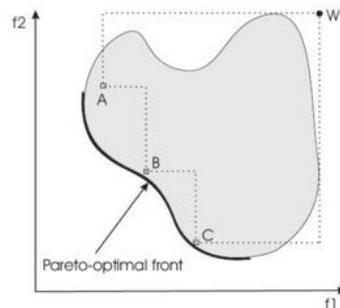
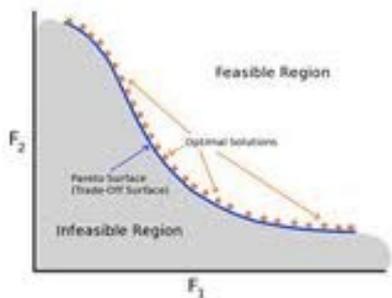
# Equilibrado de Líneas de Montaje en Nissan

- Los algoritmos de OCH se han aplicado con gran éxito al equilibrado de líneas de montaje, resolviendo problemas cada vez más complejos y realistas
- Trabajamos con la **Cátedra Nissan de la UPC** para resolver el **problema multiobjetivo de minimizar el número de estaciones y su área para un tiempo de ciclo dado** en la línea de montaje del **motor del Nissan Pathfinder:**



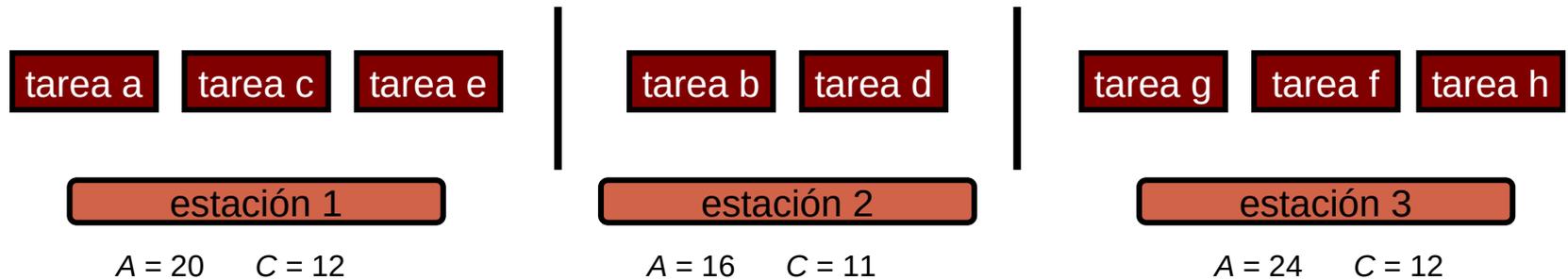
# Equilibrado de Líneas de Montaje en Nissan

- Es un **problema multiobjetivo** con **muchas restricciones fuertes y un espacio de búsqueda de gran dimensión**: oportunidad para metaheurísticas como los algoritmos de OCH
- El objetivo es proporcionar al ingeniero de planta con diversas opciones de diseño optimales con distinto equilibrio entre ambos objetivos en una sola ejecución del algoritmo



# Equilibrado de Líneas de Montaje en Nissan

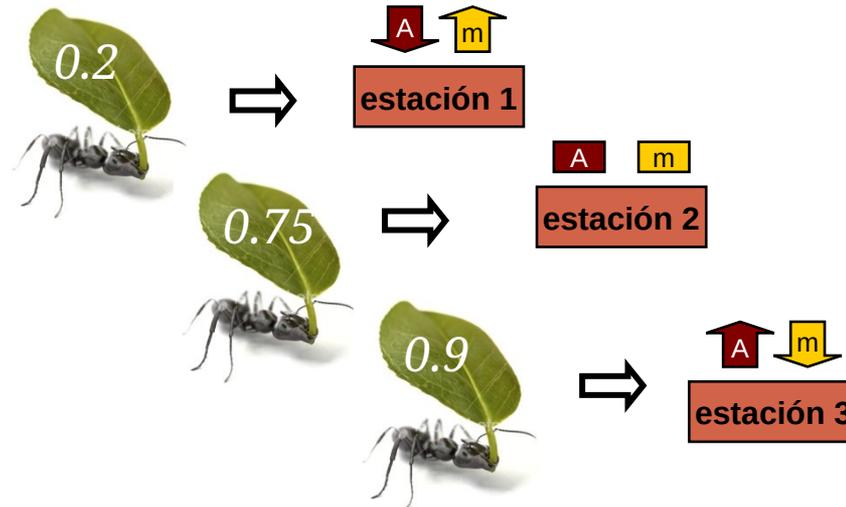
- Una solución a este problema (**TSALBP**) es una asociación de tareas a las distintas estaciones que cumpla las restricciones



- Hemos diseñado un **algoritmo de Optimización de Colonias de Hormigas multiobjetivo** que proporciona varias soluciones con un equilibrio distinto entre el área y el número de estaciones al ingeniero de la planta
- El rastro de feromona se asocia al **par (tarea, estación)**

# Equilibrado de Líneas de Montaje en Nissan

- Introducimos una **filosofía multicolonias** para obtener un mayor abanico de soluciones posibles: **cada hormiga utiliza distintos umbrales de llenado de estación**

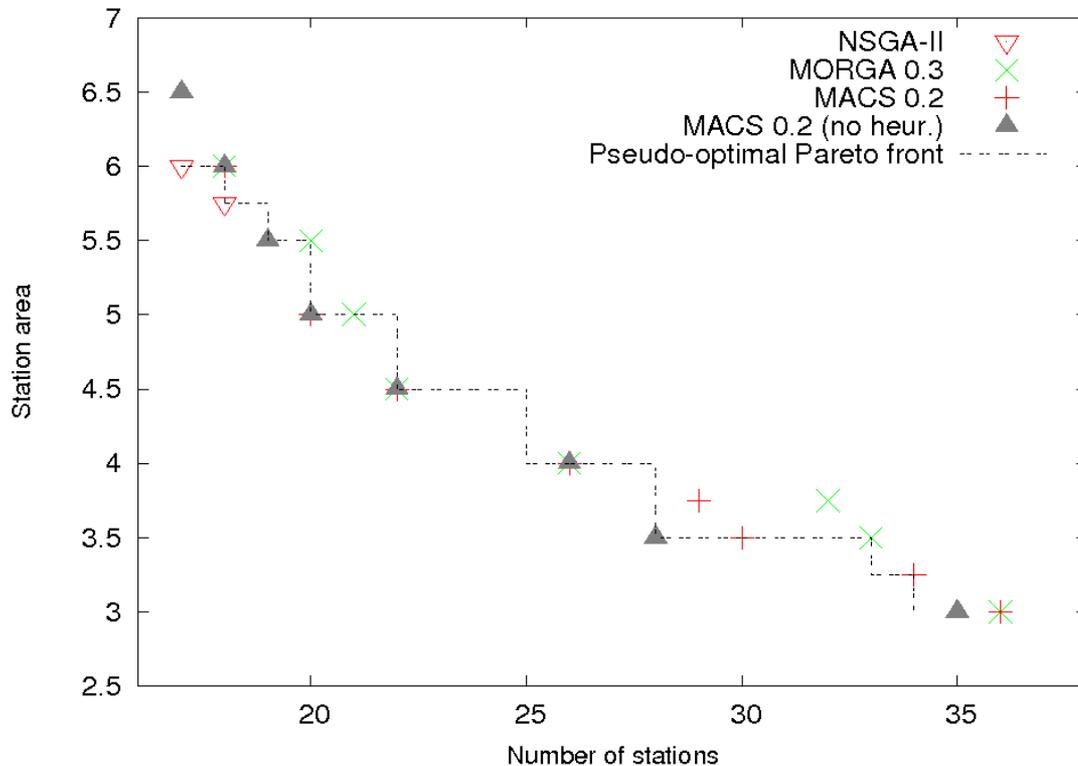


- Nuestra propuesta obtiene muy buenos resultados. El algoritmo de hormigas mejora a otras técnicas de búsqueda

