Artículo **Revista de Análisis Cuantitativo y Estadístico**
Junio, 2016 Vol.3 No.7 45-52

Sistema de programación dinámica de trayectorias para asistencia de nodos bajo demanda en redes inalámbricas de sensores

GONZÁLEZ-SILVA, Marco Antonio*†, FRANCO-MORENO, Juan José, BAEZA-REGALADO, Michelle y MORENO-ZAMORA, Brenda Paola

Universidad Politécnica Metropolitana de Hidalgo

Recibido Noviembre 25, 2015; Aceptado Abril 01, 2016

Resumen

Proponer un sistema dinámico capaz programar recorridos eficientes donde se asista a los nodos de una red de sensores según las necesidades actuales de éstos. Lógico-deductiva, se parte de análisis de variables, se utilizan como base algoritmos de trazado de rutas, abstracción y uso de pruebas en casos particulares. Existen varios sistemas informáticos que emplean el uso de sensores para monitoreo y sensado de variables que son medibles en objetos o espacios, ejemplo de ellas son la temperatura de una habitación, la cantidad de humedad de un cultivo, el nivel de agua de un contenedor, la presión arterial de una persona, entre otros. En redes de sensores es común la planeación de trayectorias para visitar a los nodos con un determinado fin. Sin embargo, muchos de los algoritmos de trazado de rutas no toman en cuenta la prioridad y demanda actual de cada nodo en la red. En este artículo se presenta un sistema de programación de trayectorias para visitar nodos de una red según los valores que se obtengan de sus sensores, creando niveles de prioridad en cada uno de ellos. El sistema aquí presentado permite optimizar tiempos y recursos en la asistencia de nodos que requieran ser visitados según sus necesidades, aprendiendo de sí mismo para aumentar la precisión. Pruebas realizadas muestran como el sistema propuesto encuentra trayectorias óptimas para visitar nodos según demanda y prioridad. Estos resultados permiten considerar la creación de varios tipos de aplicaciones donde se optimicen recursos en el recorrido de una red.

Demanda, IoE, Sensores, wsn, Trayectorias

Abstract

There are several computer systems that make use of sensors in order to be able to monitor and sensing variables that can be measured in objects or spaces, precedent of them are the temperature of a room, the moisture of a crop, the water level of a container, blood pressure of a person, among others. In sensor networks is common the trajectory planning to visit nodes with a particular purpose. However, many of the traceroute algorithms do not take into account the priority and demand of each node in the network. This article presents a path programming system to visit nodes in a network according to the values obtained from the sensors, creating priority levels in each of them. The system here presented allows to optimize time and resources in support of nodes that require to be visited according to their needs, learning from itself to increase further accuracity. Tests show how the proposed system finds optimal paths to visit nodes on demand and priority. These results allow to consider the creation of various types of applications where it is required to optimize resources in the path of a network.

IoE, On-demand, Sensors, Trayectories, wsn

Citación: GONZÁLEZ-SILVA, Marco Antonio, FRANCO-MORENO, Juan José, BAEZA-REGALADO, Michelle y MORENO-ZAMORA, Brenda Paola. Sistema de programación dinámica de trayectorias para asistencia de nodos bajo demanda en redes inalámbricas de sensores. Revista de Análisis Cuantitativo y Estadístico. 2016. 3-7: 45-52

*Correspondencia al Autor (Correo Electrónico: maagonzalez@upmh.edu.mx)
† Investigador contribuyendo como primer autor.

Introducción

El concepto de Internet de las cosas (IoE, por sus siglas en inglés Internet of Everything) se refiere al hecho de la interacción entre objetos, personas y máquinas conectadas a Internet. Su objetivo es el uso de tecnología como sensores, controladores y actuadores para el desarrollo de soluciones que ayuden a resolver distintos tipos de problemas, muchos de ellos de la vida cotidiana.

