Modelo teórico y generación de instancias de prueba para problemas RCCN (Reducing Collisions in Constrained Navigation)

Josué Oregel, Héctor J. Puga, Martín Carpio, Víctor Zamudio, Manuel Ornelas, and Luis Mancilla Espinoza

División de Estudios de Posgrado e Investigación
Instituto Tecnológico de León
León, Guanajuato, 37290, México
{josue-oregel,hector.puga}@itleon.edu.mx
http://posgrado.itleon.edu.mx

Resumen La presencia de colisiones o encuentros entre objetos móviles se considera un problema que genera un impacto negativo en sistemas de navegación reales. En este artículo se introducen los problemas de navegación a los que hemos llamado *Reducing Collisions in Constrained Navigation(RCCN)*, los cuales están relacionados con problemas que buscan satisfacer restricciones de enrutamiento o planificación en navegación como Problemas de Enrutamiento de Vehículos (VRP) y Vehículos Guiados Autónomos (AGV), donde RCCN por sus características es diferente de ambos. Se presenta un modelo que permite detectar colisiones o encuentros entre objetos móviles en sistemas de navegación con restricciones RCCN y provee una plataforma para la reducción de las colisiones o encuentros entre los móviles del sistema. También se presenta un generador de instancias de prueba con las características de los problemas RCCN.

Palabras clave: Navegación, Colisiones, Instancias de Prueba.

1. Introducción

En este trabajo se entiende un sistema de navegación como un sistema que está compuesto por un entorno de navegación, rutas para navegar, objetos móviles (o simplemente móviles) y una planificación para navegar por las rutas.

En sistemas inteligentes de navegación se pretende que un móvil esté dotado con cierto nivel de autonomía para reaccionar y tomar decisiones basándose en las condiciones del entorno, pero contrario a lo que sucede con los seres vivos, el desplazamiento de un ente artificial no se considera una tarea trivial dado que se enfrenta a problemas como percepción y modelado del entorno y planificación del desplazamiento por sus trayectorias evitando conflictos como colisiones, puntos muertos y congestiones, requiriendo ninguna intervención humana [3,8,11,12,13,18].

La principal aportación de este artículo es la construcción de estructuras esenciales para la futura resolución del problema de reducción del total de colisiones o encuentros entre móviles mediante técnicas de optimización inteligente en sistemas de navegación con las restricciones de una clase específica de problemas de satisfacción de restricciones de navegación a la que hemos llamado *Reducing Collisions in Constrained Navigation (RCCN)*, que son problemas relacionados con Problemas de Enrutamiento de Vehículos (Vehicle Routing Problems, VRP) y Vehículos Guiados Automatizados (Automated Guided Vehicles, AGV), pero con características propias que los diferencían de estos [1,2,4,5,6,7,9,10,15,16,17].

Este documento está estructurado de la siguiente manera: en la Sección 2 se presentan las restricciones de un problema RCCN y el modelo propuesto, en la Sección 3 se presenta el generador de instancias de prueba además de una nomenclatura de instancias generadas utilizando el generador programado en Java. Finalmente, se presentan conclusiones y se menciona el trabajo desarrollado actualmente y su aplicación futura.

2. El modelo para problemas Reducing Collisions in Constrained Navigation (RCCN)

2.1. Restricciones para problemas RCCN

- El entorno de navegación es modelado como un mapa de rutas considerado como un grafo no dirigido que conecta sus nodos (o vértices) con aristas navegables y es generado a partir de una nube de puntos.
- Una ruta es una sucesión de aristas conectadas por nodos del grafo navegables por objetos móviles.
- Un nodo puede pertenecer a más de una ruta de navegación.
- Una arista pertenece a máximo una ruta y tiene definido únicamente un sentido de navegación.
- Una ruta es navegable únicamente por un móvil y este pertenece únicamente a una ruta, por lo tanto, existen tantas rutas como móviles en el sistema.
- Las rutas están predefinidas y son fijas para todos y cada uno de los móviles dentro del sistema de navegación.
- Para propósitos de estudio, los móviles son considerados como elementos puntuales, es decir, sin dimensión.
- La velocidad de recorrido de las rutas es la misma para todos los móviles, es constante y toma valores discretos.
- Un evento de colisión ocurre si hay un encuentro entre móviles en el mismo nodo al mismo tiempo dentro del mapa de ruta, es decir, no está permitido alojar más de un móvil en un nodo a la vez excepto en nodos de inicio o fin de ruta, similar a lo que ocurre en terminales reales.
- Es posible asignar tiempos de espera ($te \mid te \in \mathbb{N} \cup \{0\}$) para cada móvil en cada nodo de su ruta, y puede ser cambiado en intervalos discretos de tiempo ($\Delta te \in \mathbb{N}$), para evitar posibles colisiones en nodos comunes de su ruta con otras rutas.
- El inicio del recorrido de las rutas es simultáneo para todos los móviles, es decir, no están permitidos los tiempos de espera en nodos de inicio de ruta.

