Problemas en la implementación de algoritmos de *routing* de alta complejidad en dispositivos móviles: el caso Itiner@

Laia Descamps-Vila

UOC Estudis d'Informàtica, Multimèdia i Telecomunicació (UOC)

ICA Grupo ICA (Informática y Comunicaciones Avanzadas S.L.)

VI Jornadas SIG Libre
21, 22 y 23 marzo 2012
Girona

TSI-020110-2009-442

plan avanza2

Algoritmos

- Personalización
  - Selecciona los puntos de interés más atractivos para un usuario concreto

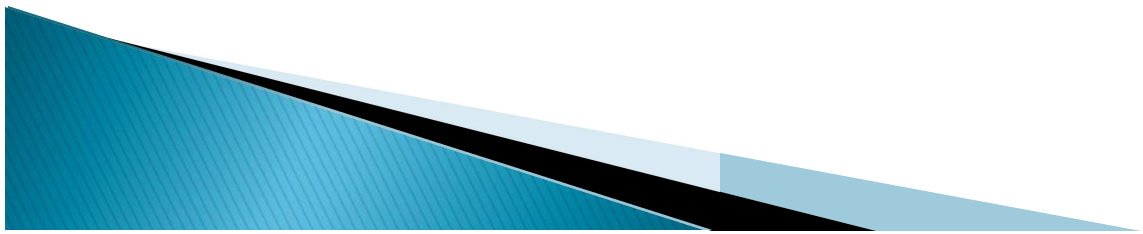
- Routing
  - Selecciona el camino para ir de un punto a otro.

Algoritmo personalización

- Características
  Diseño
  Ejemplo rutas

Características

- Rutas viables→ se ofrece una visita si el usuario puede llevarla a cabo.
  - Por ejemplo, no ofrece visitar un lugar que estará cerrado cuando el usuario llegue a visitarlo.
- Ruta más atractiva según preferencias.



Algoritmo personalización

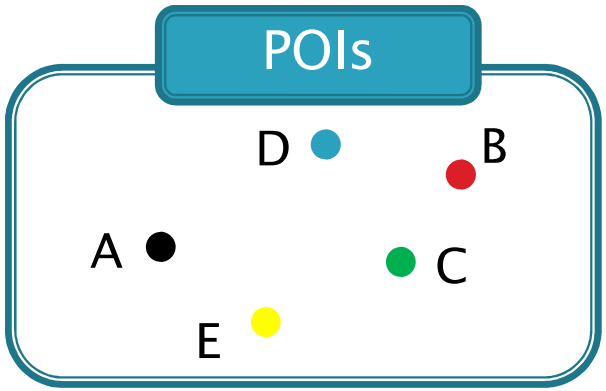
- Características
- Diseño
- Ejemplo rutas

Diseño algoritmo

- Para generar rutas viables:

1. Selecciona grupo de POIs según preferencias usuario.
   - Se generan combinaciones sin repetición de POIs que forman rutas aleatorias: [A, B, C, D, E], [A, B, C, D, F], [A, B, C, E, F], etc.

2. Se ordenan los puntos de cada ruta con una función TSP, según distancia más corta.



Antes ordenar

[A, B, C, D, E]

Después ordenar

[A, E, C, B, D]

CONJUNTO RUTAS

Restricciones

- Se **DESCARTAN** rutas si:

1. El tiempo de ruta es superior al tiempo del visitante.

$$r(t) = \begin{cases} 1 & si \quad t_{ruta} \leq t_{visitante} \\ 0 & si \quad t_{ruta} > t_{visitante} \end{cases}$$

$$t_{ruta} = t_o + \sum_{i=0}^{s} t_{visita}(i) + \sum_{i=0}^{s-1} t_{viaje}(i, i+1)$$

Restricciones

- Se **DESCARTAN** rutas si:

