Paralelismo de datos en la obtención de Tablas de Control de Tráfico con información de llegada

Juan F. R. Herrera,[1] Eligius M. T. Hendrix,[2] Leocadio G. Casado,[3] y René Haijema[4]

Resumen—**Los semáforos se pueden controlar de forma dinámica a través de varias reglas que dictaminan el color del semáforo según el número de vehículos a la espera. Estas reglas o acciones se recogen en lo que se conoce como una Tabla de Control de Tráfico (TCT). Se ha calculado una TCT óptima sólo para infraestructuras simples mediante el método de Iteración de Valores, que se puede enmarcar dentro de la Programación Dinámica Estocástica. Como novedad de este trabajo, se añade información de la llegada de nuevos vehículos para el cálculo de una TCT óptima, en el caso de una intersección simple, o un conjunto de éstas. La dificultad de este problema reside en la complejidad computacional que conlleva el proceso de Iteración de Valores.**

Para infraestructuras simples, con sólo unos pocos carriles de circulación, se desea explotar la estructura del problema mediante el uso de un algoritmo paralelo. En este trabajo se formula el problema como un caso de decisión de Markov y se explican los pasos seguidos para la paralelización del método de Iteración de Valores aplicado a este problema.

Palabras clave— **Problema de decisión Markoviano, Programación Dinámica Estocástica, Optimización Global, Control de Tráfico.**

I. Introducción

La toma dinámica de decisiones representa una extensa área en la vida real, donde el objetivo principal es tomar una decisión a partir de un cierto estado, es decir, decidir la acción x a tomar estando en el estado s y en el tiempo t. Si el tiempo y el estado son continuos, este problema se clasificaría dentro de los problemas de control óptimo. Sin embargo, el problema estudiado en este trabajo no presenta las características anteriormente mencionadas, ya que el tiempo es discreto y el número de estados es finito. Además, se estudia el problema desde una perspectiva estacionaria, es decir, la decisión no depende del tiempo t. Considerando el problema como un proceso a largo plazo, nuestro principal interés es determinar la mejor acción x para cada estado s. Específicamente, para la situación estudiada, la distribución de la cantidad de vehículos entrantes en el sistema no dependería del tiempo (estacionario), aunque la densidad del tráfico puede variar debido a un proceso estocástico. Este problema es conocido como un problema de decisiones de Markov.

[1]Dpto. de Informática, Universidad de Almería (ceiA3), e-mail: juanfrh@ual.es.
[2]Dpto. de Arquitectura de Computadores, Universidad de Málaga, e-mail: Eligius.Hendrix@wur.nl.
[3]Dpto. de Informática, Universidad de Almería (ceiA3), e-mail: leo@ual.es.
[4]Operations Research and Logistics Group, Wageningen University, e-mail: Rene.Haijema@wur.nl.

La Programación Dinámica Estocástica (SDP, *Stochastic Dynamic Programming*) consiste, en términos de optimización, en la determinación de la mejor decisión mediante su descomposición en una secuencia de decisiones en el tiempo. La base teórica de la optimalidad de una regla es debida a la existencia de una función de valor v que calcula el valor (a largo plazo) de un sistema cuando se encuentra en el estado s. Si se tiene una función de valor v correcta, también se puede conocer la regla de decisión óptima. Mediante la ecuación de Bellman se puede usar el proceso de Iteración de Valores en inducción hacia atrás para encontrar V, véase [1], [2].

Los semáforos se introdujeron a principios del siglo XX para hacer la circulación más segura, en lugares donde el tráfico desde diferentes direcciones se une en un cruce o intersección. Para dar prioridad de paso al tráfico en varias direcciones, los vehículos que se aproximan en otras direcciones han de esperar antes de obtener prioridad. Se puede reducir el tiempo de espera de los vehículos en un semáforo si se establece un control adecuado del mismo. En la literatura, este problema ha sido estudiado tanto por teóricos de colas como por ingenieros [3], [4], [5], [6]. Este trabajo es una continuación de [7], [8], [9].