2.2. Modelo para detección y estrategia para eliminar colisiones

El modelo está estructurado en un procedimiento de 3 pasos:

Paso 1. Cálculo de tiempo de ocupación de nodos: Suponiendo que se cuenta con un entorno de navegación y además se cuenta con rutas para navegar en el entorno, entonces se cuenta con un grafo del sistema con n nodos y con este grafo se obtienen dos conjuntos de datos que son los insumos del modelo:

- Una "*Matriz de Conectividad Pesada*" (Mc) de tamaño n x n, que es una matriz de adyacencia con pesos que representa al entorno de navegación y que se obtiene del grafo. Cada entrada de esta matriz contiene la información de conectividad y tiempo de recorrido $t_{(i,j)}$ [1] entre los nodos i y j del entorno de navegación (Tabla 1).

Tabla 1. Información de $t_{(i,j)}$ contenida en las entradas de la matriz Mc.

Valor	Representación
-1	No hay una arista entre los nodos i y j (no hay conectividad).
0	Es el mismo nodo ($i = j$) (no hay conectividad).
$t_{(i,j)} \in \mathbb{N}$	Tiempo para recorrer la arista entre los nodos i y j (hay conectividad).

- Un conjunto de m rutas R_k, donde cada ruta es representada por una sucesión de s nodos $n_{(k,l)}$ del grafo (Ecuación (1)). El valor de s es independiente para cada ruta.

$$R_k = n_{(k,1)}, n_{(k,2)}, \ldots, n_{(k,l)}, \ldots, n_{(k,s)} \text{ donde } 1 \leq k \leq m, 1 \leq l \leq s,\ k, s \in \mathbb{N} \ . \tag{1}$$

A partir de la información anterior, se calcula el tiempo de ocupación de un nodo de las rutas y se registra en una *Matriz de Colisiones* M_{col}. En esta matriz cada entrada representa el tiempo que toma a un móvil llegar a un nodo de su ruta más el tiempo que permanece el móvil ocupando el nodo. Para generar la matriz M_{col}, se utilizan las siguientes tres matrices de tamaño m x n: la *Matriz de Tiempo de Recorrido Original* (Mto), la *Matriz de Tiempo de Espera* (Mte) y la *Matriz de Tiempo de Retardo* (Mtr).

- *Matriz Mto*: Cada entrada $to_{(k,i)}$ de esta matriz contiene el tiempo de recorrido desde el nodo inicial hasta cada uno de los s nodos en cada ruta k (Ecuación (2)):

$$to_{(k,i)} = to_{(k,1)} + to_{(k,2)} + \cdots + to_{(k,i-2)} + to_{(k,i-1)} \ . \tag{2}$$

[1] Aunque es común en lenguaje matemático el uso de la notación t_{ij} para hacer referencia a las entradas de una matriz, en este documento se ha utilizado una notación para subíndices usando paréntesis y coma $t_{(i,j)}$ con la intención de facilitar a los autores un manejo menos confuso de los mismos.

Los posibles valores contenidos en las entradas de la matriz Mto son los presentados en la Tabla 2:

Tabla 2. Posibles valores de $to_{(k,i)}$ en las entradas de la matriz Mto.

Valor	Representación
-1	El nodo i no pertenece a la ruta k.
0	Es el nodo de inicio de la ruta k ($to_{(k,i)} = 0$, $i = 1$).
$to_{(k,i)} \in \mathbb{N}$	Tiempo para recorrer la ruta k desde el inicio hasta el nodo i.

