Planificación de trayectorias mediante Programación Dinámica

Maikel O. Torres Piñeiro[1], Valery Moreno Vega [2]

[1] *Instituto Superior Politécnico José Antonio Echevarria. maikel@electrica.cujae.edu.cu*

[2] *Instituto Superior Politécnico José Antonio Echevarria. valery@ electrica.cujae.edu.cu*

RESUMEN / ***ABSTRACT***

En este trabajo se realiza un análisis de dos métodos de PD utilizados en la planificación de trayectorias de robots móviles: uno propuesto por Kwok y Driessen (1999) y el otro el algoritmo de Bellman-Ford. A este último, se le realizan modificaciones para incrementar aún más su velocidad de búsqueda de la solución.

Palabras claves**:** planificación de trayectorias, programación dinámica.

Path planning with Dynamic Programming

In this work are studied two methods of dynamic programming that can be used in path planning of mobile robots. The first one is proposing by Kwok and Driessen (1999) and the other one is the Bellman-Ford algorithm. The last one will be modified to obtain better result.

Key word: *path planning, dynamic programming.*

INTRODUCCIÓN

La programación dinámica (PD) fue introducida por Bellman [1] y la misma consiste en el proceso matemático de tomar una serie de decisiones óptimas para obtener una solución global óptima.

Este principio ha sido usado en la planificación para buscar dentro de un conjunto de estados del espacio de configuración, aquellos que sean óptimos según determinado criterio, para luego conformar una trayectoria óptima.

Una de las principales ventajas que conlleva el uso de la PD es que las trayectorias que se obtienen por este método siempre son óptimas y se puede usar cualquier tipo de función de costo u optimización.

Son diversos los algoritmos donde se puede encontrar la programación dinámica, aunque en el área de la robótica móvil, específicamente la planificación, su versión más conocida es el llamado algoritmo de Bellman-Ford [2], no obstante existen otros trabajos donde se aplican otras versiones de la PD [3, 4, 5].

En este trabajo se hará un análisis de dos métodos de PD utilizados en la planificación de trayectorias de robots móviles: uno propuesto por Kwok y Driessen (1999) y el otro el algoritmo de Bellman-Ford [2, 6]. A este último, se le harán modificaciones para incrementar aún más su velocidad de búsqueda de la solución.

MATERIALES Y MÉTODOS

Algoritmo de Bellman-Ford.

Este algoritmo determina la trayectoria más corta entre el punto inicial y todos los restantes puntos del entorno. El algoritmo propuesto por Bellman [1] se basa en la programación dinámica y su formulación recursiva es la siguiente:

$$F_0(v) = \begin{cases} 0, \ si\, v = s \\ \infty, si\, v \neq s \end{cases} \quad 1)$$

$$F_k(v) = \min\left\{ F_{k-1}(v), \min_{w \in Q_{k-1}, (w,v) \in E} \{F_{k-1}(w) + C_{vw}\} \right\} \quad 2)$$

donde:

Q_k: Es el conjunto de estados que fueron actualizados en la etapa *k*.

E : Es el conjunto de enlaces simples entre los estados. Se considera que dos estados están enlazados de manera simple si se puede ir de uno al otro en una etapa.

S : Estado inicial.

v,*w* : Estados cualesquiera.

C_{vw}: Costo de ir del estado *w* al estado *v*.

El seudo código algoritmo puede ser definido como sigue:

n: Cantidad de estados.

V: Conjunto de todos los estados.

Determinar $F_0(v) \forall v \in V$ y Q_0

Mientras $Q_k \neq 0 \; y \; k \leq n$ hacer

Calcular $F_k(v) \forall v \in V$ y determinar Q_k

Fin Mientras

Si $Q_k = \varnothing$ entonces $F_k(v)$ define la longitud de la trayectoria más corta desde S hasta $v \forall v$.

Si k = n y $Q_k \neq \varnothing$ entonces existen enlaces entre estados con costo negativo y no pueden ser definas las trayectorias más costas.

Aunque este algoritmo fue concebido para determinar la trayectoria óptima según la distancia no hay pérdida de generalidad si se toma $F_k(v)$ como otro tipo de función objetivo. Siempre se obtendrán las trayectorias óptimas según la función objetivo $F_k(v)$.

La aplicación de este método en la planificación de trayectoria para un robot móvil es inmediata, ya que el mismos es un método de búsqueda en una red de estados o grafo y en el caso de la robótica móvil los estados serían un conjunto de localizaciones pertenecientes al entorno donde se encuentra el robot, es decir, configuraciones del espacio de configuración.