El control de tráfico óptimo en una intersección se basa en lo que se conoce como una Tabla de Control de Tráfico (TCT, *Traffic Control Table*), que establece qué combinación de carriles tienen prioridad dado el número de coches esperando en cada carril. Para encontrar una TCT que minimice el tiempo de espera de los vehículos, se puede aplicar el algoritmo de Iteración de Valores (*Value Iteration*), basado en Inducción hacia Atrás (BI, *Backward Induction*). La idea de la BI fue introducida por Bellman allá por el año 1953. BI se puede aplicar a problemas de optimización dinámicos que son discretos en el tiempo. En un sistema estacionario sin un horizonte de tiempo, el proceso se reduce a lo que se denomina en este trabajo Iteración de Valores.

El documento se estructura de la siguiente forma. En la sección II, se describirá el modelo de decisiones de Markov para las acciones recogidas en una TCT. En la sección III, se expondrá el algoritmo de Iteración de Valores para el problema descrito. En la sección IV, se aplicará el método a un caso simple. El algoritmo paralelo y los resultados obtenidos se expondrán en las secciones V y VI respectivamente. Las conclusiones de este artículo y las posibles líneas futuras de trabajo se recogen en la sección VII.

II. Descripción del modelo

Deseamos encontrar una TCT que indique cómo cambiar los colores de los semáforos dependiendo del tráfico, de tal forma que a la larga se minimice el tiempo de espera. La tabla ha de determinar qué carril debe tener prioridad dado el estado actual del tráfico. El problema se puede formular como un problema de decisión Markoviano (MDP, *Markov Decision Problem*).

A. Características del modelo

El problema de optimización de determinar el control de los semáforos es modelado como un proceso de control estocástico y discreto en el tiempo. Dado el estado s del tráfico, se debe elegir una acción x entre las disponibles. La elección óptima $x(s)$ es calculada mediante un proceso de Iteración de Valores.

El modelo se basa en las siguientes premisas:

- El tiempo se divide en intervalos. Un intervalo es el tiempo necesario para que un vehículo abandone una cola o cruce una intersección y determina la distancia de seguridad entre dos vehículos.
- Los semáforos pueden cambiar sólo al final de un intervalo.
- Cambiar de verde a rojo tarda dos intervalos en amarillo: $Y1$ e $Y2$.
- La decisión de cambiar el estado de un semáforo se implementa de forma instantánea.
- Mientras que el semáforo esté en verde o en amarillo, como máximo un vehículo cruza la intersección en un periodo de tiempo.
- Los vehículos se mueven a una velocidad constante. Los vehículos en la cola no se mueven. Sólo el primer vehículo de la cola se mueve cuando el semáforo lo permite.
- Para facilitar el análisis, la longitud de los vehículos es despreciable: por lo tanto un cruce jamás será bloqueado por el tráfico.

A continuación se define la notación que se usará a lo largo de este artículo. Un cruce o intersección consiste en F flujos de tráfico o carriles de circulación, donde cada flujo f consistirá en un carril y una cola asociada a este. El vector $q = (q_1, \ldots, q_F)$ almacena el número de vehículos esperando en cada cola. El conjunto $\mathcal{F} = \{1, \ldots, F\}$ de todos los carriles se puede particionar en C subconjuntos disjuntos llamados combinaciones. Las combinaciones son establecidas de tal forma que una de ellas puede recibir prioridad de paso sin que haya colisiones entre los vehículos de los distintos carriles que la forman. Flujos de tráfico en conflicto no forman parte de una misma combinación. Las distintas combinaciones son establecidas de antemano, por lo que no son objeto de estudio en este problema. Los carriles dentro de una combinación reciben a la vez la misma luz verde, roja o amarilla. Si una combinación está a verde o amarillo, las combinaciones restantes están a rojo.

B. Formulación como un problema de decisión Markoviano

Para describir el problema, se usan los siguientes parámetros:

- $\mathcal{S}$: un conjunto finito de estados. Cada estado se determina principalmente por el color de las luces del semáforo y el número de vehículos esperando en las distintas colas asociadas al cruce. El número de estados es finito porque el número de cruces es limitado y las colas asociadas a cada cruce se limitan en capacidad.
- $c(s)$: el coste del estado s, determinado por el número de vehículos esperando en las colas que describen el estado s.
- λ_f: es la probabilidad de que un vehículo llegue a la cola f en un intervalo de tiempo y por lo tanto $(1-\lambda_f)$ es la probabilidad de que no llegue.
- $\mathcal{E}$: un conjunto finito de posibles eventos. Cada evento representa la incorporación o no de un nuevo vehículo a la cola f. Nótese que el evento no depende del estado.
- $\mathcal{X}(s)$: Un conjunto finito de posibles acciones para controlar los semáforos. El conjunto de posibles acciones depende del estado s.

