XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

# ESTRATEGIAS PARA EL CONTROL DE SEÑALES DE TRAFICO URBANO

SERGIO NICOLAS DI GERONIMO
MARIBEL FERNANDEZ ROMERO

E.S.L.A.I

SUMARIO

Este trabajo examina el problema de optimización del flujo de vehículos en una red de tráfico urbano, aplicando control "off-line" y "on-line".

En la primera parte se estudian los problemas de la hora pico. Se presentan modelos matemáticos y se desarrollan estrategias de control, primero para cruces simples de calles y luego para redes urbanas. En la segunda parte se realiza un análisis similar para las horas de tráfico normal.

Por último se presenta una generalización de los modelos y estrategias propuestas, considerando cruces de avenidas y calles de varios carriles.

SERGIO NICOLAS DI GERONIMO _ 5º año de la Licenciatura en Informática - E.S.L.A.I. - Argentina, dirección : E.S.L.A.I. , casilla de correo 3193 - (1000) Correo Central, Bs. As. Argentina, teléfonos: (021) 870216 - (0223)3307

MARIBEL FERNANDEZ ROMERO _ 5º año de la Licenciatura en Informática - E.S.L.A.I. - Argentina, dirección : E.S.L.A.I. , casilla de correo 3193 - (1000) Correo Central, Bs. As. Argentina, teléfonos: (021) 870216 - (0223)3307

## 1_INTRODUCCION

Los automóviles son una gran fuente de contaminación en las ciudades. Por eso se intenta mejorar el flujo de vehículos, ya que esto decrementa el desgaste de los mismos y reduce la contaminación ambiental. Se pueden conseguir mejoras en el flujo, decrementando el tiempo de espera en los cruces con semáforos. Hasta el presente los esquemas utilizados para computar secuencias de señales de control, se basan en datos históricos ("off-line"), por lo cual no pueden manejar alteraciones imprevistas en el tránsito. Sin embargo, existen facilidades para monitorear en tiempo real la mayoría de las variables involucradas en el control de tráfico, lo que hace pensar que el control "on-line" podría disminuir las demoras.

Desde el punto de vista del control de tráfico, hay dos problemas diferentes:

* la hora "pico" (período sobresaturado): Aquella en que en la mayoría de las intersecciones con semáforos, las colas de vehículos en espera no desaparecen al finalizar la luz verde.
* la hora "no pico" (período no saturado): Aquella en que en la mayoría de los cruces con semáforos, no quedan vehículos al finalizar la luz verde.

El primero es más crítico, pues es cuando los servicios están sobrecargados y las demoras son muy grandes. En este sentido el problema del período no saturado es menos importante.

En este trabajo se analizan los dos problemas, considerando primero cruces simples y luego su generalización para casos más complicado. A partir de modelos matemáticos, que representan los cruces en los distintos períodos, se desarrollan estrategias de control que minimizan las demoras, tanto para el período sobresaturado como para el no saturado.

El siguiente cuadro esquematiza el análisis del problema hecho a lo largo de este trabajo:

* Periodo Sobresaturado _ Modelo
  _ Estrategias de control off-line y on-line

* Periodo No saturado _ Modelo
  _ Estrategias de control on-line

En todos los casos se analizó primero un cruce simple y luego una red.

Además de lo esquematizado en el cuadro anterior se abordó una generalización del modelo para avenidas y calles con varios carriles.

## 2 _ PERIODO SOBRESATURADO

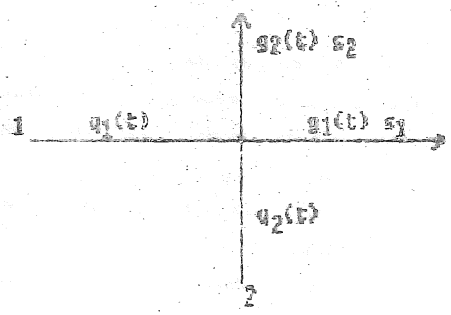
Se llama **período de tráfico sobresaturado** (o simplemente período sobresaturado) a las horas en que en la mayoría de los cruces señalizados, las colas de vehículos en espera de luz verde son tan grandes que el tiempo que dura la misma no es suficiente para que todos crucen (las colas de espera no desaparecen al finalizar la luz verde).

## 2# _ MODELOS

### CRUCE SIMPLE SOBRESATURADO

Bajo la hipótesis de que las colas de espera no desaparecen al finalizar la luz verde, se propone un modelo matemático para este tipo de cruces. [SIN-73]

Por simplicidad se analizan primero cruces de calles de una sola mano, como el que muestra la figura:




$q_i(t)$ indica la tasa de arribo de vehículos (número de vehículos por minuto) en la dirección i

$s_i$ flujo de saturación en la dirección i: máximo número de vehículos por ciclo que pueden pasar a través de la intersección en la dirección i, si se dispone del máximo de luz verde (todo el ciclo). Es una constante para todos los ciclos.

ciclo comprende desde que se prende la luz verde en una dirección, hasta que lo vuelve a hacer en la misma dirección.

C es la duración del ciclo. Se asume constante (Esta suposición se basa en el experimento de Webster [SIN-73] que mostró que variar la duración del ciclo no disminuye las demoras).

$$C = G_1 + G_2 + L$$

donde

$G_i$ es el tiempo de luz verde en la dirección i

L es la suma del tiempo de luz amarilla en ambas direcciones

$g_i(t)$ es la tasa promedio de salida de vehículos por ciclo, en la dirección i (o sea: número de vehículos por C)

Las variables de control $u_i(t)$ están definidas como la fracción de luz verde en la dirección i.

$$u_i(t) = G_i / C \qquad i=1,2$$

Por ser el cruce sobresaturado las colas nunca desaparecen, siempre hay vehículos para pasar, por lo tanto es válida la relación:

$$u_i(t) = g_i(t) / s_i \qquad i=1,2$$

Por razones de seguridad es necesario tener un mínimo y un máximo de luz verde por ciclo, lo cual impone restricciones a las variables de control:

(1) $U_i <= u_i(t) <= V_i \qquad i=1,2$

$0 < U_i, V_i < 1$ puesto que $V_i = (\text{Max } G_i) / C$

$U_i = (\text{Min } G_i) / C$

Un resultado que se obtiene a partir de la definición de las variables de control es el siguiente:

(2) $u_1 + u_2 = (G_1 + G_2) / C = (C - L) / C = 1 - L / C = EG$ ("efective green")

De acuerdo a esto basta con tener una sola variable de control por intersección. (Si $u_1 = u$ entonces $u_2 = EG - u$)

Como las demoras están dadas por el tiempo pasado en cola se puede representar el estado del cruce en cada ciclo por las longitudes de las colas en cada dirección. El vector de estado X se define como: $X(k) = (X_1(k), X2(k))$

donde $X_i(k)$ es la longitud de la cola en la dirección i, en el ciclo k.