Como los entornos por los que se mueve un robot pueden tener obstáculos entonces el conjunto de localizaciones del entorno (estados) se encuentra dividido en dos tipos:

1. estados prohibidos (ocupados por obstáculos)
2. estados libres (el robot puede ocupar uno de ellos sin problemas)

Luego para hacer la formulación del algoritmo más completa y acorde a las condiciones de la robótica móvil las ecuaciones 1) y 2) quedan como se muestra a continuación:

$$F_0(v) = \begin{cases} 0, \; si \; v = s \; y \; v \notin O \\ \infty, si \; v \neq s \; ó \; v \in O \end{cases} \qquad 3)$$

$$F_k(v) = \min \left\{ \begin{array}{ll} F_{k-1}(v), & \\ \min\limits_{w \in Q_{k-1}, (w,v) \in E} \{F_{k-1}(w) + C_{vw}\}, & \forall v \notin O \\ \infty, & \forall v \in O \end{array} \right\} \qquad 4)$$

donde:

O: Conjunto de estados ocupados por obstáculos (estados prohibidos).

De esta manera la formulación recursiva del algoritmo está representada por las ecuaciones 3) y 4) manteniendo la misma definición.

Bellman-Ford Optimizado

El algoritmo de Bellman-Ford tal y como se plantea en la sección anterior tiene el inconveniente de que en cada etapa explora todos los estados del entorno. Esto puede traducirse en un mayor tiempo de ejecución, por tal razón, a continuación se plantea la formulación recursiva del método de Bellman-Ford optimizado.

$$F_0(v) = \begin{cases} 0, \; si \; v = s \; y \; v \notin O \\ \infty, si \; v \neq s \; ó \; v \in O \end{cases} \qquad 5)$$

$$F_k(v) = \min \left\{ \begin{array}{ll} F_{k-1}(v), & \\ \min\limits_{v \in Q_k, \, w \in G_{k-1}(v)} \{F_{k-1}(w) + C_{vw}\}, & \forall v \notin O \\ \infty, & \forall v \in O \end{array} \right\} \qquad 6)$$

donde:

Q_k: Es el conjunto de estados que se van a explorar en la etapa *k*.

$G_{k-1}(v)$: Es el conjunto de padres de *v* que fueron actualizados en la etapa *k-1*. Se define como padre de un estado *v* cualquier estado *w* del cual se pueda llegar a *v* en una etapa.

S: Estado inicial.

v, w: Estados cualesquiera.

C_{vw}: Costo de ir del estado *w* al estado *v*.

O: Conjunto de estados ocupados por obstáculos (estados prohibidos).

El seudo código del algoritmo puede ser definido como sigue:

n: Cantidad de estados.

V: Conjunto de todos los estados.

P_v: Conjunto de padres óptimos. Un padre óptimo del estado *v* es el estado *w* que hace que se minimice el costo de *v*.

Determinar $F_0(v) \forall v \in V$ y Q_1

Mientras $Q_k \neq \varnothing \; y \; k \leq n$ hacer

Calcular $F_k(v) \forall v \in Q_k$,

$P_v \forall v \in Q_k$ y determinar Q_{k+1}

Fin Mientras

Si $Q_k = \varnothing$ entonces $F_k(v)$ define el costo de la trayectoria óptima desde *S* hasta $v \forall v$. Considerando v_{end} como el estado al cual se quiere llegar desde *S*, entonces la secuencia de estados es dada por:

$v_i = v_{end}$

Mientras $v_i \neq S$ hacer

$v_i = P_{vi}$

Fin Mientras

Si k = n y $Q_k \neq \varnothing$ entonces existen enlaces entre estados con costo negativo y no pueden ser definas las trayectorias óptimas.

Es importante destacar que en esta versión del algoritmo se disminuye el tiempo de ejecución en gran mediada al reducir el espacio de estados con posibilidades de ser óptimos. Esto se debe a que en cada etapa solo se exploran aquellos estados cuyo estado padre haya sido actualizado en la etapa anterior. Al hacerse de esta manera no se tienen que explorar todos los estados en cada etapa, como sucede en el algoritmo de Bellman-Ford sin optimizar, ya que solo es de interés explorar aquellos estados que realmente tengan la posibilidad de mejorar su costo en la etapa *k*.

No obstante, aún se le puede realizar una modificación que tributaria en una disminución del espacio de búsqueda.