C. Objetivo

El tiempo de espera de un vehículo se define como el tiempo que transcurre desde el momento en que un vehículo se une a la cola hasta que cruza la línea. El objetivo práctico es encontrar un control que minimice el tiempo de espera estimado (EW) a largo plazo. La ley de Little formaliza la relación entre la tasa de llegada λ de vehículos a la cola, el tiempo de espera esperado EW y el número de vehículos esperado en una cola EQ de la siguiente forma:

$$EQ = \lambda \times EW.$$

Si se minimiza el número de vehículos esperando en las colas, también se minimiza el tiempo de espera. El algoritmo MDP no guarda un registro de los tiempos de espera de un vehículo en particular, pero sí lo hace del número de vehículos esperando en el comienzo de un intervalo. Por tanto, el objetivo del modelo MDP es minimizar el número medio de vehículos esperando en cada intervalo sobre un horizonte infinito. Sobre un horizonte de n intervalos, esto implica minimizar la suma de vehículos esperando en el comienzo de cada intervalo.

III. Iteración de Valores a través de inducción hacia atrás

A continuación se describe el proceso de Iteración de Valores para un MDP estacionario[1]. Hay que tomar una decisión x, dentro de un espacio de decisiones $\mathcal{X}$ que depende del estado actual $s \in \mathcal{S}$. El sistema dinámico se describe como la función de transición $T(s, x, e)$ que describe el siguiente estado a ser

[1]Un proceso estacionario es aquel en el que la función de distribución de la probabilidad es constante en el tiempo.

Algoritmo 1 Iteración de valores

1: Inicializar a 0 el vector V
2: **repeat**
3: Copiar el vector V en W
4: Para $j = 1, \ldots, N$ determinar (2)
5: **until** $\max_j(V_j - W_j) - \min_j(V_j - W_j) < \epsilon$

alcanzado después de tomar la decisión x en el estado s habiendo tenido lugar el evento estocástico $e \in \mathcal{E}$. Una acción $x(s)$ indica qué decisión tomar en el estado s. El coste asociado en un MDP genérico normalmente depende tanto del estado s como de la decisión x. En el modelo en que estamos interesados, el coste sólo depende del estado s, por tanto la función de coste es $c(s)$.

La estrategia $x(s)$ es óptima si existe una función $v(s)$ y un escalar d tal que $\forall s \in \mathcal{S}$

$$v(s) - d = c(s) + \min_{x \in \mathcal{X}(s)} [E_e\{v(T(s, x, e))\}], \quad (1)$$

donde E_e simboliza el valor esperado con respecto al evento estocástico e, que explicaremos con más detalle en la sección IV-A.4. La idea de la inducción hacia atrás en (1) es que la evaluación de s dependerá de las valoraciones de los estados que se que alcanzarán en la siguiente etapa. Bellman introdujo el término inducción hacia atrás donde (1) se repite de forma iterativa recibiendo el nombre de Iteración de Valores.

Lo esencial en el modelo descrito es que el número de eventos es finito, de tal forma que pueden ser numerados e_i con una probabilidad de ocurrencia p_i. El número de estados también es finito, así como la función de valor v_j para $j = 1, \ldots, N$. De esta forma, la solución de (1) consiste en encontrar un vector $V = (V_1, \ldots, V_j, \ldots, V_N)$ tal que

$$V_j - d = c(s_j) + \min_{x \in \mathcal{X}(s_j)} \sum_i p_i V_k, \quad (2)$$

donde

$$V_k = v(T(s_j, x, e_i)). \quad (3)$$

La Iteración de Valores para encontrar el vector V se muestra en el algoritmo 1 donde se copian los valores de V en otro vector W y se usan estos valores para encontrar un nuevo valor de V. Para cumplir (2), la iteración debe diferir un valor escalar d para todos los estados s_j. La convergencia al escalar se mide a través de

$$span(V, W) = \max_j(V_j - W_j) - \min_j(V_j - W_j) \quad (4)$$

hasta una precisión ϵ.

