Ajuste del control semafórico automático.

Gabriel Luque (gabriel@lcc.uma.es)

Transporte y Tráfico

Contenido


Introducción
• Problemática
• Soluciones
• Propuesta
Ajuste Semafórico
• Conceptos Previos
• Sistemas actuales
• Sistema automático propuesto
Resolviendo
• Modelado
• Datos y realismo
Algoritmos
• Implementando una propuesta
• Algoritmos evolutivos
• Multi-objetivo

Introducción

- Problemática
- Soluciones
- Propuesta

Problemática

- Evolución de las ciudades
  - Actualmente, la mayor parte de la población mundial vive en grandes ciudades
  - Se espera que en un futuro próximo el 80% de la población mundial resida en una gran ciudad

- Esto produce una gran cantidad de nuevos retos y problemas como los relativos al excesivo tráfico de vehículos:
  - Atascos
  - Contaminación
  - Seguridad
  - ...

Posibles soluciones

- Soluciones “clásicas”:
  - Infraestructuras
  - Promocionar el transporte público u otros (bicicletas)
  - Promover el uso compartido del vehículo (VAO)
  - Limitar el acceso a vehículos

- Soluciones “inteligentes”: uso más eficiente y efectivo de lo que se dispone
  - Ofrecer información precisa y actual al usuario para la toma de decisiones (tráfico, plazas de aparcamiento disponibles, …)
  - Toma de decisiones automatizadas: rutas adaptables y/o personalizadas
  - Mejor ajuste de los elementos actuales: rutas y frecuencia del transporte público, ajuste de los semáforos, …

Propuesta

Ajuste inteligente de los tiempos de los semáforos de forma automática

- Menos atascos
- Menos esperas
- Rutas más rápidas
- Menos contaminación

Ajuste Semafórico

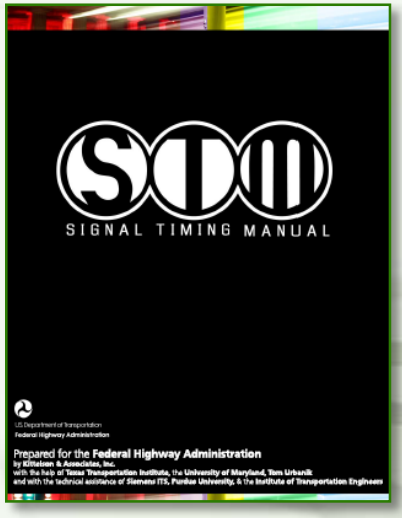
- Conceptos Previos
- Sistemas actuales
- Sistema automático propuesto

Conceptos previos

- Se debe conocer bien los elementos que intervienen en el problema a resolver

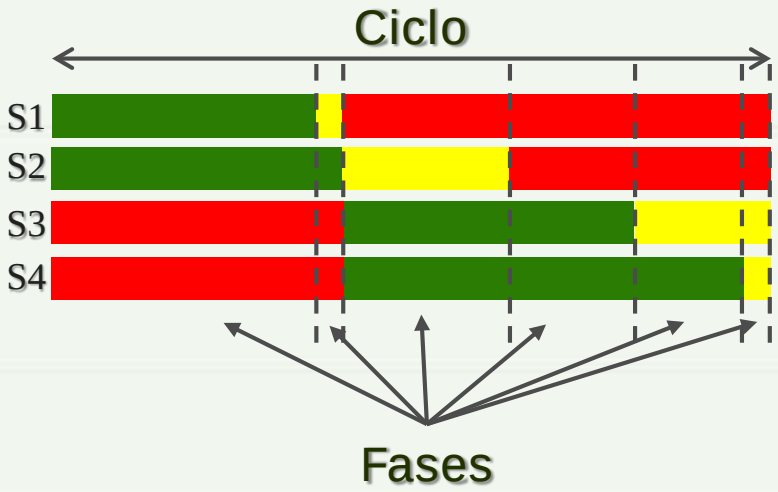
- Múltiples fuentes de información:
  - Reglamentos internacionales. Ej (U.S. Transport Deparment):
    - Manual on Uniform Traffic Control Devices (862 páginas)
    - Traffic Signal Timing Manual (274 páginas)
  - Reglamentos nacionales. Ej (DGT):
    - Regulación semafórica (32 páginas)
    - Cruces semafóricos y sincronismo (32 páginas)
  - Personal especializado (gestores de tráfico)
  - Literatura científica

- Filtrado de la información



Conceptos previos

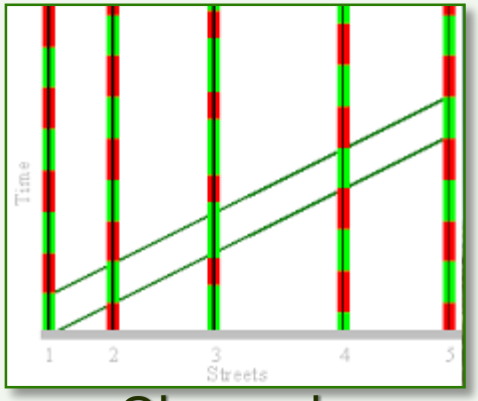
- Definiciones importantes:
  - Cruce semafórico
  - Ciclo (y tiempo de ciclo)
  - Fase (y tiempo de fase)
  - Plan semafórico



Cruce semafórico

Conceptos previos

- Otra información útil:
  - Se puede modificar tanto el tiempo de ciclo como el de fase
  - No se pueden cambiar las fases
  - Tiempos de ciclo: 60-120 segundos
  - Fases amarillas (de precaución antes de rojo): 4 segundos
  - Existen tiempos mínimo de fase (ejemplo: las fases para peatones –rojas y algunas amarillas- debe dar tiempo a cruzar la calle a una persona que vaya a 1m/s)
  - Promocionar olas verdes en grandes avenidas
  - Planes dependiente del tráfico (hora/tipo de día/estación)
  - ...