# Equilibrado de Líneas de Montaje en Nissan

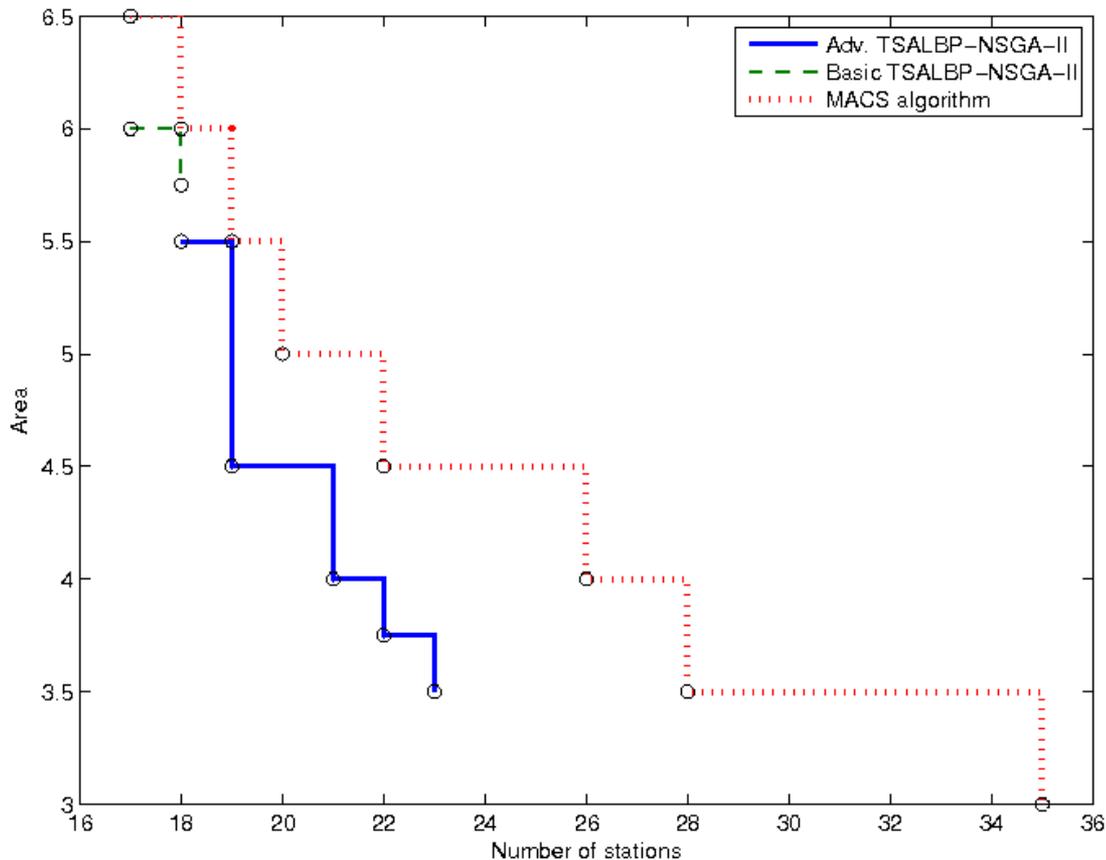
## Motor del Nissan Pathfinder:

- 747 piezas y 330 referencias en 6 versiones del motor diesel
- 378 operaciones de montaje (prueba rápida incluida) agrupadas en 140
- 79 operarios para un turno de 301 motores



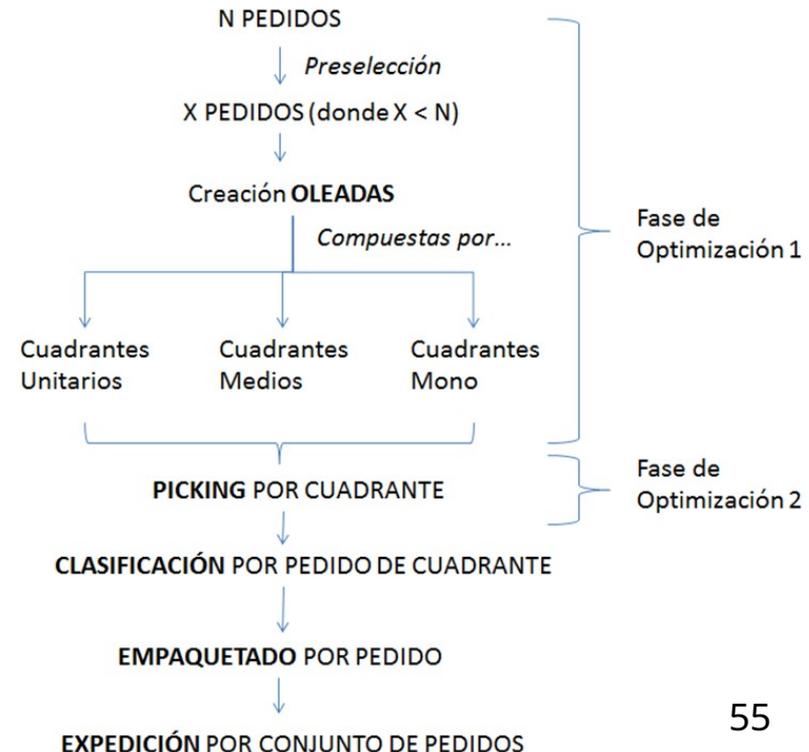
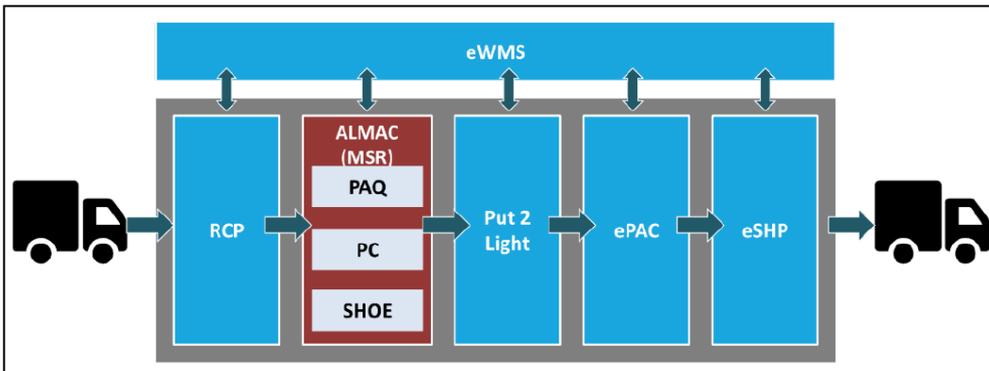
# Equilibrado de Líneas de Montaje en Nissan

Posteriormente diseñamos un algoritmo genético multiobjetivo específico para el TSALBP que mejora los resultados del algoritmo de OCH:



# Preparación de Envíos de Venta Online en Almacenes

- **Optimización del proceso de recogida de los productos para los pedidos de compra online**
- Inditex (Zara): de 30,000 a 500,000 pedidos diarios. Picos en los periodos de rebajas (ej. Black Friday en electrónica)
- Almacenes de gran tamaño y gran diversidad de productos → número exponencial de combinaciones



# Preparación de Envíos de Venta Online en Almacenes

---

## ■ **Datos de entrada:**

- Listado de pedidos online a gestionar
- Fecha de entrega (***due date***) de los pedidos
- Información detallada de cada **SKU** (stock-keeping unit = «código de artículo») de los pedidos
- **Localización** de cada SKU en el almacén
- **Matriz de distancias** de cada localización en el almacén

## ■ **Variables de decisión:**

- Pedidos a preparar en la siguiente “tacada” (**oleada**)
- Pedidos asociados con cada grupo
- Enrutamiento de recogida de cada grupo

# Preparación de Envíos de Venta Online en Almacenes

## ■ Restricciones:

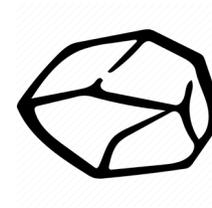
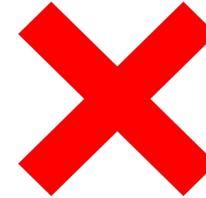
- **Número de pedidos por grupo** (depende del tamaño del área de preparación)
- **Nº de grupos por oleada**
- **Nº de empleados**
- **Nº de SKUs por grupo**
- **Fecha de entrega en el área de empaquetado**
- **Otras restricciones especiales**
- ...



- **Solución con Iterated Local Search diseñada por Helena Ramalinho de la Universidad Pompeu Fabra**

# Simplificando Redes Neuronales

- Buscando mejorar acierto.
- Distintos problemas de clasificación: médicos, plantas, ...



# Simplificando Redes Neuronales

- Distintos problemas de clasificación: médicos, plantas, ...
- Mejora un 5% de acierto (83-88%, 93-98%, ...).

Dataset	Measure	Not Pruned	Best Fixed	Feature Selection
SRSMAS	Accuracy	0.832	0.866	<b>0.884</b>
	% Active neur.	100	20	60
RPS	Accuracy	0.938	0.938	<b>0.985</b>
	% Active neur.	100	40	45
LEAVES	Accuracy	0.923	0.927	<b>0.943</b>
	% Active neur.	100	80	59
PAINTING	Accuracy	0.939	0.945	<b>0.958</b>
	% Active neur.	100	60	55
CATARACT	Accuracy	0.703	0.719	<b>0.747</b>
	% Active neur.	100	70	55
PLANTS	Accuracy	0.432	0.432	<b>0.472</b>
	% Active neur.	100	10	68

