Shortening the Tour-length of a Mobile Data Collector in the WSN by the Method of Linear Shortcut

Md. Shaifur Rahman and Mahmuda Naznin

Bangladesh University of Engineering & Technology, Dhaka, Bangladesh
shaifur.at.buet@gmail.com, mahmudanaznin@cse.buet.ac.bd

Abstract. In this work, we present a path-planning method for the Mobile Data Collector (MDC) in a wireless sensor network (WSN). In our method, a tour for the MDC is generated such that the latency of delivering data to the sink is reduced. We show that the TSP tour covers all the nodes of the WSN. However, the latency for the TSP tour may be prohibitively high for delay-sensitive real-time WSNs. We reduce the latency by shortening the given TSP tour using a method called Linear Shortcut. We observe that the MDC need not visit the exact location of the node. It only need to be in the proximity of the node as required by the transmission radius. We present an algorithm that iteratively shortens tour-length and hence, reduces the latency. We term the resulting tour as Tight Label Covering (TLC) tour. Experimental results show that TLC tour reduces latency by a significant margin.

Keywords: Wireless Sensor Network, Path-planning, Mobile Data Collector, TSP tour

1 Introduction

Wireless Sensor Network (WSN) is widely used for tracking, monitoring and other purposes. The problem of collecting data packets from the sensor nodes and depositing those to the sink node is known as *Data Gathering Problem* [1,2]. Using mobile elements for data gathering in the WSN is a recent trend [3]. It has many advantages. For example, it increases connectivity, reduces cost of deployment of a dense WSN and increases the lifetime of the WSN. However, it has the disadvantage of high latency as these mobile elements have limited speed (compared to the high speed data gathering by routing). The dedicated mobile element in the WSN which collects data packets and brings those to the static sink is called *Mobile Data Collector* (MDC). A convenient way to control the latency in the case of data collection by the MDC is to carefully plan its path so that the path is as short as possible. In this work, we present a path-planning method that shortens the path of the MDC iteratively. We test our path-planning method in a simulation which is run on a realistic testbed.

Experimental results show that the shortening of the path indeed translates into decreased latency and other improvements.

Rest of the sections are organized as follows. Related works are discussed in Section 2. The problem is formulated in Section 3. Our method is presented in Section 4. Experimental results are presented in Section 5. The prospect of future work is presented in Section 6.

2 Related Works

A complete survey on using mobile elements for data collection in the WSN is presented in [3]. Earlier works on mobile elements can be found in [4], [5], [6] and [7]. However, random motion of the mobile elements in these works is not suitable for optimization. Mobile sinks have been considered in [8] and [9]. Mobile relay based approaches for opportunistic networks have been surveyed in [10]. However, these methods are not suitable for the WSN because of its difference with such networks. In [11], an energy-efficient data gathering mechanism for large-scale multi-hop network has been proposed. The inter-cluster tour proposed in this work is NP-hard. Latency issue is not addressed in it. One of the heuristics used in this work produces edges which are not connected with any nodes of the WSN. Latency is considered while planning path for the mobile collector in [12] and [13]. Methods presented in these works produce a shorter tour termed *Label Covering* tour from a *TSP-tour*. However, the transmission range of the sensor nodes is ignored in the shortening process. In [14], authors propose an approximation algorithm which is based on the *TSP-tour* of the MDC. Although, the computation time ($O(n)$) is impressive, the solution is applicable to only certain kind of *TSP-tour* (tours for which the centroid of the tour-polygon lies within that polygon). If a condition regarding the concavity of the given *TSP-tour* is not met, the problem of finding the optimal solution becomes NP-hard. In [15], authors address the problem of planning paths of multiple robots so as to collect the data from all sensors in the least amount of time. The method presented here exploits earlier work on TSP*-tour with neighborhood* problem. However, this work does not utilize the available location information of the sensor nodes to the fullest as it allows the traversal of the full boundary of the transmission region of a node. In sparse network, one or more sensor nodes have no neighbors at all. As a result, the traversal of the boundary those nodes is futile and adds up to the tour-length.

3 Preliminaries

We represent a WSN with n nodes by a complete graph K_n where the graph-nodes represent the sensors and the sink. The edges in this graph represent the Euclidian distances between two nodes. We adopt the *disk model* of the given transmission range TXR. A circle with radius TXR centered at a node represent the area of radio transmission of that node. We assume that there are one sink and one MDC in the WSN and that both the sink and the sensor nodes are static. A *tour* or *cycle* for the MDC is a closed path in the graph K_n which starts and ends at the sink node.