Modificación para reducir el espacio de búsqueda

Para reducir aún más el espacio de búsqueda, y por tanto, el tiempo de ejecución del algoritmo, se propone que se exploren solo aquellos estados que cumplen con las condiciones del método de Bellman-Ford optimizado y además que *el costo del estado padre sea menor que el menor costo encontrado hasta el momento para el estado objetivo*. Esto elimina aquellos estados que no aporten una solución óptima.

En la sección 3 se hará un análisis del efecto que provoca la reducción del espacio de búsqueda de las diferentes versiones de los métodos de PD presentados en este trabajo.

Método de Kwok-Driessen

El algoritmo que a continuación se presenta fue creado por Kwok y Driessen [3] para planificar trayectorias de robots móviles. En el mismo se definió una matriz ***M*** donde a cada elemento de la matriz se le asocio un estado del entorno según su localización dentro del mismo. Un estado se consideró como un punto con localización *(i,j)* dentro del entorno.

$$M_{i,j} = \begin{cases} 0, si\ el\ estado(i,j)\ es\ prohibido \\ 1, si\ el\ estado(i,j)\ es\ libre \end{cases} \quad 7)$$

La formulación recursiva propuesta por Kwok y Driessen es la siguiente:

$$C_{1ij} = \left\{ \begin{array}{l} -1, \forall (i,j) \neq (i_{end}, j_{end}) \\ 0, \forall (i,j) = (i_{end}, j_{end}) \end{array} \right\} \quad 8)$$

$$C_{k+1,i,j} = \left\{ \begin{array}{ll} \min\limits_{m \in S_{kij}} \left\{ \begin{array}{l} F(i,j,m,i_{\max}, j_{\max}) + \\ C_{k,L1(i,j,m),L2(i,j,m)} \end{array} \right\}, & \\ & \forall (i,j)\ y\ S_{kij} \neq \varnothing \\ -1, & \forall (i,j)\ y\ S_{kij} = \varnothing \end{array} \right\} \quad 9)$$

$$m^*_{k+1,i,j} = \arg \min_{m \in S_{kij}} \left\{ \begin{array}{l} F(i,j,m,i_{\max}, j_{\max}) \\ + C_{k,L1(i,j,m),L2(i,j,m)} \end{array} \right\} \quad 10)$$

$$I_{k+1,i,j} = L_1(i,j,m^*_{k+1,i,j}) \quad 11)$$

$$J_{k+1,i,j} = L_2(i,j,m^*_{k+1,i,j}) \quad 12)$$

donde:

m: Tipos de movimientos que se pueden realizar para llegar a un estado desde otro entre dos etapas consecutivas. Ver Figura 1.

$m^*_{k+1,i,j}$: Es el movimiento óptimo para llegar en la etapa *k+1* al estado *(i,j).*

$L_1(i,j,m)$: Indica el valor *i* del estado $(\hat{i},\hat{j})$ del que se proviene usando el movimiento *m.*

$L_2(i,j,m)$: Indica el valor *j* del estado $(\hat{i},\hat{j})$ del que se proviene usando el movimiento *m.*

S_{kij} : Es el conjunto de movimientos permitidos para llegar al estado *(i,j)* en la etapa *k.* S_{kij} se define como:

$$S_{kij} = \left\{ \begin{array}{l} m : m \in (1,...,8), \\ C_{k,L1(i,j,m),L2(i,j,m)} \neq -1, \\ F(i,j,m,i_{\max}, j_{\max}) \neq -1 \end{array} \right\} \quad 13)$$

$F(i,j,m,i_{max},j_{max})$: Función objetivo y se define como:

$$F = \left\{ \begin{array}{l} -1, \\ \quad si\ L_1(i,j,m) \notin (1,...,i_{\max})\ ó \\ \quad L_2(i,j,m) \notin (1,...,j_{\max})\ ó \\ \quad M(L_1(i,j,m), L_2(i,j,m)) = 0 \\ dis\tan cia\ entre\ (i,j)\ y \\ (L_1(i,j,m), L_2(i,j,m)), \\ \quad para\ otro\ caso \end{array} \right\} \quad 14)$$

$I_{k,i,j}$: Es la localización *i* óptima para moverse hacia el estado $(\hat{i},\hat{j})$ regresando una etapa desde *k.*

$J_{k,i,j}$: Es la localización *j* óptima para moverse hacia el estado $(\hat{i},\hat{j})$ regresando una etapa desde *k.*

