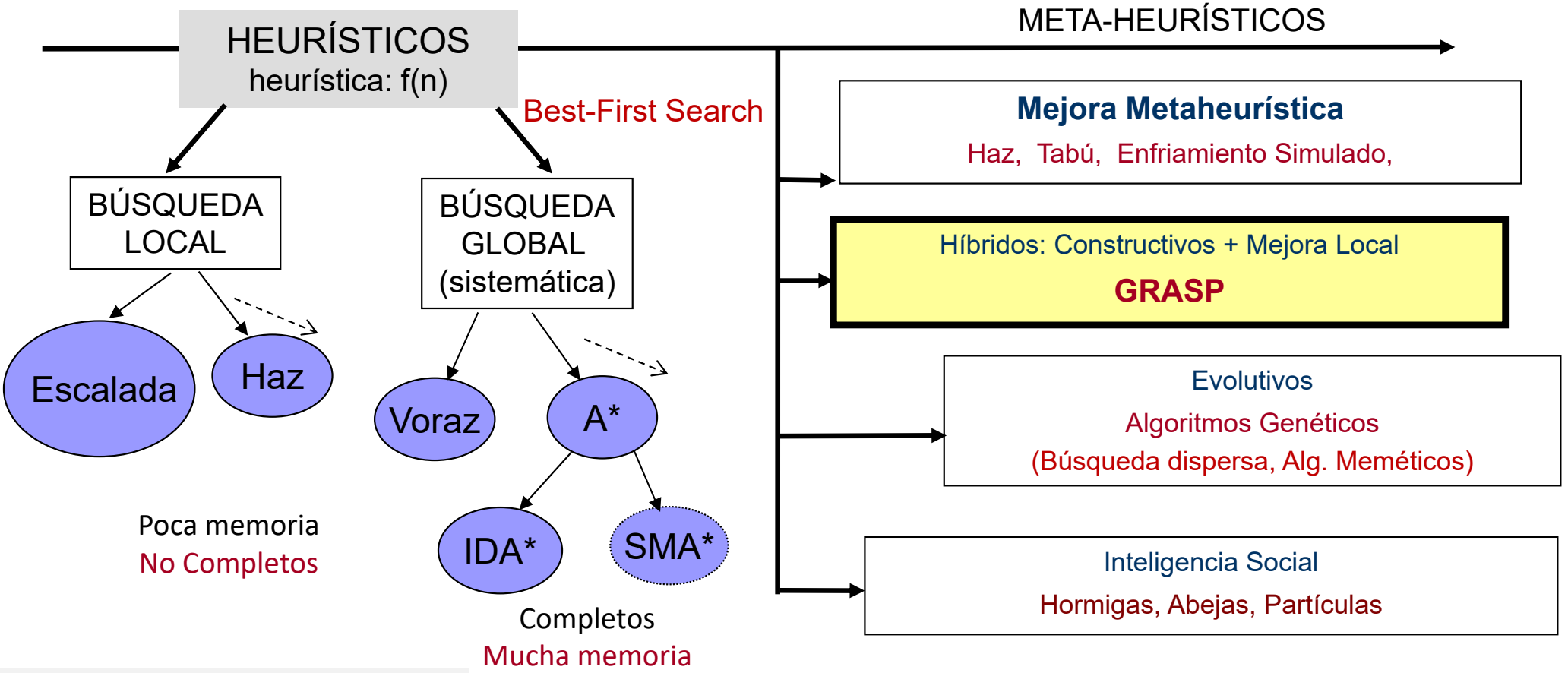
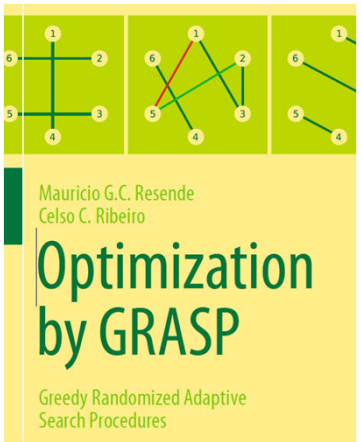
3.- Metaheurísticas Híbridas: Constructivos + Mejora Local



POLIFORMAT

- **GRASP: Greedy Randomized Adaptive Search Procedures.** M. Resende, J.L. González. Inteligencia Artificial, Inteligencia Artificial. No.19
- **Optimization by GRASP**. M. Resende, C. Ribeiro. Springer (2016).
- **Machine Learning into Metaheuristics.** E.Talbi ACM Computing Surveys, 2021
- **Greedy Randomized Adaptive Search Procedures.** T.A. Feo, M. Resende. J. of Global Optimization 6 (1995)



Búsqueda Heurística en IA: Constructiva

a) Búsqueda Global/Sistemática (típica)#

Búsqueda Voraz (Greedy).

- Caso típico de "búsqueda el primero mejor", donde no importa g(n):

$$f(n) = h(n)$$

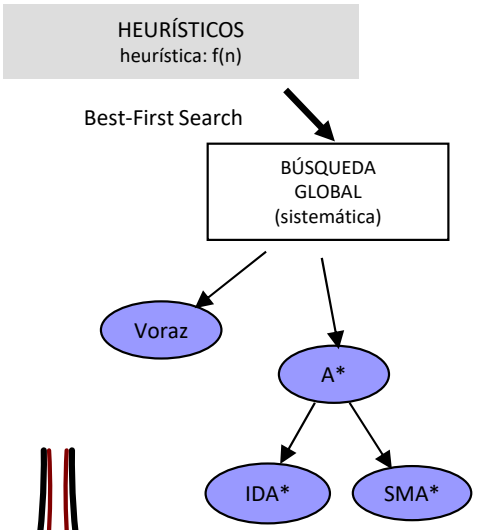
- No siempre encuentran solución o acaban, no completos, no admisibles#

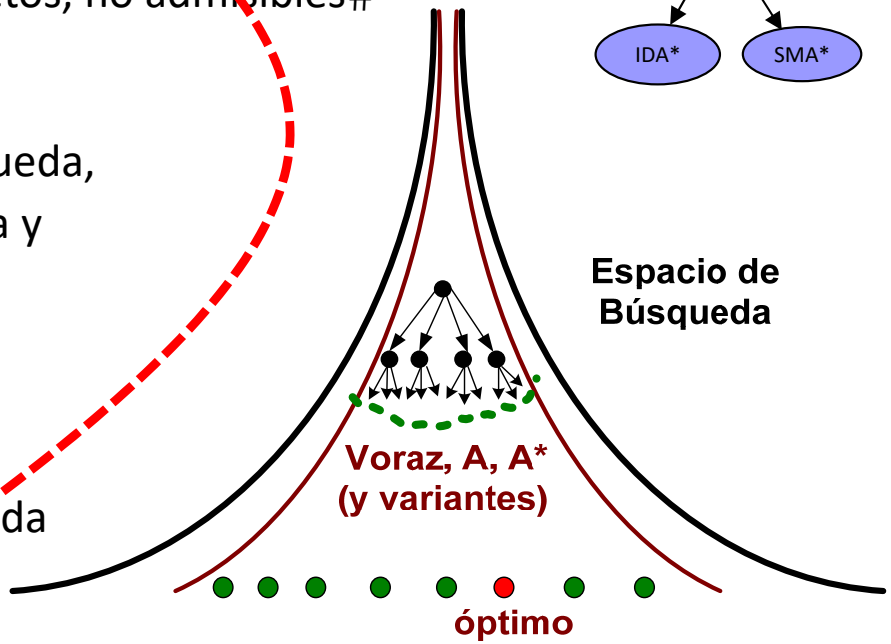
Algoritmo A.