La evolución dinámica de las colas se describe por la ecuación:

$$X_i(k+1) = X_i(k) + q_i(k) - g_i(k) \qquad i=1,2$$

donde $g_i(k)$ está definida como antes (tasa de salidas en el ciclo k)

$q_i(k)$ es la cantidad de arribos en el ciclo k.

Como $g_i(k) = s_i * u_i(k)$ entonces:

(3) $X_i(k+1) = X_i(k) + q_i(k) - (s_i * u_i(k))$

Además se imponen restricciones sobre la longitud de las colas:

(4) $0 < X_i(k) < X_i \qquad i=1,2.$

## OPTIMIZACION EN UN CRUCE SIMPLE SOBRESATURADO

Como el objetivo de cualquier esquema de control en la hora pico es reducir las demoras, y éstas se miden por el tiempo de espera en cola, la suma de las longitudes de las colas es una medida apropiada de la performance de una secuencia de control. Por esto razón, se tratará de minimizar la sumatoria de longitudes de cola. Un objetivo secundario es el grado de utilización de los servicios. De acuerdo a esto la función objetivo es:

$$(5) \qquad I = \min_{u_1, u_2} \sum_{k=1}^{N} \sum_{i=1}^{2} K1 * X_i(k) + K2 * s_i * (u_i(k) - V_i)$$

donde

N es la cantidad de ciclos a optimizar

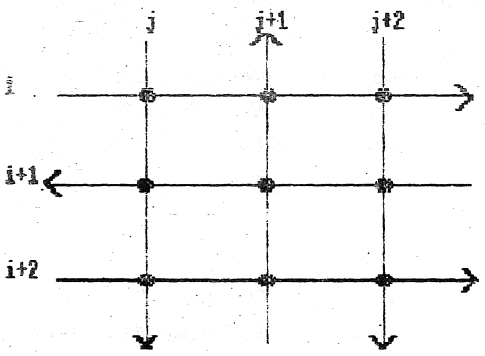
K1,K2 son factores de ponderación (en este caso K2=0, puesto que para cruces sobresaturados el desperdicio es nulo)

$V_i$ es el máximo porcentaje de luz verde en la dirección i

El problema consiste en encontrar la secuencia de valores para las variables de control $u_1$ y $u_2$ que satisfagan (5), cumpliendo la ecuación (3) y teniendo en cuenta las restricciones dadas por las desigualdades (1), (2) y (4).

## REDES SOBRESATURADAS

Las redes están formadas por intersecciones y calles que las interconectan, en una cascada de dos dimensiones. Se asume que los cruces son de calles de una mano y que los vehículos no doblan. Las calles que interconectan los cruces son modeladas como elementos de demora.




Como muestra la figura, la salida de la calle i del cruce (i, j) coincide con la entrada del cruce siguiente, luego de p períodos de demora ( p >= 1).

$$g_{i\,j}(k) = s_{i\,j} * u_{i\,j}(k) \quad \text{luego} \quad q_{i\,j+1}(k+p) = s_{i\,j} * u_{i\,j}(k)$$

El estado de la red en cada ciclo estará representado por una matriz, en la cual la posición i j representa el estado del cruce (i, j) en dicho ciclo, por medio de las longitudes de las colas en cada dirección de la intersección. Dicha matriz tiene la siguiente forma:

$$X(k) = \begin{bmatrix} X_{1,1}(k) & \cdots & X_{1,n}(k) \\ & \cdots & \\ X_{n,1}(k) & \cdots & X_{n,n}(k) \end{bmatrix}$$

donde

$X_{i\,j}(k) = (X_{i\,j(z)}(k),\ X_{i\,j(j)}(k))$

siendo

$X_{i\,j(z)}(k)$ la longitud de la cola en la intersección (i, j) en la dirección z, en el ciclo k ($z=i,j$).

La evolución dinámica de las colas está dada por la siguiente ecuación:

(6) $X(k+1) = (X(k) - (S \# U(k)) + C) [+] ((B_1 \# U(k-1)) + \ldots\ldots + (B_p \# U(k-p)))$ $k=0,1,\ldots N-1$

donde

$B_j$ $j=1, \ldots p$ son matrices de ponderación de control

S matriz en la cual cada posición indica el flujo de saturación de la calle de la intersección correspondiente

U(k) es la matriz de control (cada elemento es un par $u_i,u_j$ de control para cada dirección de la intersección (i, j) )

U(k-j) $j=1, \ldots p$ son usados para simular las demoras por las calles que interconectan los cruces

C matriz de entradas desde fuera de la red

+ es la suma común de matrices, componente a componente

# es un producto de matrices componente a componente

[+] es una suma de matrices pero con la siguiente salvedad:

sea $A = B [+] C$ luego

$a_{i\,j} = b_{i\,j} + c_{i\,j-1}$ si hay una calle con sentido (i, j-1) (i, j) ó

$a_{i\,j} = b_{i\,j} + c_{i\,j+1}$ si hay una calle con sentido (i, j+1) (i, j)

Esto es teniendo en cuenta los cruces en la dirección i. Lo mismo vale para los cruces en la dirección j.