- *Matriz Mte*: Esta matriz contiene los valores de tiempo de espera $te_{(k,i)}$ asignados a cada uno de los s nodos de las k rutas.
- *Matriz Mtr*: En esta matriz cada entrada $tr_{(k,i)}$ contiene el tiempo de retardo en los s nodos de las k rutas, provocado por la acumulación del tiempo esperado por los móviles en nodos anteriores de la ruta k (Ecuación (3)):

$$tr_{(k,i)} = te_{(k,1)} + te_{(k,2)} + \cdots + te_{(k,i-2)} + te_{(k,i-1)} \ . \tag{3}$$

El intervalo de tiempo total $tt_{(k,i)}$ que un móvil permanece ocupando un nodo i de su ruta k, es el comprendido entre el tiempo acumulado $ta_{(k.i)}$ (Ecuación (4)) y la suma del tiempo acumulado con el tiempo de espera asignado en el nodo i (Ecuación (5)).

$$ta_{(k,i)} = to_{(k,i)} + tr_{(k,i)} \ . \tag{4}$$

$$tt_{(k,i)} = [ta_{(k,i)}, ta_{(k,i)} + te_{(k,i)}] \ . \tag{5}$$

La matriz M_{col} almacena los intervalos de tiempo $tt_{(k,i)}$ y es el arreglo que permite al modelo realizar la detección de colisiones en un sistema de navegación con las características de los problemas RCCN.

Paso 2. Cálculo de eventos de colisión: Cuando un nodo i común a dos rutas k y l no es inicio o final de una de las rutas y existe una intersección en los intervalos de tiempo total de recorrido hasta el nodo, se identifica un evento de colisión por $c_{([k,l],i)}$ (Ecuación (6)). Para obtener el total de colisiones TCS en el sistema de navegación, se calcula la suma del total de eventos de colisión en cada nodo del sistema (Ecuación (7)). El Algoritmo 1 presenta el pseudocódigo para el cálculo de eventos de colisión.

$$c_{([k,l],i)} = \begin{cases} 1 \text{ si } tt_{(k,i)} \cap tt_{(l,i)} \neq \emptyset \\ 0 \quad \text{de otra forma} \end{cases} \text{, donde } i \neq 1, i \neq s \ . \tag{6}$$

$$TCS = \sum_{i=1}^{n} \sum_{k=1}^{m-1} \sum_{l=k+1}^{m} c_{([k,l],i)} \ . \tag{7}$$

Algoritmo 1 Pseudocódigo del cálculo de eventos de colisión

Entrada: Una Matriz de Conectividad Mc
Entrada: Un conjunto de m rutas de navegación
1: Calcular los intervalos de tiempo $tt_{(k,i)}$ de las entradas de la Matriz de Colisiones $Mcol$
2: Inicializar contador de colisiones $TSC = 0$
3: **para** (cada nodo i del sistema de navegación; $1 \leq i \leq n$) **hacer**
4: **para** (cada ruta k del sistema de navegación; $1 \leq k \leq m-1$) **hacer**
5: **para** (cada ruta l del sistema de navegación; $j+1 \leq l \leq m$) **hacer**
6: **si** $(tt_{(k,i)} \cap tt_{(l,i)} \neq \emptyset)$ **entonces**
7: Aumentar el contador de colisiones $TSC = TSC + 1$
8: **fin si**
9: **fin para**
10: **fin para**
11: **fin para**
12: **devolver** Contador de colisiones TSC

Paso 3. Estrategia para eliminación de eventos de colisión: Para eliminar un evento de colisión entre objetos móviles en un nodo común a dos rutas diferentes, se selecciona una de las rutas y se establece un cambio en el tiempo de espera Δte del nodo anterior al nodo en conflicto (Ecuación (8)). Este cambio en el tiempo de espera produce el retardo en la llegada del móvil de la ruta elegida hacia el nodo común con evento de colisión, llegando después del móvil que se desplaza por la otra ruta, eliminando el evento de colisión.

$$te_{(k,i-1)} = te_{(k,i-1)} + \Delta te, \text{ donde } i \text{ es un nodo con evento de colisión } . \quad (8)$$

Es posible asignar en los nodos valores arbitrarios de tiempo de espera para reducir colisiones; sin embargo, como es deseable que en todo sistema de navegación se cumpla con el recorrido de las rutas en el menor tiempo posible, es recomendable asignar tiempos de espera cercanos a cero para afectar lo menos posible el tiempo de recorrido original.