2. Hay un POI cerrado cuando el visitante llega a visitarlo.

$$t_{llega}(j) = t_o + \sum_{i=0}^{j-1} t_{visita}(i) + \sum_{i=0}^{j-2} t_{viaje}(i, i+1)$$

$$r(t,j) = \begin{cases} 1 & si \quad t_{abrir}(j) < t_{llega}(j) < t_{cerrar}(j) \\ 0 & si \quad t_{abrir}(j) > t_{llega}(j) \\ 0 & si \quad t_{llega}(j) > t_{cerrar}(j) \end{cases}$$

Restricciones

- Se **DESCARTAN** rutas si:

3. No hay un POI donde comer a la hora que el visitante quiere comer.

$$r(t,j) = \left\{ \begin{array}{ll} 1 \;\; si & t_{inicio\_comida} < t_{llega}(j = poi_{tipo\_comida}) < t_{final\_comida} \\ 0 \;\; si & t_{inicio\_comida} < t_{llega}(j \neq poi_{tipo\_comida}) < t_{final\_comida} \\ 0 \;\; si & t_{abrir}(j) > t_{llega}(j = poi_{tipo\_comida}) \\ 0 \;\; si & t_{llega}(j) > t_{cerrar}(j = poi_{tipo\_comida}) \end{array} \right\}$$

Rutas que pasan las 3 restricciones

CONJUNTO RUTAS VIABLES

Siguiente paso

Seleccionar más atractiva

Funciones

1. Importancia media

$$c_{medio} = \sum_{i=0}^{s} poi_{puntuación\_media}$$

2. Importancia intrínseca

$$poi_{puntuación\_intrínseca} = \begin{Bmatrix} poi_{coste\_familia} & si & usuario_{viaja} = familia \\ poi_{coste\_pareja} & si & usuario_{viaja} = pareja \\ poi_{coste\_amigos} & si & usuario_{viaja} = amigos \\ poi_{coste\_solo} & si & usuario_{viaja} = sólo \end{Bmatrix}$$

$$c_{intrínseco} = \sum_{i=0}^{s} poi_{puntuación\_intrínseca}$$

3. Puntos interés preferidos

$$c_{tipo\_punto}(i) = \left\{ \begin{array}{cccc} 5 & si & poi_{tipo\_punto}(i) & \in ruta \\ 0 & si & poi_{tipo\_punto}(i) & \notin ruta \end{array} \right\}$$

$$c_{tipo} = \sum_{i=0}^{s} c_{tipo\_punto}(i)$$

4. Sitios comida preferida

$$c_{tipo\_comida}(i) = \left\{ \begin{array}{cccc} 5 & si & poi_{tipo\_comida}(i) & \in ruta \\ 0 & si & poi_{tipo\_comida}(i) & \notin ruta \end{array} \right\}$$

$$c_{comida} = \sum_{i=0}^{s} c_{tipo\_comida}(i)$$

5. Importancia general

$$c_{interés\_general}(i) = \left\{ \begin{array}{ccc} 5 & si & poi(i) = imprescindible \\ 3 & si & poi(i) = interesante \\ 0 & si & poi(i) = prescindible \end{array} \right\}$$

$$c_{general} = \sum_{i=0}^{s} poi_{interés\_general}$$

Función de coste

Rutas más atractiva → máximo coste

$$c_{ruta} = c_{medio} + c_{intrínseco} + c_{tipo} + c_{comida} + c_{general}$$

RUTA VIABLE MÁS ATRACTIVA

Algoritmo personalización

- Características
- Diseño
- Ejemplo rutas

Preferencias a seleccionar

- Horario de comida y cena
- Hora final de ruta
- Incluir o no puntos de comida en la ruta
- Modo de transporte
- Puntos de interés preferidos
- Tipo de comida preferida
- Con quién viaja el usuario
- Viaja con animales
- Viaja con personas con problemas de movilidad

Ejemplo rutas

- Hora comida: *13-15h*
- Hora fin ruta: *15.00h*
- Modo transporte: *a pie*
- Viaja *solo*
- Puntos de interés: *histórico y turismo*
- Comida preferida: *restaurante*
- *No viaja con animales*