Las matrices de estado y de control están sujetas a las siguientes restricciones:

(7) $\underline{U} <= X(k) <= \overline{X}$

(8) $\underline{U} <= U(k) <= \overline{Y}$

OPTIMIZACION EN REDES SOBRESATURADAS

Al igual que en el caso de un único cruce, el costo está determinado por la suma de las longitudes de las colas en cada intersección. La función de costo es:

$$(9) \qquad I = \min_{U(0),\ldots,U(N-1)} \sum_{k=0}^{N-1} \| X(k) \|$$

donde N es la cantidad de ciclos que se van a optimizar.

Luego, el problema de optimización en una red sobresaturada consiste en resolver la ecuación (9) sujeta a las restricciones (6), (7) y (8).

Observación: Desarrollando la ecuación (9), es posible cancelar algunos términos, pero la función resultante no es más fácil de usar, por lo tanto se trabajará con la función objetivo (9).

## 2.2 _ ESTRATEGIAS DE CONTROL

### 2.2.1 _ CONTROL OFF-LINE

#### CRUCE SIMPLE SOBRESATURADO

El problema consiste en encontrar la secuencia óptima de tiempos de luz verde para N ciclos. El objetivo es:

$$\min_{u_1,u_2} \sum_{i=1}^{2} \sum_{k=1}^{N} X_i(k)$$

sujeto a las restricciones dadas por las ecuaciones (1), (3), (4).

Se verá si es posible representar el problema como búsqueda de caminos en un grafo. Primero es necesario analizar si es un problema discreto, para ello se han realizado medidas de

_duración del ciclo de un semáforo: 60 a 70 segundos.

_duración de la luz amarilla: 3 a 4 segundos.

_cotas del tiempo de luz verde: mínimo 20 segundos, máximo 40 seg.

_tiempo que tarda un auto en cruzar: 4 a 5 segundos.

De éstas se deduce que es posible representarlo con un grafo. La cantidad de sucesores de un nodo está dada por los posibles tiempos de luz verde en cada dirección, y debido a que un coche tarda 4 o 5 segundos en cruzar, los tiempos de luz verde pueden variar "discretamente" ( de a 3 o 4 segundos). Entonces, la cantidad de sucesores es 6 o 7 (según se varíe de a 3 o de a 4 segundos).

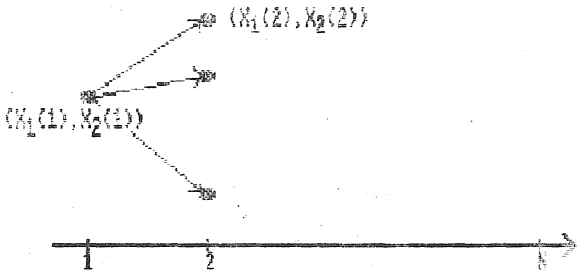
Por lo tanto el problema de encontrar la secuencia óptima de tiempos de luz verde puede representarse con un grafo en el que cada nodo tiene 6 sucesores, y los nodos están dispuestos en "camadas" (o etapas) correspondientes a los ciclos. El número de etapas es:

(duración del período de sobresaturación) / C

DESCRIPCION DEL GRAFO

Cada nodo representa un estado $(X_1(k),X_2(k))$. Cada arco representa un control $(u_1(k),u_2(k))$, o solamente $u_1(k)$, ya que $u_2(k)$ se calcula a partir de $u_1(k)$ con la ecuación (2).

Hay un arco por cada vector de control factible ¿que cumpla las restricciones). El estado al cual pase está dado por la ecuación (3).

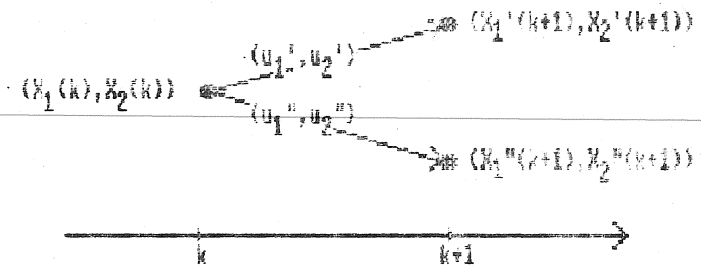



PROPIEDADES

1) No hay arcos múltiples.

Dem.

Se quiere probar que con dos controles distintos no se puede pasar al mismo estado.




Supongamos $(u_1',u_2')$, $(u_1'',u_2'')$ distintos.

$q_i(k)$ = cantidad de arribos en el período $k$ en la dirección $i$, por lo tanto es independiente de $u'$ y $u''$

De acuerdo a la ecuación (3)

$$X_i'(k+1) = X_i(k) + q_i(k) - s_i * u_i'(k)$$

$$X_i''(k+1) = X_i(k) + q_i(k) - s_i * u_i''(k)$$

Por lo tanto $u_i'$ *distinto de* $u_i''$ implica $X_i'$ *distinto de* $X_i''$, lo cual demuestra que no hay arcos múltiples. LQQD.

2) El grafo no es un árbol.

Dem.

Es posible llegar al mismo estado por dos caminos distintos (con dos secuencias de control distintas).

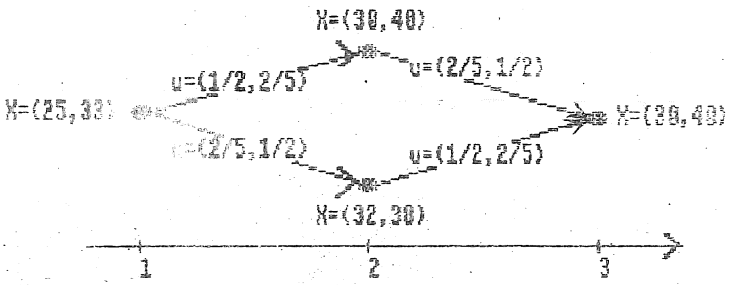
Ej.

C = 60 seg, L = 6 seg, $s_1 = s_2 = 20$, $20/60 < u_1, u_2 < 40/60$

q(1) = (15,15)

q(2) = (8,10)

EG = 1 - L/C = 9/10, $u_2 = 9/10 - u_1$




Los valores verifican $X_i(k+1) = X_i(k) + q_i(k) - s_i * u_i(k)$ :

Pasaje de 1 a 2

(30,40) = (25,33) + (15,15) - (20,20)*(1/2,2/5)

(32,38) = (25,33) + (15,15) - (20,20)*(2/5,1/2)

Pasaje de 2 a 3

(30,40) = (30,40) + (8,10) - (20,20)*(2/5,1/2)

(30,40) = (32,38) + (8,10) - (20,20)*(1/2,2/5)

LQQD.