Charith, Chi y Srimal (2015), realizaron un estudio de distintos tipos de sistemas y aplicaciones desarrollados con IoE que tienen un alto impacto de utilidad en diferentes categorías como inteligencia portátil, casas, empresas y ciudades inteligentes (smart cities), motivando a desarrolladores a realizar trabajos sobre estos temas. Un ejemplo de ciudades inteligentes se puede ver en la propuesta de Asensio, Trasviña-Moreno, Blasco, Marco y Casas (2015) donde por medio de sensores se monitorea el nivel de tráfico de una avenida para programar cambios automáticos en los semáforos y agilizar el tránsito.

Como parte importante de IoE están las redes inalámbricas de sensores (WSN, por sus siglas en inglés Wireless Sensor Networks). Una WSN es una colección de sensores capaces de medir diferentes variables físicas como temperatura, ritmo cardiaco, nivel de líquido en un contenedor, humedad, aceleración, dirección de movimiento, presión de aire, etc., y por medio del uso de una señal de radiofrecuencia formar una red inalámbrica donde se puedan enviar datos hacia algún punto para ser procesados y/o analizados. De esta manera, acceder a datos sensados de algún objeto o persona puede realizarse desde diversos sitios remotos, utilizando la red de Internet como una opción y ampliando el panorama de aplicaciones al extender la cobertura de conectividad de las cosas y personas.

De acuerdo con Shane (2013), en un futuro una persona podrá tomar una píldora o tener sensores que detectarán su estado de salud e informar éste a través de Internet.

Muchas de las aplicaciones donde se hace uso de una WSN requieren que los nodos que forman parte de la red sean visitados con cierta periodicidad y propósito. Shue y Conrad (2013) presentaron un estudio del uso de robots como asistentes en las redes de sensores que visitan a los nodos con varios fines como localización, recolección datos, reemplazar o reparar nodos dañados, realizar carga de baterías, entre otros. En un enfoque de aplicación con IoE, Vargheese y Dahir (2014) proponen el uso de sensores y análisis de datos para predecir información de productos fuera de stock en tiendas, y notificar a los asociados de ellas para programar visitas de sus proveedores para abastecimiento, de esta manera se prevé que los clientes siempre encuentren el producto buscado.

El problema de planear rutas para visitar diversos lugares donde se optimicen tiempos y recorridos se conoce como VRP (por sus siglas en inglés, Vehicular Routing Problem) y se planteó por primera vez por Dantzig y Ramser (1959). En aquél trabajo se presentó una solución para programar trayectorias de vehículos y satisfacer demandas de abastecimiento de gasolina en estaciones de servicio. A raíz de esto muchas variantes de VRP han aparecido en la literatura, por ejemplo se han agregado restricciones de tiempo para abastecer demandas en determinados puntos (VRP with Time Windows, conocido en inglés), y considerarse ésto en la planeación de trayectorias (Oliveira, Vasconcelos, Alvarenga, Mesquita y Souza, 2007). De manera similar, uno de los algoritmos más utilizados para programar recorridos de manera óptima y visitar nodos de una red conexa y no dirigida es el de caminos mínimos (Dijkstra, 1959).

ISSN 2410-3438

Este algoritmo utiliza pesos en cada camino y explora todas las posibles rutas desde un origen hacia un destino buscando el menor costo según la suma de los pesos totales. Finalmente vale la pena citar al problema del agente viajero (TSP, en inglés Travelling Salesman Problem) citado en teoría de grafos y expuesto por W. Hamilton en el mismo año que Dijkstra.

TSP plantea el caso de encontrar la ruta óptima para recorrer un grafo donde se visiten todos los nodos y regresar al punto de partida, muchas soluciones computacionales se han expuesto para solucionar este problema (Reinelt G, 1994).