¿Cómo se traslada esta propiedad al modelo de las TCTs? En primer lugar, nótese que la estrategia $x(s)$ representa la idea de una tabla de control de tráfico. En el modelo, el conjunto de estados $\mathcal{S}$ es contable ya que está basado en las luces del semáforo y el número de vehículos en espera. Además, se ha establecido un número máximo de vehículos en una fila, por lo que las funciones $v(s)$ y $x(s)$ se representan por el vector V y la TCT final estará determinada por:

$$X_j = x(s_j) = \arg \min_{x \in \mathcal{X}(s_j)} \sum_i p_i V_k, \quad (5)$$

con V_k definida en (3). El reto computacional viene determinado por el valor de N que es normalmente alto. Un aspecto importante del modelo definido es que la computación de V_j requiere un conjunto finito de valores W_j, determinado por el conjunto de índices

$$\mathcal{K}_j = \{k : s_k = T(s_j, x, e_i) \forall i, \forall x \in \mathcal{X}(s_j)\}. \quad (6)$$

Por lo tanto, la actualización de los elementos de V en el algoritmo 1 debe realizarse en un orden que permita maximizar la localidad espacial de los datos.

IV. Casos estudiados del modelo TCT

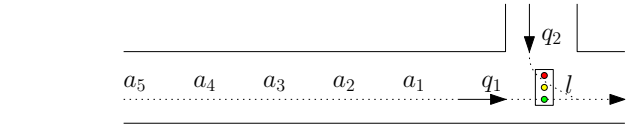
Cada infraestructura será referenciada mediante un código. Por ejemplo, I1C2F2 para una intersección con $C = 2$ combinaciones y $F = 2$ flujos de circulación o carriles.

En la figura 1 se puede observar dos infraestructuras simples:

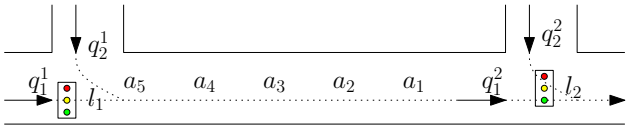
I1C2F2 Un cruce simple en T con sólo dos carriles de tráfico, 1 y 2. Los vehículos en el carril 1 se desplazan desde el oeste hasta el este, mientras que los vehículos del carril 2 lo hacen desde el norte al este, es decir, giran a la izquierda. Estos dos carriles no pueden estar en verde simultáneamente. Por lo tanto, puede ocurrir que se formen colas en alguno de los dos carriles. Cada dirección tiene sólo un carril y cada carril tiene asociada una cola en la línea de parada del cruce. Adicionalmente, el carril 1 tiene información de llegada de nuevos vehículos.

I2C2F2 Un tándem de dos cruces en T con dos flujos cada uno: los vehículos del cruce 1 se desplazan al cruce 2, es decir, sendos carriles del cruce 1 alimentan la cola 1 del cruce 2. Para este caso asumimos que el tiempo necesario para viajar desde el cruce 1 hasta el cruce 2 es de cinco intervalos de tiempo.

En este trabajo, se implementará la infraestructura I1C2F2, dejando la implementación de I2C2F2 como trabajo futuro.



(a) I1C2F2



(b) I2C2F2

Fig. 1. Infraestructuras.

A. I1C2F2

A continuación se describen las características especificas de este caso.

A.1 Estado s

Para el control dinámico óptimo del tráfico, las luces se cambian en base a la información actual del estado de los semáforos y del estado de los flujos de tráfico. El conjunto de posibles estados de los semáforos se denota por $l \in \mathcal{L}\{0,1,2,3,4,5,6\} = \{Red, GF1, GF2, Y1F1, Y2F1, Y1F2, Y2F2\}$.

Red: Semáforo en rojo para todos los carriles.
GF1: Semáforo en verde para el carril 1.
GF2: Semáforo en verde para el carril 2.
Y1F1: Semáforo en amarillo para el carril 1 (intervalo 1).
Y2F1: Semáforo en amarillo para el carril 1 (intervalo 2).
Y1F2: Semáforo en amarillo para el carril 2 (intervalo 1).
Y2F2: Semáforo en amarillo para el carril 2 (intervalo 2).

El estado de los flujos de tráfico para este caso es:

- $q = (q_1, q_2)$. El número de vehículos esperando en cada carril f. Q representa el número máximo de vehículos en una cola determinada.
- $a = (a_1, \ldots, a_M)$. M intervalos de información de llegada están disponibles para el carril 1, donde $a_t \in \{0,1\}$ determina la presencia o no de un vehículo en el intervalo t. Por ejemplo, si $a_t = 1$, un vehículo llegará a la cola o pasará la línea de parada en t intervalos de tiempo.