3. Generador de instancias de prueba

Para probar el rendimiento de propuestas de solución a problemas RCCN es necesario contar con instancias de prueba adecuadas. En el estado del arte existen instancias de prueba para problemas VRP, AGV y más variantes, que se han utilizado en solución a problemas de navegación (e.g. generación de rutas de navegación óptimas, planificación de navegación sobre las rutas, etc.). Sin embargo, no se han encontrado instancias que pudieran usarse para probar el rendimiento de propuestas de solución a problemas RCCN conservando las características del modelo presentado, por lo que ha sido necesario generar un conjunto de instancias de prueba confiable y adecuado.

En esta sección se presenta un generador de instancias de prueba que cumple con las restricciones de problemas RCCN, la construcción se explica en dos pasos: la generación de rutas de navegación y la generación de tiempos de recorrido de las rutas de navegación.

3.1. Generación de rutas de navegación

El generador se basa en una nube de n puntos (o nodos) como entorno para la generación de rutas de navegación. Las rutas de navegación están definidas como una sucesión de nodos y cada ruta se compone de una cantidad independiente de nodos, lo que permite la aplicación de una entre dos opciones de configuración: la misma cantidad de nodos para cada ruta o cantidades diferentes de nodos para cada ruta.

Si se considera el conjunto de rutas generadas como parte de un sistema de navegación, entonces el número de *aristas totales* At requeridas para generar las rutas del sistema se calcula a partir de la cantidad s_k de nodos de cada ruta, (Ecuación 9):

$$At = \sum_{k=1}^{m} (s_k - 1) \ . \tag{9}$$

Por otro lado, la cantidad de *aristas útiles* Au para generar rutas a partir de un nube de puntos de n puntos es (Ecuación (10)):

$$Au = \frac{n(n-1)}{2} \ . \tag{10}$$

Si $Au \geq At$, teóricamente es posible con los n nodos del grafo completo generar el sistema de navegación con At aristas, de otra forma, el sistema no se podrá generar. El Algoritmo 2 presenta el pseudocódigo para la generación de rutas de navegación.

3.2. Generación de tiempos de recorrido de rutas

Para generar colisiones usando las rutas generadas, se ubican los nodos comunes y se asigna tiempo de recorrido a cada segmento de las rutas, de tal forma que la suma de tiempo de recorrido hasta los nodos comunes sea la misma para cada ruta involucrada. El algoritmo 3 presenta el pseudocódigo para la asignación de tiempos de recorrido a los segmentos de ruta del sistema de navegación.

Algoritmo 2 Pseudocódigo para la generación de rutas de navegación

Entrada: n: cantidad de nodos del sistema de navegación
Entrada: m: cantidad de rutas que se van a generar
Entrada: s_k: cantidad de nodos de R_k

para cada R_k desde 1 hasta m **hacer**
  seleccionar aleatoriamente un nodo y asignarlo a R_k como nodo inicial ($n_{(k,1)}$)
  para cada i desde 2 hasta s_k **hacer**
    repetir
      seleccionar aleatoriamente un nodo ($n_{(k,i)}$)
      si $n_{(k,i)}$ es diferente de $n_{(k,i-1)}$ (nodo anterior en la ruta) **entonces**
        si aún no existe una arista entre $n_{(k,i)}$ y $n_{(k,i-1)}$ **entonces**
          asignar $n_{(k,i)}$ a R_k
        fin si
      fin si
    hasta que ha sido asignado un $n_{(k,i)}$ a R_k
  fin para
fin para
devolver m *rutas* de navegación.