En general, existen varios sistemas y algoritmos para planeación de trayectorias propuestos en gran parte en áreas como teoría de toma de decisiones y puestos en práctica en redes de sensores. Sin embargo, estos algoritmos basan su planeación en estudios previos de demanda con datos no cambiantes y no toman en cuenta variaciones en los datos que pueden ser medidas en tiempo real y que afecten los requerimientos de asistencia en diferentes instantes de tiempo.

En este artículo se presenta un sistema llamado DynaSTy (por sus siglas en inglés Dynamic Schedule of Trayectories) para la programación dinámica de rutas y asistir a nodos que requieran ser visitados según los valores de sus variables medidas en tiempo real.

DynaSTy se basa en crear una red de sensores donde se mida algún parámetro que ofrezca información sobre una demanda existente y que interactúe con el entorno de un nodo cuyo estado sea dinámico respecto a ciertas entradas y salidas. Estos datos son enviados a través de una conexión inalámbrica hacia un servicio de base de datos en línea (cloud computing) donde podrán ser consultados en diversos instantes.

A su vez, un servidor local hace uso de la información en línea para programar rutas de asistencia mediante un algoritmo cuyos fundamentos son TSP y VRP donde las ponderaciones estarán en los nodos y no en los trayectos. El monitoreo en tiempo real permite al sistema crear curvas de aprendizaje útiles para predictibilidad, control y alcance, previniendo que los nodos sean asistidos según su demanda.

El contenido de este artículo se distribuye en cuatro secciones, en la sección 1 se mostró la introducción, en la sección 2 llamada metodología se presentaran los tres subsistemas de DynaSTy: red inalámbrica de sensores; cómputo en la nube y algoritmo de trazado de ruta. En la sección 3 se presentan pruebas y resultados de DynaSTy en simulador, y finalmente en la sección 4 se presentan las conclusiones.

Metodología

Como se mencionó anteriormente DynaSTy requiere de tres subsistemas para su funcionamiento los cuales se mencionan a continuación.

Subsistema de red inalámbrica de sensores

Una de las bases de DynaSTy es que los nodos que requieran ser asistidos deben sensar alguna variable física de su entorno y enviar los datos de ella a través de una conexión inalámbrica hacia una base de datos, donde serán almacenados y con base en ella crear una demanda de asistencia.

Utilizando una conexión por medio de tecnología 802.11 un sensor debe conectarse a la red de Internet ya sea de manera independiente o con ayuda de otro nodo.

ISSN 2410-3438

En la segunda situación un sensor puede no tener la cobertura o capacidad de conectarse a Internet, pero deberá tener la posibilidad de enviar sus datos hacia otro nodo intermedio que a su vez se conecte a otro nodo intermedio y así sucesivamente hasta alcanzar una estación base o nodo sink, como se les conoce en WSN. Un nodo sink es capaz de recibir datos de otros nodos y tener conexión a Internet. En la figura 1 se aprecia como los datos de sensado del nodo A tienen que recorrer una trayectoria pasando por los nodos B, C (intermedios) y D donde éste último envía la información recibida hacia algún sitio en Internet. De esta manera se requiere que en el diseño de la red de sensores que participen exista una conexión a Internet de forma independiente o de manera asistida.

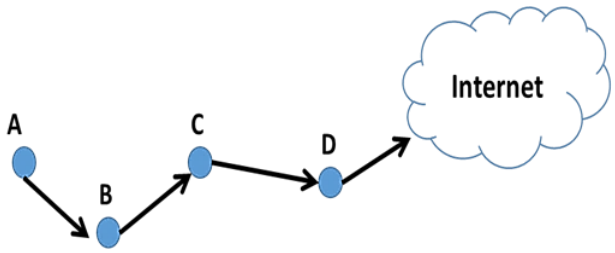

Figura 1 Conexión asistida hacia Internet en una red de sensores

Cómputo en la nube

Una vez que los sensores han enviado su información de sensado a una base de datos destinad, éstos representarán un valor de demanda (VDemanda). Esto es, si por ejemplo un sensor mide que la capacidad de un contenedor de líquido ha alcanzado el 70% de su capacidad éste tendrá un valor de demanda mayor que uno que tenga un 60%, tomando en cuenta que la demanda es recolectar el líquido. Para fines de DynaSTy, VDemanda será calculado siempre en porcentaje.

