Multi-goal Path Planning Autonomous System for Picking up and Delivery Tasks in Mobile Robotics

K. Hernández, B. Bacca and B. Posso, *Member, IEEE*

***Abstract*— Intelligent service robots are needed in office-like environments to perform common tasks of picking up, and delivering things such as mail, goods, trash recycled paper, etc. These tasks are challenging since robots must avoid static and dynamic obstacles, and mainly robots have to perform path planning considering multiple goals as saving energy. This work proposes an autonomous multi-goal path planning system for picking up or delivering tasks in mobile robotics. The multi-goal path planning method is based on the *Lin-Kernighan Heuristics* (LKH) algorithm [1], which was modified in order to implement an autonomous system for picking up/delivering tasks using non-Euclidean distances, Hamiltonian paths, and a Pioneer 3DX mobile robot. This work proposes a client – robot system ARMM [2] where many clients request pickup / delivery services, then the robot continuously plan a Hamiltonian path to visit each one of the requested pickup / delivery goals, and return to its base station. To validate the results of this work two well-known metrics were performed [3]: distance traveled and time elapsed. A comparison between the nearest goal, random selection, the LKH with Euclidean distances, and the LKH with non-Euclidean distances algorithms were performed.**

***Keywords*— Multi-goal planner, traveling salesman problem, pickup or delivery tasks, service robotics.**

I. INTRODUCCIÓN

LA ROBÓTICA de servicio ha hecho parte de la comunidad científica y programas gubernamentales para la innovación y la ciencia en la última década. Por ejemplo, la agenda de investigación estratégica de la Red Europea de Investigación en Robótica (EURON) [4] compiló el esfuerzo de la industria y la academia en dos grandes proyectos: la Coordinación de Acción para la Robótica en Europa (CARE) y la Plataforma de Tecnología Robótica Europea (EUROP), para fortalecer la competitividad Europea en investigación, desarrollo y mercadeo global en robótica hasta el 2020. Los sectores económicos relacionados en este documento son: industria, servicios profesionales y domésticos, seguridad y espacio. Cada uno con escenarios de aplicación como: trabajadores robóticos, compañeros de trabajo robóticos, robots para logística, vigilancia e intervención, exploración e inspección, y entretenimiento. Este trabajo se enfoca en tareas de recolección y entrega, muy comunes en sectores industriales y de servicios profesionales. Estas tareas tienen diversos aplicaciones como: manipulación automática de bienes en almacenes [5], entrega de medicamentos [6], recolector de basura urbana [7] y robots mensajeros [8]. Estas aplicaciones son un reto para los robots de servicio ya que involucran entornos dinámicos, métodos adecuados de representación de entornos, navegación autónoma y estrategias de planificación de rutas especialmente diseñadas. En este último aspecto, las tareas de recolección y entrega poseen escenarios que involucran ir del punto A al B, y luego al C, D, etc. Lo cual se conoce como planificación de múltiples metas. Además, el camino seleccionado necesita ser cercano a una solución óptima, ya que el robot tiene recursos energéticos limitados.

En este artículo, se propone un sistema autónomo de planificación de rutas multi-metas para tareas de recolección o entrega (ARMM) [2]. Se usa un mapa 2D construido usando un sensor láser (LRF) en el que se posicionan diferentes metas. Muchos clientes pueden solicitar servicios de entrega o recogida en cualquier momento; luego el sistema planificador propuesto calcula el camino Hamiltoniano más corto en tiempo real, lo que significa que todas las metas de la ruta serán visitadas solo una vez. Después, el robot navega por la ruta calculada. En cada meta, se requiere una validación humana para recoger o entregar bienes. Al final, el robot termina su recorrido en la position *home.*

Este artículo está organizado de la siguiente manera: la sección II describe los trabajos relacionados; la sección III describe la propuesta; en la sección IV están las pruebas realizadas y la sección V muestra las conclusiones.

II. TRABAJOS RELACIONADOS

Muchos métodos de planificación se han propuesto para ir del punto A al B como A* [9], B* [10] y D* [11]. Otras propuestas involucran restricciones cinemáticas en los métodos de planificación obteniendo trayectorias suaves ajustadas a la geometría del robot [12]. Los robots de servicio no son requeridos para ir del punto A al B, sino que también al C, D, E, etc. viajando en rutas óptimas considerando métricas como distancia, tiempo o consumo de energía. Esto se conoce como el problema del agente viajero o *Traveling Salesmen Problem (TSP)* [13].

Este trabajo está enfocado en desarrollar un planificador de rutas multi-metas autónomo para aplicaciones de recogida o entrega usando un robot móvil. La Tabla 1 resume los trabajos más relevantes en esta temática, donde se considera: si la solución está basada en TSP, el enfoque de optimización escogido, si el método considera o no distancias no Euclidianas, si el trabajo propuesto genera caminos Hamiltonianos, el campo de aplicación, y si el trabajo ha sido implementado en un robot móvil.

Observando la Tabla 1, vemos que los métodos propuestos por [14] y [15] usan algoritmos basados en inteligencia computacional. A pesar que estos trabajos tienen implementaciones en el mundo real, no consideran caminos

K. Hernández, Universidad del Valle, Cali, Colombia, kevin.a.hernandez@correounivalle.edu.co
B. Bacca, Universidad del Valle, Cali, Colombia, bladimir.bacca@correounivalle.edu.co
B. Posso, Universidad del Valle, Cali, Colombia, breyner.posso@correounivalle.edu.co

Hamiltonianos. En [16] los autores proponen una solución de optimización tipo *Pareto* satisfaciendo 5 criterios a saber: el número de vehículos, la distancia recorrida, el número de solicitudes, el tiempo de espera, y la distancia de la ruta más larga. En [17] se presentan resultados de una simulación de un algoritmo de asignación de múltiples tareas para entregar suministros médicos.