- Combinación de *búsqueda voraz* (reduce coste de búsqueda, pero no óptima ni completa) y *coste uniforme* (completa y óptima, pero ineficiente).

$$f(n)=g(n)+h(n)$$

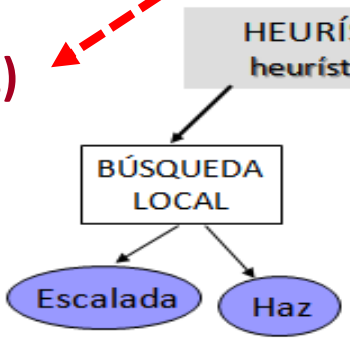
- Algoritmo A*: $\forall n, h(n) \leq h^*(n)$ Completa y admisible.
- Pero requiere mucha memoria (un algoritmo que expanda menos nodos que A* no tendrá admisibilidad).

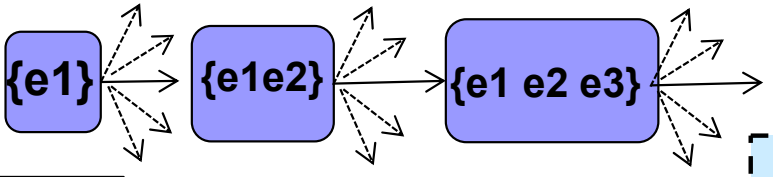




b) Busqueda Local (Escalada, Haz)

- Ciclos, Óptimos Locales, etc.





Construcción Solución Optimizada
(Pasos iterativos: +Elementos Solución)

Nuevos elementos se añaden iterativamente a la solución.

Métodos Constructivos:
Algoritmo A, A* ⇒ GRASP

Mejora de una Solución
Requiere Solución Inicial
(Pasos iterativos: Mejora Solución)
Búsqueda en un espacio de Soluciones

Mejora Iterativa de UNA SOLUCIÓN → Enfriamiento Simulado, Búsqueda Tabú

Mejora Iterativa CONJUNTO SOLUCIONES (no colaborativa) → Búsqueda en Haz

Mejora Conjunto Soluciones
(Pasos: Evoluciones Cooperativas)

Control Centralizado:
Evoluciones poblacionales
(Pobl. Individuos ~ Soluciones) → Alg. Genéticos, Alg. Meméticos, Búsqueda Dispersa

Generación & mejora

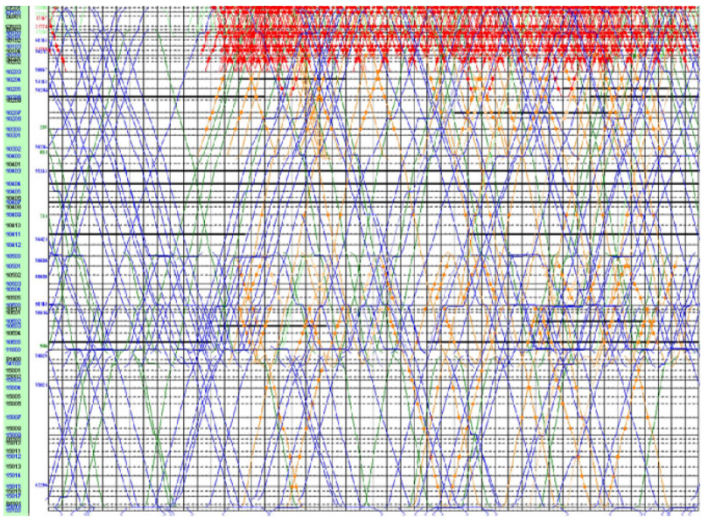
Control Autónomo:
Inteligencia de Enjambre
Enjambre de Individuos (Soluciones) → Alg. Hormigas, Colonia de Abejas, Enjambre de Partículas…

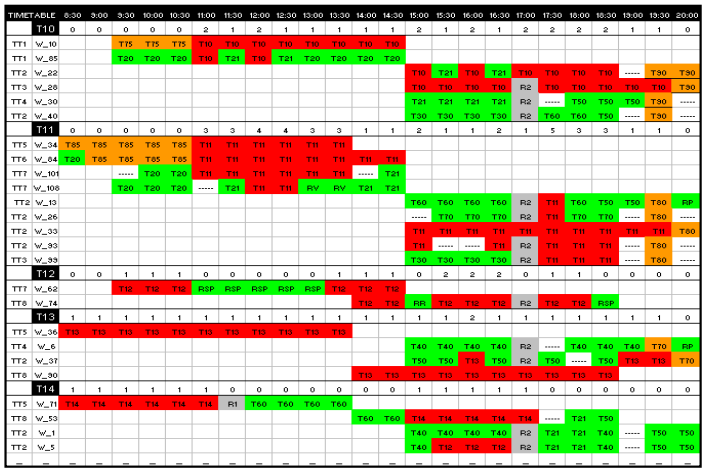
Métodos Constructivos:

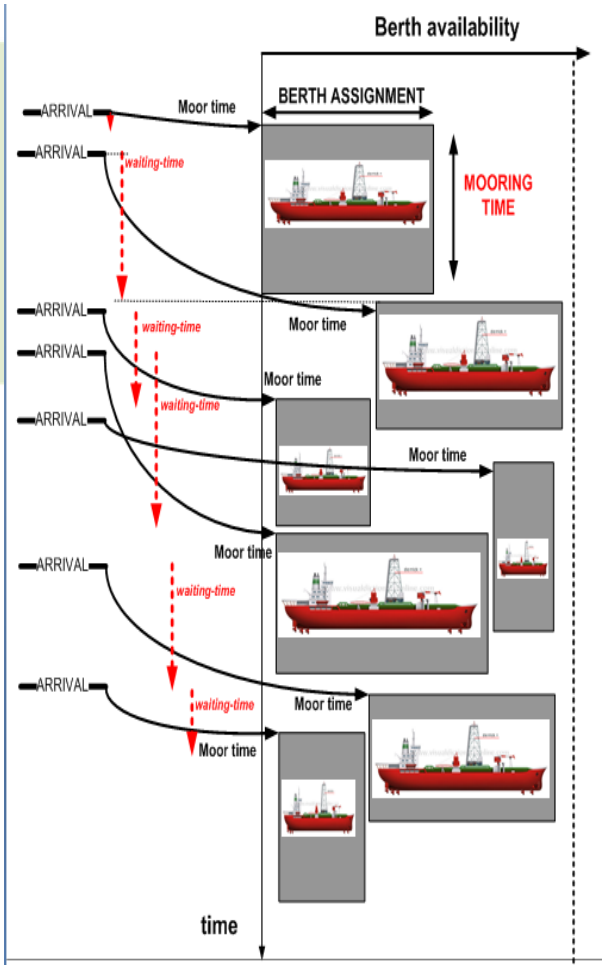
Búsqueda de la solución en un espacio de estados: nuevos elementos *se añaden iterativamente a la solución*.

Apropiados para:

- Problemas en los que es difícil obtener soluciones iniciales (factibilidad)
- Vecindario muy amplio o poco estructurado (metaheurísticas de mejora)
- Complejidad de la representación de la solución (en metaheurísticas poblacionales)







Greedy Randomized Adaptive Search (GRASP)

Feo & Resende, 1989

"A probabilistic heuristic for a computationally difficult set covering problem"

Op. Res. Letters,8:67–71

Metaheurística de multi-inicio (***multi-start***) o iterativo, donde cada **iteración** tiene dos fases:

(i) Fase Constructiva: se encuentra una solución factible, mediante una heurística constructiva.