**Algoritmo**

El problema consiste en encontrar el camino mínimo en el grafo. Para esto se necesita la lista de arribos para cada ciclo (k=1,...,N), los cuales se suponen conocidos a priori (se usan datos históricos, o valores dados por alguna distribución de probabilidades).

Para hallar el óptimo como camino en el grafo, se aplican técnicas de programación heurística, las cuales se basan en el uso de información (heurística) que facilite la búsqueda. Concretamente se utiliza el algoritmo de búsqueda de camino mínimo en grafos A* [NIL-71]. La heurística $h$ es un estimado del costo futuro del camino:

$$h(n_k) = \sum_{i=1}^{2} [(N-k) * X_i(k) + (\sum_{j=k}^{N-1} q_i(j) * (N-j)) - s_i * v_i * (N-k) * (N-k+1) / 2]$$

donde

$n_k$ es un nodo del grafo en la etapa k.

$q_i(k)$ son los arribos esperados en la dirección i para el ciclo k

$V_i$ es el máximo tiempo de luz verde que se le puede dar a la dirección i, de acuerdo a la restricción (1).

El estimado de un nodo de la etapa k es la suma de los ciclos en las etapas restantes suponiendo que a dicho cruce se le da el máximo de luz verde en cada dirección.

Un estimado $h$ se dice admisible si $h(n) <= h(n)$ para todo n, siendo h(n) el costo real del camino a partir del nodo n. Es conveniente elegir un estimado admisible puesto que en este caso el algoritmo A* termina encontrando un camino óptimo, si existe [NIL-71].

El estimado $h$ es admisible (es subestimado).

Dem.

$$h(n_k) = X_1(k+1) + \dots + X_1(N) + X_2(k+1) + \dots + X_2(N) =$$

$$= (N-k) * X_1(k) + \sum_{j=k}^{N-1} q_1(j) * (N-j) - \sum_{j=k}^{N-1} s_1 * u_1(j) * (N-j) + (N-k) * X_2(k) +$$

$$+ \sum_{j=k}^{N-1} q_2(j) * (N-j) - \sum_{j=k}^{N-1} s_2 * u_2(j) * (N-j)$$

$$h(n_k) - h(n_k) = s_1 * V_1 * \sum_{j=k}^{N-1} (N-j) + s_2 * V_2 * \sum_{j=k}^{N-1} (N-j) -$$

$$\sum_{j=k}^{N-1} s_1 * u_1(j) * (N-j) - \sum_{j=k}^{N-1} s_2 * u_2(j) * (N-j) =$$

$$= \sum_{j=k}^{N-1} (N-j) * s_1 * (V_1 - u_1(j)) + \sum_{j=k}^{N-1} (N-j) * s_2 * (V_2 - u_2(j))$$

Por la restricción (1), $V_1 - u_1(j) >= 0$ para todo j

$V_2 - u_2(j) >= 0$ para todo j

por lo tanto

$h(n_k) - h(n_k) >= 0$, entonces $h(n_k) <= h(n_k)$.

LQGD.

Un estimado es consistente si es admisible y cumple $h(m) - h(n) <= K(m,n)$ para todo par de nodos m, n; siendo K (m,n) el costo del camino entre m y n (*camino* si no hay camino ). Es conveniente que el estimado sea consistente pues en este caso se minimiza el número de operaciones realizadas por el algoritmo A*, ya que no se reabren nodos [NIL-71].

El estimado *h* es consistente.

Dem.

Hay que probar: $h(n_k) - h(n_j) < K(n_k,n_j)$, k *distinto de* j puesto que entre dos nodos de la misma etapa no hay camino.

$K(n_k,n_j) = X(k+1) + \ldots + X(j)$

$h(n_k) = X'(k+1) + \ldots + X'(j) + \ldots + X'(N)$

$h(n_j) = X''(j+1) + \ldots + X''(N)$

La diferencia entre X' y X'' está en la forma de calcularlos:

$$X'(i+1) = X'(i) + q_i(i) - s * V$$

$$X''(i+1) = X(i) + q_i(i) - s * V$$

Como $X'(i) < X(i)$ (porque el valor estimado es menor que el real) entonces

$X'(i+1) <= X(i) - s * V$ luego

$$h(n_k) - h(n_j) <= X'(k+1) + \ldots + X'(j)$$

Además

$$X(k+1) + \ldots + X(j) < X(k+1) + \ldots + X(j) = K(n_k,n_j)$$

entonces

$$h(n_k) - h(n_j) <= K(n_k,n_j)$$

luego es consistente.LQQD.

Observación: El estimado *h* de un nodo se puede calcular fácilmente a partir del de su predecesor (lo cual es importante para la eficiencia del algoritmo):

Sean $n_k$ y $n_{k+1}$


$h(n_k) = X'(k+1) + X'(k+2) + \ldots + X'(N)$

$h(n_{k+1}) = X''(k+2) + \ldots + X''(N)$

Entonces conociendo el estimado de $n_k$ calculamos el de su sucesor de la siguiente forma:

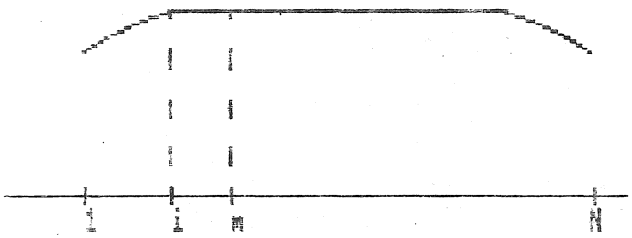
$$h(n_{k+1}) = h(n_k) - \sum_{i=1}^{2} (X_i(k) + q_i(k) - s_i * V_i) + \sum_{i=1}^{2} s_i * (V_i - u_i(k)) * (N-k-1)$$

EFICIENCIA DEL ALGORITMO

Es necesario almacenar los nodos abiertos. Si tenemos 60 etapas, en el peor caso el número de abiertos podría ser del orden de 6**60, por lo tanto hay que buscar alguna simplificación

Hipótesis sobre el comportamiento del tránsito:

En la hora saturada el comportamiento del tráfico varía al principio hasta alcanzar un equilibrio, que se pierde hacia el final de la hora saturada.