TABLA I
ESTADO DEL ARTE EN PLANIFICACIÓN MULTI-OBJETIVOS

Ref.	*TSP*	*Enfoque*	*No-Euc.*	*Ruta Hamilt.*	*Aplicación*	*Implem. Real*
[14]	NO	Prog. Subastas	NO	NO	Recogida/ entrega vehículos	SI
[15]	NO	Opt. Colonia Hormigas	SI	NO	Mensajería	SI
[16]	NO	Opt. Evolutiva	SI	NO	Recogida-entrega	NO
[17]	NO	Asignación de Tareas	SI	NO	Recogida-entrega en Hospitales	NO
[18]	SI	Mapa Auto-Organizado	SI	SI	Movimiento robot hexápodo	SI
[19]	SI	Opt. Colonia Hormigas	NO	NO	Recogida de mercancía / entrega a clientes	NO
[20]	SI	A* / planif. Local	SI	SI	Exploración planetaria	NO
[21]	SI	Mapa Auto-Organizado	SI	SI	Inspección	NO
[22]	SI	Opt. Colonia Hormigas	SI	SI	Inspección submarina	NO
[23]	SI	Mapa Auto-Organizado	NO	SI	Problema de la ruta safari	NO

Otros enfoques como [18], [23] y [21] se basan en redes neuronales auto-organizadas, las cuales resuelven naturalmente el TSP. Pero, los autores afirman que se necesitan modificaciones al calcular pesos para introducir distancias no-Euclidianas y caminos Hamiltonianos. Se necesitan distancias no-Euclidianas ya que el robot debe saber la distancia real de recorrido entre dos puntos del mapa, la cual incluye a los obstáculos del ambiente de trabajo. En [18] y [21] se considera la planificación de rutas simples usando TSP; las pruebas se realizaron usando un robot hexápodo navegando en un entorno limitado, lo cual no permite concluir si el enfoque funcionaría en entornos más grandes.

En [19] y [22] se proponen métodos basados en optimización por colonia de hormigas para la planificación multi-metas. Aunque estos algoritmos se usan para tratar con problemas no lineales y de alta dimensionalidad, un conjunto de reglas heurísticas son necesarias para reducir la dimensionalidad e incrementar convergencia.

Los métodos en la Tabla I basados en mapas auto-organizados y optimización por colonia de hormigas, trabajan sin ninguna pista inicial de la solución, lo cual es una propiedad deseable. Pero, según los autores, la mayoría de estos métodos tiene problemas de escalabilidad, lo cual sería una razón por la que no tienen implementaciones reales. TSP es difícil de resolver, ya que es no lineal y de alta dimensionalidad, lo cual se incrementa con la escala. En este trabajo se propone un sistema autónomo usando un robot móvil para tareas de recogida o entrega satisfaciendo los siguientes requerimientos: solución heurística basada en TSP; generación de caminos usando distancias no-Euclidianas; generación caminos Hamiltonianos; implementación en un robot real (Pioneer 3DX) usando un mapa de aproximadamente 52m por 41m.

III. Planificador de Rutas Multi-metas para Tareas de Recogida o Entrega (ARMM)

El ARMM se concibió para atender las solicitudes remotamente (ver Fig. 1). El servidor ARMM se encuentra en el robot, entonces usando el enfoque propuesto en este trabajo se realiza la planificación multi-metas y navegación para visitar todas las solicitudes de recogida/entrega recibidas (paso 2, Fig. 1). Luego, la interfaz humano-máquina (HMI) maneja la recogida o entrega (paso 3, Fig. 1). Finalmente, el robot móvil retorna a *home.* Para implementar este proceso, esta sección describe la configuración del sistema, las suposiciones asumidas, el TSP y la solución propuesta.

A. Configuración del Sistema

El robot móvil utilizado fue un Pioneer 3DX equipado con un computador con 1.6Ghz, 4Gb de RAM y Ubuntu 12.04 LTS. El sensor principal que se usó fue un láser SICK LMS200, y la odometría del robot móvil. Además, el robot posee una comunicación inalámbrica para manejar las solicitudes de recogida o entrega.

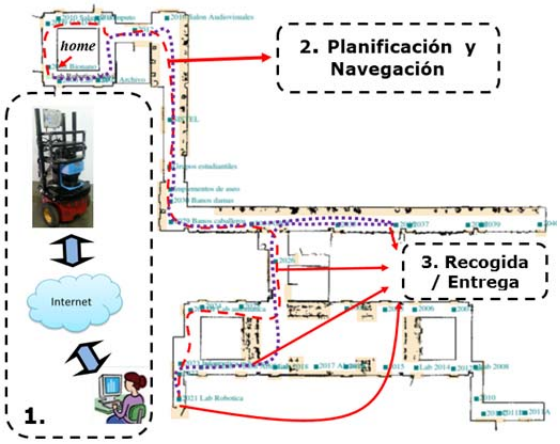


Figura 1. Diagrama conceptual del ARMM usando un robot móvil.

Como se observa en la Fig. 1, el mapa 2D incluye las metas que pueden ser alcanzadas por el robot. También se marcaron zonas prohibidas con el fin de evitar que el robot quedara atrapado entre materas, o que se fuera por las escaleras. El mapa fue construido usando una herramienta de mapeo basada en filtro de partículas de Mobile Robots Inc. [8].