- *No necesita una solución inicial,*
- *La longitud del camino (solución) debe ser limitada.*
- *Los elementos a añadir a la solución en cada paso son limitados (baja ramificación).*

(ii) Fase de Mejora con Búsqueda local:

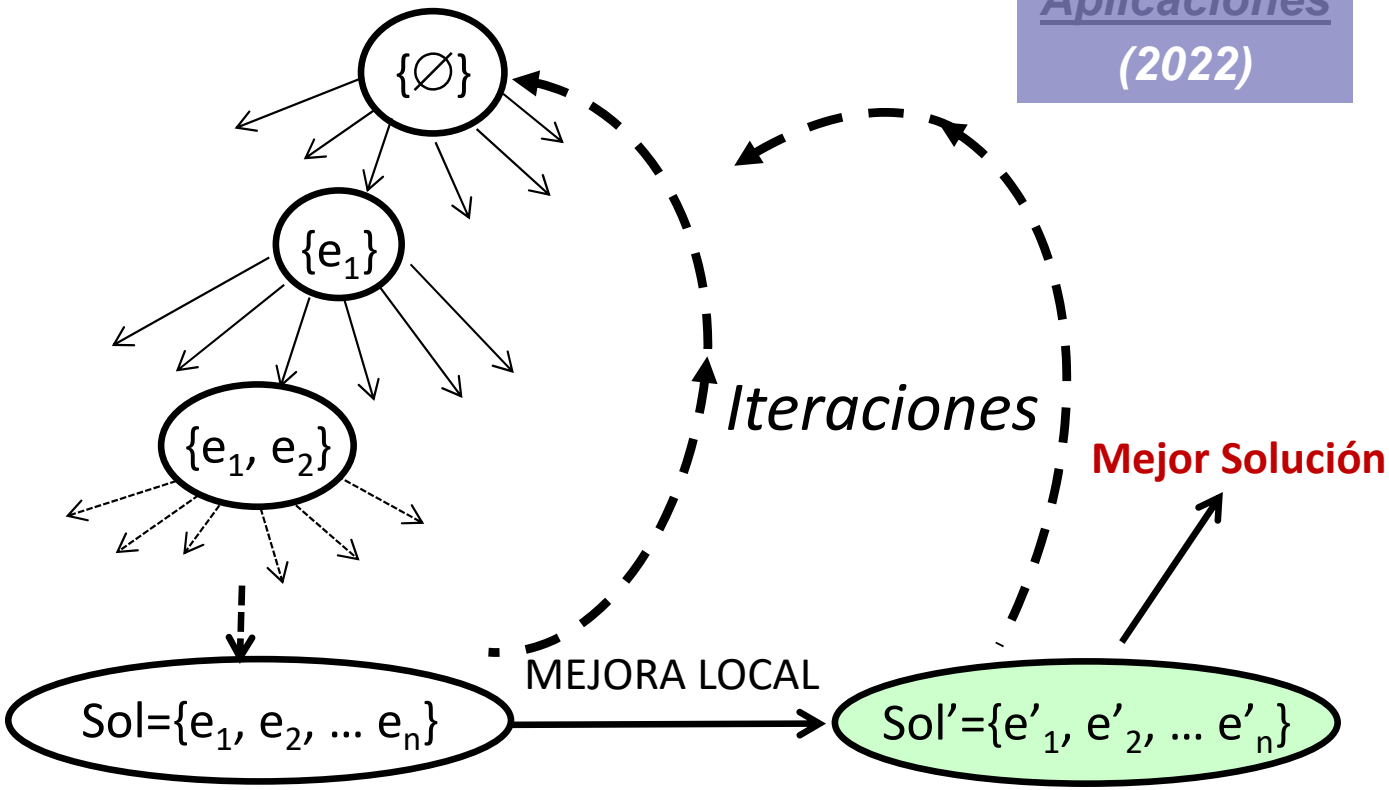
- La vecindad de la solución se explora para mejorarla, típicamente, mediante escalada: mejor candidato *(best-improvement)* o primer candidato *(first-improvement)*.

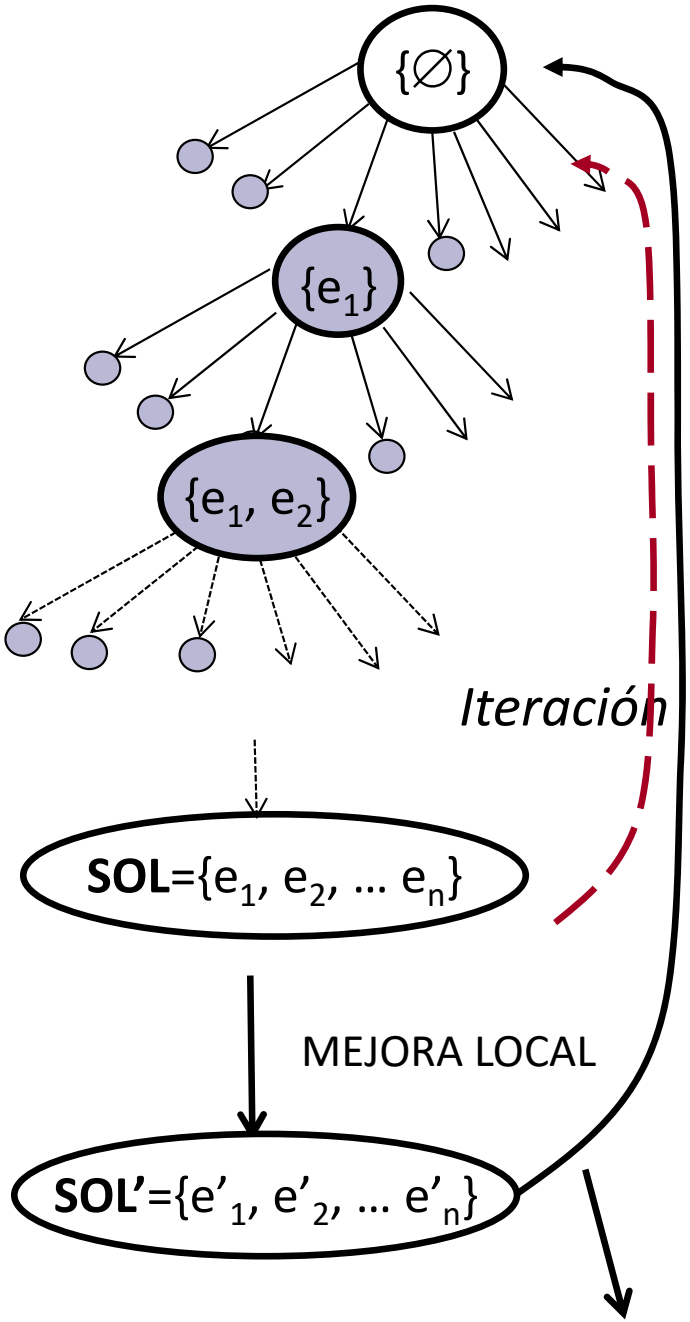
Al final de todas las iteraciones, se guarda la mejor solución

Aplicaciones típicas:

Viajante, Recubrimiento, Scheduling, Diseño, Path-finding, Asignaciones, etc.