Entonces, de acuerdo al comportamiento anterior, se puede buscar el camino óptimo hasta la mitad de la hora saturada, y luego repetir la secuencia de controles óptima, en sentido inverso en la otra mitad del período. Hay dos opciones:

a) Buscar una solución óptima hasta la etapa m (m <= N/2). Repetir la secuencia de los últimos m-i controles hasta llegar a la etapa N-i. Finalmente poner los i primeros controles en orden inverso, desde la etapa N-i+1 hasta la N. (Se elige m tal que la cantidad de nodos abiertos sea manejable, según la capacidad de la máquina disponible).

b) Asignar todo el espacio de memoria disponible, y si en algún momento no es suficiente, se podan los nodos menos prometedores. Si el estimado es bueno, al eliminar los menos prometedores, la solución que se obtiene estará cercana al óptimo.

En ambos casos se está reduciendo el número de caminos a explorar, pero bajo los supuestos realizados, la solución obtenida estará muy cerca de la óptima.

RED SOBRESATURADA

En una red de n intersecciones, como hay 6 o 7 posibles valores de u para cada cruce, existen 6**n controles posibles para la red, lo que se traduce en esa cantidad de sucesores para cada nodo.

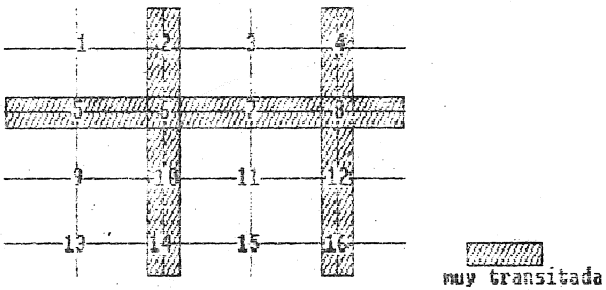
El problema en este caso también se puede representar por un grafo, pero antes de aplicar algoritmos de búsqueda de caminos es necesario reducir la cantidad de sucesores.

Una manera de lograr reducir el grafo es dividir la red en grupos de cruces, y asignarle a todos los cruces de un grupo el mismo vector de control. Hay varios criterios posibles para agrupar intersecciones, por ejemplo:

- igual número de cruces
- cruces con arribos y salidas similares
- cruces con restricciones similares (en cuanto a longitud de cola)

Como a todos los cruces del grupo se les va a asignar el mismo vector de control, el criterio de agrupación más conveniente es el segundo.

Ej:




Los grupos para esta red son

- ( 6, 8 )
- ( 2, 4, 5, 7, 10, 12, 14, 16 )
- ( 1, 3, 9, 11, 13, 15)

De 6**16 vectores posibles se redujo a 6**3.

Sabiendo que con grupos grandes se pierden muchas soluciones, podría ser conveniente subdividir los grupos grandes (que aparecen cuando hay muchos cruces con arribos y salidas similares).

Una vez realizada la división en grupos, hay distintas formas de resolver el problema:

**a)** Aplicar aproximaciones sucesivas **[BRO-87]** sobre los grupos: para la elección de una solución inicial hay varias opciones, una posibilidad sería dar a todos los grupos el vector de control que corresponde al valor mínimo de luz verde.

A partir de la solución inicial, se busca la secuencia de control ( u (k) ) óptima para un grupo, manteniendo constantes los controles de los otros grupos. ( Para encontrar el óptimo de cada grupo se busca el camino mínimo en un grafo en el que cada nodo tiene 6 sucesores, debido a que se le asigna la misma secuencia de control a todos los cruces)

**b)** Basándonos en la Hipótesis de comportamiento del tránsito en la hora saturada, vista en la solución para un cruce simple sobresaturado se hace lo siguiente: Buscar una solución hasta la etapa m. Repetir la secuencia de m - i controles hasta llegar a la etapa N-i. Luego poner los i primeros controles en sentido inverso.

Buscar la solución hasta la etapa m, con h grupos, es buscar el camino óptimo en un grafo donde cada nodo representa el estado de los h grupos de la red, con 6**h sucesores por nodo, y m etapas.

El número de nodos abiertos en el algoritmo puede ser en el peor caso del orden de (6**h)**m. Por lo tanto hay que elegir h y m convenientes de acuerdo a la disponibilidad de memoria y tiempo de máquina, pero sin perder de vista que cuanto menor sea la cantidad de grupos, más posibilidades hay de perder el óptimo.

La solución para el caso **(b)** sería:

_ Elegir N y m.

_ Buscar el camino mínimo en el grafo, para m etapas (dándole a todos los cruces del grupo el mismo valor de u).

_ Repetir los últimos m - 1 vectores de control en las siguientes etapas (hasta la N-1)

_ Dar los controles hasta el ciclo N en sentido inverso a los encontrados para las etapas 1 a i

Se pueden combinar los dos métodos, aplicando (b) en **(a)**.

2.2.2 _ CONTROL ON-LINE

CRUCE SIMPLE SOBRESATURADO

Para resolver el problema en tiempo real se deben tratar las etapas en forma independiente, porque no se puede conocer los $q_i$ futuros. Por esta razón se debe hallar la mejor solución para una etapa dada, optimizando el tiempo de espera en esa etapa (ciclo).

El modelo visto anteriormente describe la evolución de las colas en el cruce mediante la siguiente ecuación

$$X_i(k+1) = X_i(k) + q_i(k) - g_i(k)$$

siendo $g_i(k) = s_i * u_i(k)$ .

El objetivo es

**(10)** $$I = \min_{u_1,u_2} \sum_{i=1}^{2} X_i(k+1)$$

Como en cada ciclo hay 6 o 7 posibles valores para $G_i$ (ya que $u_i = G_i / C$ ), hallar los valores $u_i$ que satisfacen la ecuación **(10)** consiste en buscar el mínimo para 6 o 7 valores.