Algoritmo 3 Pseudocódigo para la asignación de tiempo de recorrido de rutas

Entrada: m rutas de navegación

para cada R_k desde 1 hasta m **hacer**
  para cada $n_{(k,i)}$ desde 1 hasta s_k **hacer**
    si $n_{(k,i)}$ es un nodo común de R_k y otra ruta **entonces**
      aumentar el contador de cantidad de nodos comunes en R_k
    fin si
  fin para
fin para
mnc = índice de la ruta con la mayor cantidad de nodos comunes
asignar un tiempo de recorrido igual a cero a $n_{(mnc,1)}$
para cada $n_{(mnc,i)}$ desde 2 hasta s_{mnc} **hacer**
  asignar aleatoriamente un tiempo de recorrido a $n_{(mnc,i)}$
fin para
para cada R_k desde 1 hasta m excepto mnc **hacer**
  para cada $n_{(k,i)}$ desde 1 hasta s_k **hacer**
    si $n_{(k,i)}$ es un nodo común entre R_k y R_{mnc} **entonces**
      generar un tiempo de recorrido hasta $n_{(k,i)}$ igual al de $n_{(mnc,i)}$
    fin si
  fin para
fin para
si aún hay nodos comunes **entonces**
  para cada R_k desde 1 hasta $m-1$ excepto mnc **hacer**
    para cada R_l desde R_{k+1} hasta m excepto mnc **hacer**
      para cada $n_{(k,i)}$ desde 1 hasta s_k **hacer**
        para cada $n_{(l,i)}$ desde 1 hasta s_l **hacer**
          si $n_{(k,i)} = n_{(l,i)}$ **entonces**
            generar un tiempo de recorrido hasta $n_{(k,i)}$ igual al de $n_{(l,i)}$
          fin si
        fin para
      fin para
    fin para
  fin para
fin si
si hay trayectos sin tiempos de recorrido **entonces**
  asignar tiempo de recorrido para los trayectos faltantes
fin si
devolver una Matriz de Conectividad Pesada Mc

3.3. Instancias de prueba generadas

La generación de instancias de prueba se llevó a cabo con un programa en Java [14] que entrega dos archivos para cada instancia: un archivo `.mcc` que contiene la Matriz de Conectividad Pesada Mc y un archivo `.rtc` que contiene m rutas de navegación. Estos dos archivos son las dos estructuras de datos que se usan como insumo en el modelo para problemas RCCN.

Las instancias tienen la nomenclatura: `#N-#R-#C-ver#`, donde N hace referencia al número de nodos, R al número de rutas, C al número de colisiones del sistema y puesto que es posible generar diferentes mapas de rutas con la misma configuración de nodos, *ver* hace referencia a la versión de la instancia generada, comenzando con 1.

El valor de semilla utilizado en el método *Random* del programa Java ha sido almacenado en los archivos generados para brindar la posibilidad de reproducir cada instancia de prueba.

Algunos ejemplos de nomenclatura de instancias de prueba generadas se muestran en la Tabla 3.

Tabla 3. Ejemplos de nomenclatura de instancias de prueba RCCN generadas con un programa en Java.

Nombre de Instancia	Semilla Java Random	Nodos por ruta
27N-5R-5C-ver1	-3770157802445094294	27
27N-5R-7C-ver1	-6075749904103534460	27
27N-5R-8C-ver1	4480157765928300129	27
27N-5R-11C-ver1	53830088977007794	27
27N-5R-12C-ver1	-4538243863454794414	27
100N60R-676C-ver1	1242326594992500141	Entre 60 y 90
150N-90R-941C-ver1	242298037008080666	Entre 80 y 105
200N-80R-1089C-ver1	-7332834085541032903	Entre 120 y 200
250N-200R-4879C-ver1	3618874879153313709	Entre 100 y 130
300N-120R-1739C-ver1	6519364232554299876	Entre 200 y 300
400N-300R-10541C-ver1	2333333709128786646	Entre 150 y 250
500N-300R-10293C-ver1	4932274200351035357	Entre 250 y 390
1000N-650R-75841C-ver1	-5907896972978185640	Entre 700 y 770
2000N-1000R-91390C-ver1	-9174493999035787771	Entre 1,000 y 1,500

4. Conclusiones

La presencia de conflictos, como colisiones, en sistemas de navegación se considera un problema debido al impacto negativo que los conflictos provocan en los sistemas reales. En sistemas de navegación donde la cantidad de colisiones entre móviles es grande puede ser necesario contar con herramientas de planificación que mejoren el desempeño del sistema.

En este trabajo se ha presentado una propuesta que puede permitir un mejor desempeño de un sistema de navegación. La propuesta contiene los siguientes dos elementos:

- Un modelo para problemas a los que se ha llamado *Collision Reduction in Constrained Navigation (RCCN)*, que permite detectar colisiones o encuentros entre objetos móviles y provee una plataforma para reducir estas utilizando técnicas de optimización inteligente.
- Un generador de instancias de prueba con las características de problemas RCCN, que son la base para comparar el rendimiento de técnicas de optimización inteligente aplicadas a problemas RCCN.