Aplicaciones (2022)





Procedimiento GRASP $SOL_{MEJOR}=\emptyset$, α, *fcoste*

Mientras (no condición de parada, o k iteraciones)

a) Fase Constructiva: $SOL=\emptyset$

Greedy Randomized Construction (Semilla)

Iterativamente:

- Formar lista de elementos candidatos "C". Evaluar ∇coste.
- Formar (α) lista restringida de los mejores candidatos (RCL).
- Seleccionar un elemento aleatoriamente de la lista restringida.

hasta encontrar solución (SOL).

b) Fase de Mejora: $SOL' = busqueda\ local\ (SOL)$

Proceso de búsqueda local a partir de la solución construida (SOL).

c) Actualización: $SOL_{MEJOR} = Actualiza\ (SOL', SOL_{MEJOR})$

Si la solución obtenida mejora a la mejor almacenada, actualizarla.

return (SOL_{MEJOR});

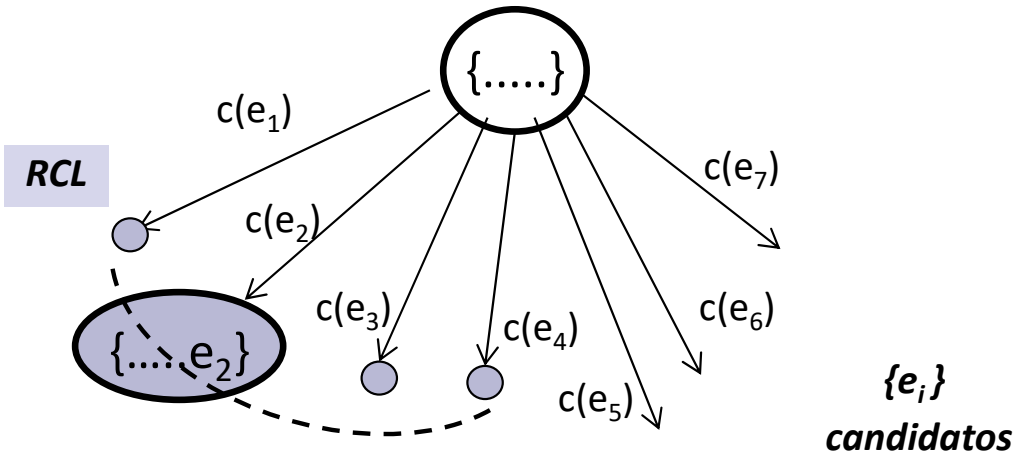
Fase Constructiva (greedy)

En cada iteración:

- **Elementos_Candidatos C={e_i}:**

  Conjunto de elementos factibles a incorporar a la solución parcial que se está construyendo.

  La incorporación de cada elemento de {e_i} tiene un **coste adicional c(e_i)**.



- **RCL:** lista de candidatos restringida (mejores elementos que incrementan menos, α, la función de coste).

- Elegir aleatoriamente un elemento de **RCL.**

La RCL esta formada por aquellos elementos e_i, tal que $c(e_i) \in \{c^{min}, \; c^{min} + \alpha(c^{max} - c^{min})\}$

α sirve para diversificar la construcción y no repetir soluciones:

α=0 da un algoritmo completamente *greedy*, con α=1 es una estrategia aleatoria.

➢ En cada iteración *no se selecciona al mejor candidato , sino un elemento aleatorio del conjunto de mejores candidatos (RCL).*

Fase Constructiva (greedy)

procedimiento Greedy Randomized Construction(Semilla)

$s \leftarrow 0$

Inicializa el conjunto candidato: $C \leftarrow E$

Evalua el costo incremental $c(e) \forall e \in C$

while $C \neq 0$ **do**

  $c^{min} \leftarrow min\{c(e) | e \in C\}$

  $c^{max} \leftarrow max\{c(e) | e \in C\}$

  $RCL \leftarrow \{e \in C | c(e) \leq c^{min} + \alpha(c^{max} - c^{min})\}$

  selecciona un elemente t de RCL aleatoriamente

  $s \leftarrow s \cup \{t\}$

  actualiza el conjunto candidato C

  re-evalua los costos incrementales $c(e) \forall e \in C$

return s

- *$C(e_i)$ representa el coste incremental de añadir e_i a la solución parcial.*
- En cada iteración *no se selecciona al mejor candidato , sino un elemento aleatorio del conjunto de mejores candidatos (RCL).*
- *En el caso que existan **restricciones de factibilidad**, $c(e)=\infty$ si $s \cup \{e_i\}$ no satisface restricciones.*
- *Coste incremental $c(e_i)$ puede ser obtenido como $h(e_i)$*

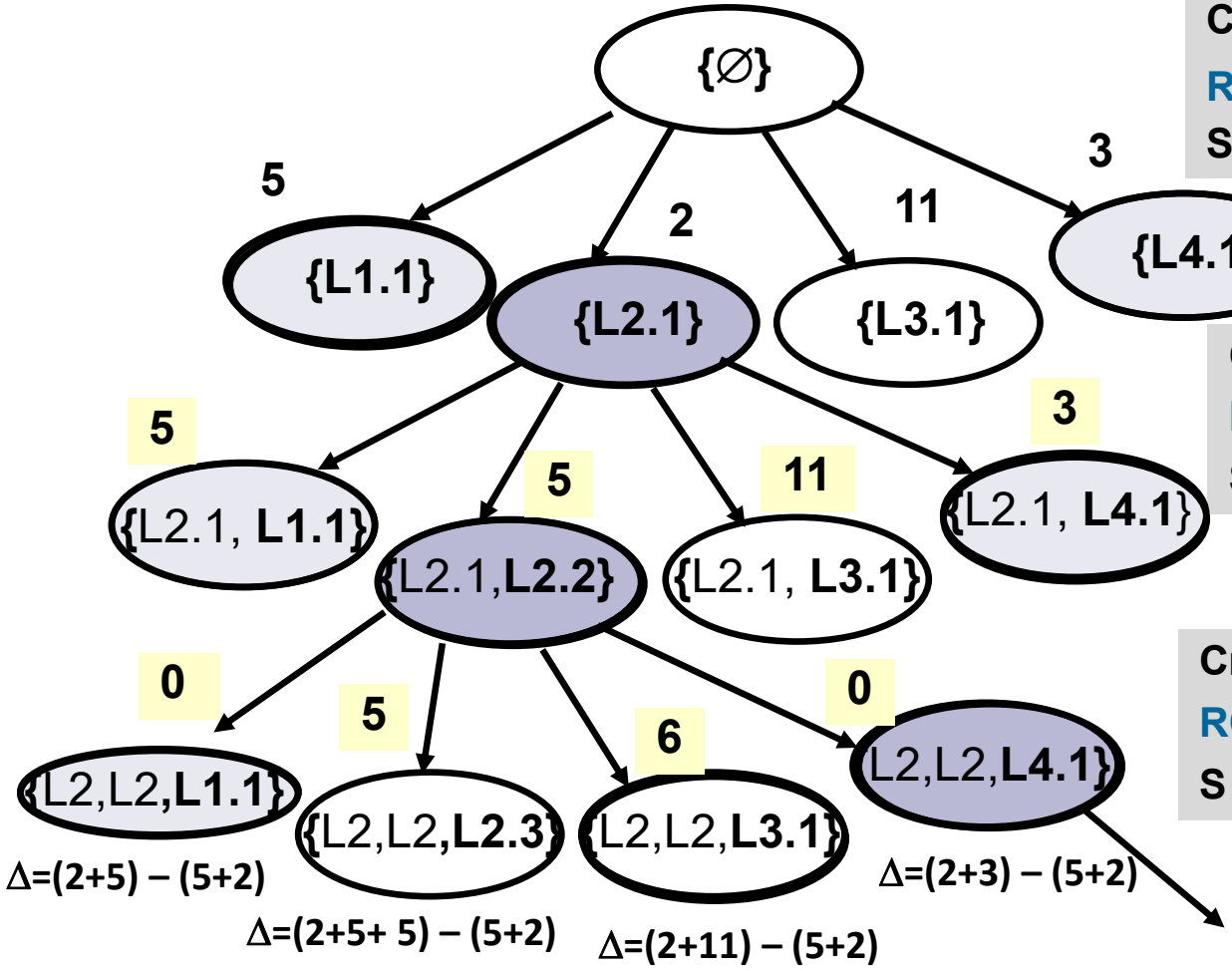
EJEMPLO-1

4 lectores (L1, L2, L3, L4) desean leer los periódicos (P1, P2, P3) en el mismo orden.