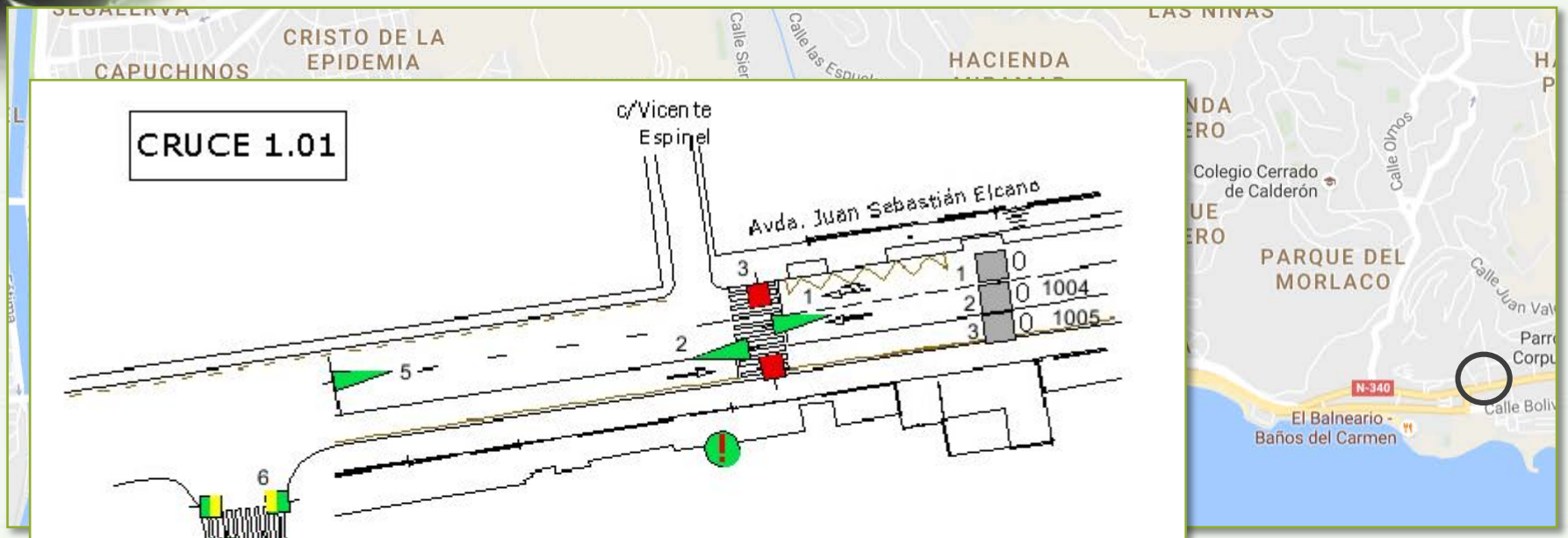
Ola verde

Cómo se configuran actualmente

- Ubicación/tipo:
  - Los reglamentos fija dónde colocar los semáforos obligatorios
  - También ofrece otras ubicaciones no obligatorias pero sí recomendadas

- Definición de las fases:
  - Fijadas por el reglamento
- Duración de ciclo/fase:
  - Definido por el gestor atendiendo a restricciones
  - Habitualmente se hace manualmente por cruce (y en algunos casos por avenida)
  - Basado en la experiencia y conocimiento acumulado
  - Hay sistemas dinámicos (reaccionan ante el tráfico actual). Problema: cambios rápidos que no mejoran el tráfico

Cómo se configuran actualmente



- 3 Flujos de tráfico
- 6 Semáforos:
  - 4 de vehículos (1 de bus)
  - 2 de peatones

Cómo se configuran actualmente

- Generan varios planes semafóricos que se eligen atendiendo:
  - Estación (verano – invierno)
  - Día de la semana
  - Franja horaria (de 3 a 9)

SELECCIÓN HORARIA - VERANO 2015

SUB 1 LABORABLE

Hora	Plan	Ciclo	Sentido
0:00	12	80	Oeste
2:00	13	70	Oeste
5:00	12	80	Oeste
6:30	20	115	Oeste-Simultáneo (118 a 115)
11:00	5	110	Oeste
16:00	2	115	Oeste
20:30	5	110	Oeste
22:30	7	100	Oeste

SUB 1 SABADO

Hora	Plan	Ciclo	Sentido
0:00	7	100	Oeste
3:00	12	80	Oeste
10:00	5	110	Oeste
14:00	7	100	Oeste
17:30	5	110	Oeste
22:30	7	100	Oeste