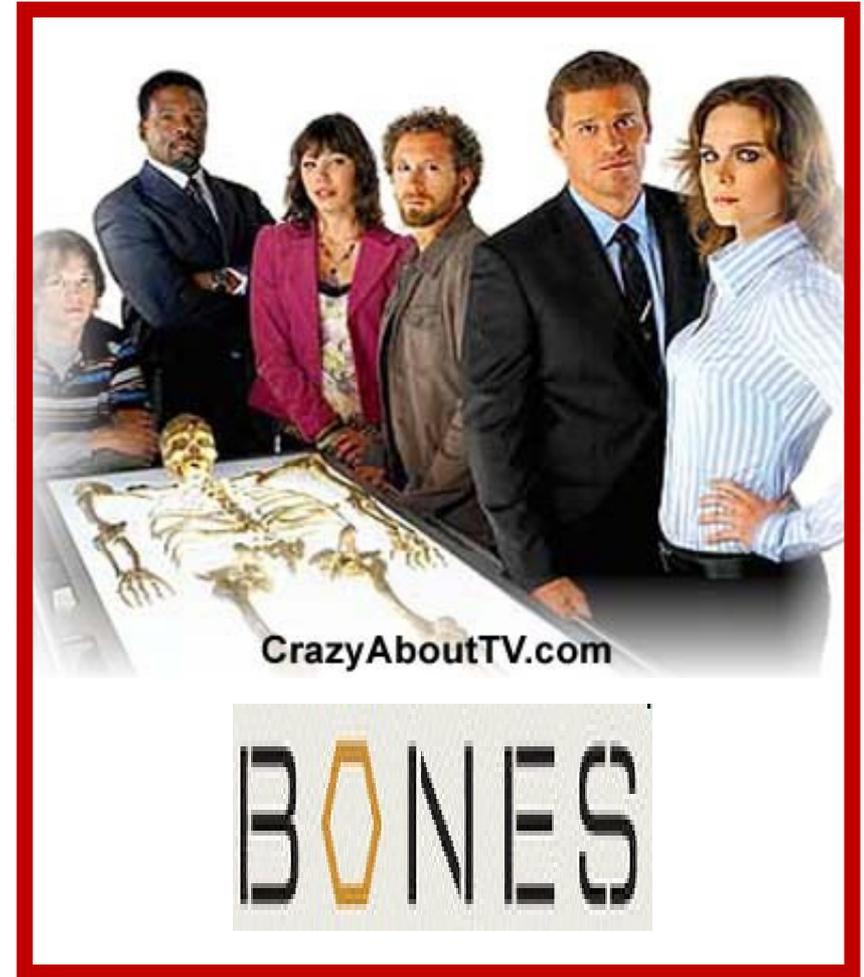
# Simplificando Redes Neuronales

---

- Aplicar un multi-objetivo para mejorar también la robustez de precisión.
- Más interpretable

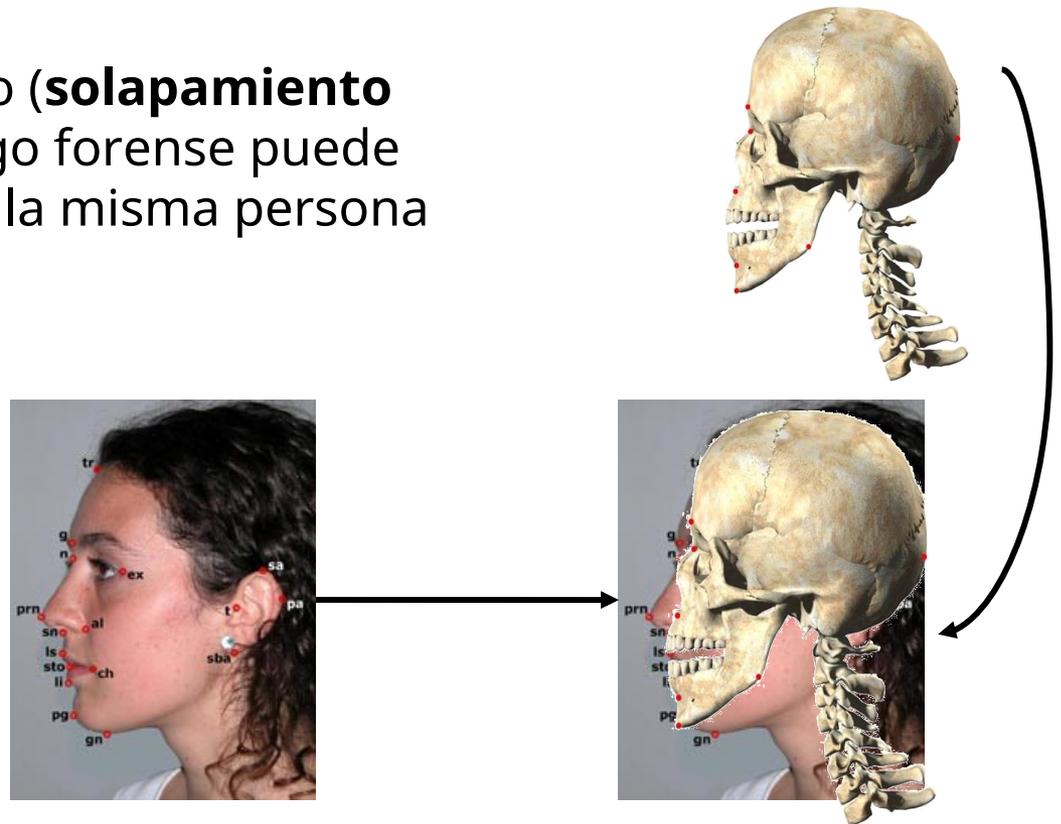


# Identificación Forense de Personas Desaparecidas

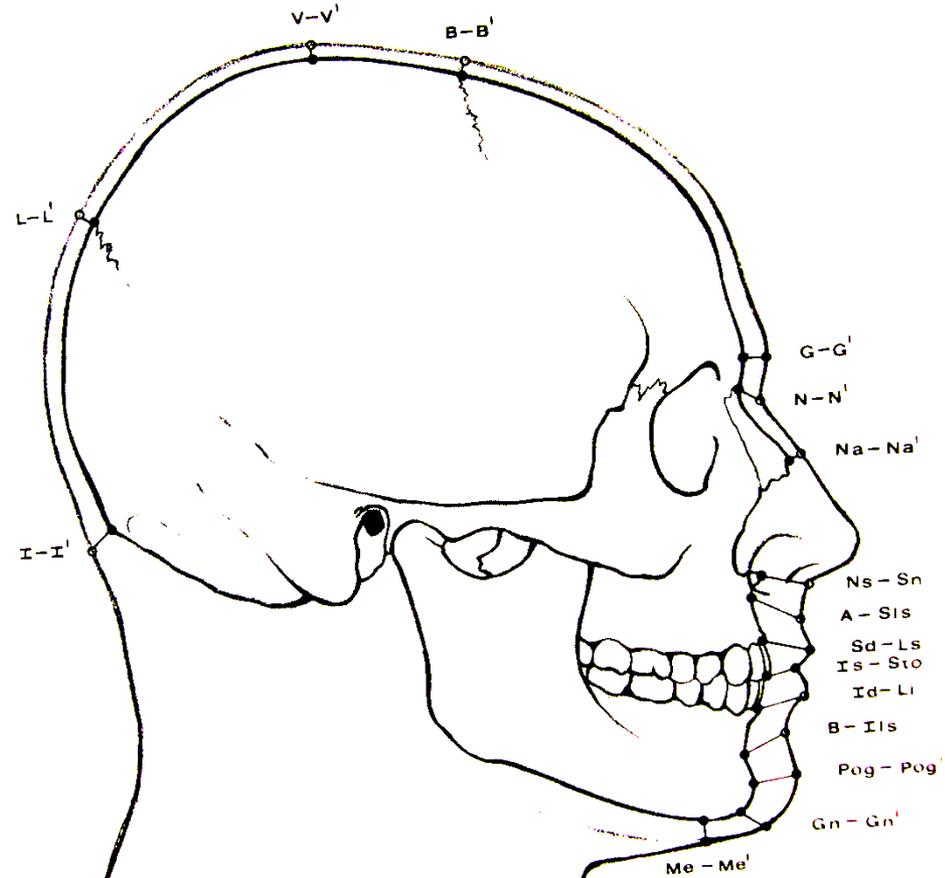
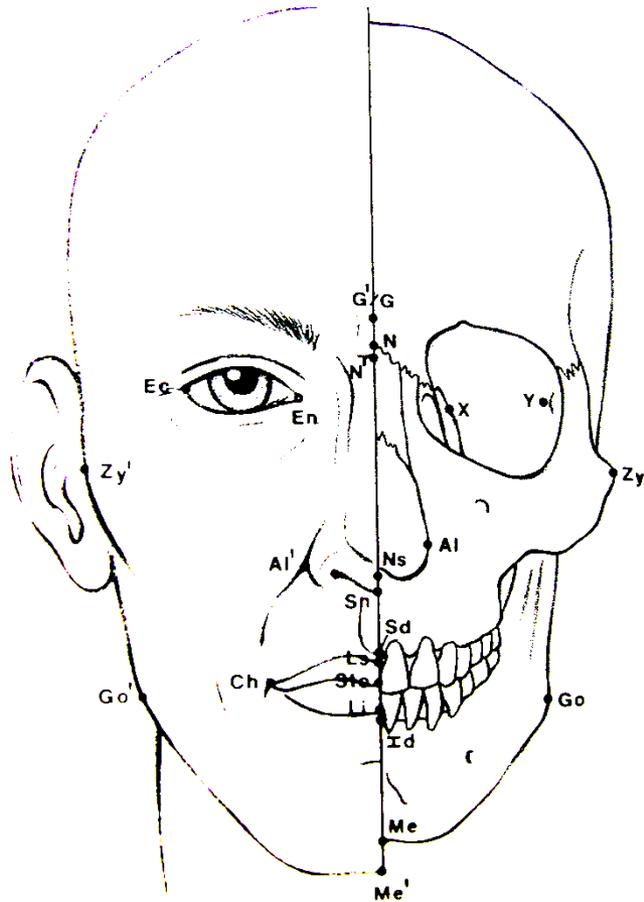


# Identificación Forense de Personas Desaparecidas

- **La superposición craneofacial** es una técnica de identificación forense basada en la comparación de un “modelo” del cráneo encontrado y una foto de una persona desaparecida
- Proyectando uno sobre otro (**solapamiento cráneo-cara**), el antropólogo forense puede determinar si pertenecen a la misma persona