Los demás valores que requiere DynaSTy para su operación son los siguientes:

A. Ponderación. Esto es la razón de cambio de una variable respecto a otra variable (para el caso de DynaSTy el tiempo). Se calcula con la fórmula de razón de cambio promedio de la ecuación (1).

$$P= |\Delta x| / \Delta t \quad (1)$$

Donde:

P = Variación promedio (ponderación).

Δx = Valor final de x – Valor inicial de x. (valor absoluto)

Δt= Valor final de t – Valor inicial de t. (variación del tiempo).

Nótese que Δx está dado en valor absoluto porque para el funcionamiento de DynaSTy solo es relevante la variación promedio en términos positivos. Para ejemplificar la importancia de esta fórmula y concepto, el cual se ocupará en la sección 2.3, se considera el siguiente caso. Si en un contenedor se sensa que en 2 días el líquido que almacena subió del 50% al 70%, entonces aplicando la fórmula 1 tenemos:

$$P=(|50\%-70\%|)/(2 \text{ días}-0 \text{ días})=10\%/\text{día} \quad (2)$$

Entonces el resultado de (2) se puede interpretar como que el líquido contenido en un contenedor sube una razón promedio de 10% al día, lo que dará un valor de ponderación al nodo cuando se decida si debe ser visitado antes que otro nodo.

B. Identificador único (IdNodo). Número único asignado a un nodo para diferenciarlo de los demás.

C. Ubicación (Ubic). Es la información de latitud (Norte, Sur) y longitud (Este y Oeste) donde se encuentra localizado el nodo.

ISSN 2410-3438

D. Fecha de asistencia (FechA). Indica en qué fecha el nodo fue asistido y su demanda fue atendida por última vez.

E. Asitencia (a). Este valor indica si el nodo fue asistido en su demanda, y cambiará de acuerdo a la fecha de asistencia.

F. Contenido (ContNodo). Es el valor real en litros, kilogramos o cualquier unidad de medida según la variable que se esté sensando en el nodo. A diferencia de VDemanda este valor será representado en la unidad de medida usada.

Como medida adicional se considera un valor que define la capacidad de asistencia (CapA) del nodo que realizará la trayectoria calculada por DynaSTy. Dicho valor ayudará a obtener un estimado de cuántos nodos podrán ser asistidos antes que la capacidad del nodo visitante sea cubierta.

Algoritmo de trazado de ruta

Después de que todos los datos han sido almacenados y algunos de ellos estimados en la base de datos, se requiere calcular la ruta que deberá seguir un nodo que asiste para cubrir la demanda solicitada en la red. Este cálculo se realiza mediante el algoritmo de trazado de ruta que se explica a continuación.

Una premisa del algoritmo es que es necesario especificar los valores de Δt de la fórmula (1). El usuario puede decidir qué variación en el tiempo desea utilizar y que afectará en el valor de "P". Tal vez su variable principal, como puede ser el tiempo en el que un nodo agota su batería no represente ningún cambió si se toma un Δt con diferencia de unas cuantas horas, pero habrá una demanda distinta con un factor de variación de varios días o semanas, lo que obviamente afectaría en la programación de una ruta.

Otro requisito del algoritmo es que es necesario establecer un rango R[a,b] respecto a VDemanda que indique cuando un nodo debe ser asistido. Por ejemplo siguiendo el mismo caso de demanda de abastecimiento de batería en un nodo, un valor de VDemanda dentro del rango de R[0%,69%] indicará que un nodo debe ser asistido, porque se considera que no se tiene energía suficiente. Por lo tanto este valor tendrá un sentido distinto según los valores y criterios usados.