- No quiere comer
- Hora fin ruta: *18.00h*
- Modo transporte: *bici*
- Viaja *con la familia*
- Puntos de interés: *deporte y turismo*
- Viaja *con animales*

Visitante A

Visitante B

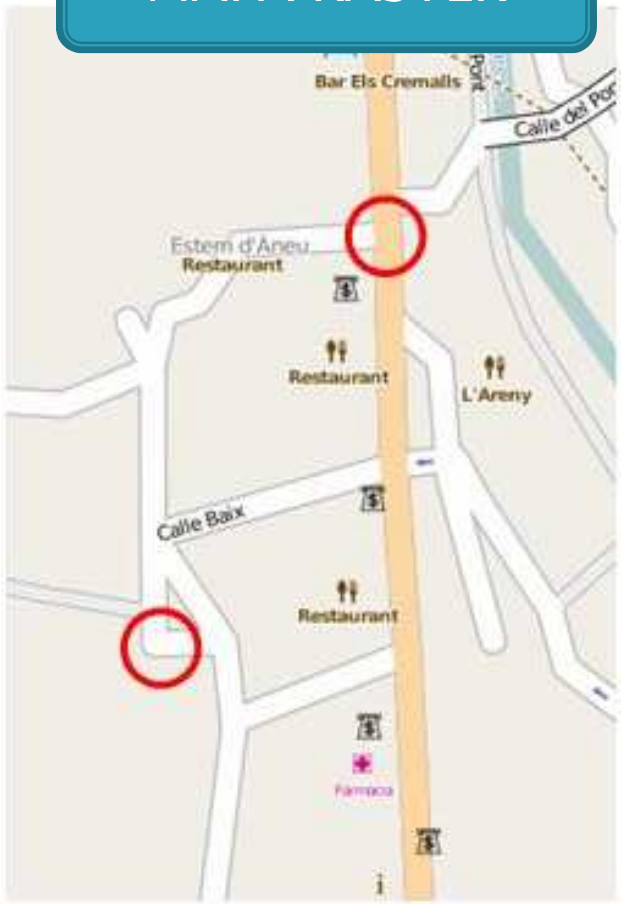
Visitante A

Visitante B

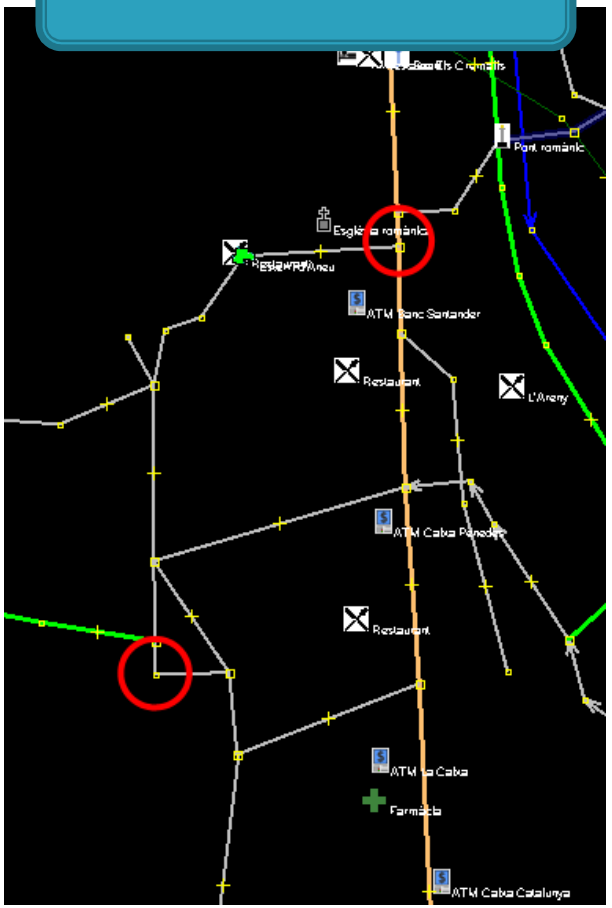
Algoritmo routing

¿Qué hace?