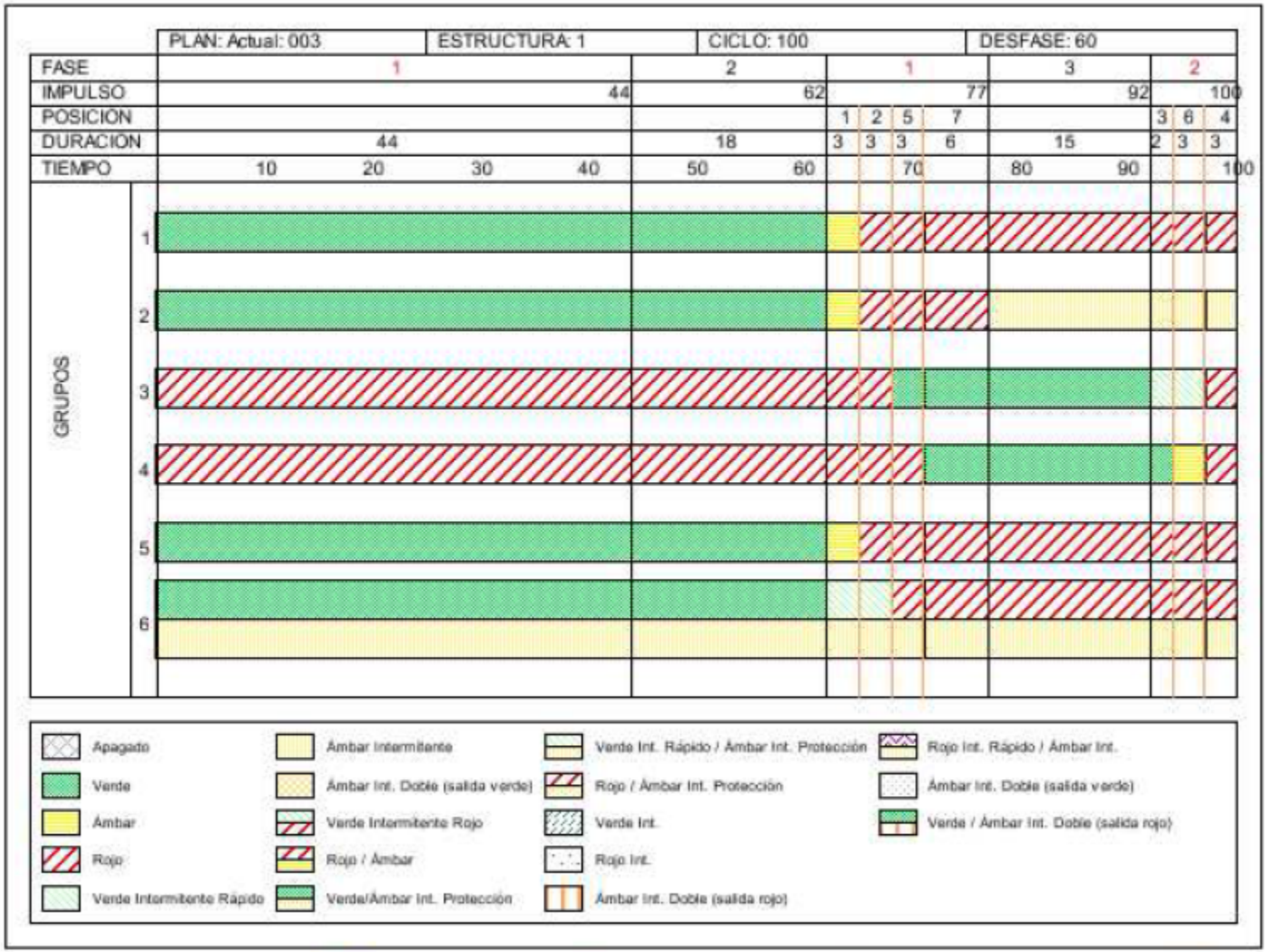
SUB 1 DOMINGO

Hora	Plan	Ciclo	Sentido
0:00	7	100	Oeste
3:00	12	80	Oeste
10:00	11	90	Oeste
11:30	10	95	Oeste
16:00	7	100	Oeste
17:00	20	115	Oeste
22:30	7	100	Oeste

Cruce: 01010 PLAN: Actual: 003

Descripción: Juan Sebastián Elcano - c/ Vicente Espinel

Comentario:



06-jun-2016 14:31:43

Sistema propuesto

- Generar de forma automática los tiempos de ciclo y fase:
  - Esos tiempos deben respetar las restricciones
- Considerar simultáneamente todos los semáforos de la ciudad o del área definida por el centro de control de tráfico
- Obtener diferentes planes (*offline*) atendiendo a las intensidad de tráfico
- El objetivo final es obtener un tráfico más fluido y que produzca menos contaminación

Resolución

Modelado

Datos y realismo

Modelado del problema

- Dada la siguiente información:
  - Los cruces semafóricos a mejorar (localización, fases)
  - Y el tráfico (o tipo de tráfico existente)

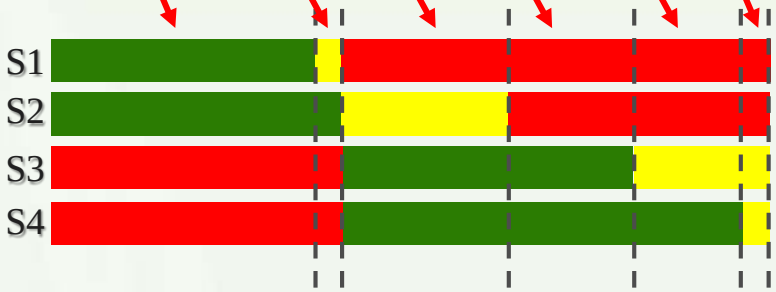
- El objetivo es: encontrar la configuración (solución) para los tiempos de fase y ciclo que mejore el resto de soluciones existentes