Por tanto, un estado s está definido por $s = (l, q, a)$ y el número de posibles estados es

$$N = |\mathcal{S}| = |\mathcal{L}| \cdot (Q+1)^2 \cdot 2^M. \tag{7}$$

La complejidad de I2C2F2 es mayor debido a la existencia de más carriles y por tanto de un mayor número de combinaciones de luces en $\mathcal{L}$.

A.2 Acción de control x

Dado un estado s, la acción $x \in \mathcal{X}(s)$ inmediatamente ajusta las luces:

$$\mathcal{X} = \begin{cases} \{1,2\} & = \{GF1, GF2\} & \text{si } l = 0 = Red \\ \{1,3\} & = \{GF1, Y1F1\} & \text{si } l = 1 = GF1 \\ \{4\} & = \{Y2F1\} & \text{si } l = 3 = Y1F1 \\ \{0\} & = \{Red\} & \text{si } l = 4 = Y2F1 \\ \{2,5\} & = \{GF2, Y1F2\} & \text{si } l = 2 = GF2 \\ \{6\} & = \{Y2F2\} & \text{si } l = 5 = Y1F2 \\ \{0\} & = \{Red\} & \text{si } l = 6 = Y2F2 \end{cases}$$

A.3 Función objetivo

En un intervalo de tiempo, con $q_1 + q_2$ vehículos esperando, la función objetivo es

$$c(s) = q_1 + q_2. \tag{8}$$

A.4 Transformación de estados

Durante un intervalo, un vehículo llega al carril f con una probabilidad λ_f. Como este caso sólo consta de dos carriles, se define el evento como un vector de dos elementos: $e = (e_1, e_2)$, donde $e_f \in \{0,1\}$ denota el número de vehículos que llega al carril f. Para el caso simple de dos carriles se tienen cuatro posibles eventos: $e \in \{(0,0),(0,1),(1,0),(1,1)\}$. La probabilidad p_i relacionadas con cada evento es:

$$e = \begin{cases} i=1, & e=(0,0) & \text{con} & p_1 = (1-\lambda_1)(1-\lambda_2) \\ i=2, & e=(0,1) & \text{con} & p_2 = (1-\lambda_1)\lambda_2 \\ i=3, & e=(1,0) & \text{con} & p_3 = \lambda_1(1-\lambda_2) \\ i=4, & e=(1,1) & \text{con} & p_4 = \lambda_1\lambda_2 \end{cases}$$

Durante un intervalo de tiempo, la información de llegada $a = (a_1, a_2, a_3, a_4, a_5)$ es actualizada con $a \leftarrow (a_2, a_3, a_4, a_5, e_1)$, es decir, los vehículos que llegan al carril se aproximan y a_1 representa el vehículo que se añade a la cola. Si la cola se incrementa en número o no depende del estado del semáforo. La función $\Delta_f(x)$ indica si un carril f obtiene prioridad ($\Delta_f(x) = 1$) o no ($\Delta_f(x) = 0$) cuando las luces se cambian debido a la acción x. Si el carril 1 está a rojo, un vehículo que está a un intervalo de distancia de la cola 1, incrementa el número de vehículos en la cola 1, es decir, cuando $\Delta_1(x) = 0$, por tanto $l \in \{0,2,4,6\}$. Cuando x establece el color del semáforo a verde o amarillo para el carril 1 ($\Delta_1(x) = 1$ y $l \in \{1,3,5\}$) y la cola 1 no está vacía, el vehículo se une a la cola. En caso de que la cola esté vacía, el coche cruza la línea de parada de la cola 1 sin retraso. La transición de la cola 1 será $q_1 \leftarrow q_1 + a_1 - \Delta_1(x)$.

Para el carril 2, no hay información sobre la llegada de vehículos, por tanto el siguiente estado de la cola 2 dependerá solo de e_2 y de la acción x. La transición de la cola 2 es $q_2 \leftarrow q_2 + e_2 - \Delta_2(x)$.