B. Supuestos

TSP es conocido como un problema NP completo [1]. Por lo cual, se necesitan supuestos para garantizar la convergencia. En este trabajo se hicieron las siguientes suposiciones: i. Un mapa 2D se calculó y etiquetó previamente; ii. La distancia no-Euclidiana entre cualquier meta A y otra meta B es la misma desde B hacia A; iii. El método de localización está basado en un filtro de partículas implementado por [8]; iv. Se usa el método de ventana dinámica para planificación local [24]; v. El proceso de desarrollo para el ARMM usó la metodología RUP (Rational Unified Process) [25]; vi. El software cliente es una interfaz basada en texto (ver Fig. 2).

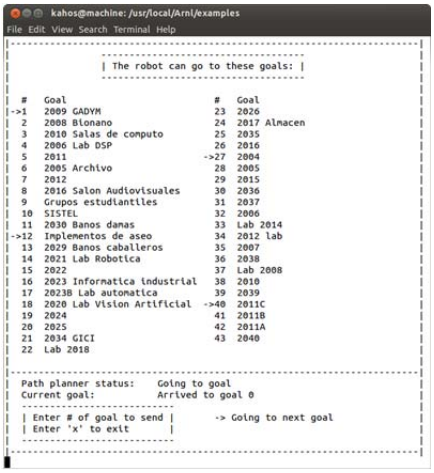


Figura 2. Interfaz de usuario para el aplicativo cliente.

C. Formulación del Problema del Agente Viajero (TSP)

Distintas soluciones a este problema fueron implementadas usando distancias Euclidianas [1]. La Tabla 1 muestra algunas soluciones implementadas usando rutas no Hamiltonianas; y todavía menos estudios fueron realizados implementándose en plataformas robóticas reales. En este trabajo, la solución al TSP descrita en [1] es adaptada para tareas de recogida o entrega usando un Pioneer 3DX, distancias no-Euclidianas y rutas Hamiltonianas. Hoy en día el algoritmo heurístico de Lin-Kernighan [1] es uno de los métodos más exitosos para generar soluciones cercanas a ser óptimas para el TSP. La formulación original del TSP puede describirse así:

"*Dada una matriz de costos C = [c_{ij}], donde c_{ij} representa el costo de ir de una ciudad i a una ciudad j, (i, j = 1, 2, 3,…, n), entonces encontrar una permutación (i_1, i_2, i_3,…, i_n) de los enteros desde 1 hasta n tal que minimice la cantidad* $q = c_{i_1 j_2} + c_{i_2 j_3} + \cdots + c_{i_n j_1}$".

La matriz de costos C tiene algunas propiedades, las cuales se usan para clasificar la solución a diferentes problemas: i. Si $c_{ij} = c_{ji}$ para toda i y toda j, se dice que el problema es *simétrico*, de otra manera es *asimétrico*; ii. Si la desigualdad triangular se mantiene ($c_{ik} \leq c_{ij} + c_{jk,}$ para toda i, j, y k) se dice que el problema es *métrico*; iii. Si c_{ij} son distancias Euclidianas entre metas, entonces el problema es *Euclidiano*.

El TSP es un problema de optimización combinacional, y difícil de resolver. Los algoritmos de solución se clasifican en 2 tipos: métodos exactos y aproximados (heurísticos). El método de Lin-Kernighan es un algoritmo heurístico, el cual se subdivide en tres clases: construcción de tours, mejoramiento de tours y algoritmos compuestos. La construcción de tours gradualmente agrega una ciudad en cada iteración; el mejoramiento de tours mejora el recorrido actual de las ciudades realizando varios intercambios; y los compuestos combinan estos dos tipos de algoritmos. El algoritmo Lin-Kernighan es un mejoramiento de tours, el cual implementa la optimalidad variable λ. En general , entre mayor sea el valor de λ, mayor es la probabilidad de que el recorrido de la ciudad sea óptimo, para más detalles consultar [1].

D. Recogida o Entrega Multi-Objetivos para Robots Móviles

Considerando los aspectos descritos anteriormente, el método propuesto en este trabajo y basado en el algoritmo de Lin-Kernighan tiene las siguientes propiedades: es un problema combinacional simétrico y métrico; trata con distancias no-Euclidianas; y genera una ruta Hamiltoniana en la cual el robot siempre retorna a la posición *home* en lugar de retornar a la posición actual del robot.

Lo último es una contribución importante de este trabajo, y se detalla a continuación: la Fig. 3 muestra la posición actual del robot R, cuatro objetivos (*A, B, C* y *D*) y la posición *home*. En tareas de recogida/entrega, los robots móviles retornan a la posición *home* para recargar baterías o para la carga o descarga de objetos. El algoritmo original de Lin-Kernighan puede generar rutas Hamiltonianas, pero la matriz de costos almacena distancias Euclidianas. Lo cual no es válido para tareas de recogida o entrega usando robots móviles, ya que no se consideran los obstáculos en el cálculo de la ruta. Así, en este trabajo la matriz de costos es modificada para incluir distancias no-Eeuclidianas entre la posición actual del robot y cada uno de los objetivos del mapa, y entre objetivos.

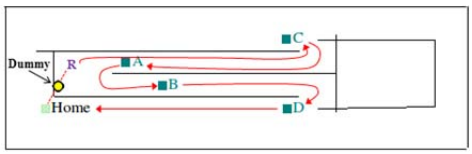


Figura 3. Ruta Hamiltoniana obtenida después de la introducción de un nodo "*dummy*".

En el algoritmo original de Lin-Kernighan la ruta termina en R. Sin embargo, en este trabajo el robot móvil tiene que terminar en la posición *home.* Por esta razón en ARMM la matriz de costos es modificada para incluir una última fila perteneciente a un nodo "*dummy"* con distancia igual a cero entre él, *home* y R, y distancias muy grandes entre él y cualquier otro nodo. Con estas modificaciones al algoritmo de Lin-Kernighan, ARMM genera una ruta Hamiltoniana que incluye *home* y considera distancias no-Euclidianas entre objetivos. Finalmente, el nodo "*dummy*" es removido sin afectar las propiedades de la ruta del robot descritas anteriormente, ya que la posición R, *home* y *dummy* están muy alejadas de los otros nodos.