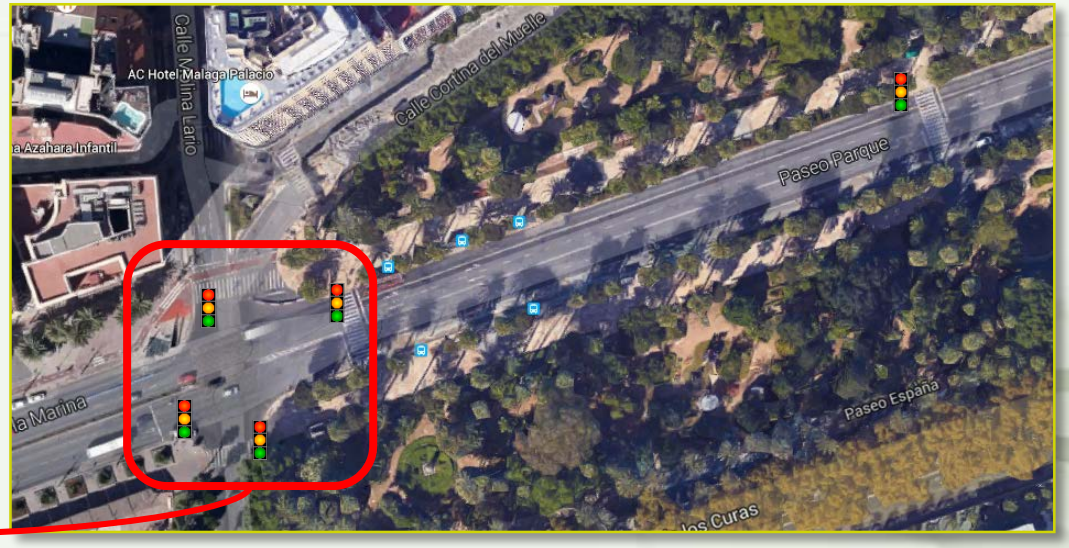
- Dificultades:
  - ¿Qué es computacionalmente una solución (representación)?
  - ¿Qué es ser mejor (punto de vista numérico)?
  - ¿De dónde obtenemos la información?
  - ¿Cómo encontramos la mejor?

Cómo representar una solución

- Existen múltiples formas de representarla
- Vector de naturales:
  - Cada valor representa una fase

20	8	47	23	30	10	60	4	30	29	18	50	70	...	32





Cómo representar una solución

- Existen múltiples formas de representarla
- Vector de naturales:
  - Cada valor representa una fase

20	8	47	23	30	10	60	4	30	29	18	50	70	...	32

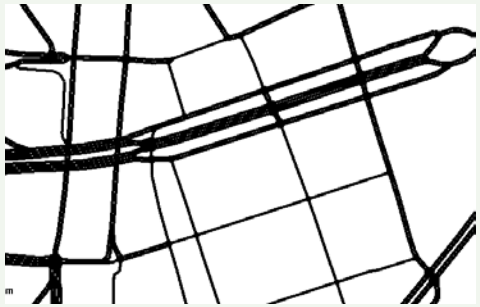
Cómo representar una solución

- Restricciones que no cumple por defecto:
  - Fases amarillas a 4
  - Ciclos > 60 y < 120
  - Promoción de olas verdes

- Otras representaciones alternativas:
  - Añadir *offset*
  - Cada cruce: tiempo de ciclo + porcentaje que ocupa cada fase
  - Reducir el número de semáforos:
    - Agrupación de cruces en grupos
    - Solo se optimiza uno del grupo
    - El resto se ponen como desplazamiento de la optimizada
  - ¿Otras?

Cómo evaluar una solución

- Necesitamos un valor numérico (o varios) que nos permita comparar soluciones
- Sistema muy complejo
  - No existen modelos matemáticos para calcularlo
  - Uso de simuladores
- SUMO (Simulator of Urban Mobility)
  - Recibe un mapa, rutas de vehículos y la configuración de los semáforos
  - Devuelve estadísticas de la simulación



SUMO

Vehículos finalizan: 30
Duración media: 120
CO2: 543452
Espera media: 10
...

Cómo evaluar una solución

- Valores devueltos:
  - Números de vehículos que llegaron a su destino
  - Duración media de las rutas
  - Emisiones (CO2, CO, NOx, PMx, HC, ...)
  - Tiempo de espera

- Objetivo a optimizar:
  - Cada uno independientemente
  - Combinación de varios
  - Varios simultáneamente
  - ¿Otros? (Olas verdes, ...)

- Dificultad
  - SUMO devuelve estadísticas de vehículos que acaban su ruta

Cómo evaluar una solución

- Valores devueltos:
  - Números de vehículos que llegaron a su destino
  - Duración media de las rutas
  - Combinación de varios
  - Varios simultáneamente
  - ¿Otros? (Olas verdes, …)

$$f_{obj} = \frac{T_{trip} + Tsw + VNRTsim}{V_R^2 + P}$$

Duración

Tiempo de espera

Vehículos que no acaban

Vehículos que acaban

Duración de las fases verdes

- Dificultad
  - SUMO devuelve estadísticas de vehículos que acaban su ruta

Datos y realismo

- Para que el sistema sea útil, los datos usados durante las simulaciones deben ser lo más realistas posibles

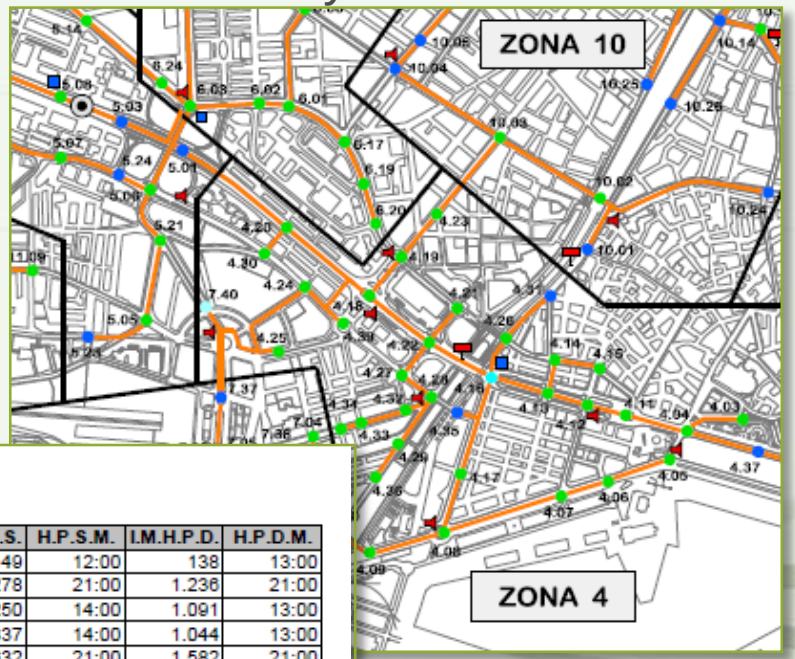
- Datos necesarios:
  - Mapa de la ciudad a optimizar
  - Situación de los semáforos y cruces semafóricos
  - Rutas de vehículos atendiendo a diferentes niveles de tráfico
  - Otras restricciones del sistema/problema