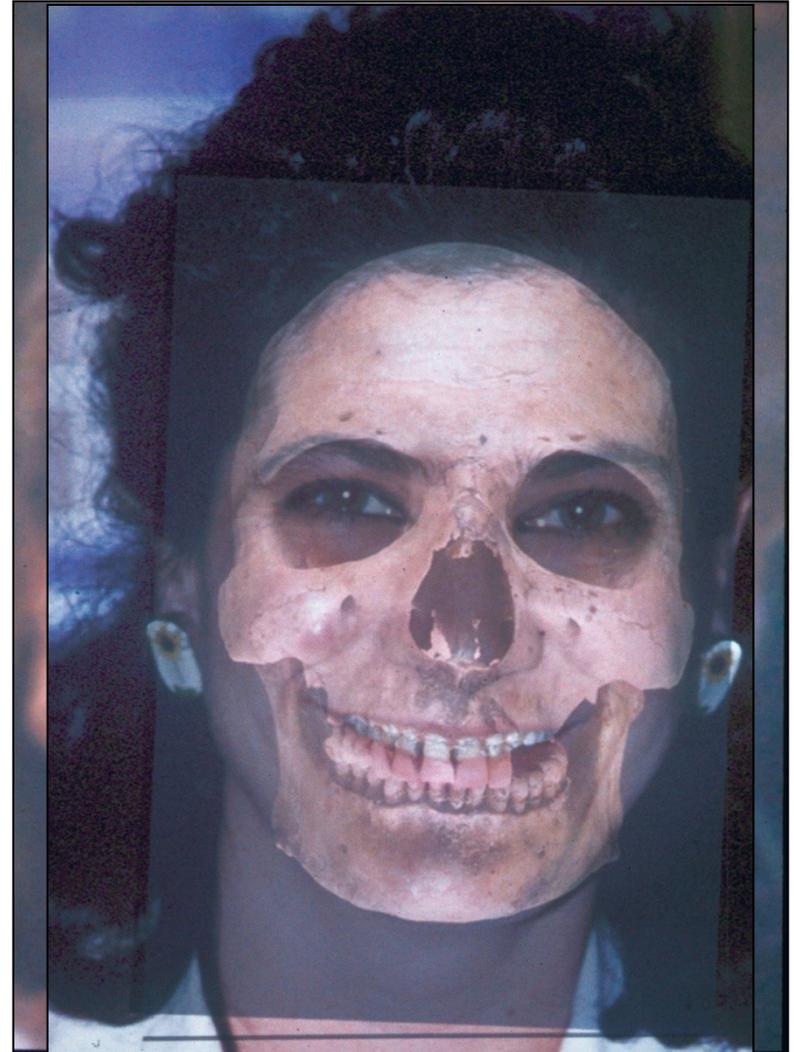
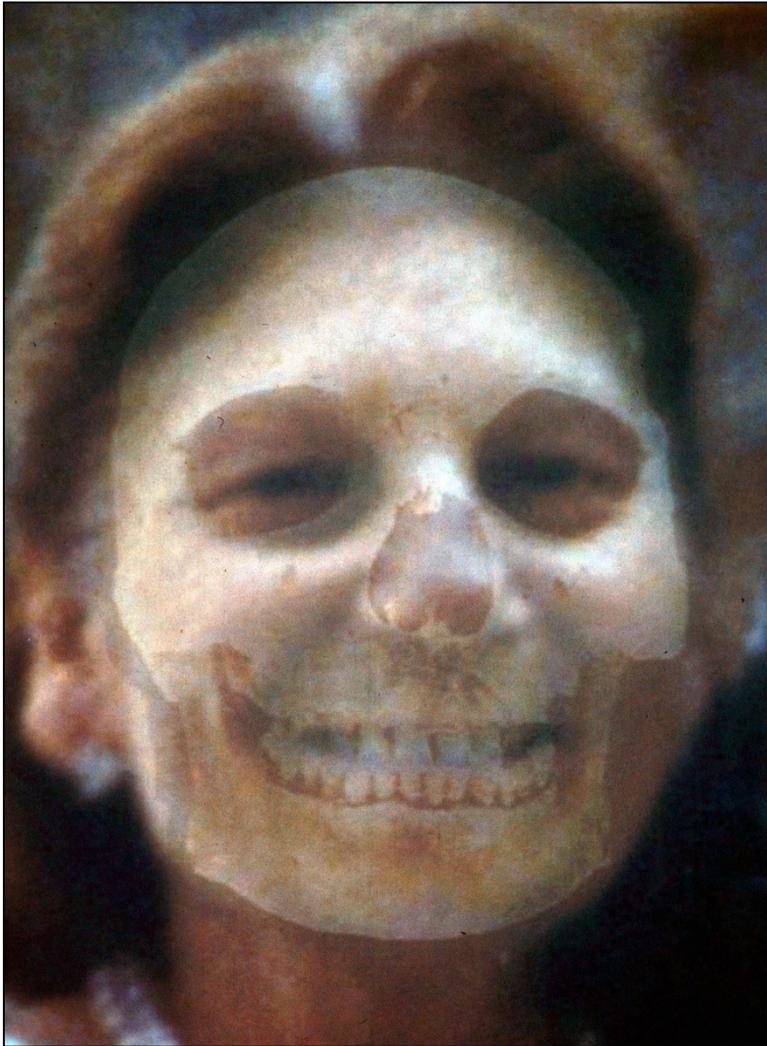