Una vez establecido R[a,b], se obtendrá el valor de VDemanda de cada nodo dentro de la red. Si el valor consultado está dentro de R[a,b] entonces este nodo ingresará a un nuevo arreglo llamado ArrayAsistibles, que indica qué nodos deben ser asistidos. Posteriormente para cada nodo dentro de ArrayAsistibles se consulta su valor de "P", calculado previamente por (1), y se crean nuevos arreglos donde se agrupen nodos con el mismo valor de ponderación. De esta manera, aquellos nodos que estén dentro de un arreglo con un valor de P más alto estarán el array1[a] y serán los primeros en ser asistidos, después aquellos que hayan sido clasificados con la segunda ponderación más alta y que estarán en el array2[b] y así sucesivamente hasta arrayN[n]. Para saber el orden de asistencia que deberá seguirse en estos últimos arreglos se considera un nodo inicial (el más cercano a un punto de partida donde el nodo asistente comenzará su recorrido, array1[1]ϵ array1[a]), posteriormente se visitará un segundo nodo (array1[2]) cuyos valores de longitud y latitud sean los más cercanos al primer nodo, y así sucesivamente hasta agotar el primer arreglo. Para el resto de los nodos incluidos en otros arreglos, por ejemplo el array2[b], se verificará desde un inicio si el nodo que asiste tiene la capacidad para visitarlos (CapA) y cumplir la demanda de sus nodos, de ser así se tomarán en cuenta éstos nodos en la ruta con la misma ponderación que array1[a], considerando a ambos arreglos como uno solo.

ISSN 2410-3438

Sin embargo, de no tener capacidad suficiente para cubrir la demanda de nodos de segundos, terceros u otros arreglos, entonces se considera la opción de asignar un segundo nodo asistente o simplemente programar más trayectorias respetando el orden establecido.

El pseudocódigo del algoritmo de trazado de rutas en presentado a continuación.

Pseudocódigo de algoritmo de trazado de rutas

Entrada: Arreglo de nodos en la red "A[x]", valores "P" para cada nodo, rango R[a,b].
Salida: Arreglo ordenado de nodos a asistir.
Inicio
//Obtener el arreglo ArrayAsistibles
1. Asignar: cont1 < -- 1, cont2 < -- 1
2. Asignar: aux < -- P(A[cont1])
3. If (aux >a)and (aux <b) then
4. Asignar: ArrayAsistibles[cont2]<-- A[cont1]
5. Asignar: cont2 < -- cont2+1
end if
6. Asignar: cont1 < -- cont1+1
7. Regresar al paso 2 hasta que cont1 > x

//Obtener arreglos array1[a], array2[b]…arrayN[n] con distintas ponderaciones.
8. Asignar N <-- 1
9.-Determinar cuantos valores de P distintos hay en ArrayAsistibles (var1)
10.-Obtener el valor más alto de P (ValorP) asignado a un nodo dentro de ArrayAsistibles según la variable N. Si N=1 entonces se busca el primer valor más alto, si N=2 se busca el segundo valor más alto y así sucesivamente.
ValorP <-- mayorP(N,ArrayAsistibles[cont2])
//La función "mayorP" encuentra el valor de P según N buscado dentro del arreglo ArrayAsistibles.
1 1.-Buscar nodos dentro de ArrayAsistibles con P=ValorP.
12.- Crear el arreglo arrayN con los nodos encontrados en el paso 11.
13.- Asignar N<-- N+1
14.- Regresar al paso 10 hasta que N>var1

//Obtener ruta
15. Elegir el nodo inicial "i" de array1[a] y visitarlo
16. Crear lista de nodos visitados
17. Si el nodo más cercano a "i" no se ha visitado entonces
18. Hacer: visitar nodo j
19. Asignar: i <-- j
20. Actualizar lista de nodos visitados
21. Regresar al paso17 hasta que todos los nodos sean visitados
22.- Repetir los pasos 15-21 para los arreglos restantes array2[b], array3[c]…… arrayN[n]