MAPA RASTER



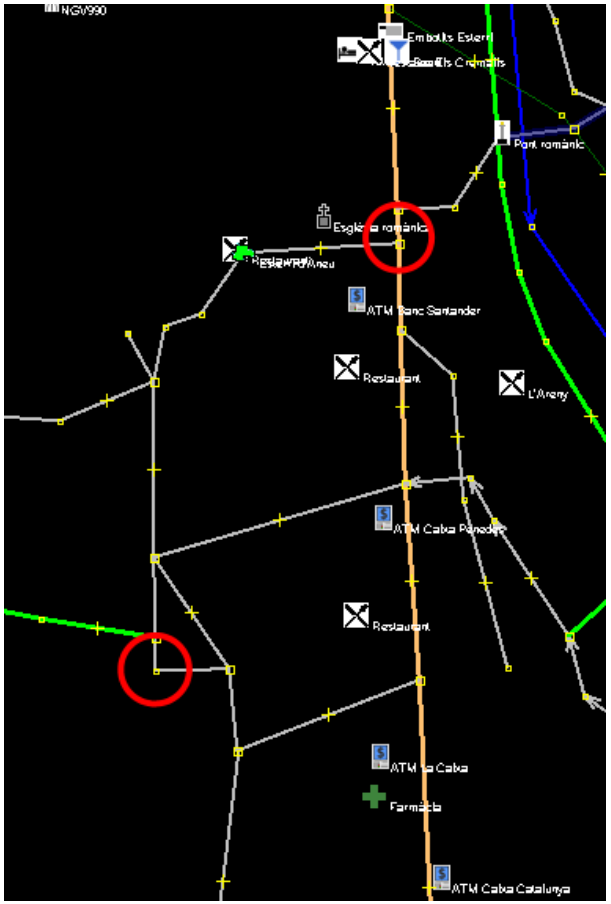
MAPA VECTORIAL



Características

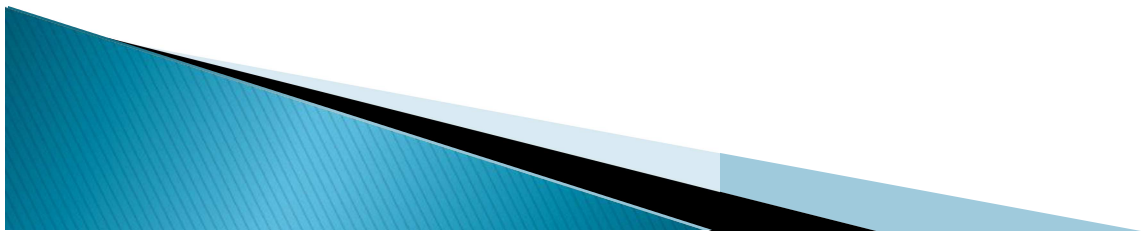
- Camino aceptable
- No se repiten nodos
- Minimizar caminos

BACKTRACKING



Datos vectoriales

- Almacenar en BD SQLite
  - Consultas lentas 
- Almacenar en memoria en una matriz de adyacencia.
  - Acceso muy rápido. 
  - Problemas de memoria en una aplicación Android. 