Definition 1. *A TSP-tour is a tour in which the MDC visits the exact location of all the nodes in the WSN exactly once. The min-cost TSP tour*[1] *is a TSP-tour in which the MDC covers the minimum Euclidian distance.*

Definition 2. *A tour T by the MDC is complete if each sensor node of the WSN can send data packets to the visiting MDC directly or via a neighboring which forwards packets. Otherwise, the tour is incomplete.*

By definition, a *TSP-tour* is complete. However, it is not a good choice for a delay-sensitive WSN as explained in the following section.

Definition 3. *Data Delivery Latency (DDL) of a packet is the time-difference between the generation and delivery of the packet.*

Let a packet i be generated at t_g time after the MDC has set out from the sink node. The MDC completes the current tour in t_T time according to some tour plan T. DDL for a packet i is given by:

$$t_l(i) = t_T - t_g(i) \tag{1}$$

If n packets are collected in the current tour by the MDC, average packet delivery latency t_{avg} is given by:

$$\begin{aligned} t_{avg} &= \frac{\sum_{i=1}^{n}[t_T - t_g(i)]}{n} \\ &= t_T - \frac{\sum_{i=1}^{n} t_g(i)}{n} \end{aligned} \tag{2}$$

The term $\frac{\sum_{i=1}^{n} t_g(i)}{n}$ in Equation 2 is the *average packet generation time*. This parameter is not controllable as it depends on the sampling rate of the sensor nodes and the event frequency. However, we may improve both per packet DDL (t_l) and average DDL (t_{avg}) by decreasing the tour-time t_T (See Equation 1 and 2). The tour-time of the *TSP-tour* i.e. t_{TSP} has two components: the fraction of tour-time t_h when the MDC halts and collects data from the nearby nodes and the fraction of tour-time t_m when the MDC travels between the node positions.

$$t_{TSP} = t_h + t_m \tag{3}$$

When the number of nodes is very high and/or the network is sparse, $t_h << t_m$ and thus t_m dominates tour-time t_{TSP}. This assumption is logical for practical scenario where the speed of a commercially available robotic car used as MDC usually is $5\ ms^{-1}$ whereas packet transfer from a sensor node to the MDC happens in the order of miliseconds [16]. Thus, decreasing the motion time t_m

[1] We use *TSP-tour* to denote the minimum cost *TSP-tour* in this work

contributes to improving latency. If the speed of the MDC is v_{MDC}, and if we assume that it accelerates to this speed instantly and stops instantly, then

$$t_m = \frac{|t_{TSP}|}{v_{MDC}} \tag{4}$$

where $|t_{TSP}|$ is the path-length of the *TSP-tour*. Given a particular MDC, v_{MDC} is fixed. The only way to decrease tour-time is decreasing the length of the tour i.e. $|t_{TSP}|$ (See Equation 4). However, decreasing the tour-length arbitrarily has the risk of making the resulting tour incomplete. Therefore, we address the issue carefully so that, the resulting tour is complete and shorter than the *TSP-tour*.

Problem Statement

Given a TSP-tour of the MDC in a WSN, find a tour T_d that is complete and shorter than the TSP-tour.

4 Improving Latency by Means of Linear Shortcut

4.1 Linear Shortcut of a Tour

Definition 4. *A linear shortcut of given a tour is derived by choosing 0, 1 or 2 points (called Anchor Points) from each tour-edge according to some strategy and connecting those points by straight lines in the order of visiting those edges. It is called linear as only new straight lines are introduced in the resulting tour instead of any curves.*

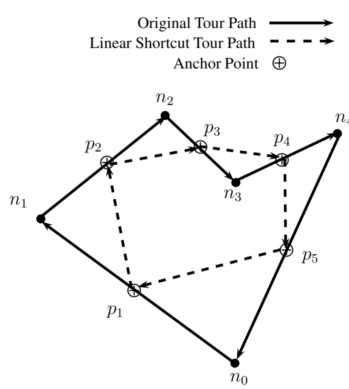
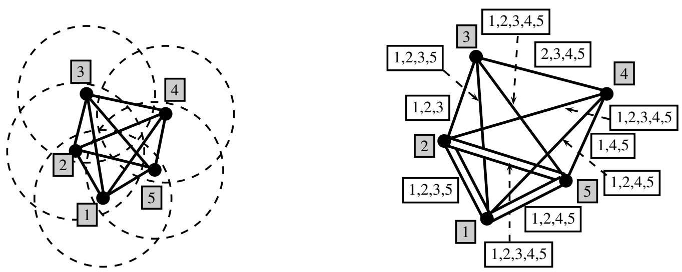