# Identificación Forense de Personas Desaparecidas



**Correlación entre los puntos craneométricos y cefalométricos**

# Identificación Forense de Personas Desaparecidas



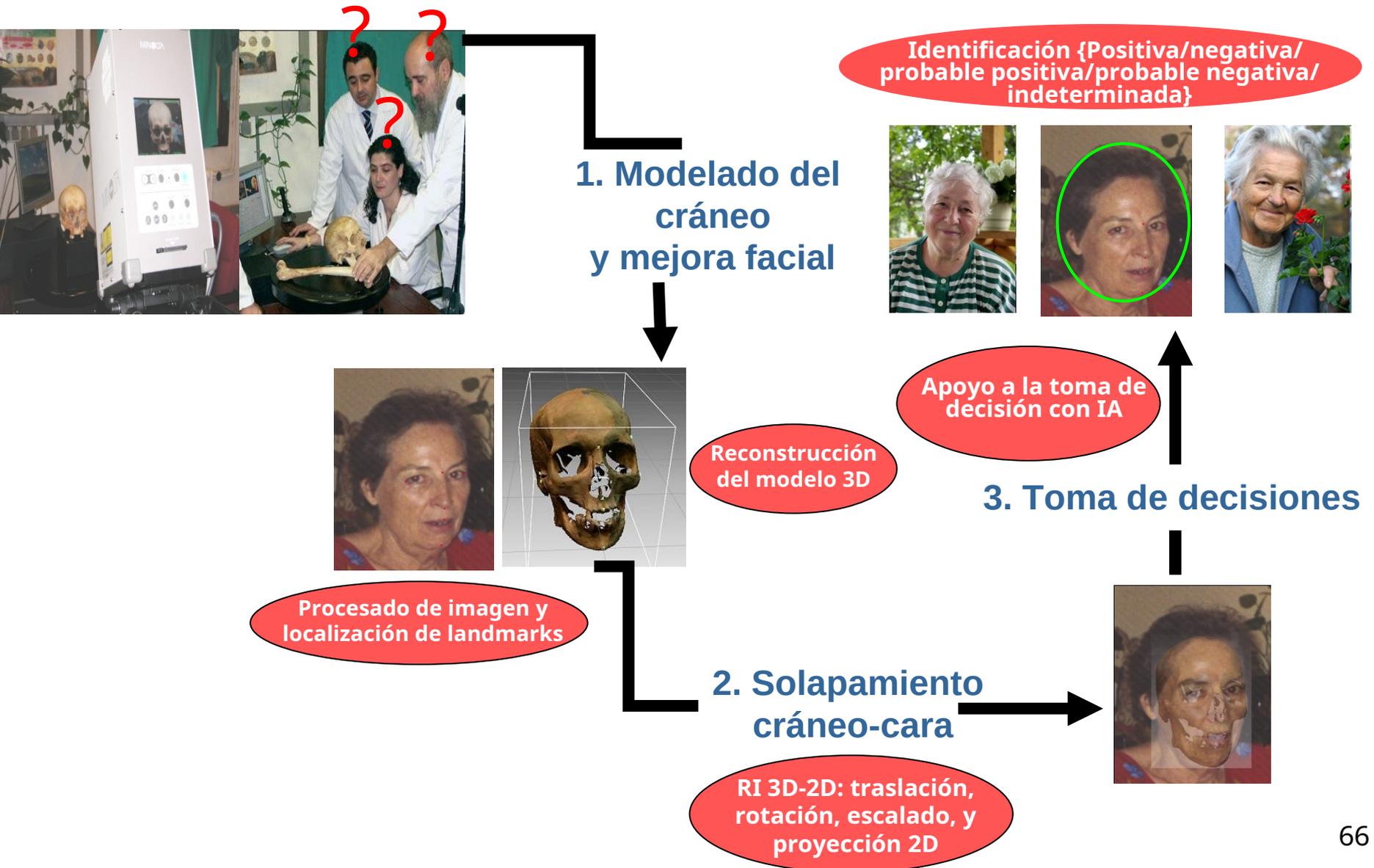
**Ejemplo real**

# Identificación Forense de Personas Desaparecidas

---

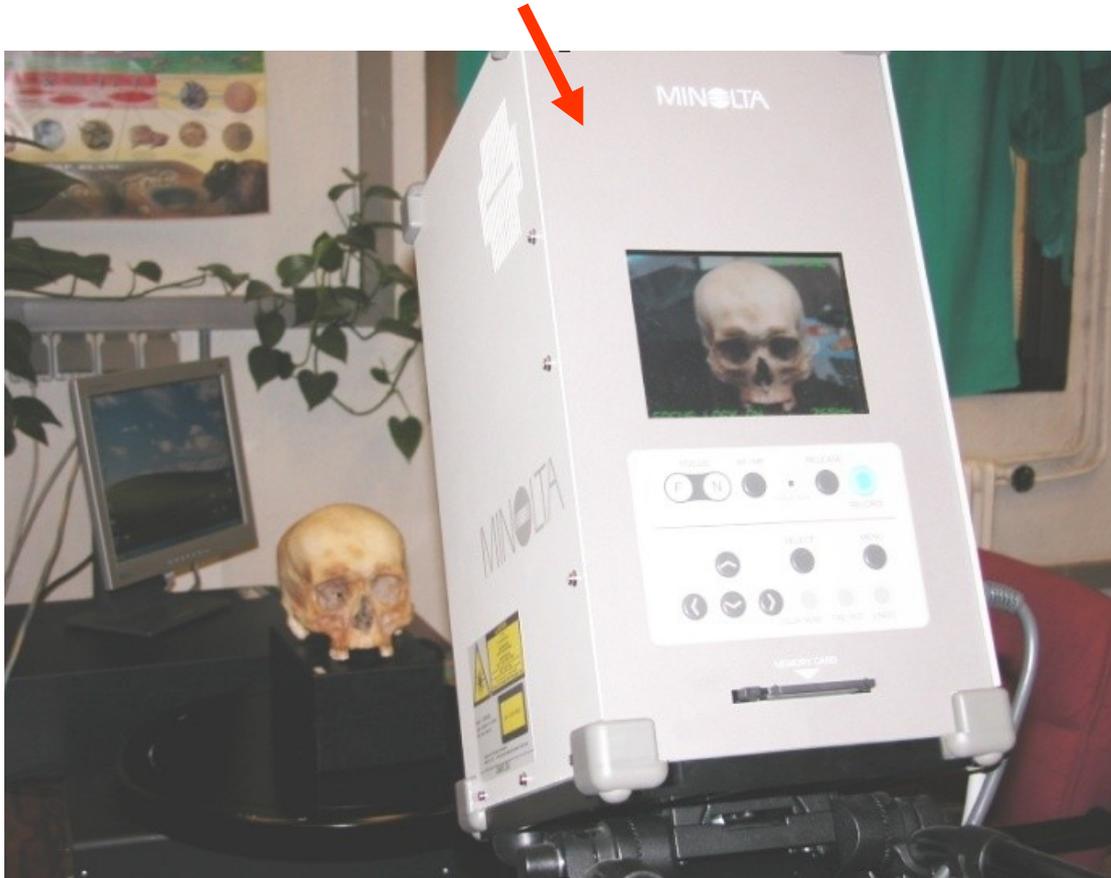
- Diseño de un **procedimiento automático basado en el ordenador** para automatizar el proceso de identificación forense por **superposición craneofacial**:
  - Diseño de métodos automáticos de registrado de imágenes de rango para obtener **modelos 3D** de cráneos (con algoritmos evolutivos, AEs)
  - Diseño de métodos de registrado 3D/2D automáticos para el **solapamiento cráneo-cara** (con AEs y conjuntos fuzzy)
  - Diseño de un **sistema de ayuda a la decisión** para asistir al antropólogo forense en la decisión final de la identificación (con visión por ordenador y operadores fuzzy)
- Proyectos Plan Nacional I+D+I (2006-09, 2009-12, 2013-15, 2016-18, 19-21) y Excelencia Junta de Andalucía (2007-10, 2013-18, 2020-22). **Patente internacional comercializada 2011. Proyecto Europeo FP7-Security MEPROCS (2012-14). Premios Internacionales**

# Identificación Forense de Personas Desaparecidas



# Identificación Forense de Personas Desaparecidas

Escáner de rango



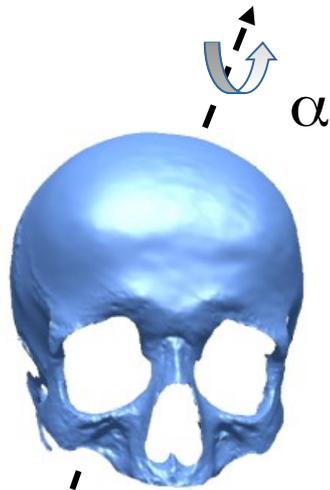
# Identificación Forense de Personas Desaparecidas

- Algoritmos Meméticos con codificación real para el modelado 3D de cráneos. Representación de una solución a este problema:

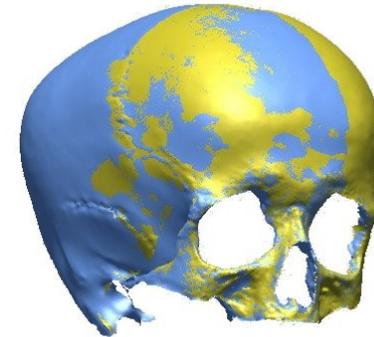


**Rotación**

**Traslación**



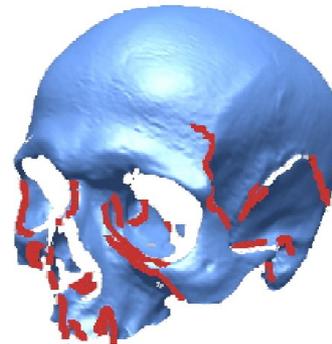
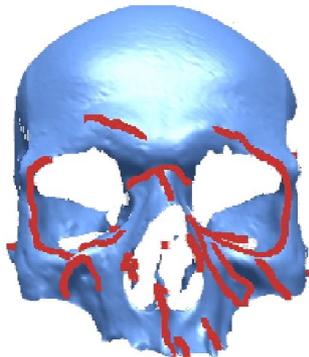
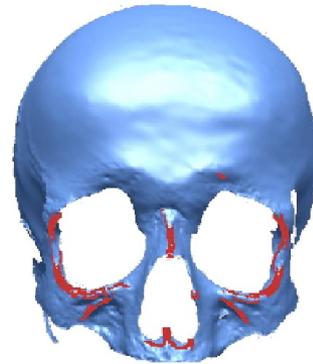
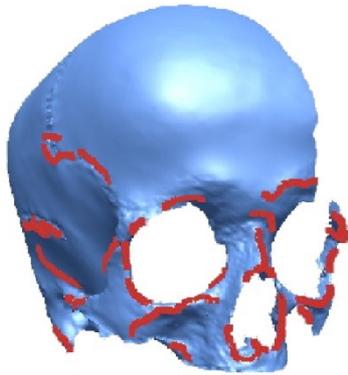
Eje ( $Eje_x, Eje_y, Eje_z$ )



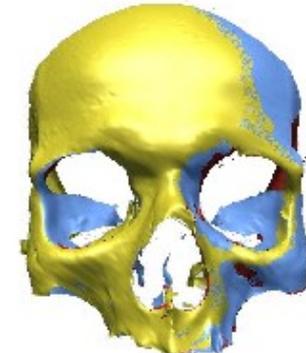
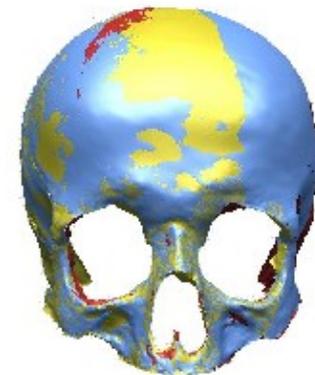
**Reconstrucción  
parcial**

# Identificación Forense de Personas Desaparecidas

***Entrada: vistas 3D***

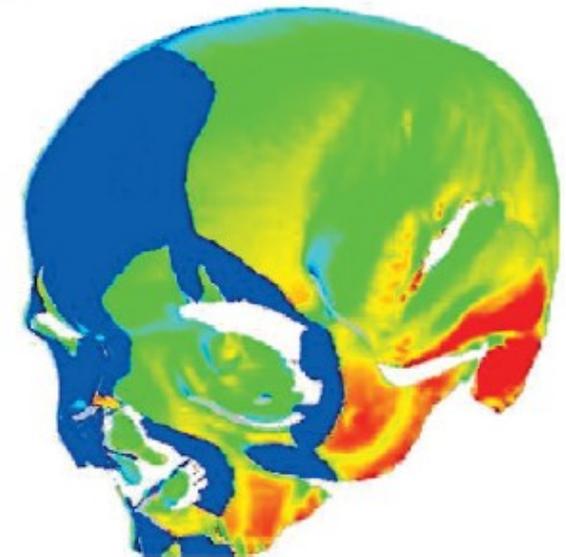
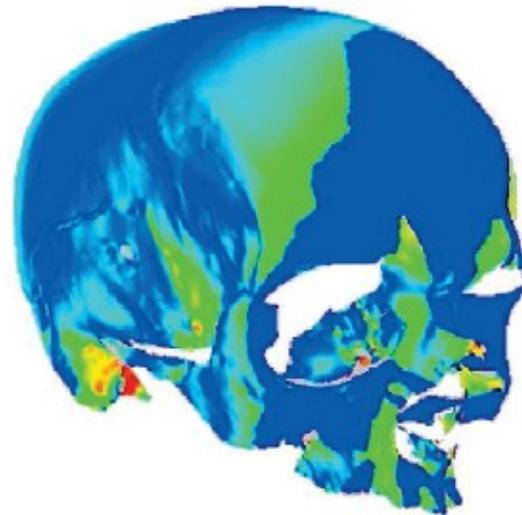
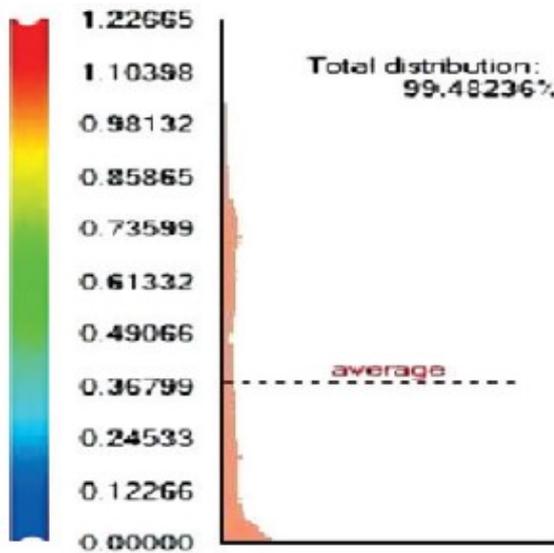


