HQuad: Statistics of Hamiltonian Cycles in Wireless Rechargeable Sensor Networks

Yanmao Man
Dept. of ECE
University of Arizona
Tucson, AZ, U.S.A.
yman@email.arizona.edu

Jing Deng
Dept. of CS
UNC at Greensboro
Greensboro, NC, U.S.A.
jing.deng@uncg.edu

George T. Amariucai
Dept. of CS
Kansas State University
Manhattan, KS, U.S.A.
amariucai@ksu.edu

Shuangqing Wei
School of EECS
Louisiana State University
Baton Rouge, LA, U.S.A.
swei@lsu.edu

***Abstract*—The rise of wireless rechargeable sensor networks calls for an analytical study of planned charging trips of wireless charging vehicles (WCVs). Often times, the WCV receives a number of charging requests and form a Hamiltonian cycle and visit these nodes one-by-one. Therefore, it is important to learn the statistics of such cycles. In this work, we use a heuristic algorithm, which we term HQuad, that takes $\mathcal{O}(N)$ to generate a Hamiltonian cycle in a 2-D network plane before we analyze its statistics. HQuad is based on a recursive approximation of dividing the region into four quadrants and the non-empty quadrants will be visited one-by-one. Our analysis is based on Poisson point distribution of nodes and models such Hamiltonian cycles surprisingly well in both expected values and the distribution functions of lengths as a function of different network parameters. Numerical results of our analysis model are compared with simulations and demonstrated to be accurate.**

***Index Terms*—wireless rechargeable; Hamiltonian Cycle; wireless sensor networks**

I. Introduction

Wireless devices usually run on energy stored on batteries. Due to their physical constraints, these batteries will run out of energy sooner or later. Wireless power transfer technology is expected to pave the way for the so-called wireless rechargeable networks, in which wireless devices will have their batteries replenished from time to time, virtually extending their lifetimes forever [1]–[3].

Most often, there is one or multiple WCVs in such networks. WCV receives charging requests from wireless devices and chooses a list of them to visit/charge one after another [4], [5]. In this work, we try to answer the following fundamental question: based on a certain node density and random distribution, how to compute or estimate the lengths of the charging cycle without knowing the exact location of these nodes on the plane? Any of such charging cycles will be considered a Hamiltonian cycle, but we are interested in the shortest one that will save time and travel distance for WCV.

Answers to the above question obviously have many practical applications such as network planning and scheduling. For instance, the expected length of such shortest Hamiltonian cycles in a known network setting (of certain node density and network region) can ensure that the battery capacity of WCV is able to sustain such a charging trip before returning for its own charge or battery replacement, or maybe an additional WCV is required for the normal operation of the entire network.

Y. Man's work was done while with UNCG, supported in part by NSF grant CCF-1320428.

The problem is closely related to Traveling Salesman Problem (TSP) but different. TSP aims to give shortest Hamiltonian cycles and sometimes approximates with short running times [6], while this paper targets the estimation of such cycles by giving a probabilistic model [7].

In this work, we propose HQuad, Quadtree-based heuristic algorithm, that takes $\mathcal{O}(N)$ to generate a Hamiltonian cycle given a network within a square plane. HQuad is an approximation method to come up with a Hamiltonian cycle that is very close to the actual shortest Hamiltonian cycle. With the help of HQuad, we can derive charging path statistics such as average length and probability density function (PDF).

The main objective of this work is to provide an efficient method to compute the statistics of Hamiltonian paths so that networks can be planned with sufficient resources, e.g., charging capacity, number of charging vehicles, and/or number of devices to be charged.

II. Related Work

The well-known Traveling Salesman Problem (TSP) is certainly related to what we are trying to achieve. Many works have been done on TSP [8]. Arora [9] proposed an approximation algorithm for Euclidean TSP problem, where coordinates of nodes are at first perturbed, then a shifted Quadtree is constructed. Finally, dynamic programming is performed to find the optimal Hamiltonian cycle. However, we focus on path statistics here.