El algoritmo de ARMM se muestra en Alg. 1. Este algoritmo se encarga de las siguientes tareas: inicializar la

conexión del robot, sus sensores de rango, el mapa del entorno, los manejadores de comandos para atender los comandos de los clientes, y la lista de objetivos del robot. Después, ARMM carga la matriz de distancias; en caso de que no exista, se calcula sin mover el robot. Esto se realiza una sola vez al principio del servicio de recogida/entrega. Luego, ARMM atiende a los clientes, y cada uno es manejado por un hilo independiente. Para cada cliente, se manejan los siguientes comandos: '*stop*' para detener el movimiento del robot; '*getGoals*' para enviar a los clientes los objetivos actualmente presentes en el mapa; '*currentGoal*' para enviar a los clientes el objetivo actualmente visitado; '*pathPlannerStatus*' para enviar a los clientes una realimentación sobre la tarea actual del algoritmo de planificación; '*dismiss*' para despachar el robot del objetivo actual y seguir con el siguiente; '*goal*' para añadir otro objetivo a la lista de objetivos a visitar, con lo cual la ruta del robot móvil tiene que ser recalculada usando el método propuesto en este trabajo; '*exit*' para terminar el aplicativo. Comandos internos son emitidos por la tarea de planificación de rutas, éstos son: '*goalReached*' es emitido por la tarea de planificación en el momento en el que llega a un objetivo solicitado, si éste es *home* el robot móvil es detenido. De otra manera el robot espera el comando '*dismiss*' y cuando lo recibe el objetivo actual se elimina; finalmente, si la lista de objetivos no está vacía, el robot móvil es conducido al siguiente objetivo y el planificador de ruta emite el estado '*goingGoal*' a todos los clientes conectados.

```
gatheringServer (Map, distanceMatrix)
doRobotConnection();                    // Inicialización del Robot
doRangeSensorsConnection();
robotArServerTask = getArServer();  // Tareas de la aplicación servidor
robotMapServer = getArMap(Map);
robotLocalizationTask = getArLocalization(robotMapServer);
robotPathPlannerTask = getPathPlanner(robotMapServer);
robotCommands = getInitCommandHandlers(); //Manejador de comandos
robotStateCB = doInitRobotPlannerState();
robotGoalList = init();
if (distanceMatrix)  // Matriz de distancias
    Load(distanceMatrix);
else
    calculateDistanceMatrix(robotMapServer, robotStateCB,
robotCommands);
while(true)  // Atendiendo a los clientes
    clientThreadRequest = acceptRequest();
    foreach (clientThreadRequest)
     clientCommand = getClientCommand();
     if (clientCommand == robotCommands('stop'))
       doArServerStopRobot();
     if (clientCommand == robotCommands('getGoals'))
       sendGoalsToClient(robotMapServer);
     if (clientCommand == robotCommands('currentGoal'))
       sendGoalToClient(robotLocalizationTask);
     if (clientCommand == robotCommands('pathPlannerStatus'))
       sendPathPlannerStatusToClient(robotStateCB);
     if (clientCommand == robotCommands('dismiss'))
       doRobotGoalDismiss(robotPathPlannerTask);
     if (clientCommand == robotCommands('goal'))
        robotGoalList = recalculateRobotPath(robotMapServer,
robotStateCB, robotCommands, robotGoalList, robotPathPlannerTask,
distanceMatrix);
     if (clientCommand == robotCommands('exit'))
       exitFromClient();
     if (robotCommands('currentGoal') == 'home' & robotStateCB
('goalReached'))
        doArServerStopRobot();
        clear(robotGoalList);
     if (robotStateCB ('goalReached'))
       doArServerStopRobot();
       waitDismiss();
       robotGoalList = removeGoal(robotGoalList);
     elseif (! isEmpty(robotGoalList))
       setCurrentState(robotStateCB('goingGoal'));
       driveRobotToGoal(robotGoalList);
    endForeach
endWhile
```

Algoritmo 1. Algoritmo del lado servidor.

En ARMM, uno de sus componentes fundamentales es la matriz de distancias. La cual es calculada por ARMM cuando el robot móvil inicia el servicio de recogida/entrega. El Alg. 2 muestra el algoritmo para calcular la matriz de distancias sin mover el robot móvil. Al inicio, se obtienen todos los objetivos del mapa y la matriz de distancias se inicializa. Después, se construye una matriz triangular con todas las distancias entre objetivos, y para hacer esto, el robot se posiciona virtualmente en cada *i-ésimo* objetivo del mapa, desde donde las distancias se calculan hasta al *j-ésimo* objetivo; esto se hace hasta que *i==j*. Todas las distancias calculadas son no-Euclidianas, ya que cada vez que un conjunto de objetivos *i* y *j* se seleccionan, el *framework* de ARIA calcula el camino mediante el algoritmo A* [8].

```
calculateDistanceMatrix (robotMapServer, robotStateCB,
robotCommands)
mapGoals = getMapGoals(robotMapServer);
distanceMatrix = init();
iGoal = 0;
foreach (mapGoals)
  putRobotAtGoal(mapGoals(iGoal));
  jGoal = 0;
  foreach (mapGoals)
    distancesToGoal = computeDistancesToGoal(jGoal);
    updateDistanceMatrix(distancesFromGoal, distanceMatrix, iGoal, jGoal);
    if (iGoal == jGoal) break;
    jGoal += 1;
  endForeach
  iGoal += 1;
endForeach
return distanceMatrix;
```