Fig. 1. Example of *Linear Shortcut* of a tour

An example of linear shortcut of a tour is shown in Figure 1. Five Anchor Points p_1, p_2, p_3, p_4 and p_5 are chosen from five tour-edges. Those are connected in the order of their visiting in the given tour to produce the linear shortcut. The first and the last points are also connected to make the path a cycle. The tour $< p_1, p_2, p_3, p_4, p_5, p_1 >$ is a linear shortcut but $< p_1, p_3, p_2, p_4, p_5, p_1 >$ is not. Using principle of triangle inequality, Lemma 1 can be easily proved.

Lemma 1. *If at least one anchor point is not coincident with the endpoint of the tour-edge, the linear shortcut is shorter than the given tour. (Proved using Triangle Inequality)*

4.2 Linear Shortcut of the *TSP-tour*



(a) Complete Graph derived from the Connectivity Graph (b) Label Covering Tour (tour-edges makred in double-line)

Fig. 2. Label Covering tour in a cluster with five nodes

In [12] and [13], a tour known as *Label Covering* or *LC* tour is derived from the *TSP-tour*. As shown in Figure 2, the tour-edges between nodes 1 and 2 are also within the range of Node 3 and 5. Therefore, the label set of this edge is $\{1, 2, 3, 5\}$. Similarly, the sets of labels of all other edges are determined. the minimum length label covering tour is determined by making shortcut of zero or more edges of the given *TSP-tour*. For the graph shown in Figure 2, the *TSP-tour* is $< 1, 2, 3, 4, 5, 1 >$ and the *LC*-tour derived from it is $< 1, 2, 3, 1 >$.

4.3 Linear Shortcut of the *LC*-tour

Definition 5. *The segment of the tour-edge which is within the circle representing the transmission area of a node is called the Contact Interval (CI) of that node on that edge.*

As shown in Figure 3, four nodes are covered by an edge connecting Node n_i and n_j. Each of their CI's is represented by two points on the edge i.e the l (which is encountered first by the MDC on this edge) and the r points. For example, the CI of Node n_{i+1} is given by (ln_{i+1}, rn_{i+1}). If the intersection of the circle and the straight line is beyond the edge, the end-point of the CI is the nearest end-point of the edge. For example, in Figure 3, r-point of Node n_{i+5} is the location of Node n_j.

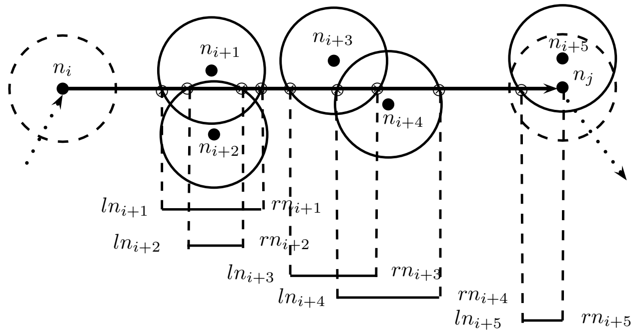


Fig. 3. *Contact Intervals* of an edge

Definition 6. *Given a list of contact intervals CI_e of a tour-edge e, Critical Contact Interval or CCI is the interval of the minimum length which has at least one point from each contact interval.*

Critical Contact Interval or CCI of i-th edge is represented by two points- $lcci$ and $rcci$ on that edge. These points can be determined as follows:

$$lcci \leftarrow r\text{-point closest to the first end-point along the tour-edge}$$
$$rcci \leftarrow l\text{-point closest to the last end-point along the tour-edge}$$

For example, the CCI of the edge shown in Figure 3 is determined in Figure 4.