***Reconstrucción***



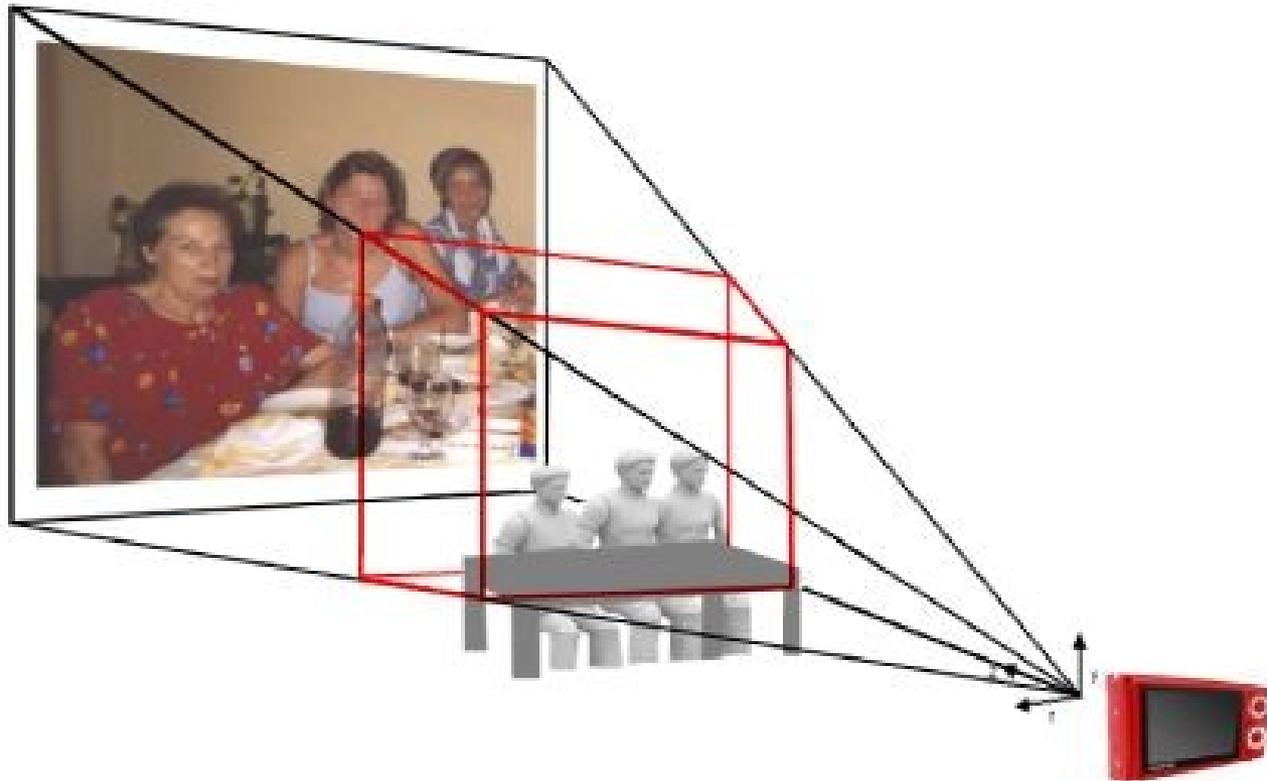
# Identificación Forense de Personas Desaparecidas

- Error global del modelo 3D: menor de 1 milímetro
- Tiempo de reconstrucción 3D: 2 minutos
- Robustez del método: baja desviación típica en 30 ejecuciones distintas



# Identificación Forense de Personas Desaparecidas

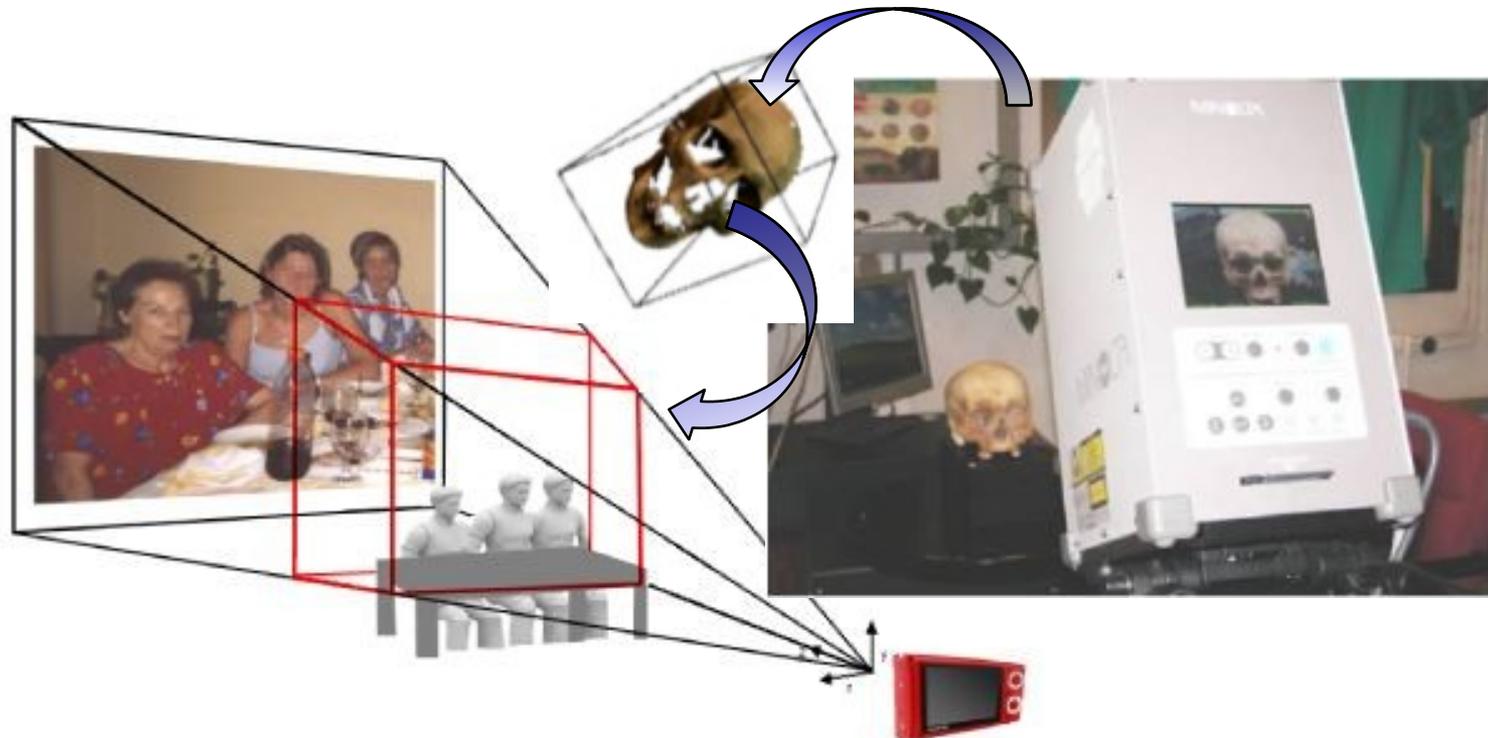
## 1. Adquisición de la fotografía



Solapamiento automático cráneo-cara mediante  
Registrado de Imágenes (RI) Evolutivo