Algoritmo 2. Algoritmo para el cálculo de la matriz de distancias.

```
recalculateRobotPath (robotMapServer, robotStateCB, robotCommands,
robotGoalList, robotPathPlannerTask, distanceMatrix)
setCurrentState(robotStateCB('planningPath'));
addGoalToPlanner(robotGoalList);
[distancesFromRobotPose, currentRobotPose] =
computeDistancesFromCurrentPose(robotMapServer, robotGoalList);
addGoalToPlanner(robotGoalList, currentRobotPose);
distanceMatrix = updateDistanceMatrix(distanceMatrix,
distancesFromRobotPose, currentRobotPose, robotGoalList);
distanceMatrix = updateDistanceMatrixWithDummyPose(distanceMatrix);
robotGoalList = TSPupdate(robotGoalList, distanceMatrix);
return robotGoalList;
```

Algoritmo 3. Algoritmo planificador de rutas multi-objetivos.

```
gatheringClient()
robotStateCB = doInitRobotPlannerState();
clientGoalList = init();
robotServer = getRobotServerConnection();
currentMapGoals = sendCommand('getGoals');
updateGUI(currentMapGoals);
```

```
while(true)
  robotStateCB = sendCommand(robotStateCB, 'pathPlannerStatus');
  updateGUI();
  if (getUserRequest())   // Procesando entradas locales de los usuarios.
    if (getUserRequest('exit')) sendCommand('exit');
    if (getUserRequest('goalRequest'))
      addGoal(clientGoalList);
      sendCommand('goal', clientGoalList);
    elseif (getUserRequest('dismiss'))
      clearGoal(getCurrentGoal(),clientGoalList);
      sendCommand('dismiss');
  else  // Procesando respuestas del servidor
    if (robotStateCB('getGoals'))  handleGoalList(robotGoalList);
    if (robotStateCB('pathPlannerStatus'))
      handlePathPlannerStatus(robotGoalList);
      if (robotStateCB('pathPlannerStatus') == 'goalReached')
        sendCommand('currentGoal');
    if (robotStateCB('currentGoal'))
      handleArrivedToGoal();
      processingRobotDismiss();
endWhile
```

Algoritmo 4. Algoritmo del lado del cliente.

Otra parte fundamental de ARMM es calcular la ruta del robot dado un nuevo objetivo enviado por algún cliente. Esto se realiza cuando el servidor recibe el comando '*goal*'. El algoritmo para recalcular la ruta del robot en ARMM se muestra en Alg. 3. En primer lugar, se añade el objetivo a una lista; segundo, usando el mapa, la tarea de localización obtiene la posición actual del robot. Después esta posición se usa para obtener las distancias hacia todos los objetivos del mapa como se describe en Alg. 2; tercero, una nueva fila es creada en la matriz de distancias usando la posición del robot y todas las demás distancias; cuarto, la posición "*dummy*" es añadida a la matriz de distancias usando valores muy altos para las distancias a otros objetivos excepto para *home* y la posición actual del robot; finalmente, se ejecuta el algoritmo Lin-Kernighan para obtener una nueva lista de objetivos, la cual es usada para mover al robot al siguiente objetivo.

La interfaz de usuario del cliente se muestra en la Fig. 2, e implementa el algoritmo mostrado en Alg. 4. El aplicativo cliente empieza inicializando el estado del planificador de caminos, su propia lista de objetivos y la conexión con ARMM, la cual es usada para obtener los objetivos del mapa. Se debe notar que los clientes no tienen conocimiento del mapa del entorno. El aplicativo cliente responde a dos tipos de eventos: entradas del usuario por teclado e información enviada desde el servidor. Los comandos que el usuario tiene acceso son: '*exit*' termina la aplicación; si el usuario solicita un servicio de recogida/entrega, la posición actual del cliente es adicionada a la lista de objetivos y el comando '*goal*' es enviado a ARMM; finalmente, los usuarios pueden despachar al robot usando '*dismiss*'. La interacción con ARMM incluye los siguientes comandos: '*getGoals*' está a cargo de obtener la lista de objetivos para el cliente; '*pathPlannerStatus*' continuamente actualiza el estado del planificador de rutas multi-objetivos con el fin de saber si el robot móvil se está moviendo, o alcanzó un objetivo, o si el robot ha sido despachado por otro cliente o si el planificador está ocupado; si el robot móvil ha alcanzado un objetivo, '*currentGoal*' informa de este evento a los otros clientes.

IV. Resultados y Discusión

Para validar a ARMM usando un robot móvil, se realizaron dos pruebas cuantitativas incluyendo métricas como el tiempo transcurrido y la distancia recorrida por el robot móvil [26]. Otros métodos comunes para la planificación de rutas multi-objetivos son selección del objetivo más cercano y selección aleatoria [1] [26]. El primero, realiza una planificación de ruta considerando el objetivo más cercano a la posición actual del robot. El segundo, almacena todas las peticiones de los clientes en una cola FIFO, y luego selecciona un objetivo aleatoriamente de los objetivos que no han sido visitados. En esta sección se describe la creación del mapa, y la comparación de ARMM usando distancias Euclidianas y no Euclidianas con los algoritmos de selección aleatoria, y objetivo más cercano.