	Ready-Time	P1	P2	P3
L1	0	5'	10'	2'
L2	0	2'	5'	5'
L3	0	11'	15'	15'
L4	0	3'	5'	5'

Greedy: Fase Constructiva

S={∅}; α=0,4



Costemin=2; Costemax=11;

RCL= {L1, L2, L4}; *c(e) ≤ 2 + 0,4 (11-2)= 5.6*

S = {L2};

Cmin=3; Cmax=11;

RCL= {L4, L1, L2}; *c(e) ≤ 3 + 0,4 (11-3)=6.2*

S = {L2, L2};

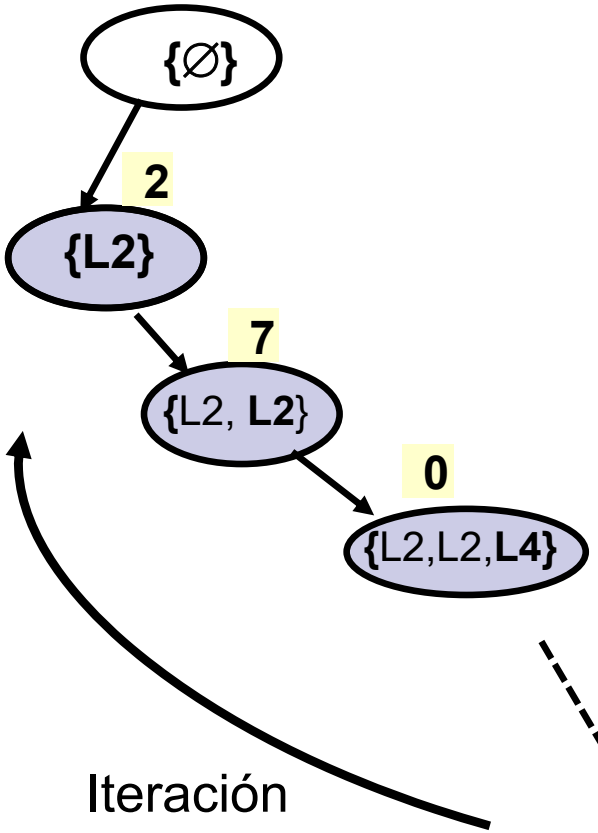
Cmin=0; Cmax=6;

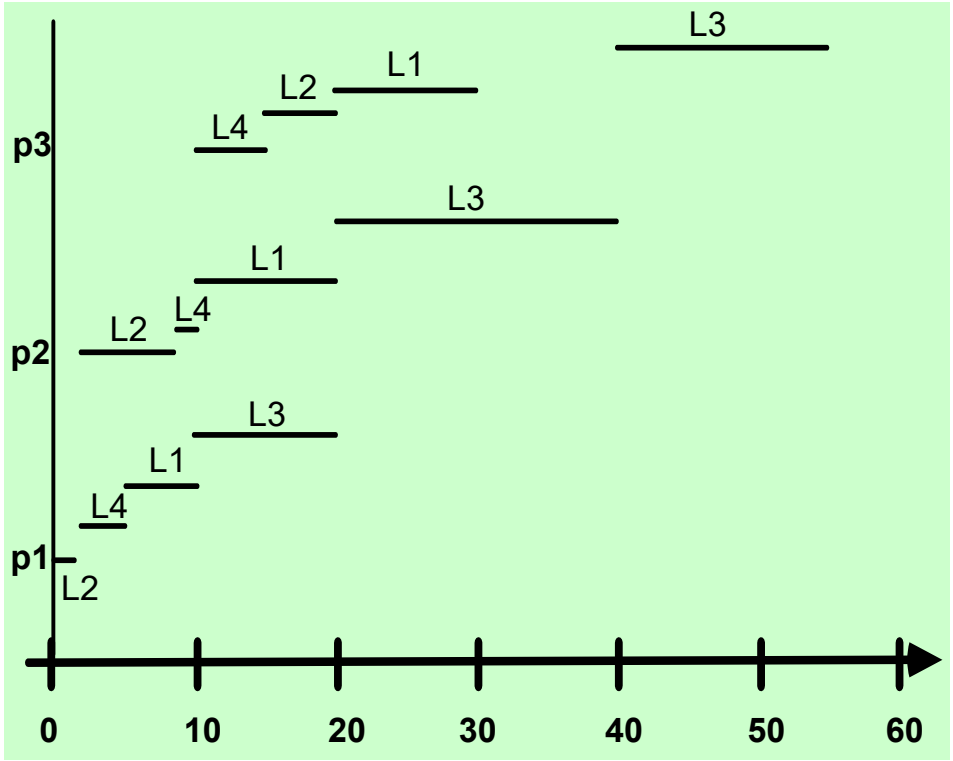
RCL= {L4, L1}; *c(e) ≤ 0 + 0,4 (6-0)=2.4*

S = {L2, L2, L4};

EJEMPLO

	Ready-Time	P1	P2	P3
L1	0	5'	10'	2'
L2	0	2'	5'	5'
L3	0	11'	15'	15'
L4	0	3'	5'	5'





SOL completa → SOL_Mejorada

GRASP: Importancia de la fase de Mejora Local

GRASP Semi-greedy: No tiene una fase de mejora local de la solución

En la fase constructiva, se obtienen soluciones optimizadas a partir de decisiones locales *'aleatorizadas'* de la RCL

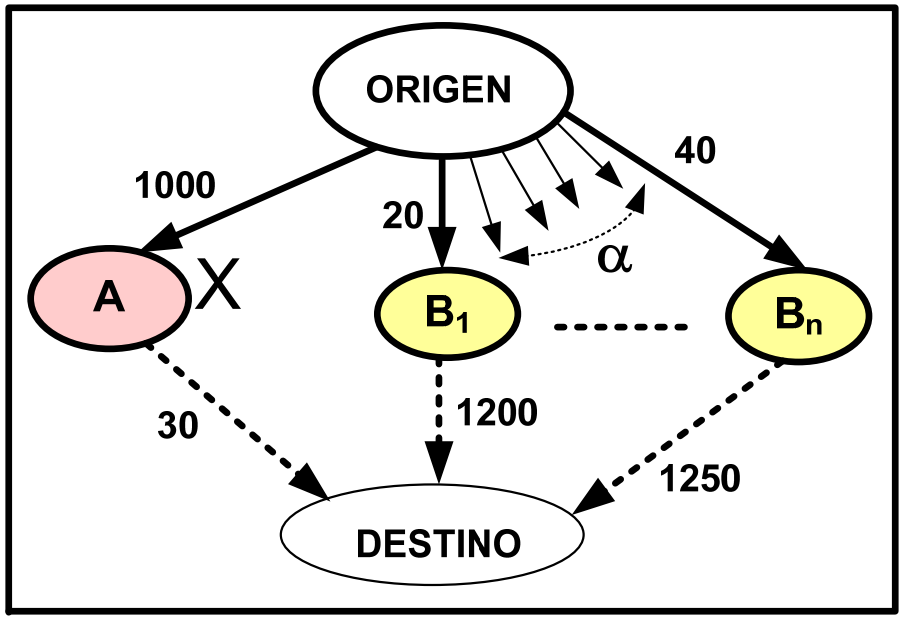
$$RCL \equiv \{c^{min}, \quad c^{min} + \alpha(c^{max} - c^{min})\}$$

Como α<1, la peor (o peores decisiones) en un cierto nivel ***van a ser siempre obviadas***

En el caso de que una 'pésima' decisión en un cierto nivel conduzca a la óptima solución, es imposible que GRASP pueda obtenerla, salvo en el proceso de mejora local final.

Cuando la solución óptima (o altamente optimizada) pueda estar compuesta de alguna decisión local donde la función de guía obtiene pésimas evaluaciones, un método constructivo puro no será adecuado.