# Identificación Forense de Personas Desaparecidas

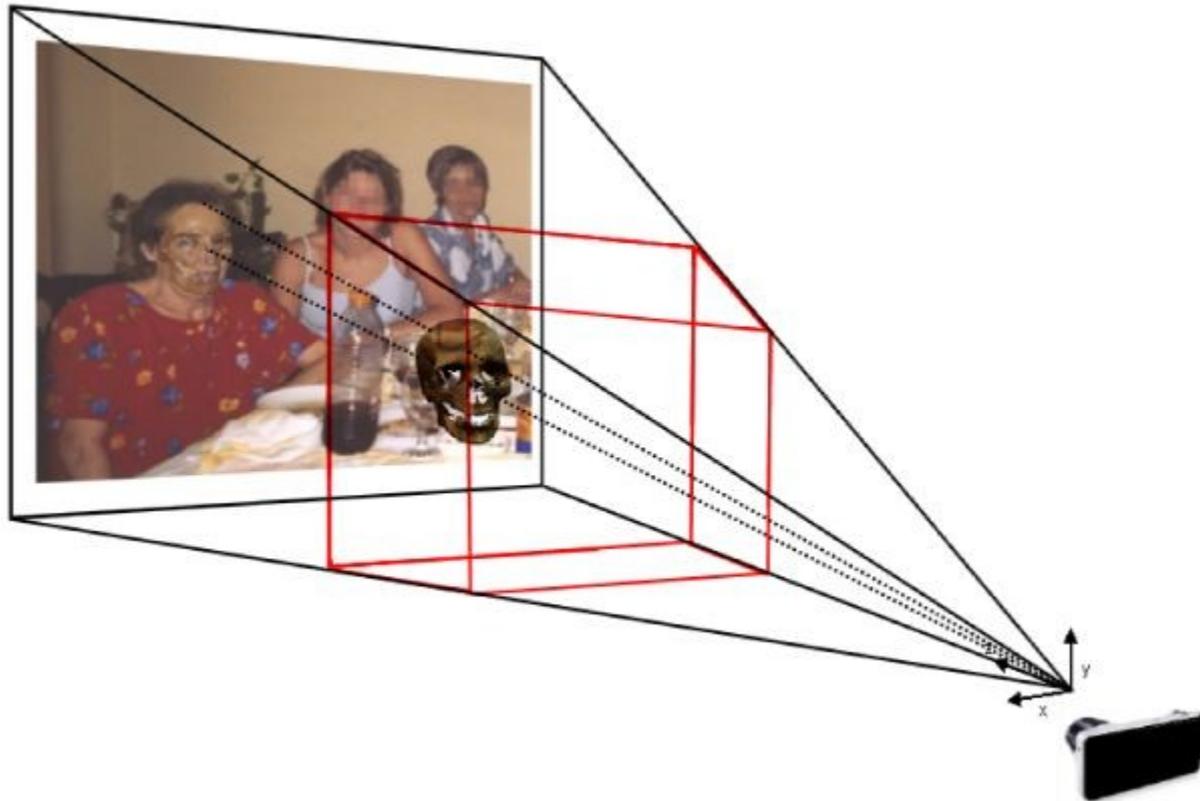
## 2. Adquisición del modelo 3D del cráneo



Solución final: **escenario original replicado mediante RI 3D-2D**

# Identificación Forense de Personas Desaparecidas

## 3. Solapamiento final cráneo-cara



Solución final: **escenario original replicado mediante RI 3D-2D**

# Identificación Forense de Personas Desaparecidas

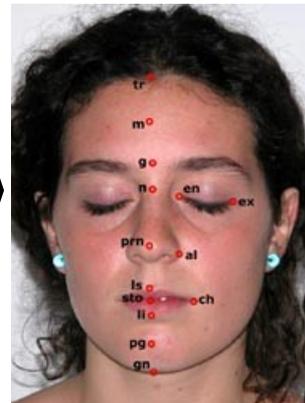
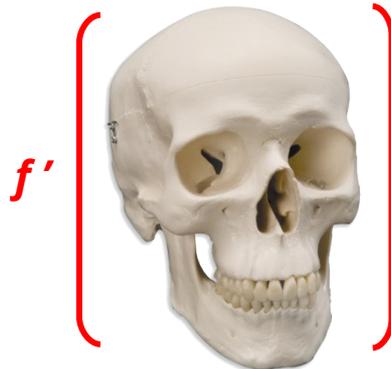
Búsqueda de la mejor superposición 3D-2D  
(Algoritmo Evolutivo con Codificación Real)

Error de Registrado

↓  $f' \cong f^*$  ← Evaluación  $f'$

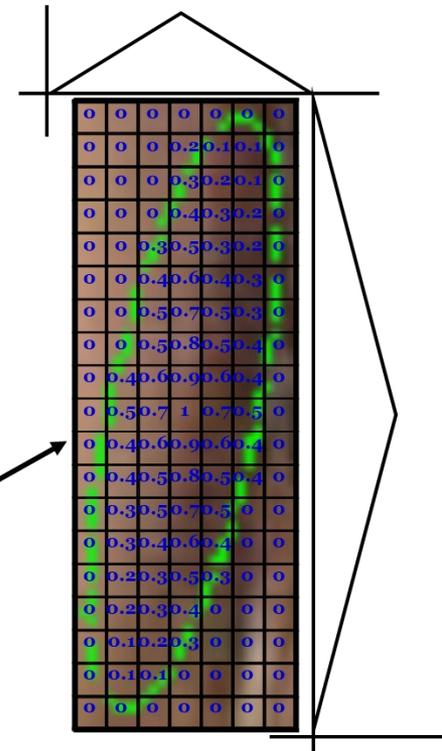
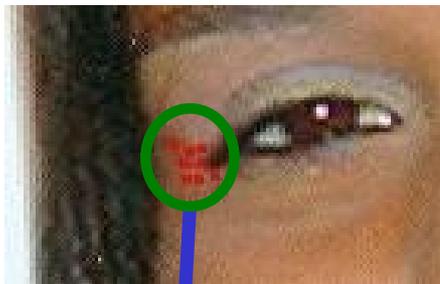
Rotación =  $\{60^\circ, (0, 1, 0)\}$   
Traslación =  $\{2, 0, 1\}$ ...

Medir la distancia  
entre cada par de  
puntos de referencia



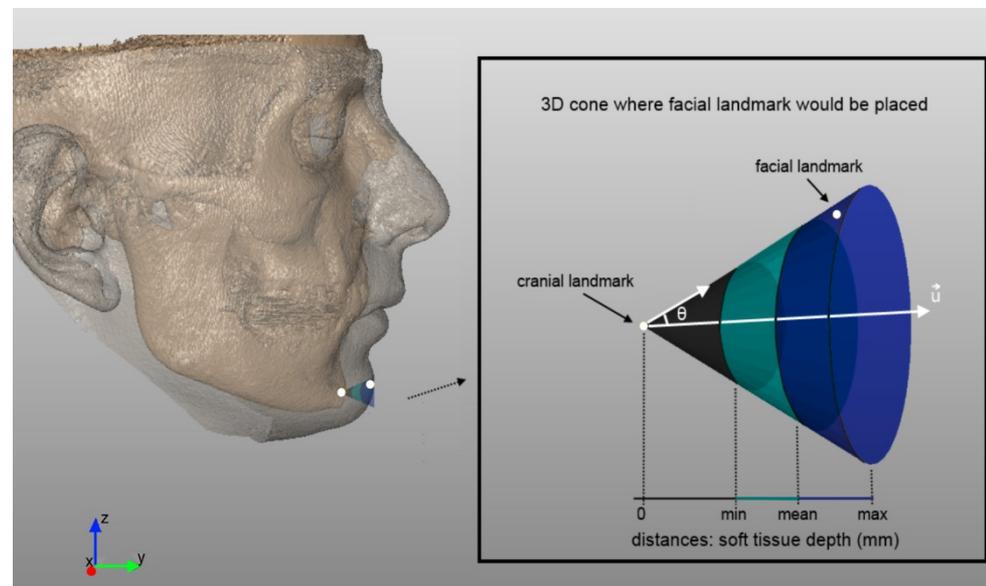
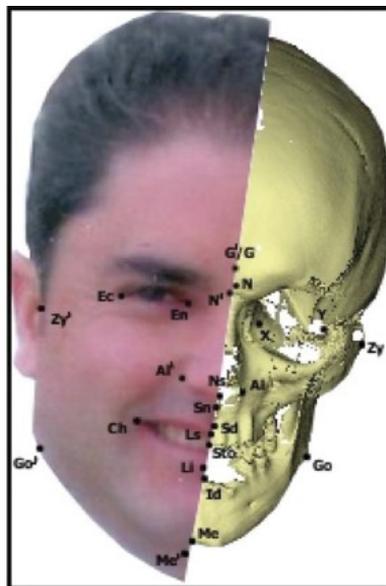
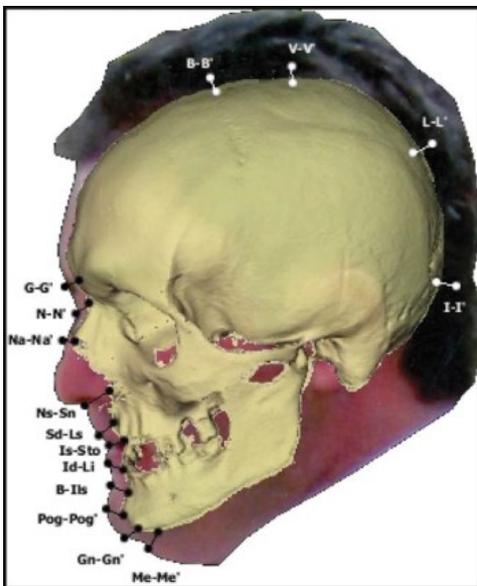
# Identificación Forense de Personas Desaparecidas

- **Landmarks imprecisos.** Cada punto cefalométrico es un área (elíptica o circular). Mayor incertidumbre asociada al landmark → mayor área. **Posibilita al forense marcar áreas cuando no se vean bien los puntos.** Modelado usando conjuntos fuzzy