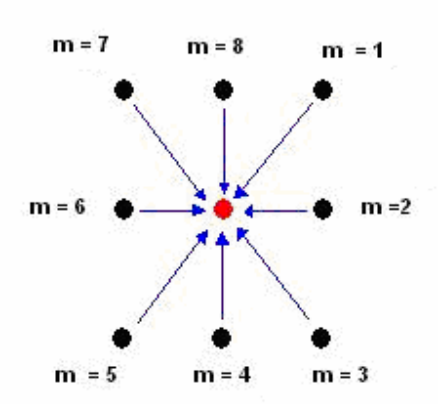

Figura 1. Tipos de movimientos que se pueden realizar para llegar a un estado dado.

El algoritmo de Kwok y Driessen consiste en repetir la formulación recursiva representada en las ecuaciones 8)-12) para *k = 1,2,3,…* desde un estado inicial (i_{start},j_{start}) hasta que suceda una de dos condiciones:

1. $C_{k+1,i_{start},j_{start}} \neq -1$
2. La cantidad de elementos del conjunto $C_{k+1,i,j}$ con $C_{k+1,i,j} \neq -1$ es igual a la cantidad de elementos del conjunto $C_{k,i,j}$ con $C_{k,i,j} \neq -1$.

La primera de las condiciones significa que ha sido encontrada una solución óptima y la secuencia de estados para ir desde (i_{start},j_{start}) hasta (i_{end},j_{end}) es dada por:

$i^* = i_{start}$

$j^* = j_{start}$

Hacer desde *p = 1* hasta *k-1*

$i^* = I_{p,i,j}$

$j^* = J_{p,i,j}$

Fin Hacer.

La segunda condición significa que no se encontró una solución y por tanto no existe un camino entre el estado inicial y el final.

DISCUSIÓN

Dentro de los algoritmos de PD que buscan un camino óptimo en un grafo o red de estados, sin dudas, el más utilizado es de Bellman-Ford. En este trabajo se ha visto este algoritmo en tres versiones diferentes: Bellman-Ford, Bellman-Ford optimizado y Bellman-Ford optimizado con una nueva modificación. Además se ha visto otra forma de emplear la PD, a la cual le llamamos Kwok-Driessen. En esta sección se

discutirá cada uno de los métodos de PD vistos y se verá el efecto que provoca reducir el espacio de búsqueda de los algoritmos.

Todas las simulaciones realizadas en esta sección se llevaron a cabo sobre el entorno mostrado en la figura 2. Este entorno tiene una dimensión de 101x101 unidades, donde cada unidad es un estado por el que puede transitar el robot. Todos los puntos negros del entorno representan obstáculos.

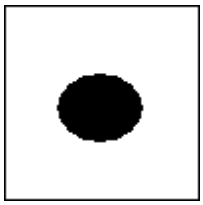

Figura 2. Entorno, dimensiones 101x101 unidades. Cada unidad constituye un estado.

A continuación se analizarán las tres versiones del algoritmo de Bellman-Ford en cuanto a la cantidad de estados explorados del espacio de búsqueda. En la figura 3 y la tabla 1 se pueden ver los resultados obtenidos de la simulación realizada.

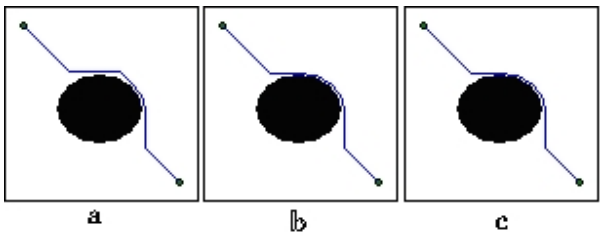

Figura 3. **a)** Resultados de la simulación con el algoritmo de Bellman-Ford. b) Resultados de la simulación con el algoritmo de Bellman-Ford optimizado. c) Resultados de la simulación con el algoritmo de Bellman-Ford optimizado y modificado. El costo de todas las trayectorias es 128.36 unidades, para la función objetivo definida como la distancia euclidiana.