El modelo se basa en un procedimiento de 3 pasos: Calcula el tiempo que cada móvil permanece ocupando cada uno de los vértices de su ruta, considerando el intervalo entre el instante en que el móvil llega al nodo hasta el instante en que lo abandona. Calcula el número de eventos de colisión presentes en el sistema de navegación, registrando un evento de colisión cada vez que se detecta más de un móvil en un nodo común a las rutas analizadas en el intervalo de tiempo de ocupación del nodo por parte de cada móvil. Finalmente, provee una estrategia para el ajuste de tiempo de espera que permanecerán los móviles en nodos anteriores a los nodos en conflicto de su ruta para evitar eventos de colisión con otros móviles.

El generador construye instancias en dos pasos: la generación en forma aleatoria de rutas en sistemas de navegación de n nodos y la generación de colisiones en nodos comunes entre las rutas mediante la asignación de tiempo de recorrido de las rutas.

Es importante mencionar que la aplicación del modelo permitirá, en primera instancia, el uso de técnicas de optimización inteligente en sistemas de navegación teóricos, para luego ser implementado en sistemas de navegación reales, como transporte de suministros en sistemas flexibles de manufactura, técnicas *store and forward* en redes de computadoras o coordinación de actividades para robots asistentes en oficinas. Actualmente se está trabajando en heurísticas basadas en técnicas de optimización como *Particle Swarm Optimization (PSO)* y *Artificial Immune System (AIS)*.

5. Agradecimientos

Los autores agradecen el soporte del Consejo Nacional de Ciencia y Tecnología (CONACYT) y de la Dirección General de Educación Superior Tecnológica (DGEST) proyecto número 4572.12-P.

Referencias

1. Baldoni, R., Bonnet, F., Milani, A., Raynal, M.: Anonymous graph exploration without collision by mobile robots. Information Processing Letters **109**(2) (2008) 98–103
2. Chiew, K., Qin, S.: Scheduling and routing of amos in an intelligent transport system. IEEE Transactions on Intelligent Transportation Systems **10-3** (09 2009) 547–552
3. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S.: Principles of robot motion: theory, algorithms, and implementations. MIT press (2005)
4. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management science **6**(1) (1959) 80–91
5. Gendreau, M., Potvin, J., Bräumlaysy, O., Hasle, G., Løkketangen, A.: Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. The vehicle routing problem: Latest advances and new challenges (2008) 143–169
6. Ghasemzadeh, H., Behrangi, E., Abdollahi, A.M.: Conflict-free scheduling and routing of automated guided vehicles in mesh topologies. Robotics and Autonomous Systems **57**(6-7) (2009) 738–748
7. Gong, F., Wang, X.: Robot path-planning based on triangulation tracing. In: Intelligent Information Technology Application Workshops, 2008. IITAW'08. International Symposium on, IEEE (2008) 713–716
8. Jianyang, Z., Wen-Jing, H.: Conflict-free routing of agvs on the mesh topology based on a discrete-time model. In: Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on. Volume 3., IEEE (2003) 3510–3516
9. Jinghui, L., Jingbo, S., Yan, Z.: A fast approach to judging basic element intersection in polygons meshing. In: Digital Manufacturing and Automation (ICDMA), 2011 Second International Conference on, IEEE (2011) 20–23
10. Jose, A.A., Adarsh, S.B., Anju C., S.P.: A novel approach for scheduling and routing of the self guided vehicles in mesh topology using velocity control and alternate path techniques. In: 2011 International Conference on Process Automation, Control and Computing (PACC), IEEE (2011) 1–5
11. Kruusmaa, M., Willemson, J.: Covering the path space: a casebase analysis for mobile robot path planning. Knowledge-Based Systems **16**(5) (2003) 235–242
12. LaValle, S.: Planning algorithms. Cambridge university press (2006)
13. Mirtich, B.: Efficient algorithms for two-phase collision detection. Practical motion planning in robotics: current approaches and future directions (1997) 203–223
14. Oracle, I.: Java se documentation at a glance. http://www.oracle.com/technetwork/es/java/javase/documentation/index.html (01 2013)
15. Peasgood, M., Clark, C., McPhee, J.: A complete and scalable strategy for coordinating multiple robots within roadmaps. Robotics, IEEE Transactions on **24**(2) (2008) 283–292
16. Peng, J., Akella, S.: Coordinating multiple double integrator robots on a roadmap: Convexity and global optimality. In: Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, IEEE (2005) 2751–2758
17. Qiu, L., Hsu, W., Huang, S., Wang, H.: Scheduling and routing algorithms for agvs: a survey. International Journal of Production Research **40**(3) (2002) 745–760
18. Rusell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. 2 edn. Pearson Education, Inc. (2002)