GRASP tenderá a a caer en máximos locales, en sucesivas iteraciones, sin que le sea posible alcanzar el máximo global



Características GRASP

- No requiere solución inicial. Aplicaciones: Problemas donde se requiere la 'construcción' de soluciones.
- Obtiene soluciones 'optimizadas'. Mejora posterior por Búsqueda Local.

Variantes Metaheurística GRASP

Semi-greedy

- No tiene una fase de mejora local de la solución.
- Peligro de perder buenas soluciones si requiere un peor paso local.

Selección RCL

a) **Reactive GRASP**: Los ajustes de la RCL (α) se determinan dinámicamente según el estado de la búsqueda ($\approx$ enfriamiento simulado). Se inicia $\alpha \approx 1$ y va disminuyendo.

b) **Selección e$\in$RCL no aleatoria**: Elitista, Ruleta, Rango, etc.

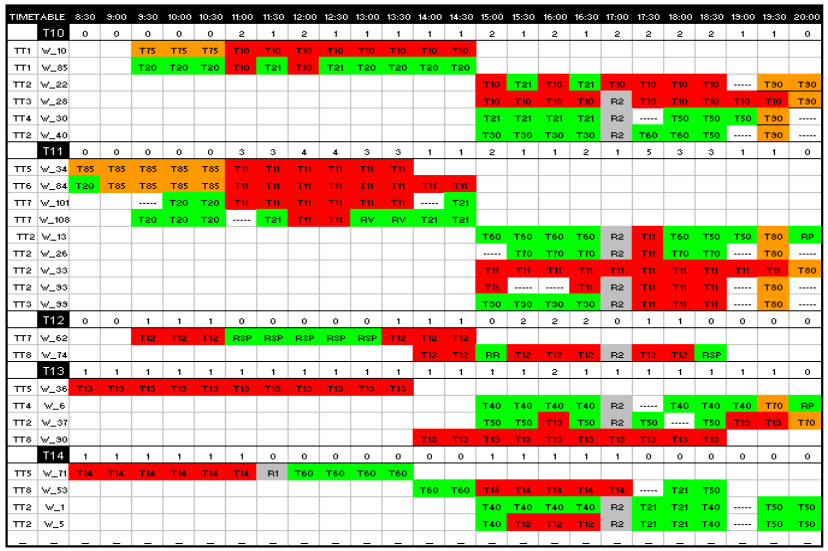
Clustering-GRASP, Path Relinking, etc.

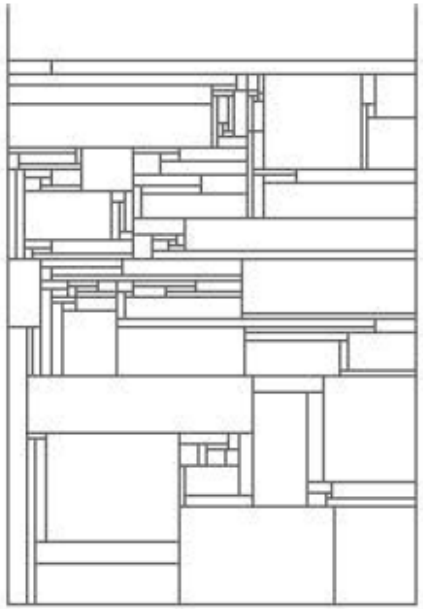
- Se guardan las mejores soluciones (no solo la mejor) de los reinicios. Las metaheurísticas de mejora se inician desde diversas mejores soluciones previas
- No formar la RCL entre sucesores inmediatos del estado actual, sino los de un nivel siguiente, etc.

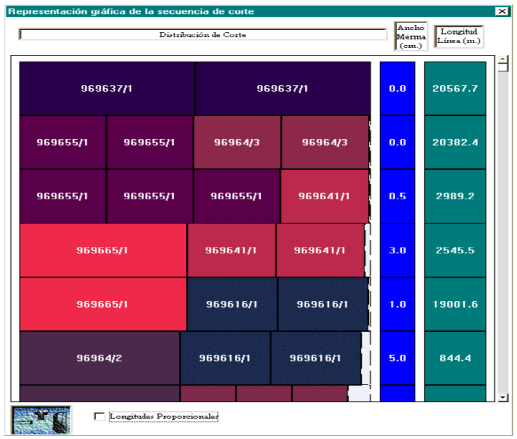
Aplicaciones

Apropiados para:

- Problemas en los que es difícil obtener soluciones iniciales (factibilidad)
- Vecindario muy amplio o poco estructurado (metaheurísticas de mejora)
- Complejidad de la representación de la solución (en metaheurísticas poblacionales)





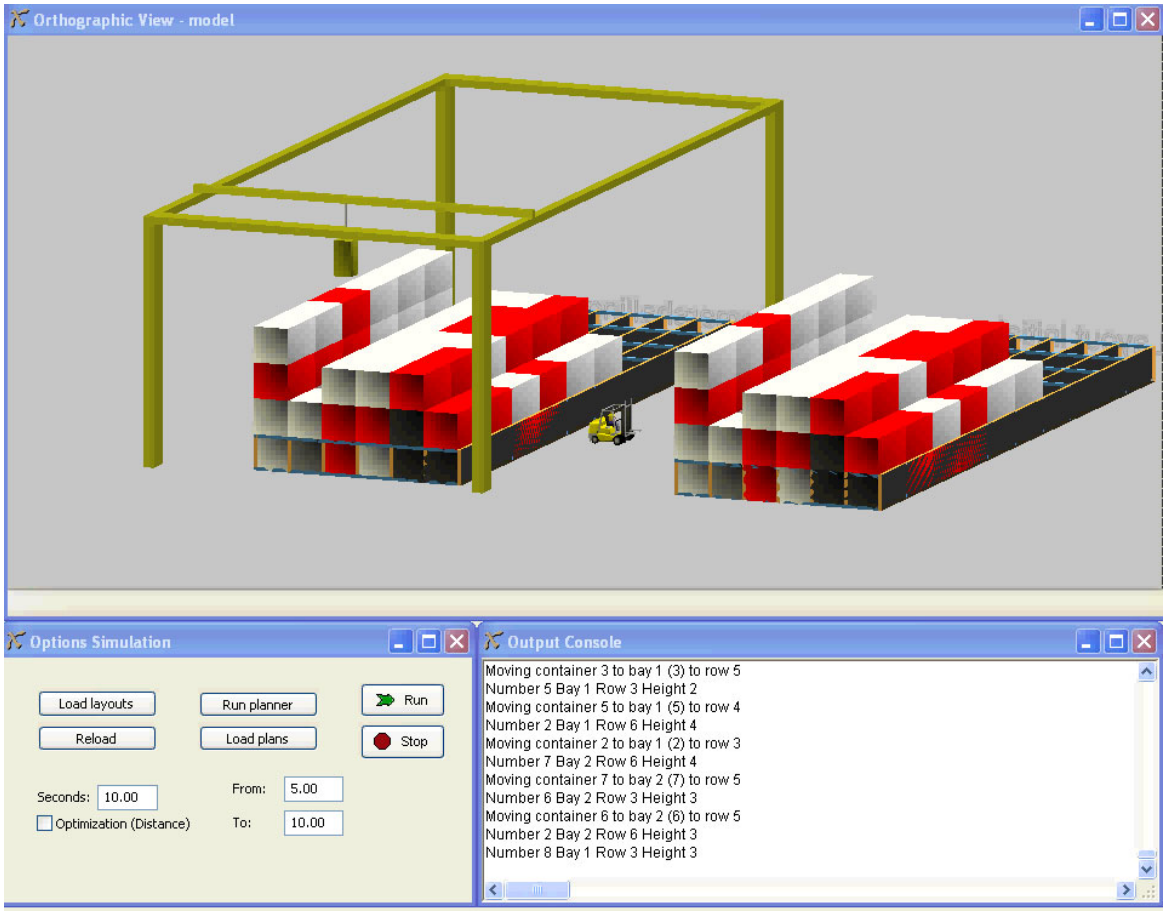


Aplicación Decisión GRASP en Planificación

Reshuffling a Container Yard with 6 rows x 6 stacks (maximum high: 4 containers):

To improve the planning performance,

- a domain-dependent heuristic function is based.
- the next action is randomly chosen from the actions that obtain a better performance of the heuristic function.