Tabla 1. Resultados de la simulación

Algoritmo	Bellman-Ford	Bellman-Ford Optimizado	Modificación de Bellman-Ford Optimizado
Estados del entorno	10201	10201	10201
Etapas	115	115	109
Estados explorados	1173115	24380	24037
Costo óptimo	128.36 unidades	128.36 unidades	128.36 unidades

La cantidad de estados que presenta el entorno utilizado en esta simulación es de 10201 ya que se asumió que cada unidad del entorno constituye un estado. Dentro de este total se encuentran los estados prohibidos y los estados libres. Las etapas no son más que la cantidad de iteraciones que realiza el algoritmo hasta que encuentra la solución óptima. Es importante destacar que el rendimiento del algoritmo no se debe valorar solo por las iteraciones que este realiza sino más bien por la cantidad de estados que tiene que explorar. La cantidad de estados explorados es el factor fundamental para evaluar la rapidez de ejecución de estos tres algoritmos ya que mientras más estados sean necesario visitar y calcular su costo, más tiempo se esta consumiendo. Debe observarse que el algoritmo de Bellman-Ford optimizado y modificado solo explora 24037 estados a diferencia de los 1173115 y 24830 estados que exploran las otras dos versiones. Como se puede ver la reducción del espacio de búsqueda es enorme. Hay que destacar también que la cantidad de estados explorados es mayor que la cantidad de estados que tiene el entorno, pues esto no entra en contradicción porque lo que sucede es que existen estados que se han explorado más de una vez, para entender esto con más claridad se debe recurrir al seudo-código del algoritmo o a su definición.

La siguiente simulación tiene por objetivo comparar el algoritmo de Kwok-Driessen con la versión modificada de Bellman-Ford optimizado. Los resultados se pueden observar en la figura 4 y la tabla 2.

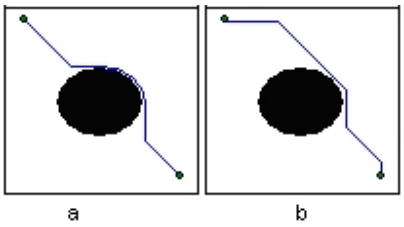

Figura 4. a) Resultados de la simulación con la modificación del algoritmo de Bellman-Ford optimizado. b) Resultados de la simulación con el algoritmo de Kwok-Driessen. El costo de todas las trayectorias es 128.36 unidades, para la función objetivo definida como la distancia euclidiana.

Tabla 2**.** Resultados de la simulación

Algoritmo	Modificación de Bellman-Ford Optimizado	Kwok-Driessen
Estados del entorno	10201	10201
Etapas	109	106
Estados explorados	24037	1081306
Costo óptimo	128.36 unidades	128.36 unidades

El método de Kowk-Driessen tiene sus orígenes en el algoritmo de Bellman de la trayectoria más corta, ahora adaptándolo a una representación matricial de los estados para hacer coincidir de esta manera cada estado con una localización (i,j) dentro del entorno por el cual se mueve el robot y definiendo además el conjunto de estados prohibidos. Pero la modificación fundamental que introducen Kwok y Driessen es la detención de la búsqueda cuando el costo del estado inicial deja de ser -1. Esto tiene una implicación importante y es que al detener la búsqueda cuando aparece el primer camino se pueden desechar soluciones mejores que la

encontrada, cuando la función objetivo no sea la definida por Kwok y Driessen. Por lo tanto el algoritmo pierde generalidad comparado con los anteriores ya que este, tal y como está definido, solo obtendrá soluciones óptimas con funciones objetivos como la distancia euclidiana.

En cuanto a la reducción del espacio de búsqueda este algoritmo introduce leves mejoras respecto al de Bellman-Ford sin optimizar ya que realiza menos etapas al detenerse cuando aparece el primer camino al estado objetivo, pero aún sigue explorando en cada etapa todos los estados. No obstante, las mejoras que puedan provocar estas modificaciones en el tiempo de ejecución son insignificantes comparadas con los algoritmos de Bellman-Ford optimizado y la modificación de Bellman-Ford optimizado. Esto se puede apreciar claramente en las tablas 1 y 2.

Por último, para poder apreciar como el algoritmo Kwok-Driessen no sirve para cualquier tipo de función de costo se muestra la siguiente simulación, igual que las anteriores solo que ahora se define como función objetivo al cubo de la distancia euclidiana entre dos estados. Los resultados se pueden observar en la figura 5 y la tabla 3.

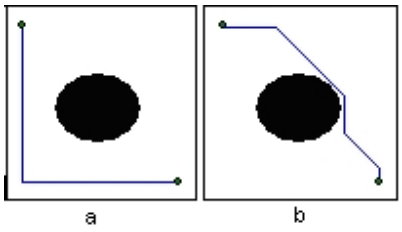

Figura 5. a) Resultados de la simulación con la modificación de Bellman-Ford optimizado. Costo de la trayectoria 160 unidades. b) Resultados de la simulación con el algoritmo Kwok-Driessen. Costo de la trayectoria 204.70 unidades. En todos los casos la función objetivo fue definida como el cubo de la distancia euclidiana.