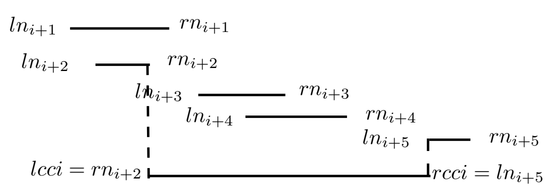
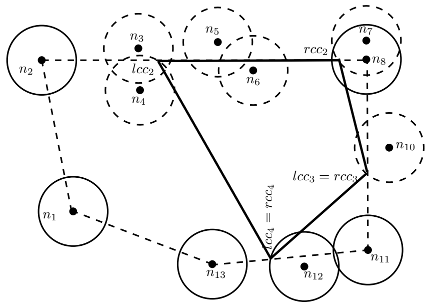


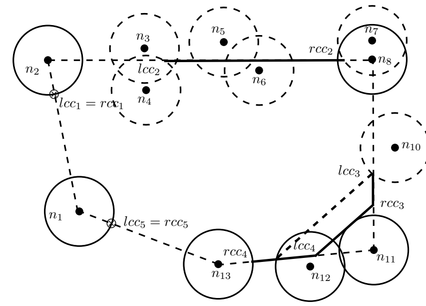
Fig. 4. *Critical Contact Interval* for a given list of intervals

After the CCI's have been computed, we can connect r point of the CCI of an edge with the l point of the CCI of the next edge. However, the nodes visited in the given tour may be missed as shown in Figure 5(a). Therefore, to cover those nodes, we apply the following method for a Node n_i:

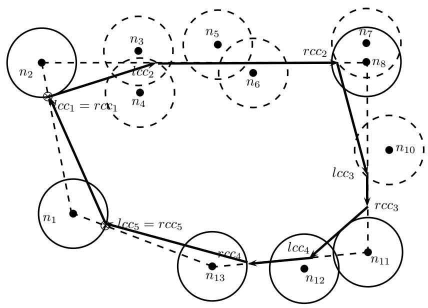
1. If both the edges have non-Null CCI's, i.e. there are intermediate nodes on both the edges, then we add the r point of the incoming edge with the l point of the outgoing edge. We call this line segment r-l line segment.
   (a) If r-l line segment intersects circle of Node n_i, then CCI's of both of the adjacent edges are kept unchanged (See Figure 5(b)).



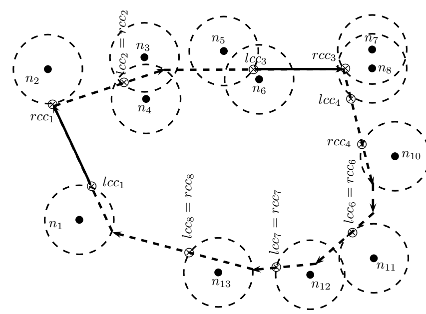
(a) Connecting the CCI's of successive tour-edges



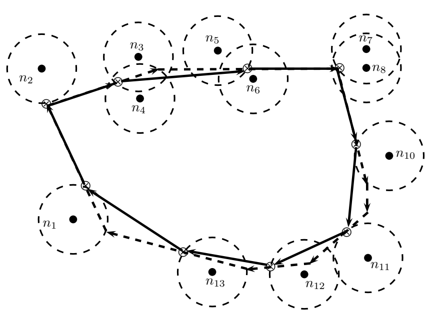
(b) Updated l and r points to cover visited nodes



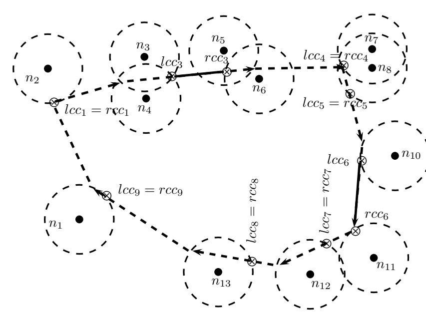
(c) TLC-tour derived in Iteration 1



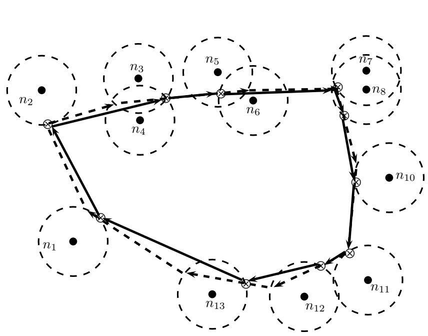
(d) Updating the l and r point after Iteration 1