En la figura 2 se muestra un bosquejo general de DynaSTy, donde se pueden apreciar los tres subsistemas. En primera instancia nodos de una red sensan ciertas variables y envían datos hacia algún sitio en Internet el cual sirve de intermediario para la consulta que se hace desde un sitio remoto para alimentar al algoritmo que calcula la ruta a seguir para asistir a los nodos según su demanda. Los nodos en la figura presentan un valor numérico que indica su valor de demanda En el lado izquierdo se visualiza una ruta que recorrerá primeramente los nodos con demanda 1.

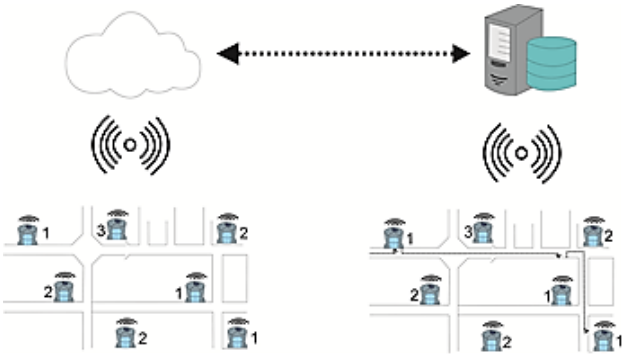

Figura 2 Bosquejo general de DynaSTy

Pruebas y resultados de DynasSTy

Para probar el desempeño de DynasSTy se programó éste en el software Matlab versión 7.6.0. con los siguientes valores.

Se creó un arreglo de 30 nodos como entrada A[30]. Se asignaron de manera aleatoria valores de VDemand inicial y final a cada uno de los nodos y se crearon dos arreglos con estos valores B[30] y C[30]. Se estableció un rango R de [0,60]. Se definió una variación de tiempo $\Delta t = 2$ días. La capacidad del nodo asistente se consideró que solo puede asistir por día a 15 nodos. Se realizaron 3 iteraciones del algoritmo de trazado de rutas donde en cada iteración se crearon nuevos valores de B[30] y C[30] respetándose los resultados de la iteración anterior, es decir si el nodo había sido asistido anteriormente entonces su valor VDemand inicial tendría 0%.

Adicionalmente se programó en Matlab recorridos utilizando el método del nodo más cercano, es decir, a partir de un nodo inicial, visitar su vecino más cercano y posteriormente el más cercano a este último y así sucesivamente hasta recorrer todos los nodos. Para este método se tomó en cuenta como inicio una versión del arreglo ArrayAsistibles que incluye todos los 30 nodos ordenados según su valor de VDemanda. Para siguientes iteraciones se contempló el mismo orden que la primera iteración y descartar así los cambios en demanda subsecuentes de los siguientes días. A este método le llamamos VCP (Vecinos Cercanos sin Ponderación).

En el Gráfico 1 se pueden apreciar los resultados de DynaSTy comparado con VCP. En el eje x se muestran los 30 nodos de las pruebas y en el eje y se muestra el valor promedio alcanzado por VDemanda de cada nodo durante los 6 días. La línea continua muestra el resultado de DynaSTy y la línea punteada de VCP. Mientras que en DynaSTy solo dos nodos estuvieron sin asistir cuando su valor de demanda bajo a menos del 70%, en VCP casi cerca de la mitad de los nodos su demanda no fue cubierta, sobre todo aquellos nodos al final del arreglo.