Soluciones problemas memoria

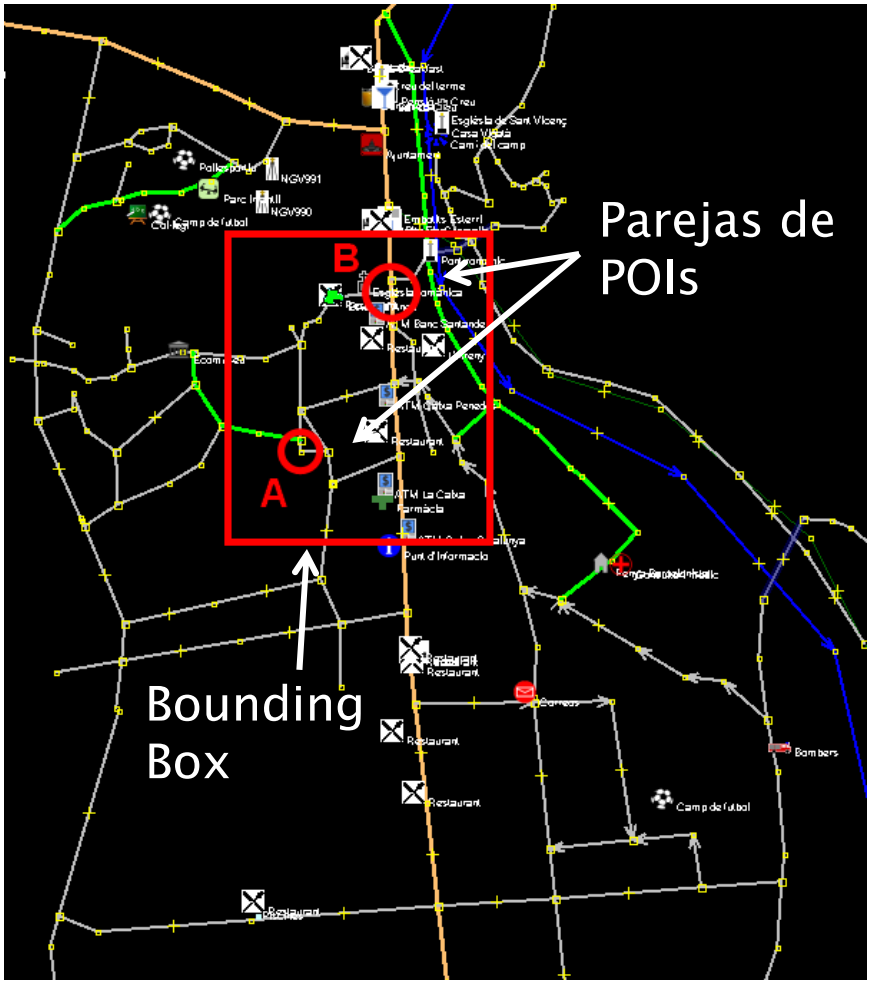
Reducir cantidad de nodos

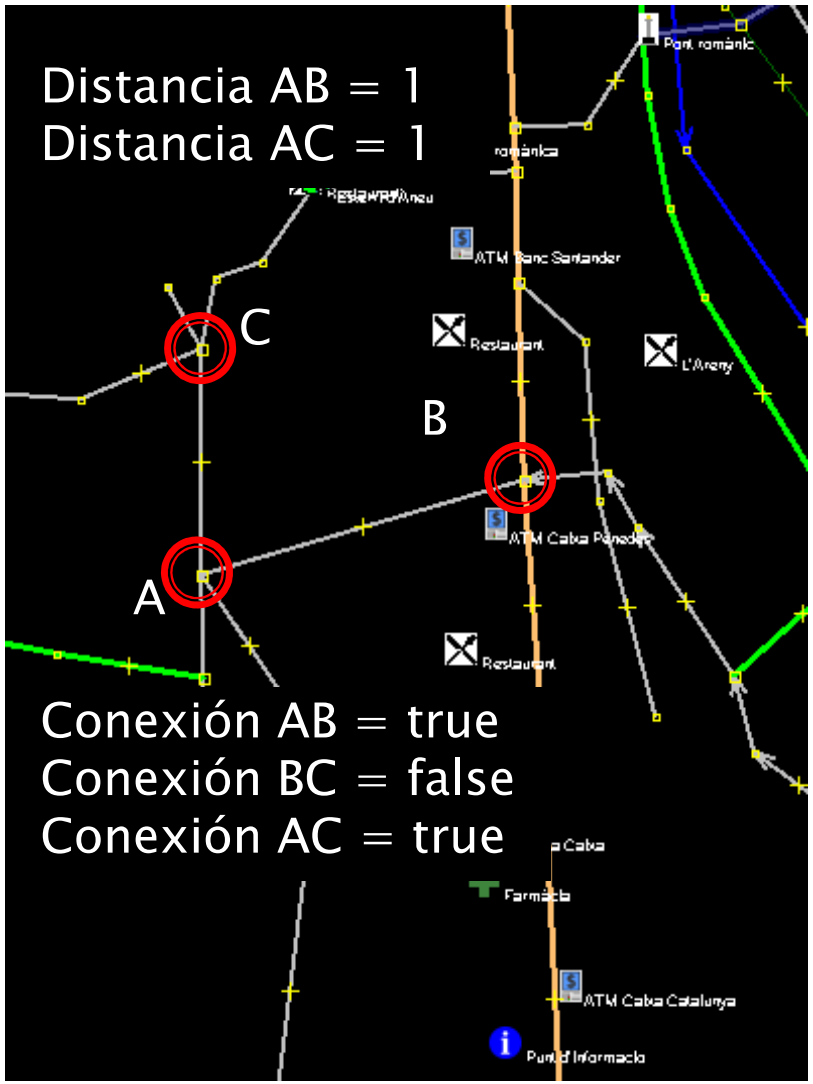
Usar datos que ocupen menos memoria

Reducir cantidad de nodos

Parejas de POIs

Bounding Box





Usar datos que ocupen menos memoria

Sólo guardar la información de si nodos están conectados

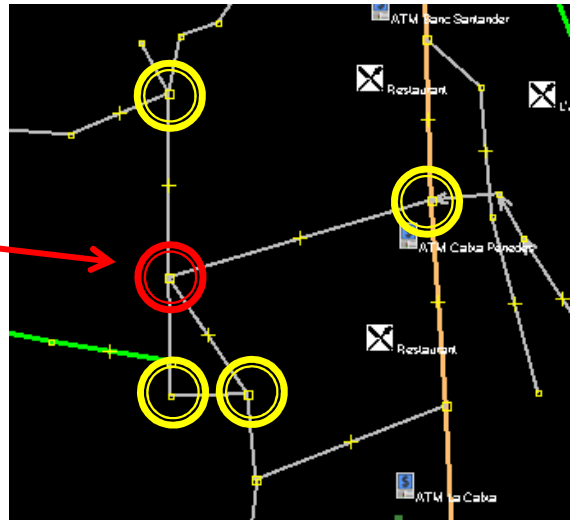
No guardar la distancia entre nodos. Distancia = 1

Mejorar eficiencia backtracking

1. Limitar el número de caminos.

$$número\ caminos/nodo\ = \frac{24}{conexiones\ nodo}$$