**Algoritmo**

Para obtener el valor $u_i(k)$ (i=1,2) óptimo, teniendo como dato el $q_i(k)$ y $X(k)$, se calcula la sumatoria de la ecuación **(10)** para cada posible valor de $u_i(k)$. El control óptimo es el que satisface **(10)**, y será aplicado en el ciclo k+1; durante éste se calcula el control óptimo para el k+2 y así sucesivamente.

Nota: No se puede calcular el control óptimo del ciclo k cuando se está en el ciclo k. Lo que se hace es calcular el óptimo para el ciclo k+1, utilizando el $q_i(k-1)$ ( debido a que $q_i(k)$ recién se conoce al final del ciclo k). Este pequeño desfasaje no es significativo en el problema.

REDES SOBRESATURADAS

Al trabajar on-line, la optimización de todos los cruces se va a hacer etapa por etapa, debido a que como en el caso anterior, no se dispone de los $q_i$ futuros. Por esta razón la función

objetivo es

(11) $$I = \min \sum_{i,j} [X_{1(i,j)}(k+1) + X_{2(i,j)}(k+1)]$$

Suponiendo una red de n cruces, la cantidad de vectores de control posibles es 6**n (6 por cada cruce). Luego, si n es grande, resulta ineficiente aplicar el algoritmo dado para un cruce simple on-line.

Por lo tanto la idea es disminuir la cantidad de vectores de control haciendo un análisis por grupo, y utilizar el método de aproximaciones sucesivas, asignando a todos los cruces del grupo el mismo vector de control.

Los criterios para dividir en grupos son los mismos que se dieron en la solución para red sobresaturada con control off-line.

Para aplicar aproximaciones sucesivas se utiliza como solución inicial, por ejemplo, los valores medios de las variables de control.

**Algoritmo**

Partiendo de la solución inicial, trabajar sobre cada grupo tomando de los vectores de control que cumplan las restricciones de ese grupo aquel que satisface (11). (Se mantienen constantes los otros grupos). Para esto hay que calcular la matriz de estado del ciclo k+1, (X (k+1) ), para cada uno de los vectores de control, y luego sumar los elementos de la matriz.

Para calcular la matriz se necesita

$q_{ij}$ (k) entradas en todos los cruces de la red

$q_{ij}$ (k-1) salidas en todos los cruces de la red que es igual a $s * u_{ij}$ (k-1)

Se necesitan los últimos r vectores de control de cada cruce, siendo r la demora más grande en la red.

El algoritmo puede terminar por dos razones:

- porque se terminó el tiempo disponible para hallar la solución (no olvidemos que estamos trabajando en tiempo real)

- porque se llegó a un equilibrio (no hay diferencia entre dos iteraciones). Si todavía hay tiempo se puede mejorar la solución con grupos más pequeños.

Nota: Al buscar una solución por grupos se corre el riesgo de perder el óptimo, pero con aproximaciones sucesivas se obtiene rápidamente una solución aceptable.

Se utiliza el método de aproximaciones sucesivas porque es un problema en tiempo real y por lo tanto se necesita tener rapidamente una buena solución para entregar, y si hay tiempo se puede seguir mejorándola.

## 3 _ PERIODO NO SATURADO

### 3.1 _ MODELO

### INTERSECCION SIMPLE NO SATURADA

Se entiende por intersección no saturada aquella en la cual al final de la luz verde no queda cola.

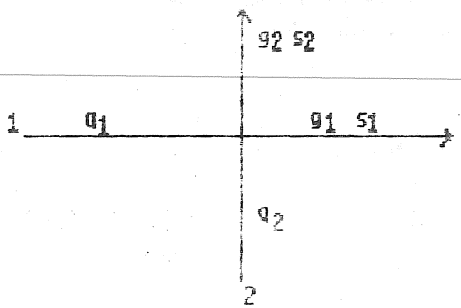
El comportamiento del tráfico en una intersección no saturada es bien conocido: cuando en una de las calles se prende la luz roja, se empiezan a formar colas. Cuando cambia a verde, las colas comienzan a disminuir hasta que desaparecen, y luego empiezan a fluir libremente los vehículos que van llegando a la intersección, hasta que se termina la luz verde, momento en que nuevamente comienzan a formarse colas.

Es el tiempo pasado en cola durante el ciclo lo que causa las demoras entonces será necesario medir en distintos momentos durante el ciclo, la longitud de las colas (por ejemplo medirlas cada segundo dentro del ciclo).

En este caso no es válido ver las llegadas y salidas como un promedio por ciclo, es necesario ver las variaciones segundo a segundo dentro del ciclo.

Debido a que en todos los ciclos al terminar la luz verde no hay cola, los ciclos se pueden analizar independientemente, por lo cual el problema que se plantea es la optimización de las longitudes de cola durante un ciclo.

Se utiliza el siguiente modelo **[SIN-73]** para una intersección simple no saturada:




Sean $q_1, q_2$ flujo de entrada en la dirección 1 y 2
$g_1, g_2$ flujo de salida en la dirección 1 y 2
$X_1, X_2$ longitud de la cola en la dirección 1 y 2
$s_1, s_2$ flujo de saturación en la dirección 1 y 2
} en cada segundo

Llamamos "períodos" a los intervalos que resultan de subdividir el ciclo en segundos.

La evolución de las colas esta regida por la siguiente ecuación:

(12) $$X_i(k+1) = X_i(k) + q_i(k) - g_i(k) \qquad i=1,2$$

donde k indica el período dentro del ciclo.

Sea $N_i$ el número de períodos durante la luz verde en la dirección i, o sea durante $G_i$. La salida se define como:

$$(13)\qquad \begin{cases} g_i(k) = s_i(k) & \text{sii } X_i(k) > 0 \text{ y } k < N_i \\ g_i(k) = q_i(k-d) & \text{sii } X_i(k) = 0 \text{ y } k < N_i \\ g_i(k) = 0 & \text{en otro caso} \end{cases} \qquad i = 1,2$$

donde $s_i(k)$ es la cantidad de vehículos que pasan en un período (flujo de saturación) en la dirección i

d es el retardo en el cruce

En otras palabras:

la salida es _ 0 si hay luz roja ($k > N_i$)

_ $s_i(k)$ si hay cola y luz verde

_ $q_i(k-d)$ si hay luz verde y no hay cola (salen a medida que llegan)

## OPTIMIZACION PARA UN CRUCE SIMPLE NO SATURADO

Igual que en el caso sobresaturado, el costo esta determinado por la suma de las longitudes de las colas en el cruce. Pero en este caso se consideran solo las que se producen durante un ciclo.