- Origen de los datos:
  - Mapas: OpenStreetMap
  - Semáforos: Centro de control de tráfico
  - Rutas: Área de Movilidad/tránsito

Datos y realismo

- Dificultades: Mapas y semáforos:
  - Mapas no completos: faltan calles, sentidos, semáforos
  - Conversión (OSM => SUMO) no del todo correcta
  - Corrección manual de los mapas. Proceso muy laborioso

- Dificultades: Rutas:
  - No suficiente detalle
  - Formato no automatizable
  - Conversión de intensidad de tráfico a rutas complejo.



Intensidad de vehículos Mayo-Agosto 2015

PM	Ubicación	I.M.D.L	I.M.D.S.	I.M.D.D.	I.M.H.P.L.	H.P.L.M.	I.M.H.P.S.	H.P.S.M.	I.M.H.P.D.	H.P.D.M.
1	Avda. Juan Sebastián Elcano - Este	2.004	1.636	1.388	169	12:00	149	12:00	138	13:00
2	Avda. Juan Sebastián Elcano - Oeste	20.383	17.890	15.186	1.465	8:00	1.278	21:00	1.236	21:00
3	Bolivia - Este	16.775	15.174	12.698	1.413	14:00	1.250	14:00	1.091	13:00
4	P.M. Pablo Ruiz Picasso - Este	19.376	15.574	12.631	1.908	14:00	1.337	14:00	1.044	13:00
5	P.M. Pablo Ruiz Picasso - Oeste	28.829	23.675	20.368	2.192	8:00	1.632	21:00	1.582	21:00
6	Pso. Reding - Este	8.528	6.767	5.296	687	14:00	554	14:00	418	13:00
7	Pso. Reding - Oeste	6.987	5.930	4.655	546	9:00	445	12:00	342	21:00
8	Victoria - Sur *	5.874	5.255	4.285	419	8:00	443	22:00	319	22:00
9	Victoria - Norte *	6.474	5.888	4.923	483	14:00	454	22:00	360	22:00
10	Túnel Alcazaba - Este	15.870	14.744	12.561	973	8:00	849	14:00	701	1:00

Algoritmos

- Implementando una propuesta
- Algoritmos evolutivos
- Multi-objetivo

Implementando una propuesta

- Código en C++
- Instancias para probar

	2 intersecciones	Málaga	París
Cruces	2	42	70
Fases	16	172	378
# vehículos	500	266	1200

Pocas fases y tráfico => más sencilla

Muchas fases y tráfico => más compleja

Implementando una propuesta

- Clase **cInstance**: manejo de los datos:
  - **Constructor**: lee los ficheros
  - **getNumberOfTLlogics()**: número de cruces
  - **getPhases()**: fases del cruce j
- Solución: **vector<unsigned>**
- Poniéndolo junto: solución aleatoria (o casi)

```
void generateSolution(vector<unsigned> &solution, const cInstance &c)
{
        solution.clear();
        vector<string> phases;
        int pos;
        for(int j = 0; j < c.getNumberOfTLlogics(); j++)
        {
                phases = c.getPhases(j);
                for(int k = 0; k < phases.size(); k++)
                {
                        if(areAllYellow(phases[k]))
                                solution.push_back(4*phases[k].size());
                        else if(isSubString(phases[k],"y",pos))
                                solution.push_back(4);
                        else
                                solution.push_back(rand()%55 + 5);
                }
        }
}
```

Implementando una propuesta

- Interacción con **sumo-wrapper**. Tres ficheros:
  - **Fichero de instancia:** instancia
  - **Fases de semáforos:** duración de las fases (fichero de números)
  - **Resultados de la simulación:** estadísticas (sexta línea => fitness)
- Poniéndolo junto: Random Search

```
void writeS
{
        ofs
        for

        f.c
}
```

```
void readFitnessFile(float &fitness)
{
        ifstream f("result.txt");
        string s;
        for(int i = 0; i < 6; i++) // skip lines
                getline(f,s);
        f >> fitness;
        f.close();
}
```

```
st cInstance &c)
```

Implementando una propuesta

```
c.read(argv[1]);

steps = atoi(argv[2]);

command = "./sumo-wrapper " + string(argv[1]) + " " + "tl.txt result.txt";

generateSolution(solution, c);
writeSolutionFile(solution, c);
system(command.c_str());
readFitnessFile(fitness);
best_sol = solution; best_fit = fitness;
cout << fitness << endl;

for(int i = 1; i < steps; i++)
{
        generateSolution(solution, c);
        writeSolutionFile(solution, c);
        system(command.c_str());
        readFitnessFile(fitness);
cout << fitness << endl;
        if(fitness < best_fit)
        {
                best_sol = solution;
                best_fit = fitness;
        }
}

cout << "Best solution: " << endl;
for(int i = 0; i < c.getTotalNumberOfPhases(); i++)
        cout << best_sol[i] << " ";
cout << endl << "Fitness: " << best_fit << endl;
```