Algorithm 1: Pseudo-code of the domain-dependent heuristic function

```
Data: s: state to evaluate
Result: h, heuristic value of s
h = 0;
if ∃ x - container / holding(x) ∈ s then
    if goal-container(x) then
        h = 0.1;
    else
        h = 0.5;
    end
end
for each row r in the yard-bay do
    Δh = 0;
    for x - container / at(x, r) ∈ s ∧ goal-container(x) do
        if ∄ y - container / goal-container(y) ∧ on(y, x) ∈ s then
            Δh = max(Δh, numContainersOn(x));
        end
    end
    h += Δh;
end
```

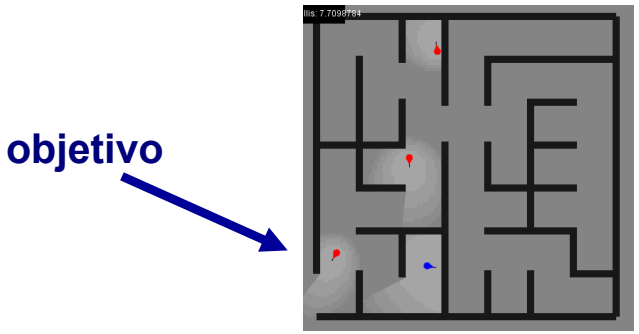
APLICACIÓN: Path-finding

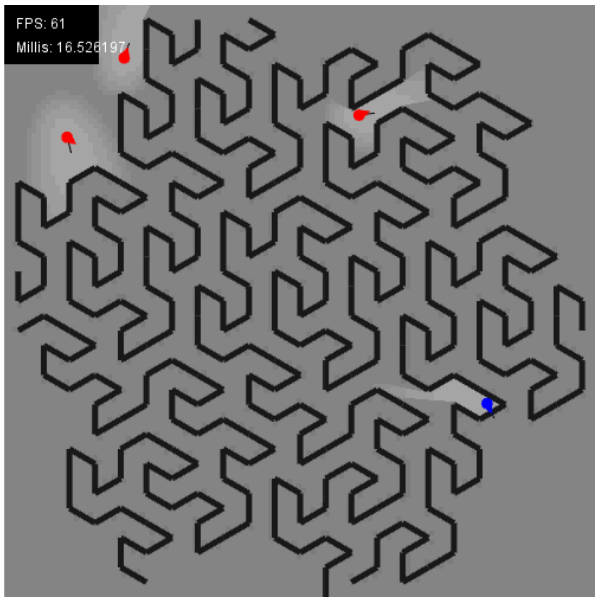
Path-finding: **Problema de encontrar el mejor camino/ruta** (habitualmente, el más corto) entre dos puntos a través de un mapa con obstáculos.

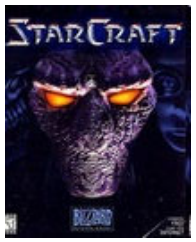
Problemas de laberinto, *Aplicaciones para videojuegos,* etc.

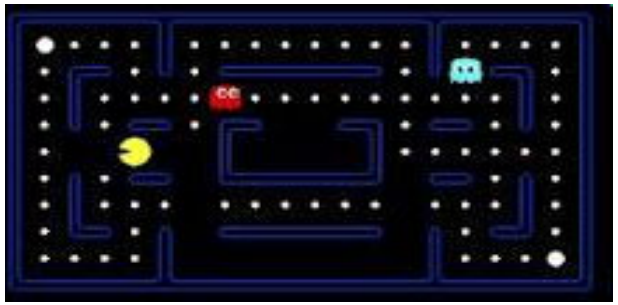
Algoritmos típicos: Dijkstra (muy costoso en laberintos complejos),
Algoritmos A (equivalente a Dijkstra, cuando h(n)=0),
A* sobre-informados, etc.

Metaheurísticas: Grasp, Inteligencia de Enjambre, Algoritmo de las hormigas, …







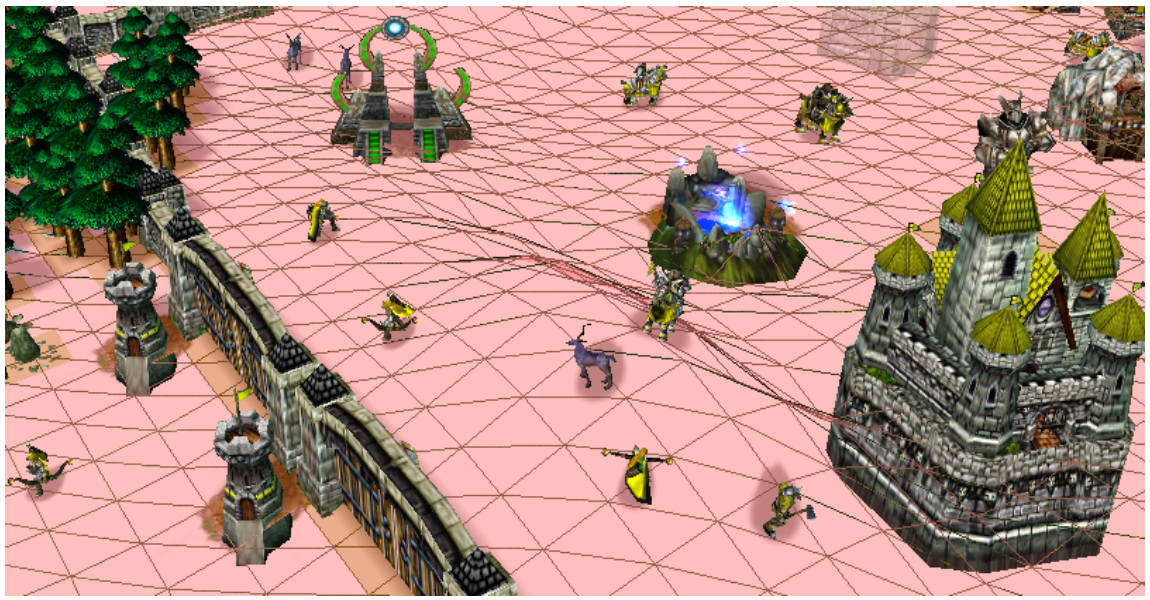


http://www.movingai.com/benchmarks/

Benchmarks for Grid-Based Path finding.