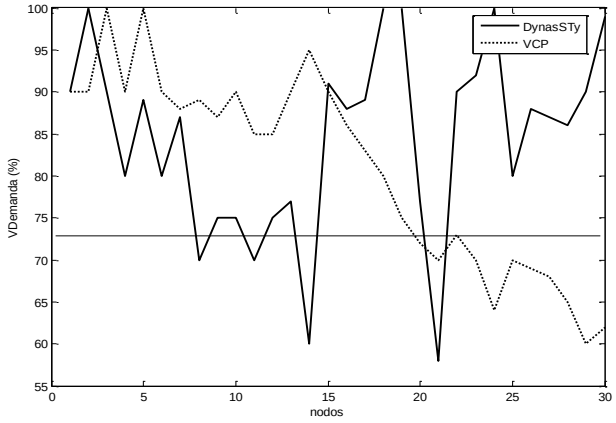

Gráfico 1 Comparativa entre DynaSTy y VCP

Lo que se puede concluir es que aquellos nodos que requerían asistencia con DynaSTy no fueron cubiertos en parte por una demanda muy alta generada aleatoriamente y en parte por la capacidad del nodo que asistía. En cambio en VCP además de estos factores influyó no tomar en cuenta la variabilidad y estancarse en rutas predefinidas con datos no cambiantes.

Conclusiones

En teoría de decisiones, redes de sensores e IoE, es importante considerar los cambios que presenten variables que influyen directamente en condiciones de entorno y que pueden modificar el comportamiento de un sistema completo.

La asistencia bajo demanda que se realice hacia nodos de una WSN puede prevenirse de manera que se eviten problemas como el que cierta sustancia se agote, que se pierda alguna conexión, que se contamine en exceso alguna región, o simplemente que los clientes de un establecimiento siempre encuentren lo que buscan.

ISSN 2410-3438

En este artículo se presentó un sistema llamado DynaSTy que combina de redes de sensores, cobertura amplia de redes como Internet y cómputo que realiza cálculos donde se tomen en cuenta valores de variables medidas en tiempo real para asistir a nodos que tienen cierta demanda y que esta puede variar con el tiempo. DynaSTy puede tener la opción de ajustarse a periodos de tiempo para predecir situaciones de demanda a futuro o asistir a aquellas en tiempo real.

DynaSTy podría tener un mayor desempeño y otras funcionalidades si se toman en cuenta más de una variable que puedan afectar el entorno de un nodo.

Agradecimientos

Agradecemos el apoyo brindado de la Universidad Politécnica Metropolitana de Hidalgo.

Referencias

Asensio A., Trasviña-Moreno, Blasco R., Marco A. & Casas R. (2015). Wireless Sensor Network in traffic management system. International Conference on Applied Informatics and Computing Theory. Salerno, Italy, (Junio 2015), 60-68

Charith P., Chi Harold L. & Srimal J. (2015). The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey. IEEE Trans. Emerg. Top. Comput. 3, 4 (October 2015), 585-598.

Dantzig G. B. & Ramser J. (1959). The Truck Dispatching Problem. Management Science 6 (1): 80–91.

Dijkstra E.W. (1959). A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, 1, 269-271.

Oliveira H.C., Vasconcelos G.C., Alvarenga G.B., Mesquita R.V. & Souza M.M. (2007). A roboust method for VRPTW with Multi-Start simulated annealing and statistical analysis. IEEE Symposium on Computational Intelligence in Scheduling. Honolulu, HI, USA, (Abril 2007), 198-205

Reinelt G. (1994). The travelling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, Berling.

Shane M., Villa N., Stewart-Weeks & Lange Anne. (2013) The Internet of Everything for Cities, Connecting People, Process, Data, and Things To Improve the "Livability" of Cities and Communities. Acceso en Mayo 2016, http://www.cisco.com/web/strategy/docs/gov/everything-for-cities.pdf

Shue S. & Conrad J.M. (2013). A Survey of Robotics Applications in Wireless Sensor Networks. Southeastcon. Jacksonville, Fla, USA, (Abril 2013). 1-5

Vargheese R. & Dahir H. (2014). An IoT/IoE enabled architecture framework for precision on shelf availability: Enhancing proactive shopper experience. IEEE International Conference on Big Data. Washingtong, DC, USA. (October2014). 21-26

ISSN 2410-3438