Wireless Sensor Recharging problem has been investigated on in recent years. The problem is to schedule a route for one or more WCVs so that they could recharge those devices in low battery energy. A hexagon-based scheme was proposed in [10]. Martello presented an algorithm to identify Hamiltonian circuits in directed graphs [11]. The algorithm used branching decisions and backtracking to filter and record potential Hamiltonian circuits. Krivelevich et al. investigated Hamiltonian cycles in random graphs, but did not focus on its length distribution [12].

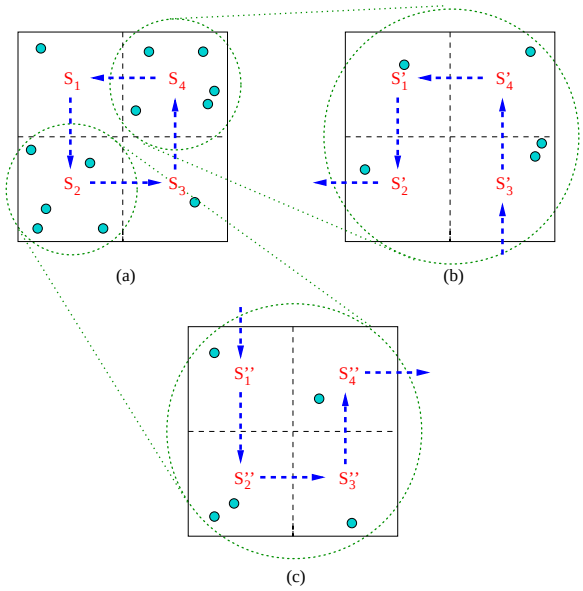


Fig. 1: HQuad Algorithm illustrations.

Vig and Palekar [13] investigated the probability distribution of estimated TSP tour lengths using heuristics and fits for the first four moments were derived.

Different from these prior arts, the main target of this paper is not to provide a recharging dispatch algorithm. Instead, we aim to give a probabilistic distribution of Hamiltonian cycle given the density and the size of square plane where the devices are located.

III. HQuad: Heuristic Quadtree-based Algorithm for Hamiltonian Cycles

A. Problem Statement

We focus on a network with a size of $r \times r$ and N nodes, each of which needs to be charged. Note that the investigation of an arbitrary rectangle area of $r \times w$ is essentially the same with each quadrant in similar shape as the original one and some of the results extended from squares. Therefore, we focus on square areas in this work:

Hamiltonian Cycles (HAM): *What is the distribution of the shortest Hamiltonian cycle in a region of $r \times r$ with N nodes distributed?*

The exact derivation of such a distribution is impossible because Hamiltonian cycle/path problem is NP-complete. Yet, such a distribution can be immensely helpful in network planning as well as algorithm design in Wireless Rechargeable Sensor Networks (WRSNs). Therefore, we use a heuristic approach, termed HQuad [9], to identify the statistics of the shortest Hamiltonian cycles.

B. Primary

HQuad is based on Divide-and-Conquer strategy that takes the locations of N nodes in a square block as input and generates the Hamiltonian Cycle for WCV to follow.

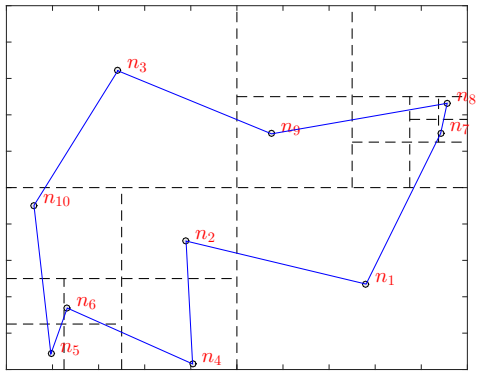


Fig. 2: An example of blocks and recharging route.