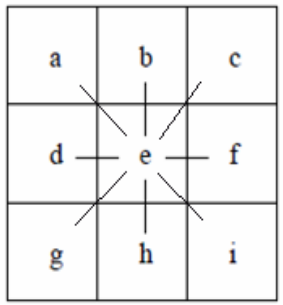
IEEE Trans. on Comp. Int. and AI in grames

Grid-based path-finding: *Descomposición del Mapa del terreno en Rejillas*

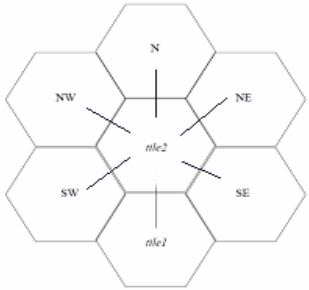


Rejilla de descomposición (Warcraft 3 Map)

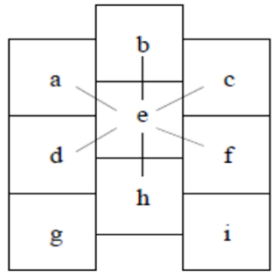
Alternativas de Descomposición:



Octiles

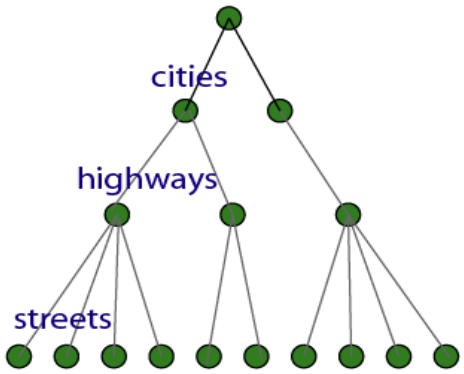


Hexes

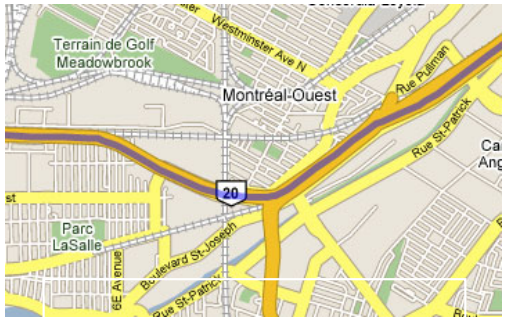


Texes

- A mayor nivel de descomposición, mayor nivel de ramificación.
- Técnica Jerárquica: Path-finding jerárquico

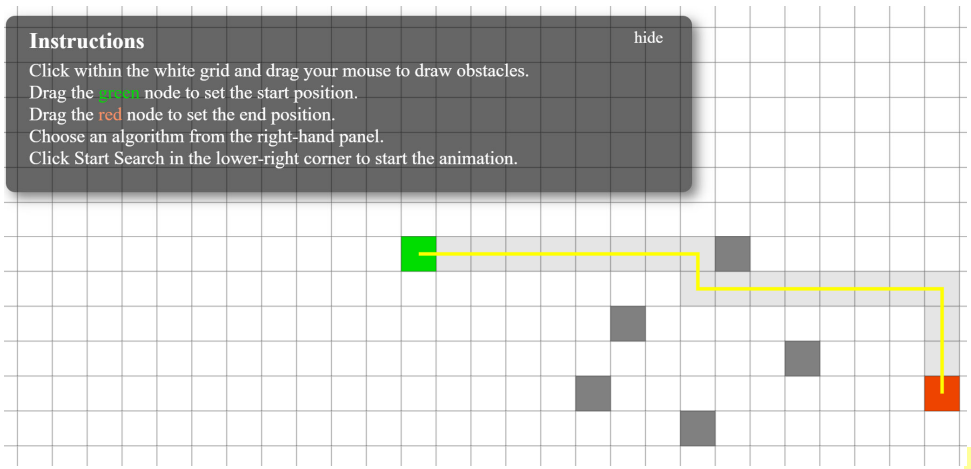


Paso 1) Búsqueda Conexiones Ciudades



Paso 2) Búsqueda de Calle

Path-finding: heurísticos básicos A, A*



Estimación distancia:

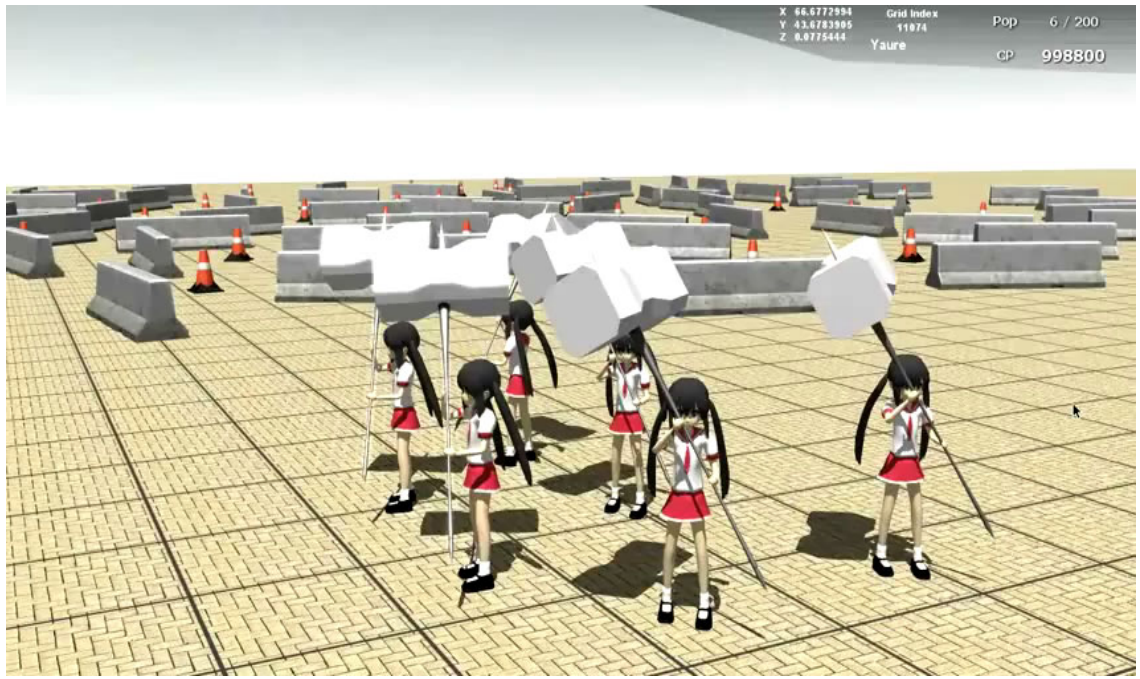
Manhatan, Euclidea, Hamiltoniana, etc.

https://qiao.github.io/PathFinding.js/visual/

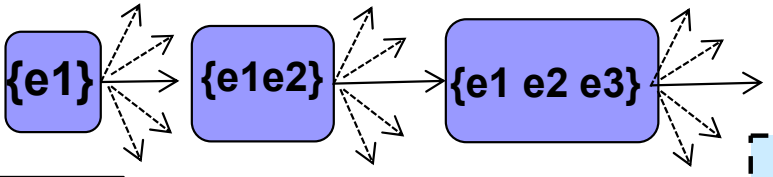
http://ashblue.github.io/javascript-pathfinding/

Path-finding con múltiples agentes

(deben mantener una cierta cohesión: PSO)



https://www.youtube.com/watch?v=nGC_kBCoHYc



Construcción Solución Optimizada
(Pasos iterativos: +Elementos Solución)

Nuevos elementos se añaden iterativamente a la solución.

Métodos Constructivos:
Algoritmo A, A* ⇒ GRASP

Mejora de una Solución
Requiere Solución Inicial
(Pasos iterativos: Mejora Solución)
Búsqueda en un espacio de Soluciones

Mejora Iterativa de UNA SOLUCIÓN → Enfriamiento Simulado, Búsqueda Tabú

Mejora Iterativa CONJUNTO SOLUCIONES (no colaborativa) → Búsqueda en Haz

Mejora Conjunto Soluciones
(Pasos: Evoluciones Cooperativas)

Control Centralizado:
Evoluciones poblacionales
(Pobl. Individuos ~ Soluciones) → Alg. Genéticos, Alg. Meméticos, Búsqueda Dispersa

Generación & mejora

Control Autónomo:
Inteligencia de Enjambre
Enjambre de Individuos (Soluciones) → Alg. Hormigas, Colonia de Abejas, Enjambre de Partículas…