A. Construcción del Mapa

En este trabajo, se construyó un mapa 2D del segundo piso de los edificios 353 y 354 de la Escuela de Ingeniería Eléctrica y Electrónica de la Universidad del Valle como se ilustra en la Fig. 4. En este mapa se observan 43 diferentes metas. El proceso para su construcción puede resumirse de la siguiente manera: primero, se capturan datos del láser cada vez que el robot se desplaza 50cm o rota 10°; segundo, se registraron los datos del láser para obtener el mapa 2D usando la aplicación Mapper3; y tercero, se definieron las zonas prohibidas y los objetivos usando Mapper3. Este procedimiento puede observarse en línea en
https://www.youtube.com/watch?v=yY-fpouhWgs.

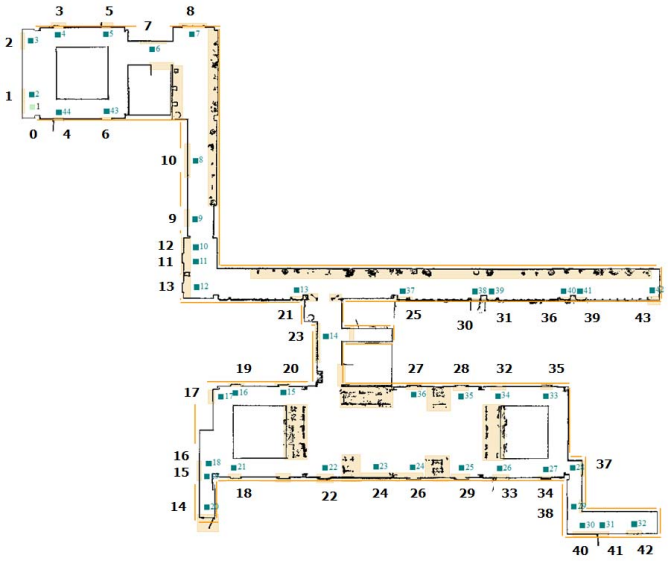


Figura 4. Mapa 2D usado para validar ARMM.

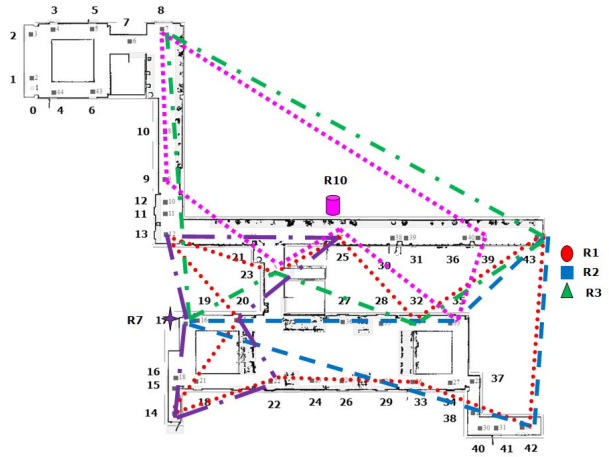
B. Resultados Cuantitativos

El tiempo y la distancia recorrida por el robot móvil se midieron para 10 peticiones diferentes en el mapa mostrado en la Fig. 4. El conjunto de metas de entrada al ARMM son:

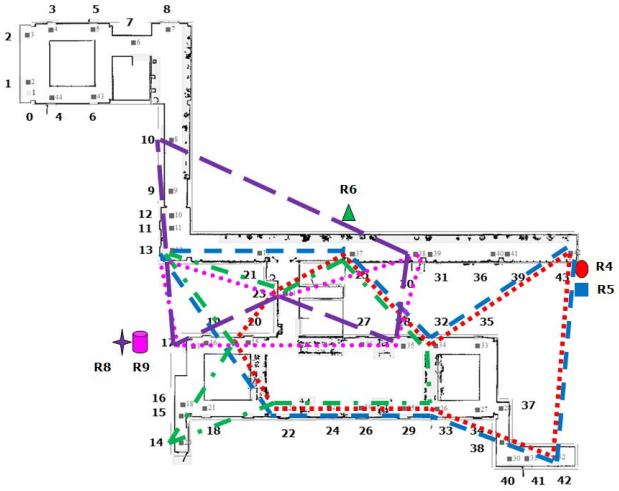
1. **No. 1**: 43, 14, 20, 32, 23, 33, 25, 42, 22, 13 y 43.
2. **No. 2**: 43, 42, 35, 17 y 43.
3. **No. 3**: 43, 17, 28, 8, 23 y 43.

4. **No**. **4**: 43, 25, 32, 20, 33, 22, 42, 23 y 43.
5. **No**. **5**: 43, 42, 25, 22, 32, 20, 33, 13 y 43.
6. **No**. **6**: 25, 32, 20, 23, 13, 22, 14, 33 y 25.
7. **No**. **7**: 17, 13, 25, 22, 14, 20 y 17.
8. **No**. **8**: 17, 13, 30, 28, 23 y 17.
9. **No**. **9**: 17, 10, 28, 23, 30 y 17.
10. **No**. **10**: 25, 11, 35, 23, 8, 36 y 25.

Todas las rutas fueron cuidadosamente seleccionadas con el fin de obtener situaciones similares a la mostrada en la Fig. 3. En estas situaciones se necesita medir las distancias no-Euclidianas para obtener soluciones que minimicen el consumo de energía, y el tiempo de espera para las tareas de recogida/entrega.



a)



b)

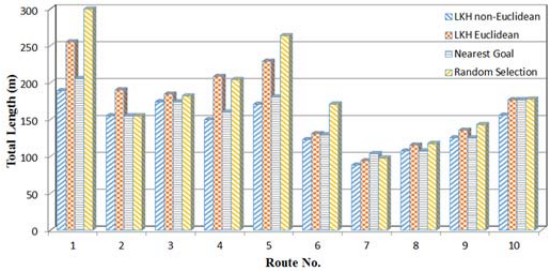
Figura 5. Diferentes rutas usadas para los experimentos en el mundo real. a) Rutas 1, 2, 3, 7 y 10. b) Rutas 4, 5, 6, 8 y 9.