- números)
- > fitness)

Ejercicios iniciales

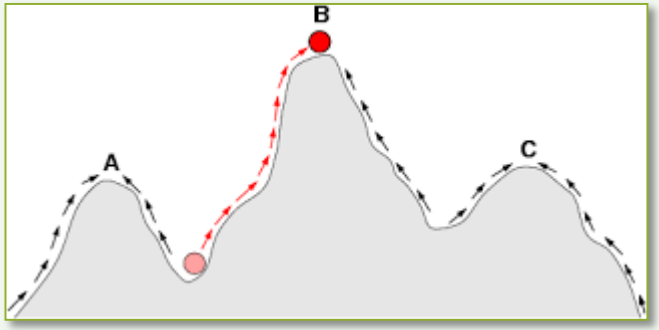
1. Descargue el código de https://github.com/GabJL/TLO
2. Compile el código:
   - cd TLO-master/code # Muévase al directorio code
   - make # Compilación
3. En el código se facilita una búsqueda aleatoria (RS). Pruébela usando la instancia de malaga:
   - cp RS sumo-wrapper .. # Desde el directorio code copie los ficheros
   - cd .. # Muévase al directorio raíz del proyecto
   - ./RS instanceFiles/malaga.txt 10 # Ejecución con 10 iteraciones
4. ¿Si hace una segunda ejecución obtenemos el mismo resultado? ¿Cómo analizar estos datos?
5. Ejecute las 3 instancias con 5 iteraciones, ¿qué parámetro cree que influye más en la duración de la ejecución?

	2 intersecciones	Málaga	París
Cruces	2	42	70
Fases	16	172	378
# vehículos	500	266	1200

Técnicas de Optimización

- Métodos de escalada (*hill climbing*)
  1. Parte de una solución inicial (generalmente aleatoria)
  2. Se analiza el vecindario
  3. Se elige el mejor vecino y se reemplaza la solución actual
  4. Se vuelve al paso 2

- Vecindario:
  - Soluciones “cercanas” en el espacio de búsqueda
  - Problemas para examinar todo el vecindario => Se examina hasta encontrar una mejor (o igual) que la actual