Nodo con 4 conexiones



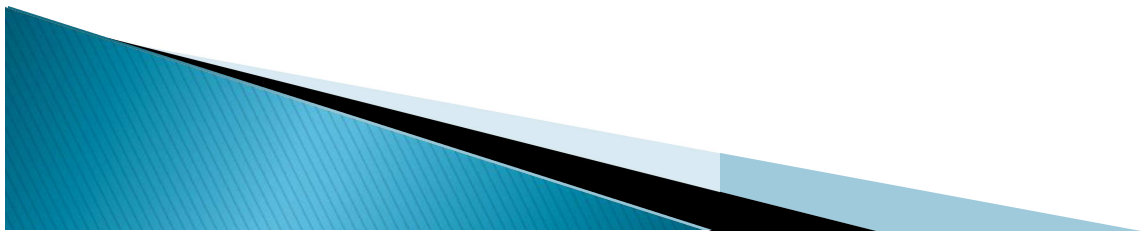
6 caminos/nodo

2. Buscar caminos en la dirección correcta

- Se empieza a buscar por los nodos que están más próximos al POI de destino.

Diseño final

1. Se seleccionan parejas de POIs
2. Se seleccionan los nodos contenidos en un rectángulo delimitado por dos POIs
3. Se pre-ordenan los nodos según distancia al POI destino
4. Se crea matriz adyacencia
5. Se aplica el backtracking para buscar caminos
6. Se obtiene el camino con menos nodos



Conclusiones

Algoritmo routing

+

Algoritmo personalización