Usando estas 10 rutas, se probaron los algoritmos selección aleatoria, objetivo más cercano, ARMM con distancias no Euclidianas y ARMM con distancias no-Euclidianas usando el robot móvil Pioneer 3DX. Un ejemplo de como el robot móvil se usó para realizar estas pruebas se presenta en https://www.youtube.com/watch?v=8Zhk9NkIgBs. En todos los casos, el robot fue despachado automáticamente una vez éste llega a su objetivo. El resultado del ARMM con distancias Euclidianas se muestra en la Fig. 5 y corresponde a la siguiente secuencia de metas:

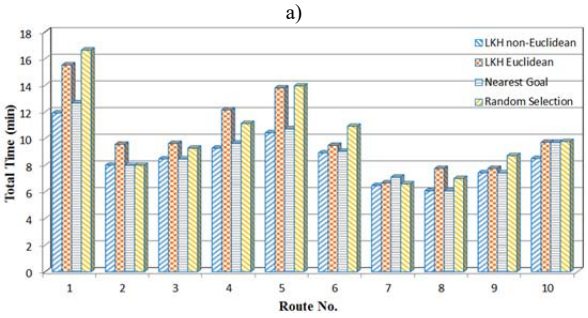
1. **Ruta No. 1**: 43, 32, 25, 23, 13, 20, 14, 22, 33, 42 y 43.
2. **Ruta No**. **2**: 43, 35, 17, 42 y 43.
3. **Ruta No**. **3**: 43, 8, 17, 23, 28 y 43.
4. **Ruta No**. **4**: 43, 32, 25, 23, 20, 22, 33, 42 y 43.
5. **Ruta No**. **5**: 43, 32, 25, 13, 20, 22, 33, 42 y 43.
6. **Ruta No**. **6**: 25, 32, 33, 22, 14, 20, 13, 23 y 25.
7. **Ruta No**. **7**: 17, 13, 25, 20, 22, 14 y 17.
8. **Ruta No**. **8**: 17, 13, 23, 30, 28 y 17.
9. **Ruta No**. **9**: 17, 23, 28, 30, 10 y 17.
10. **Ruta No**. **10**: 25, 23, 11, 8, 36, 35 y 25.

En cada ruta se marca la posición de inicio con un símbolo diferente (círculo, cuadrado, triángulo, estrella y cilindro), y en cada caso el robot móvil siempre retorna a la posición inicial. La Fig. 6a y la Fig. 6b muestran el resultado consolidado de distancia recorrida y tiempo transcurrido. Estas imágenes muestran el resultado para el algoritmo selección aleatoria mediante barras con líneas inclinadas a la derecha, el algoritmo de selección del objetivo más cercano mediante barras con líneas horizontales, ARMM con distancias Euclidianas con patrón en cuadrícula y el ARMM con distancias no-Euclidianas con barras de líneas inclinadas a la izquierda. Es fácil observar que ARMM usando distancias no-Euclidianas, recorre distancias menores en todas las 10 rutas resultantes. De la misma manera, el tiempo transcurrido para completar la ruta en los 10 casos es menor comparado con los otros enfoques. En todas las pruebas, el robot móvil navega por el entorno a la misma velocidad.

Observando la topología de las rutas mostradas en la Fig. 5, y comparando los resultados de la Fig. 6, se puede notar que ARMM con distancias no-Euclidianas ofrece mejores resultados cuando los objetivos están dispersos en un área grande del mapa. Esto puede observarse en las rutas 1, 4, 5 y 10. Lo cual evidencia la escalabilidad de ARMM.



a)



b)

Figura 6. a) Distancia recorrida por el robot usando los 4 enfoques. b) Tiempo transcurrido usando los 4 enfoques.

V. Conclusiones

Este trabajo ha presentado a ARMM, el cual aborda el problema de planificación de rutas multi-objetivos. ARMM fue concebido para satisfacer requerimientos tales como considerar distancias no-Euclidianas y proveer rutas Hamiltonianas para la navegación en robótica. Para hacer esto, ARMM se basa en el algoritmo *Lin-Kernighan Heuristics*, el cual fue diseñado originalmente como solución al TSP usando distancias Euclidianas. Adicionalmente, otra contribución importante de este trabajo es el nodo *dummy*, el cual fue introducido con el fin de generar rutas Hamiltonianas y así forzar al robot móvil para que vuelva a su posición inicial *home*. ARMM se implementó en un robot móvil real. Los resultados reportados incluyen una comparación entre cuatro algoritmos diferentes (objetivo más cercano, selección aleatoria, ARMM con distancias Euclidianas y con distancias no-Euclidianas). Se usó el robot móvil Pioneer 3DX para recorrer 10 trayectorias distintas para cada algoritmo que se probó. Como resultado, ARMM generó las menores distancias de recorrido y los menores tiempos transcurridos.

REFERENCIAS

[1] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," Eur. J. Oper. Res., vol. 126, no. 1, pp. 106–130, 2000.

[2] K. Hernandez and B. Bacca-Cortes, "Aplicación de Recolección Multi-Metas (ARMM), Registro 13-48-354." Ministerio del Interior, Dirección Nacional de Derechos de Autor, Cali, p. 1, 2014.

[3] K. Chen, C. Lin, C. Chien, J. Tsai, and Y. Liu, "Real-Time Path Planning and Navigation for a Web-Based Mobile Robot Using a Modified Ant Colony Optimization Algorithm Department of Mechanical Engineering 2 The Ant Colony Optimization," in Recent Advances in Automatic Control, Modelling and Simulation, 2013, pp. 179–184.