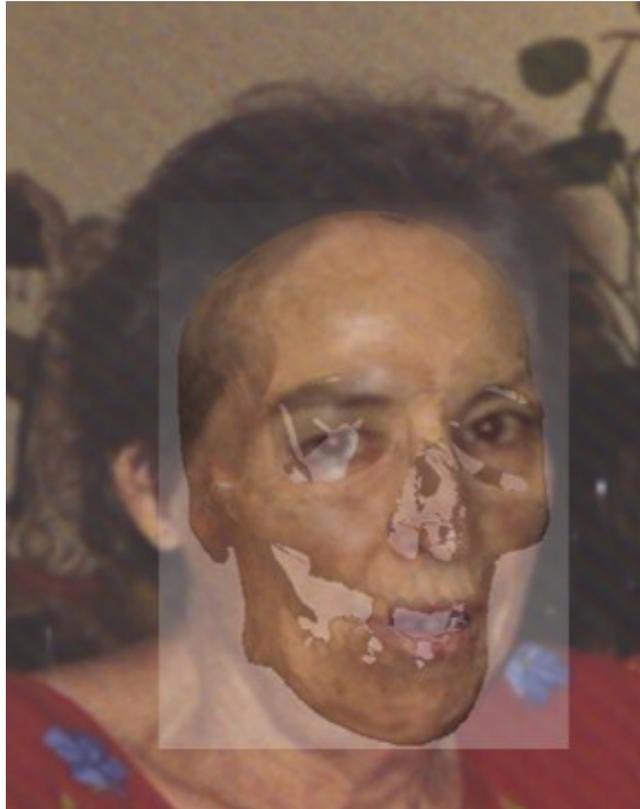
# Identificación Forense de Personas Desaparecidas

- **Distancias entre landmarks cráneo-cara.** Los puntos no casan perfectamente debido al grosor de los tejidos blandos. Las distancias varían para cada landmark en función de grupo racial, edad, género, ...
- **Nuestro método incorpora información de esas distancias para obtener mejores superposiciones cráneo-cara**



# Identificación Forense de Personas Desaparecidas

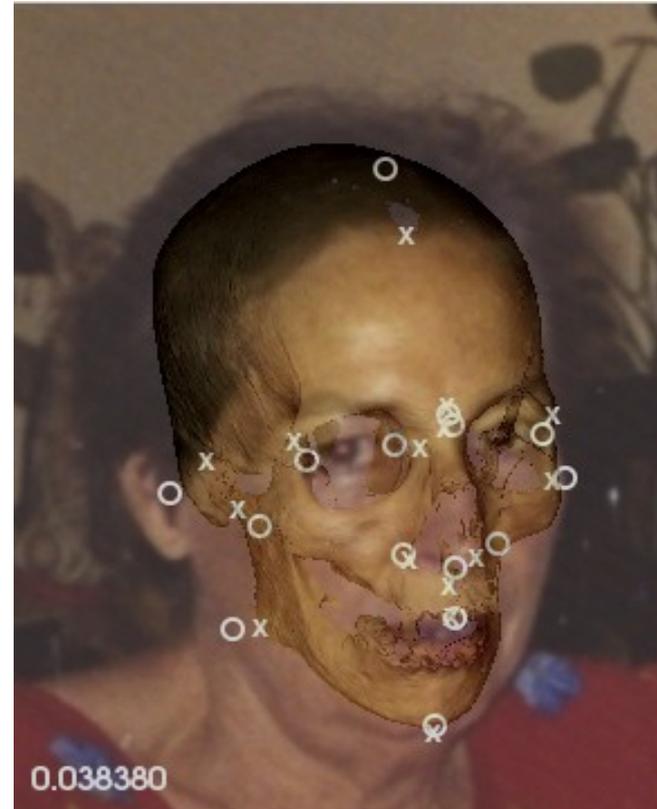
Manual



Area deviation error: 34.70%

varias horas

Fuzzy AE



Area deviation error: 13.23%

2-4 minutos

# Spin-off Panacea Cooperative

- Se ha llegado a crear una spin-off, Panacea Cooperative Research.
- Producto comercial: [Skeleton-Id](#)

<https://www.youtube.com/watch?v=4p8Ufws80Js>



# 4. Software de Metaheurísticas

---

- Existen distintos tipos de software de metaheurísticas.
- No muy maduro.
- ✓ Código separado de autores de algoritmos.
- ✓ Distintos *frameworks*, no compatibles.
- ✓ Orientados a desarrollo de algoritmos, no usuario final.
- ✓ Distintos entornos y lenguajes: Matlab (mayoritario), Java, Python, ...

# 4. Software de Metaheurísticas

---

## Metaheuristic optimization frameworks: a survey and benchmarking

José Antonio Parejo · Antonio Ruiz-Cortés ·  
Sebastián Lozano · Pablo Fernandez

Soft Comput (2012) 16:527–561  
DOI 10.1007/s00500-011-0754-8

**Abstract** This paper performs an unprecedented comparative study of Metaheuristic optimization frameworks. As criteria for comparison a set of 271 features grouped in 30 characteristics and 6 areas has been selected. These features include the different metaheuristic techniques covered, mechanisms for solution encoding, constraint handling, neighborhood specification, hybridization, parallel and distributed computation, software engineering best practices, documentation and user interface, etc. A metric has been defined for each feature so that the scores obtained by a framework are averaged within each group of features, leading to a final average score for each frame-

work. Out of 33 frameworks ten have been selected from the literature using well-defined filtering criteria, and the results of the comparison are analyzed with the aim of identifying improvement areas and gaps in specific frameworks and the whole set. Generally speaking, a significant lack of support has been found for hyper-heuristics, and parallel and distributed computing capabilities. It is also desirable to have a wider implementation of some Software Engineering best practices. Finally, a wider support for some metaheuristics and hybridization capabilities is needed.

# 4. Librerías útiles

---

- Para muchos algoritmos es muy cómodo trabajar con operaciones de vectores.
- Existen librerías para distintos lenguajes que pueden ser útiles.
- Ejemplo para distintos lenguajes (hay muchos más):
  - Java: [Jblas](#)
  - C++: [Eigen](#), [Armadillo](#)
  - Python: [Numpy](#)
  - Rust: [ndarray](#)
  - Julia, Matlab, R: en el propio lenguaje

# 4. Software de Metaheurísticas

---

- Para desarrollar las prácticas de la asignatura se podrá emplear el entorno/lenguaje que se desee.
- El profesor de prácticas proporcionará distintos códigos básicos de metaheurísticas desarrollados en C.
- El alumno deberá indicar el software considerado en su documentación de prácticas y proporcionar las fuentes y los ejecutables realizados

# 4. Software de Metaheurísticas

---

- Para desarrollar las prácticas de la asignatura se podrá emplear el entorno/lenguaje que se desee.
- ¿Puedo elegir?
  - Sí, no es una asignatura centrada en desarrollo, interesa conocer algoritmos para implementarlos.
  - Lenguaje bajo nivel: (C++,Java, Rust) adecuados pero mejor con librerías.
  - Python: Cuidado si no se conoce cómo optimizar la función de *fitness* (uso de numpy/numba).
  - Matlab: no recomendada, software libre compatible (*Octave*) lento y errores.
  - Julia: Falta de conocimiento/experiencia.

# 4. Software Destacable

---

- **jMetal (Java) ó jMetalPy (Python)**

- Algoritmos Multi-objetivo.
- Fácil de usar, problemas propios.
- Se programa, permite gráficas.



- **PlatEMO**

- Muchos algoritmos.
- Implementado en Matlab.
- Bien documentado.



# 4. Software Destacable

---

## ■ ECJ, Java

- Configurable con ficheros de texto.
- Sólo requiere programar la función objetivo.

## ■ Jenetics, Java

- Sólo de algoritmos genéticos.
- Bien documentado.

## ■ Optaplanner, Java

- Planificación y Enrutado.
- Ámbito concreto.
- Bien documentado.

OptaPlanner 

# 4. Software Destacable

---

## ■ Inspyred, Python

- Muchos algoritmos.
- Fácil de usar.

## ■ Pyswarms, Python

- Limitado a PSO, muy completo.
- Visualizaciones.

## ■ Mealpy, Python

- El más extenso.
- Implementación deja que desear.
- No validado autores.

# 4. Software Destacable

---

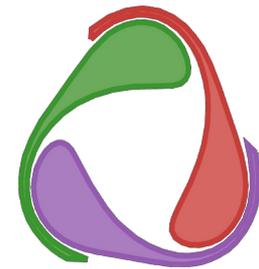
## ■ JuMP, Julia

- Optimización.
- Lenguaje de Modelado.



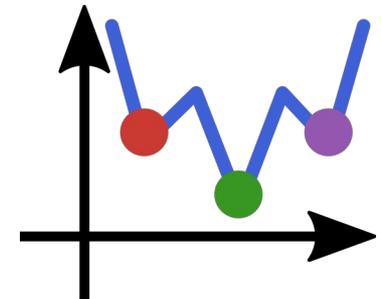
## ■ Optimization, Julia

- Muchos algoritmos.
- De distinto tipo, evolutivos y exactos.



## ■ Metaheuristics, Julia

- Distintas metaheurísticas.
- Visualizaciones.



## ■ Evolutionary, Julia

- Limitado a algoritmos genéticos.
- Visualizaciones.

# 4. Software Destacable

---

## ■ ParaDisEO, C++

- Paralelismo.
- Complejo de integrar un problema.
- Gráficas.



## ■ OptFrame, C++

- Completo.
- Complejo de usar.

## ■ Pagmo, C++

- Completo.
- Sencillo, mejorable documentación.
- Interfaz en Python.