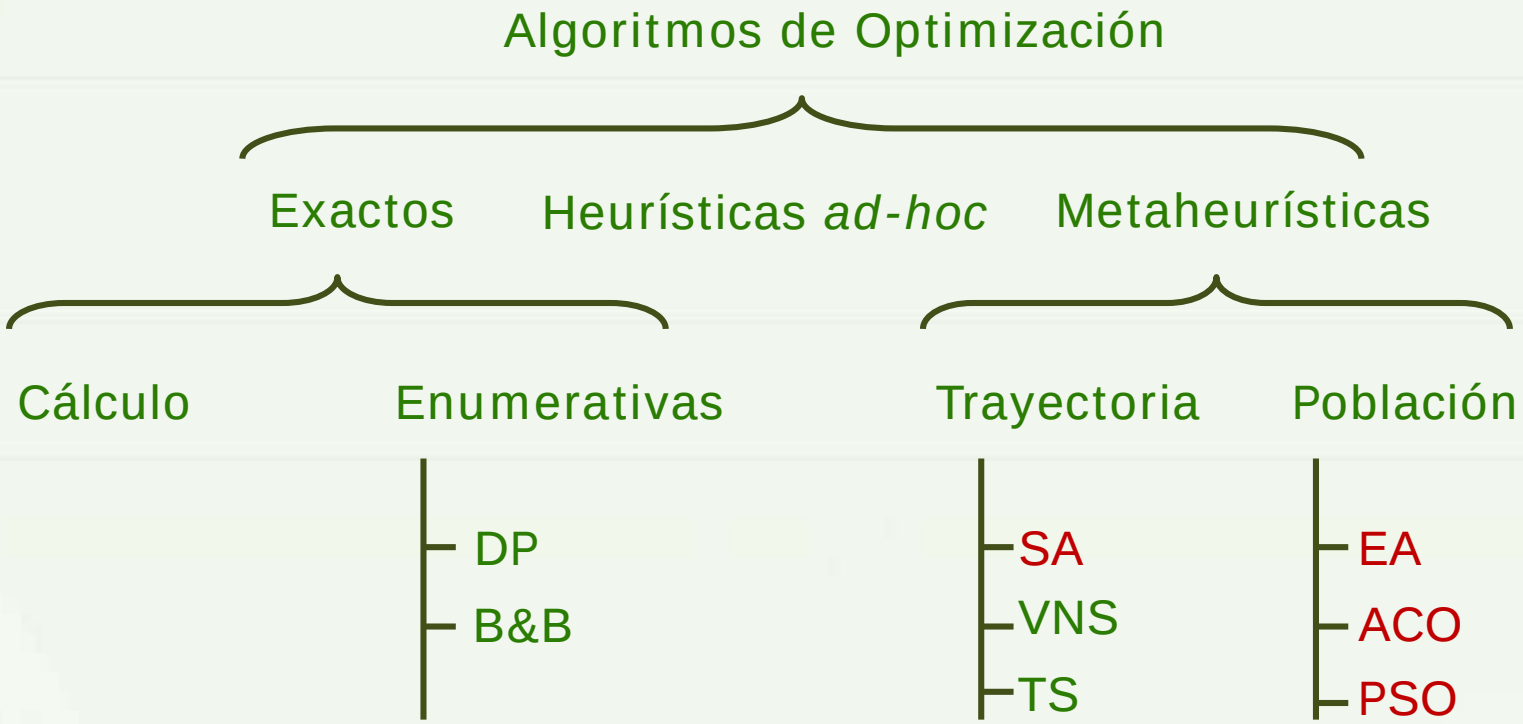


Ejercicios: métodos de escalada

Siempre trabajaremos con la instancia de malaga y 80 evaluaciones

6. ¿Qué es una solución vecina de una para este problema? Piense una operación que a partir de una solución permita obtener una solución en su vecindario.
7. Copie el código de RS a HC y modifíquelo siguiendo el siguiente esquema:
   - Cree una nueva función que a partir de una solución, calcule aleatoria una del vecindario
   - En cada iteración del algoritmo, genere un vecino de la solución actual
   - Si el vecino es mejor que la actual, reemplace la actual por el vecino
8. Híbrido RS+HC 1 (rshc1): Cree un nuevo código con el siguiente comportamiento:
   - Genere 20 soluciones aleatorias
   - Aplique HC a la mejor solución (60 evaluaciones)
9. Híbrido RS+HC 2 (rshc2): Cree un nuevo código con el siguiente comportamiento:
   - Genere una solución aleatoria
   - Aplique HC a esa solución hasta que converja (en 4 pasos no encontró un vecino mejor)
   - Reinicie la solución volviendo al inicio hasta completar el máximo de evaluaciones

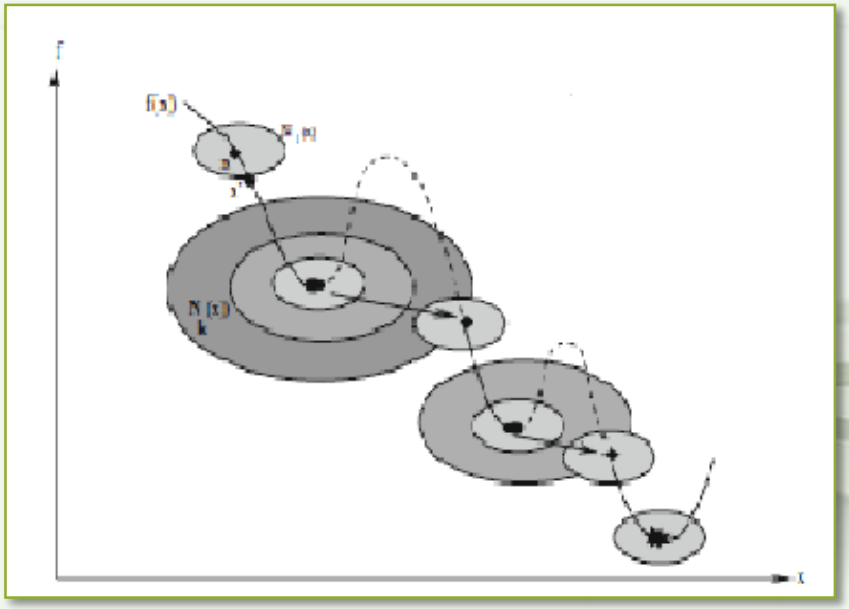
Técnicas de Optimización



Inspiradas en la naturaleza

Técnicas de Optimización

- Variable Neighborhood Search (VNS)
  1. Se definen K vecindarios (de más reducido a más amplio): V_1, V_2, …, V_k
  2. Parte de una solución inicial (generalmente aleatoria)
  3. N = 1
  4. Se busca el mejor vecino usando el vecindario V_N
     - Si es mejor que la actual, se reemplaza la actual por la nueva y N = 1
     - Si es peor, N++
  5. Se vuelve al paso 4

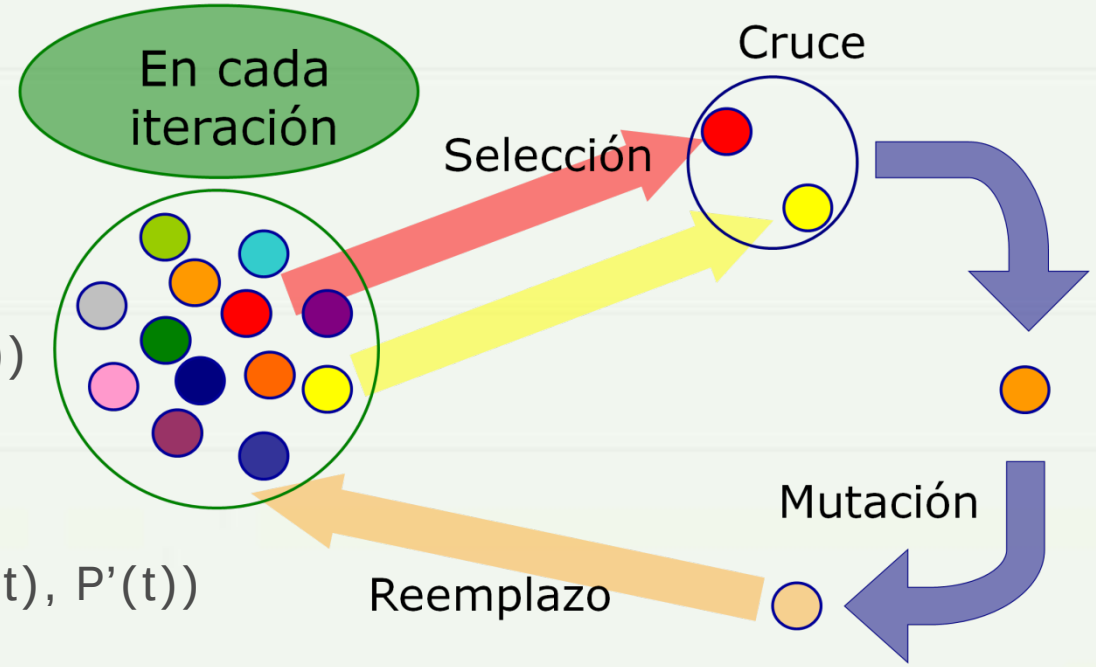


Algoritmos Evolutivos

- Basados en la evolución natural de Darwin
- Algoritmo poblacional (maneja múltiple soluciones)
- Tres pasos principales:
  - Selección
  - Reproducción (cruce y mutación)
  - Remplazo
- Múltiples familias de acuerdo a cómo realizan esos pasos:
  - Algoritmos Genéticos (GA),
  - Programación Genética (PG),
  - Estrategias Evolutivas (ES),
  - ...