Tabla 3. Resultados de la simulación

Algoritmo	Modificación de Bellman-Ford Optimizado	Kwok-Driessen
Estados del entorno	10201	10201
Etapas	161	106
Estados explorados	193887	1081306
Costo óptimo	128.36 unidades	204.74 unidades

En la figura 5 se observa como las trayectorias obtenidas por cada método son diferentes. La trayectoria obtenida con el método de Kwok-Driessen posee un costo superior por lo que no constituye la trayectoria óptima para llegar desde el estado inicia al estado final. La razón por la cual la trayectoria obtenida por este último método no es la óptima es la que se mencionaba con anterioridad, se detiene la búsqueda cuando se encuentra la primera.

Obsérvese también que la cantidad de estados explorados por el método de Bellman-Ford optimizado y modificado es muy inferior que la cantidad que explora el algoritmo Kwok-Driessen, lo cual permite verificar una vez más la superioridad de este método.

CONCLUSIONES

En este trabajo se han estudiado varios tipos de algoritmos que usan la programación dinámica para encontrar una trayectoria entre dos estados cualesquiera. Se comenzó por el algoritmo de Bellman-Ford y su versión optimizada, a la cual se le introdujo una modificación para reducir la cantidad de estados explorados. Posteriormente se analizó otro método de planificación de trayectorias propuesto en [3] que también usa la programación dinámica.

Todos los algoritmos se discutieron y compararon mediante la realización de varias simulaciones en un entorno estático.

El algoritmo de Bellman-Ford sin optimizar resultó ser el que más estados exploraba en la búsqueda de la solución por lo que fue, de todos, el que más tiempo de ejecución consumió. No obstante, las soluciones encontradas siempre fueron óptimas.

El método Kowk-Driessen resultó ser un poco más eficiente que el de Bellman-Ford sin optimizar en cuanto al tiempo de ejecución ya que explora menos estados en la búsqueda de la solución, pero las soluciones que se encuentran no siempre son óptimas, esto depende de la función objetivo que se halla usado, por cuanto este solo es válido cuando se use como función de costo la distancia.

En todos los casos el método propuesto, modificación de Bellman-Ford Optimizado resultó ser más eficiente que los restantes pues su tiempo de ejecución siempre fue menor debido a que exploraba menos estados y sus soluciones siempre fueron la óptima para las funciones objetivos utilizadas, por cuanto el método garantiza su generalidad para cualquier tipo de función de costo.

REFERENCIAS

1. **BELLMAN, R.:**, "Dynamic Programming", Princeton University Press, 1957.
2. **CORMEN, THOMAS H.; LEISERSON, C. E.; RIVEST, R. L. Y STEIN, C.:** "Introduction to algorithms". Second Edition, MIT Press and McGraw-Hill, 2001.
3. **KWOK, K.S. Y DRIESSEN, B. J. :** "Path Planning for Complex Terrain Navigation Via Dynamic Programming", in Proceedings of the American Control Conference, 1999.
4. **BARRAQUAND, JEROME Y FERBACH, PIERRE:** "Path planning through variational dynamic programming", Technical report., Paris Research Laboratory, 1993.

5. **SARKAR, SAURABH :** "Path planning and obstacle avoidance in mobile robots", Msc. thesis. University of Cincinnati., 2007.
6. **MORENO, V. Y AEYELS, D.:** "Minimización del tiempo de vuelo de satélites usando Programación Dinámica", RIAI, 2006.

AUTORES

Maikel O. Torres Piñeiro, Ingeniero en Automática, profesor instructor, Instituto Superior Politécnico José Antonio Echeverría, 266-3341, maikel@electrica.cujae.edu.cu., Investiga en el área de la robótica móvil.

Valery Moreno Vega, Ingeniero en Máquinas Computadoras, máster en Informática Aplicada, Doctor en Ciencias Técnicas, profesor titular, Instituto Superior Politécnico José Antonio Echeverría, 266-3343 valery@electrica.cujae.edu.cu., Actualmente realiza investigaciones en el área de robótica, informática aplicada a la automatización y en aplicaciones para el control utilizando métodos de inteligencia artificial.