V. Paralelización

El algoritmo (secuencial) 1 es un proceso iterativo donde el vector V de tamaño $N = |\mathcal{S}|$ se actualiza en base a los valores previos de V guardados en el vector W hasta que se cumple el criterio de terminación. La manera natural de paralelizar el algoritmo 1 es distribuir la evaluación de V entre los procesadores. El conjunto de valores de estado V se puede particionar de distintas maneras dependiendo de N y de las características de la arquitectura. Cuando se utiliza paso de mensajes, el número de mensajes debe ser el menor posible. Para arquitecturas de memoria compartida, el acceso a los valores de W debe ser realizado de tal forma que se incremente la probabilidad de que los valores se encuentren en caché.

Teniendo en cuenta estas consideraciones, una opción es guardar los valores de V y W como una matriz, donde las columnas son los valores de las luces y las filas son las diferentes combinaciones de estado del tráfico (estado de las colas e información de llegada). En una arquitectura de memoria distribuida, una solución razonable podría ser un particionamiento de la matriz en columnas. De esta manera,

el número de mensajes no es muy alto, aunque el tamaño de éstos será probablemente grande.

A. Complejidad computacional

La carga de trabajo está relacionada con el número de flujos de tráfico (F), el tamaño máximo de la cola (Q) y el número de estados (M) que describe la información de llegada. El límite impuesto en la longitud de la cola no sólo hace que el espacio de estados sea finito. El uso de longitudes de cola grandes introduce situaciones con una probabilidad de ocurrencia muy baja. Esto dificulta la convergencia en la Iteración de Valores. Por tanto, la complejidad del algoritmo viene dada por N, véase (7).

B. MPI

El esquema MPI es principalmente usado en memoria distribuida, aunque también puede ser utilizado en memoria compartida. La figura muestra el paso de mensajes para la paralelización usando MPI, donde cada proceso MPI es responsable de la evaluación de los valores de V asociados a un valor del estado del semáforo l. Se puede obtener un grado mayor de paralelismo mediante la paralelización de cada proceso MPI.

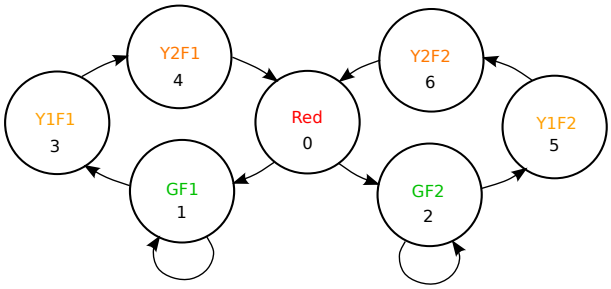


Fig. 2. Transición de luces en el I1C2F2.

El principal inconveniente del esquema MPI descrito es el desbalanceo entre procesos. Por ejemplo, los procesos encargados de $l \in \{Red, GF1, GF2\}$ requieren más datos del vector W para actualizar los valores V_j que los procesos encargados de las luces en amarillo. Adicionalmente, el máximo número de procesos en I1C2F2 está limitado al número máximo de luces, que es de siete.

C. Threads

Este esquema se utiliza únicamente en arquitecturas de memoria compartida. El hecho de tener el conjunto V como una matriz, donde cada hebra de ejecución se encarga de la evaluación de un conjunto de filas contiguo, facilita el uso de un número arbitrario de hebras. Un inconveniente es la limitación impuesta por el número de núcleos de ejecución que contenga un nodo, siendo necesario un esquema de paso de mensajes o híbrido (MPI y threads) cuando se requiere un número mayor de núcleos de ejecución. Se puede encontrar un trabajo previo donde se utiliza OpenMP para la obtención de TCTs sin información de llegada en [10].

VI. Resultados preliminares

Los experimentos se han llevado a cabo en en un nodo del *cluster BullX-UAL*, compuesto de 18 nodos de cómputo. Cada nodo dispone de dos procesadores Intel® Xeon® E5-2650 y 64 GB de memoria principal. Los algoritmos han sido codificados en C y compilados usando gcc con MVAPICH y POSIX threads API.

Además de todos los datos expuestos en la sección IV-A, se asume que el ratio de llegada para cada una de las dos colas es de 0,2 vehículos por intervalo $\lambda_1 = \lambda_2 = 0{,}2$. Esto es, el ratio sería de un 40 % cuando todos los semáforos están en rojo. Se ha establecido una precisión de $\epsilon = 0{,}01$. El caso A será ejecutado dos veces con los siguientes parámetros:

Q50M10 Tamaño de la cola $Q_1 = Q_2 = 50$. Número de intervalos de información de llegada $M = 10$.