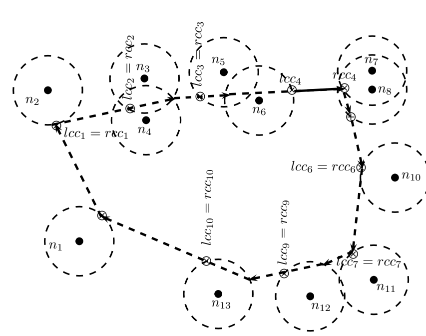
(e) TLC-tour derived in Iteration 2



(f) Updating the l and r point after Iteration 2



(g) TLC-tour derived in Iteration 3



(h) Updating the l and r point after Iteration 3

Fig. 5. Generating *TLC-tour* using Linear Shortcut

(b) If r-l line segment does not intersect the circle of Node n_i, then we draw a straight line that is parallel to the r-l line segment and tangent to that circle. Let this line intersects the incoming and outgoing edges at points p_i and p_o respectively. (see Figure 5(b) for Node n_{11}). We update the r point of the incoming edge and the l point of the outgoing edge as p_i and p_o respectively.

2. If the incoming edge does not have any intermediate node with overlapping circle or (in case it has) its r point is farther from point n_i by at least TXR, then we compute the point p_i as the intersection between the incoming edge and the circle centered at n_i. If the incoming edge has non-null CCI, then we update its r point as p_i. Otherwise, we set the incoming edge's r and l point as p_i. This case applies for Nodes n_1, n_2 and n_{13} as shown in Figure 5(b).

Now, we join the r point of the previous edge with the l point of the next edge. The final edges are shown as bold straight lines in Figure 5(c). We term this shortening as *tightening* of the given tour by the linear shortcut method. We term the shorter tour derived from the *LC-tour* as *Tight Label Covering* tour or TLC-tour.

4.4 Iterative Improvement of the *TLC*-tour

The path found in Figure 5(c) can be further shortened using the linear shortcut method. We divide each iteration of improvement into 2 steps:

1. Connect the r point rcc_i of i-th edge with l point lcc_j of the next edge (j-th edge such that $j > i$) with non-Null CCI and include the edge connecting lcc_j and rcc_j in the edge set.
2. *Re-associate* the intermediate circles with the resulting edges and *recompute* the CCI's for each edge.

Now, we use the following policy to *re-associate* the circles when existing tour-edges *break* into shorter ones and new edges are *added*:

1. For each circle adjacent with two edges, the CI's for the both the edges are calculated. Then the circle is associated with the edge on which the circle has longer CI.
2. If the CI's for both the edges are equal, the circle is associated with the incoming edge.

As shown in Figure 5(d), there are eight edges. Circle of Node n_1 has overlaps with both the outgoing and the incoming edges. However, unlike the incoming edge, the circle has a CI of non-zero length with the outgoing edge. Therefore, we associate Node n_1 with this edge. Node n_2 has CI's of zero length with both the incoming and the outgoing eges. Therefore, we associate it with the incoming edge.The CCI of the edge connecting n_1 and n_2 is updated after this *re-association*. In similar ways, we determine the l and r points of the CCI's of the remaining edges. After this round of re-associating of circles and computation of the CCI's of respective edges, we join the r point of an edge with the l point of the next edge. Thus, the tour as shown in Figure 5(e) is derived. According to the Lemma 1, it is shorter than the tour derived in the previous iteration.

We can continue this way in to more iterations to tighten the given tour. The steps are illustrated in Figure 5(f), 5(g) and 5(h).

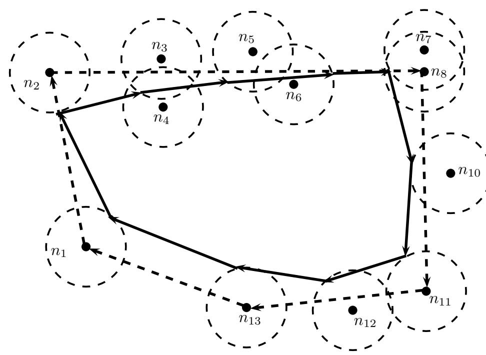


Fig. 6. Comparison between input *LC-tour* (doted path) and *TLC-tour* (solid path) derived in Iteration 4

The given LC-tour and the TLC-tour derived after Iteration 4 has been imposed on each other for comparison in Figure 6. The derived TLC-tour is a significantly shorter than the LC-tour.