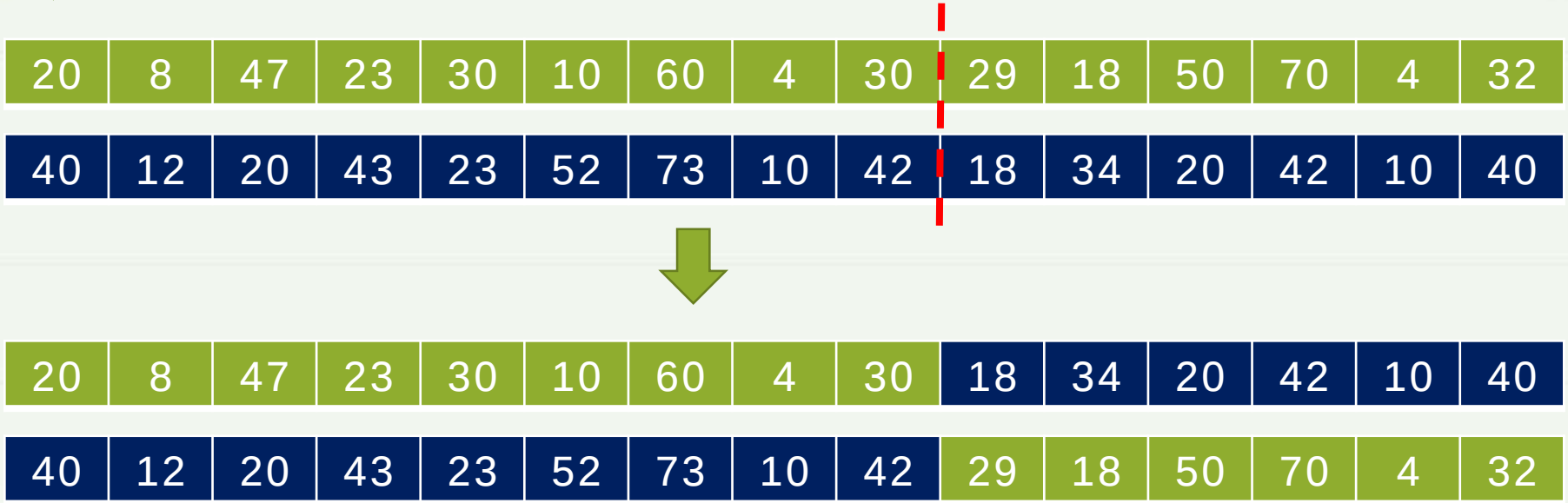
Algoritmos Evolutivos

```
t := 0
Generar(P(t))
Evaluar(P(t))
while not end condition do
        P'(t) := Seleccionar(P(t))
        P'(t) := Cruzar(P'(t))
        P'(t) := Mutar(P'(t))
        Evaluar(P'(t))
        P(t+1) := Remplazar(P(t), P'(t))
        t := t+1
end while
```

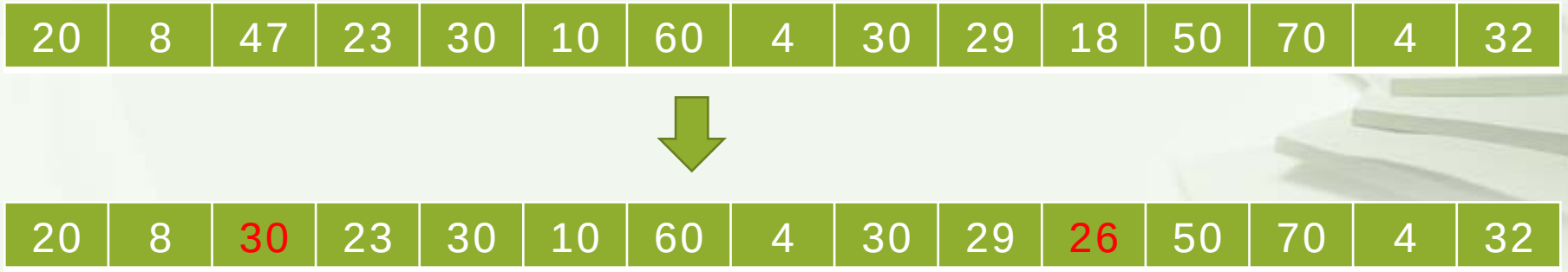


Algoritmos Evolutivos

- Recombinación/Cruce:



- Mutación:



Ejercicios: VNS y GA

Siempre trabajaremos con la instancia de malaga y 80 evaluaciones

10. Partiendo de alguno de los anteriores implemente un VNS básico:
    - Al igual que en uno anterior, considere que no hay vecinos mejores si en 3 pasos no fue capaz de generar uno mejor

11. En el código facilitado hay un ssGA implementado a falta de los operaciones de mutación y recombinación. Complételos:
    - Como mutación use algún esquema usado en los anteriores
    - Como recombinación implemente el cruce en un punto (solo debe generar una solución ya que usamos un estado estacionario)

12. Modifique la recombinación para que tenga en cuenta de no “cortar” en medio de la planificación de un semáforo

13. Piense e implemente algún otro operador de recombinación no basado en puntos de corte.

Ajuste del control semafórico automático.

Transporte y Tráfico