[4] EUROP and CARE, "Robotic Visions to 2020 and Beyond," 2010.

[5] KIVA Systems, "KIVA Warehouse Automation System," 2014. [Online]. Available: http://www.kivasystems.com/solutions/. [Accessed: 01-Jan-2014].

[6] Fraunhofer-IPA, "Care-O-Bot," Fraunhofer Institute for Manufacturing Engineering and Automation, 2015. [Online]. Available: http://www.care-o-bot-4.de/. [Accessed: 01-Jul-2015].

[7] G. Ferri, A. Manzi, P. Salvini, B. Mazzolai, C. Laschi, and P. Dario, "DustCart, an autonomous robot for door-to-door garbage collection: From DustBot project to the experimentation in the small town of Peccioli," in 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 655–660.

[8] A. MobileRobots, "Pioneer Robots," 2013. [Online]. Available: http://www.mobilerobots.com/ResearchRobots/. [Accessed: 01-Jan-2014].

[9] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," Syst. Sci. Cybern. IEEE Trans., vol. 4, no. 2, pp. 100–107, 1968.

[10] H. Berliner, "The B* tree search algorithm: A best-first proof procedure," Artif. Intell., vol. 12, no. 1, pp. 23–40, 1979.

[11] a. Stentz, "Optimal and efficient path planning for partially-known environments," Proc. 1994 IEEE Int. Conf. Robot. Autom., no. May, pp. 3310–3317, 1994.

[12] M. Dakulović and I. Petrović, "Two-way D* algorithm for path planning and replanning," Rob. Auton. Syst., vol. 59, no. 5, pp. 329–342, 2011.

[13] G. Ausiello and S. Leonardi, "Algorithms for the On-line Travelling Salesman," ALGORITHMICA, vol. 29, pp. 1–30, 2001.

[14] B. Coltin and M. Veloso, "Online pickup and delivery planning with transfers for mobile robots," in IEEE International Conference on Robotics & Automation (ICRA), 2014, pp. 5786–5791.

[15] K. Y. Chen, C. Chen, C. Y. Lin, J. H. Tsai, and H. Y. Chung, "Development of optimal path planning based on ant colony and wireless sensor network localization techniques for an autonomous mobile service robot," in Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on, 2014, vol. 2, pp. 954–959.

[16] D. H. Phan and J. Suzuki, "Evolutionary Multiobjective Optimization for the Pickup and Delivery Problem with Time Windows and Demands," Mob. Networks Appl., vol. 21, no. 1, pp. 175–190, Feb. 2016.

[17] S. Jeon and J. Lee, "Multi-robot multi-task allocation for hospital logistics," in 2016 18th International Conference on Advanced Communication Technology (ICACT), 2016, pp. 339–341.

[18] P. Vanek, J. Faigl, and D. Masri, "Multi-goal trajectory planning with motion primitives for hexapod walking robot," in Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on, 2014, vol. 2, pp. 599–604.

[19] R. Falcon, X. Li, A. Nayak, and I. Stojmenovic, "The One-Commodity Traveling Salesman Problem with Selective Pickup and Delivery : an Ant Colony Approach," in IEEE Computational Intelligence World Congress, 2010, pp. 1–8.

[20] L. Hongyun, J. Xiao, and J. Hehua, "Multi-goal path planning algorithm for mobile robots in grid space," 2013 25th Chinese Control Decis. Conf., pp. 2872–2876, May 2013.

[21] J. Faigl and J. Macák, "Multi-Goal Path Planning Using Self-Organizing Map with Navigation Functions," in ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2011, no. 1, pp. 27–29.

[22] B. Englot and F. Hover, "Multi-goal feasible path planning using ant colony optimization," 2011 IEEE Int. Conf. Robot. Autom., vol. 1, pp. 2255–2260, May 2011.

[23] J. Faigl and L. Preucil, "Artificial Neural Networks and Machine Learning–ICANN 2011," in 21st International Conference on Artificial Neural Networks, 2011, no. 1, pp. 85–92.

[24] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," Robot. Autom. Mag. IEEE, vol. 4, no. 1, pp. 23 – 33, 1997.

[25] P. Kruchten, The Rational Unified Process: An Introduction, 3rd ed. Addison-Wesley Professional, 2003.

[26] C. Sprunk, J. org R owek amper, G. Parent, L. Spinello, G. D. Tipaldi, W. Burgard, and M. Jalobeanu, "An Experimental Protocol for Benchmarking Robotic Indoor Navigation," in International Symposium on Experimental Robotics, 2014, pp. 1–15.

Kevin A. Hernández-Ossa recibió el título de ingeniero electrónico en la universidad del Valle, Cali, Colombia, en el 2014. Hizo su trabajo de grado en el campo de planificación para robótica móvil y creó el software ARMM. Hoy adelanta estudios de maestría en ingeniería en la universidad Espirito Santo, Brasil. Sus intereses de investigación incluyen la robótica móvil, bio-robótica y la inteligencia artificial.

Bladimir Bacca Cortes recibió su título de ingeniero electrónico de la Universidad del Valle en 1999, después recibió su M.Sc. en automatización de la universidad de Girona, España en el 2012. Sus principales intereses de investigación se enfocan en la robótica móvil, localización y mapeo simultáneo, visión artificial y visión omnidireccional. Hoy es profesor titular de la universidad del Valle, Cali, Colombia.

Breyner Posso Bautista recibió su título en ingeniería electrónica en 2004 de la Universidad del Valle. Después, recibió su M.Sc en automatización en la misma Universidad. Hoy en día, realiza estudios de doctorado en la universidad del Valle. Sus intereses de investigación son la robótica móvil, visión artificial y reconocimiento de patrones.