Condition For the Termination of Iterations: We can define *path gain* $g_i(t_{TLC})$ for a TLC-tour derived in iteration i as follows:

$$g_i(t_{TLC}) = \frac{|t_{TLC}|_{i-1} - |t_{TLC}|_i}{|t_{TLC}|_i} \quad (5)$$

Here $|t_{TLC}|_i$ is the length of the TLC-tour derived in iteration i. We stop the iterations for path-shortening as soon as the *path gain* is below a threshold like 1%, 5% etc.

Computation Complexity: Our method of generating *TLC-tour* is formally presented in Algorithm 1. First, we generate the CI's for all the circles in $O(n)$ time. Then, we sort the CI's in the non-decreasing order of the distance from the first endpoint of the edge in $O(n \log n)$ time. Therefore, we generate the CCI for an edge in $O(n + n \log n) = O(n)$ time. For a given tour, there are $O(n)$ edges. Therefore, we generate CCI's of all the edges in $O(n^2)$ time. Then, we connect successive r and l Points in $O(n)$ time. For each edge, we test the circles for *re-association*. There may be $O(n)$ such circles associated with each edge and a total of $O(n)$ edges. Therefore, updating the CCI's takes $O(n^2)$ time.

Algorithm 1 Generating *TLC-tour* from *LC-tour*

Input: *LC-tour* T_{LC} and a path-gain threshold g_t

Compute the CI's of all the nodes
Compute the CCI's of all the edges
while *path-gain* $> t_g$ **do**
  derive tour T_{TLC} by connecting the r point of an edge with the l point of the next edge
  for each edge e **do**
    for each Circle c (of Node n) associated with Edge e **do**
      re-associate Circle c (if necessary)
    end for
    update CCI of Edge e
  end for
  determine *path-gain*
end while

Output: *TLC-tour* T_{TLC}

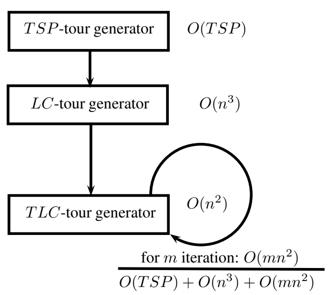
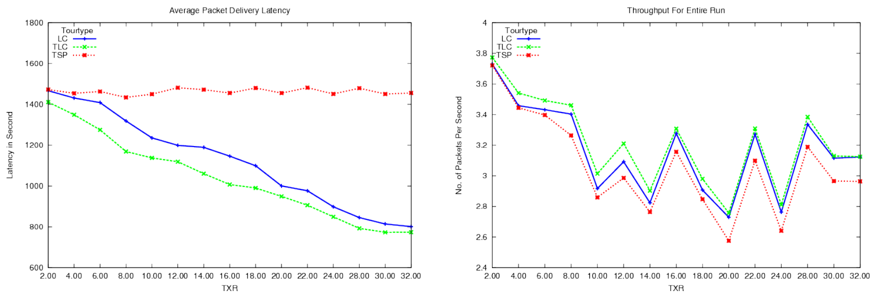


Fig. 7. Stages of computation along with the time complexity

If there are m iterations, then generating *TLC-tour* from *LC-tour* takes $O(mn^2)$ time. We illustrate the stages of computation along with the time complexity in Figure 7.

5 Experimental Results

We use *Castelia*[17] framework of *OMENT++* simulator to distribute sensor nodes randomly. The sensor nodes also generated packet randomly. We use *Concord TSP-Solver*[18] to compute the optimal *TSP-tour*. We derive the *LC*-tour from the *TSP-tour* and the TLC-tour from the *LC*-tour using *path-gain* threshold of 5%. The MDC tours continuously in *TSP-tour, LC-tour* and *TLC-tour*. During its travel, it collects the packets from the sensor nodes and deposits those to the sink node. We vary the value of TXR from $2m$ to $32m$. Low value of TXR indicates lower degree of connectivity and hence, sparse network. Similarly, higher value of TXR indicates dense network.



(a) Average Packet Delivery Latency (b) Throughput

Fig. 8. Comparison among *TSP-tour*, *LC*-tour and *TLC*-tour

As shown in Figure 8(a), *average packet delivary latency* is always the lowest for *TLC-tour* and the highest for *TSP-tour*. The value is comparatively better in the case of the sparse WSN. In Figure 8(b), *throughput* for the entire run is shown. The value is always the highest for *TLC*-tour and the lowest for the *TSP-tour*.