Q100M5 Tamaño de la cola $Q_1 = Q_2 = 100$. Número de intervalos de información de llegada $M = 5$.

El algoritmo 1 ha necesitado 361 iteraciones para resolver *Q50M10* con una precisión $\delta < 0{,}01$. En cambio, para resolver *Q100M5* se han necesitado 2.090 iteraciones.

Los resultados del esquema MPI utilizando siete procesos muestran un rendimiento pobre, obteniendo una aceleración inferior a 3×. Se están estudiando las posibles causas de esta falta de rendimiento, aunque se cree que puede ser debido al desbalanceo de la carga de trabajo entre los procesadores y el coste del paso de mensajes.

La tabla I muestra los resultados experimentales para la versión hebrada variando el número de hebras. La tabla recoge el número de hebras utilizadas, el tiempo de ejecución en segundos y la aceleración obtenida (Ac.). La aceleración alcanzada es cercana al lineal. Como trabajo futuro, se planea reorganizar las columnas de las funciones de valor V y W asignadas a cada hebra para mejorar la localidad de los datos en la caché.

TABLA I
Resultados numéricos usando hebras.

	Q50M10		*Q100M5*	
Hebras	Tiempo	Ac.	Tiempo	Ac.
1	511	–	240	–
2	260	2	119	2
4	133	3,8	56	4
8	68	7,5	31	7,7
16	36	14,2	17	14

Ambos métodos requieren un punto de sincronización al final de cada iteración con el fin de comprobar el criterio de terminación y de tener disponible los datos en W para la próxima iteración. Esto limita el grado de paralelismo de la aplicación.

VII. Conclusiones y trabajo en curso

Los problemas de optimización resueltos a través del algoritmo de Iteración de Valores son difíciles de paralelizar debido a que en cada iteración, debe existir un punto de sincronismo entre todas las unidades de proceso lanzadas en paralelo. Este trabajo estudia el algoritmo para la generación de tablas de control de tráfico óptimas. La dificultad de estos problemas se incrementa con el número de intersecciones y la inclusión de información de llegada. Una aceleración cercana a la lineal se ha obtenido con una versión hebrada para un caso sencillo del problema. Una futura investigación se centrará en la computación en paralelo de tablas de control de tráfico que implique un mayor número de estados de tráfico. Una paralelización en dos niveles (MPI para la paralelización entre nodos y threads para la paralelización dentro de un mismo nodo) sería conveniente para alcanzar un buen rendimiento en un cluster de nodos de memoria compartida.

Agradecimientos

El presente trabajo ha sido financiado mediante el proyecto TIN2008-01117, TIN2012-37483 y P11-TIC-7176, financiados parcialmente por los fondos FEDER. J.F.R. Herrera es un becario FPU.

Referencias

[1] Richard Bellman, "A Markovian Decision Process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, 1957.

[2] Martin L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, Inc., 1994.

[3] G. F. Newell, "Approximation methods for queues with application to the fixed-cycle traffic light," *SIAM Review*, vol. 7, no. 2, pp. 223–240, 1965.

[4] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Yibing Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.

[5] J. S. H. van Leeuwaarden, "Delay analysis for the fixed-cycle traffic-light queue," *Transportation Science*, vol. 40, no. 2, pp. 189–199, 2006.

[6] I. J. B. F. Adan M. S. van den Broek, J. S. H. van Leeuwaarden and O. J. Boxma, "Bounds and approximations for the fixed-cycle traffic-light queue," *Transportation Science*, vol. 40, no. 4, pp. 484–496, 2006.

[7] René Haijema, *Solving large structured Markov Decision Problems for perishable inventory management and traffic control*, Ph.D. thesis, Universiteit van Amsterdam, 2008.

[8] René Haijema and Jan van der Wal, "An MDP Decomposition Approach for Traffic Control at Isolated Signalized Intersections," *Probability in the Engineering and Informational Sciences*, vol. 22, pp. 587–602, 2008.

[9] René Haijema and Eligius M. T. Hendrix, "Traffic responsive control of intersections with predicted arrival times: A markovian approach," *Computer-Aided Civil and Infrastructure Engineering*, vol. 29, no. 2, pp. 123–139, 2014.

[10] Eligius M. T. Hendrix, Siham Tabik, and René Haijema, "Determination of traffic control tables by HPC," in *Actas de las XXII Jornadas de Paralelismo*, La Laguna, Spain, 2011, pp. 131–134.