La función objetivo es:

$$(14)\qquad i = \min_{u_i} \sum_{i=1}^{2} \sum_{k=2}^{N} X_i(k)$$

donde N es la cantidad de períodos del ciclo.

Minimizar tiempos de espera equivale a minimizar longitudes de cola, o sea que el problema es encontrar el valor de $u_i$ (tiempo de luz verde durante el ciclo, en la dirección i ), que minimice la suma de las longitudes de colas.

## 3.2 _ ESTRATEGIAS DE CONTROL ON-LINE

Por ser los ciclos independientes (como se explicó en el modelo), la secuencia de controles óptima este formada por el óptimo en cada período. Por lo tanto sólo se hará el análisis de las soluciones en tiempo real puesto que no es necesario ver los datos de todos los períodos conjuntamente.

## CRUCE SIMPLE NO SATURADO

El objetivo es:

$$\min_{u_i} \sum_{i=1}^{2} \sum_{k=1}^{N} X_i(k)$$

donde $u_i$ es la fracción de luz verde en la dirección i.

Para calcular $S = \sum_{i=1}^{2} \sum_{k=1}^{N} X_i(k)$ se necesita conocer $q_i(k)$, ya que el flujo máximo por

segundo ($s_i$) es un valor conocido, y $g_i$ (k) se calcula con la ecuación booleana (13). Como se está trabajando en tiempo real, el valor de $q_i$ (k) se obtendrá a través de "sensores".

Hay dos formas de medir $q_i$ (k):

a) antes de cada luz verde calcular el cociente (longitud de cola / tiempo de luz roja) y tomar $q_i$ (k) constante para todo k del ciclo, igual al cociente anterior.

b) mediante sensores, medir en cada período la tasa de arribo de vehículos. De esta forma obtenemos un valor de $q_i$ (k) para cada k.

Algoritmo

Para obtener el valor de $u_i$ óptimo, calculamos S para cada valor posible de $u_i$ (como se vió en el análisis que se realizó para el caso sobresaturado, hay 6 o 7 valores posibles), y tomando el $u_i$ que minimice S. Estando en un ciclo j, con la sucesión $q_i$(k) medida en el ciclo anterior (j-1), calculamos el $u_i$ que minimice S. Ese valor será el que se usará en el próximo ciclo (j+1).

Así, en cada ciclo se mide la sucesión $q_i$ (k) que se utilizará en el siguiente, para obtener el $u_i$ óptimo para el ciclo posterior. (Se asume que el tiempo de cálculo del $u_i$ óptimo es inferior al tiempo del ciclo).

Nota: El mismo algoritmo se puede utilizar en el caso de no disponer de sensores,trabajando con datos históricos.

RED NO SATURADA

En el caso de red no saturada, es importante que los ciclos de los semáforos estén sincronizados de tal forma que cuando los vehículos que salen de un cruce llegan al siguiente, encuentren luz verde.

Debido a que el período de optimización en el caso no saturado es un ciclo, y estamos trabajando en tiempo real, una vez que están sincronizados los cruces, se puede tratar cada uno independientemente aplicando lo visto para un cruce simple. Entónces, en la red no saturada el problema se resuelve en dos etapas

1) Sincronización de los cruces

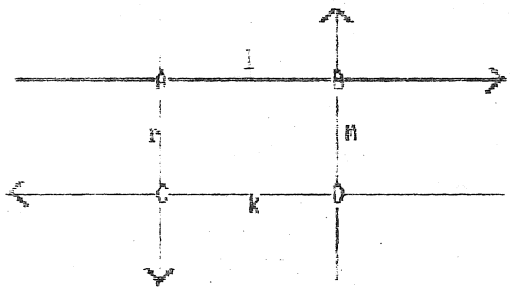
2) Optimización en cada cruce

Para el segundo punto, se puede aplicar el algoritmo de optimización dado para cruces simples.

Se analizará ahora el primer punto, o sea el problema de la sincronización de los cruces.

En calles de una única mano, la sincronización se logra desfasando los ciclos de los semáforos el tiempo suficiente para que los vehículos vayan de un cruce a otro a velocidad normal. Como el tiempo de luz verde es distinto en cada ciclo y en cada cruce, y además la distancia entre cruces no es la misma en toda la red, no es posible sincronizar todos los cruces.

Por ejemplo.

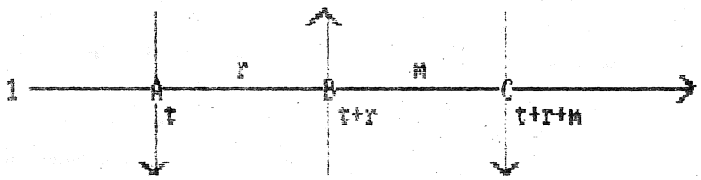



l = tiempo necesario para recorrer la distancia entre los cruces A y B a velocidad normal.

Análogamente se definen r, m y k.

Si l = k y m = r no será posible sincronizar estos cuatro cruces, y aún en el caso en que todos los cruces estén a la misma distancia, solo es posible sincronizarlos si el tiempo de luz verde por ciclo es constante para todos los cruces (no se cumple esto en nuestro caso). Luego, el objetivo no es sincronizar toda la red sino solo algunas calles, aquellas en las cuales la fluidez del tráfico sea importante.

En una red urbana generalmente se pueden distinguir calles muy transitadas y calles de poco tráfico. Nuestro problema consiste en sincronizar las primeras.