6 Conclusion

We have given a framework for shortening a given tour of the *MDC*. The resulting tour decreases packet delivery latency and increases throughput. The *TLC-tour* derived by *Linear Shortcut* of the *LC-tour* is highly suitable for real-time WSN in which high latency is undesirable.

In our future work, we shall consider more objectives besides minimizing latency, for example- facilitating multi-hop forwarding among the sensor nodes, load-balancing of the network traffic etc. We shall also extend the path-planning for a WSN with multiple *MDC*'s and multiple sinks.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**(4) (March 2002) 393–422
2. Rajagopalan, R., Varshney, P.: Data-aggregation techniques in sensor networks: a survey. Communications Surveys Tutorials, IEEE **8**(4) (quarter 2006) 48 –63
3. Di Francesco, M., Das, S.K., Anastasi, G.: Data collection in wireless sensor networks with mobile elements: A survey. ACM Transaction on Sensor Networks **8** (August 2011) 7:1–7:31
4. Zhao, W., Ammar, M.H.: Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In: Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems. FTDCS '03, Washington, DC, USA, IEEE Computer Society (2003) 308–

5. Shah, R.C., Roy, S., Jain, S., Brunette, W.: Data mules: Modeling a three-tier architecture for sparse sensor networks. In: IN IEEE SNPA WORKSHOP. (2003) 30–41
6. Zhao, W., Ammar, M., Zegura, E.: A message ferrying approach for data delivery in sparse mobile ad hoc networks. In: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing. MobiHoc '04, New York, NY, USA, ACM (2004) 187–198
7. Jun, H., Zhao, W., Ammar, M.H., Zegura, E.W., Lee, C.: Trading latency for energy in wireless ad hoc networks using message ferrying. In: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops. PERCOMW '05, Washington, DC, USA, IEEE Computer Society (2005) 220–225
8. Wang, Z.M., Basagni, S., Melachrinoudis, E., Petrioli, C.: Exploiting sink mobility for maximizing sensor networks lifetime. In: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences - Volume 09. HICSS '05, Washington, DC, USA, IEEE Computer Society (2005) 287.1–
9. Rao, J., Wu, T., Biswas, S.: Network-assisted sink navigation protocols for data harvesting in sensor networks. In: WCNC 2008, IEEE Wireless Communications & Networking Conference, March 31 2008 - April 3 2008, Las Vegas, Nevada, USA, Conference Proceedings, IEEE (2008) 2887–2892
10. Conti, M., Pelusi, L., Passarella, A., Anastasi, G.: Adaptation and Cross Layer Design in Wireless Networks, chapter: Mobile-relay Forwarding in Opportunistic Network. CRC Press (2008)
11. Ma, M., Yang, Y.: Sencar: An energy-efficient data gathering mechanism for large-scale multihop sensor networks. IEEE Transaction on Parallel and Distributed System **18** (October 2007) 1476–1488
12. Sugihara, R., Gupta, R.K.: Improving the data delivery latency in sensor networks with controlled mobility. In: 4th IEEE international conference on Distributed Computing in Sensor Systems, Berlin, Heidelberg, Springer-Verlag (2008) 386–399
13. Sugihara, R., Gupta, R.K.: Path planning of data mules in sensor networks. ACM Transaction on Senson Networks **8**(1) (August 2011) 1:1–1:27
14. Yuan, Y., Peng, Y.: Racetrack: an approximation algorithm for the mobile sink routing problem. In: Proceedings of the 9th international conference on Ad-hoc, mobile and wireless networks. ADHOC-NOW'10, Berlin, Heidelberg, Springer-Verlag (2010) 135–148
15. Bhadauria, D., Tekdas, O., Isler, V.: Robotic data mules for collecting data over sparse sensor fields. J. Field Robot. **28**(3) (May 2011) 388–404
16. Huang, P., Xiao, L., Soltani, S., Mutka, M., Xi, N.: The evolution of mac protocols in wireless sensor networks: A survey. Communications Surveys Tutorials, IEEE **PP**(99) (2012) 1 –20
17. : Castalia wsn simulator framwork for omnet++. `http://www.castelia.org`
18. : Concorde tsp solver. `http://www.tsp.gatech.edu/concorde.html`