Para sincronizar la calle 1: Si el ciclo en A empieza en el instante t con luz verde en dirección 1, y el tiempo de demora en recorrer la distancia entre A y B es r, en el instante t+r debe haber luz verde en dirección 1 en el cruce B, o sea que los ciclos en B están desfasados r con respecto a los de A, y los de C están desfasados r+m, y así sucesivamente.

Un análisis más completo de las posibilidades de sincronización tendría que hacerse sobre la red particular que se esté analizando.

## 4 _ GENERALIZACION

### 4.1_ CRUCES DE AVENIDAS SOBRESATURADAS

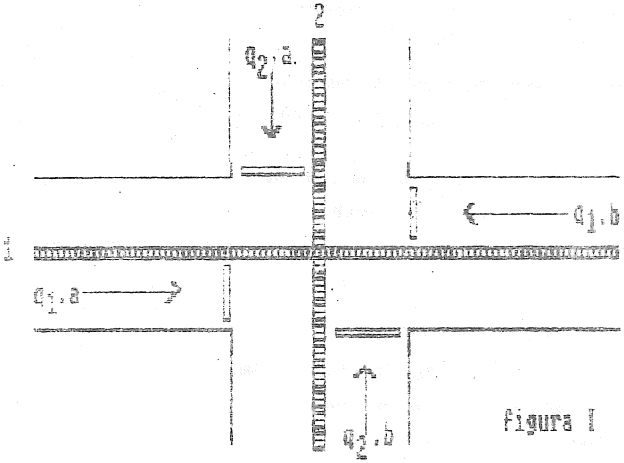



figura 1

La generalización del modelo de cruce simple a un cruce de avenidas es sencillo. Consideraremos la intersección de dos avenidas de un único carril por mano (ver fig 1). Para cada avenida se define

$u_i$, $g_i$, C análogamente al modelo para cruce sobresaturado.

$q_i.a$, $q_i.b$ tasa de arribo de vehículos en cada sentido de la avenida i.

$s_i$ flujo de saturación considerando una sola mano de la avenida i.

En cada mano de la avenida i se forman colas, cuya evolución dinámica se describe por ecuaciones similares a las del modelo para cruce sobresaturado:

$$X_i.a(k+1) = X_i.a(k) + q_i.a(k) \cdot g_i.a(k)$$
$$X_i.b(k+1) = X_i.b(k) + q_i.b(k) \cdot g_i.b(k) \qquad i=1,2$$

En una avenida pueden suceder dos cosas:

1) Las dos manos de la avenida son sobresaturadas.

2) Una de las manos es sobresaturada y la otra no.

En el caso (2) para representar el estado del cruce tomamos sólo la mano sobresaturada, y vemos a la avenida como una calle común. (Se aplican las mismas estrategias que en los casos estudiados antes para intersecciones de calles simples sobresaturadas).

En el caso (1), como en un cruce de avenidas el tiempo de luz verde para cada sentido de la avenida es el mismo

$$u_i.a(k) = u_i.b(k) \quad \text{para todo } k$$

y con [illegible] $g_i = g_i * u_i$

entonces $g_i.a(k) = g_i.b(k)$ para todo k.

Representaremos el estado de las colas en la avenida por

$$X_i(k) = \max [ X_i.a(k), X_i.b(k) ]$$

con lo cual se podrá utilizar el modelo de cruces de calles sobresaturadas, con la única salvedad de que en el vector de estados tendremos ( X1.a(k), X1.b(k), X2.a(k), X2.b(k) ), pero para trabajar igual que en los casos anteriores se toma

$$( \max_{i=a,b} ( X_1.i(k) ), \max_{i=a,b} ( X_2.i(k) ) )$$

Al generalizar de esta forma el modelo, los métodos a aplicar para encontrar la secuencia óptima de controles son iguales a los vistos anteriormente.

## 4.2_ CRUCES DE CALLES CON VARIOS CARRILES

En los modelos anteriores todas las variables están definidas sobre la hipótesis de que las calles son de una única vía. Veremos como extenderlas para el caso de calles con más de un carril. Supongamos calles de n carriles.

- Sea $q_i'(k)$ el número total de vehículos que arriban en un ciclo al cruce por los n carriles, en la dirección i. Bajo la hipótesis de que los vehículos que llegan al cruce se distribuyen homogéneamente en los n carriles, podemos decir que $q_i(k) = q_i'(k) / n$
- Si $s_i'$ es la máxima cantidad de autos que pueden salir en la dirección i en un ciclo, definimos $s_i = s_i' / n$
- $u_i$ es idéntica a la dada anteriormente en los modelos, o sea la fracción de luz verde en la dirección i.
- $g_i = u_i * s_i$

Luego, con esta nueva definición de las variables, a los cruces de calles con muchos carriles se les pueden aplicar los modelos dados anteriormente, y utilizar las soluciones ya vistas.

## 5_ CONCLUSIONES

A lo largo de este trabajo se abordaron algunos de los problemas que encierra el control de semáforos en una red de tráfico urbano. Se aplicaron conceptos y herramientas teóricas a problemas concretos obteniendo soluciones factibles, sin llegar a una implementación concreta.

En el trabajo se analizan sistemas de control de señales de tráfico off-line y on-line. Las soluciones presentadas para el primer caso son actualmente aplicables, en cambio no es común en la actualidad el uso de sistemas de control on-line (suponemos que se debe al costo, puesto que existe la tecnología adecuada para implementar tales sistemas).

Nota: Una versión anterior de este trabajo constituyó el proyecto final de la materia Algoritmos y Estructuras de Datos II [BRO-87], cursada por los autores en la Escuela Superior Latinoamericana de Informática en el año 1987.

## 6_ AGRADECIMIENTOS

A Edgardo Broner y Juan V. Echague por las valiosas ideas aportadas en la realización de la primera versión de este trabajo

## 7_ BIBLIOGRAFIA

[BRO-87] Apuntes del curso de Algoritmos y Estructuras de Datos II
E. Broner, E.S.L.A.I., 1987

[SIN-73] Hierarchical Strategies for the On-Line Control of Urban Road Traffic Signals
M. G. Singh, Lecture Notes in Computer Science, Vol 4, 1973

[NIL-71] Problem Solving Methods in Artificial Intelligence
N. Nilsson, Mc Graw-Hill, 1971