As shown in Figure 1.a, N nodes are placed in an $r \times r$ square block S_0, which is divided into four $r/2 \times r/2$ sub-blocks that are marked as S_i, $i = 1, 2, 3, 4$. WCV visits each sub-block from one to another before returning to the starting point. Each sub-block is further divided quadruply into smaller ones, where WCV follows similar route to visit the nodes and then goes to the next bigger block as illustrated in Figure 1.b and 1.c when we zoom in. Normally, some blocks may be empty and some do not need to be divided any further with only one node in each of them. More realistic example of division is shown in Figure 2, where we can observe that all blocks contain at most one node and WCV only visits the non-empty blocks rather than all (sub)blocks.

C. Algorithm Details

Because the blocks are divided into 4 equal sub-blocks when necessary, we use *Quadtree* $\mathcal{T}$ as the data structure where every parent owns exactly 4 children, similar to [14]. Each vertex in $\mathcal{T}$ represents a block and its children represent the sub-blocks. In this subsection, the terms "block" and "vertex" are interchangeable, so are the terms "sub-block" and "child".

The biggest block is represented by vertex $root$, which is the root of $\mathcal{T}$. For each vertex $parent$, it contains two elements, seq and $nodes$. seq decides the visiting sequence of its sub-blocks. For example, if $seq = (S_3, S_4, S_1, S_2)$, then the route is shown in Figure 1.b. $nodes$ is the set of nodes that locate within this block. Take S_4 in Figure 2 for an example, $nodes_4 = \{n_7, n_8, n_9\}$.

The Quadtree $\mathcal{T}$ is built up in Algorithm 1 from Line 1 to Line 20 in the way of Breadth-first Search (BFS) with a queue Q. For every vertex $parent$ in Q, we first test if its $nodes$ set contains only one node or nothing at all. If so, then we continue to the next iteration because we do not have to further divide this block. Otherwise, we have to at first derive the sequences belonging its children seq_i, $i = 1, 2, 3, 4$.

We need to determine the seq of every parent. Suppose sequence of the $parent$ is $seq = (S_a, S_b, S_c, S_d)$, the sequence of its first child S_a is $seq_a = (S_a, S_d, S_c, S_b)$. Note that the first child might not be S_1. For example, in Figure 1.b,

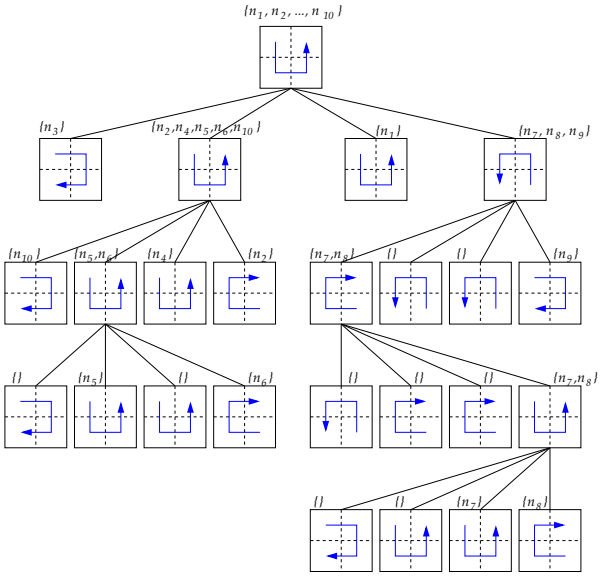


Fig. 3: An example of Quadtree $\mathcal{T}$ generation.

the first child S_a is S_3'. The second and third children share the same sequence with seq. The fourth one depends on whether the $parent$ is the root of $\mathcal{T}$ or not. If so, then $seq_d = (S_c, S_d, S_a, S_b)$. Else, then $seq_d = (S_c, S_b, S_a, S_d)$. The reason of such a difference is that if the $parent$ is the root, which represents the biggest block, then when WCV finishes the fourth sub-block, it goes back to the starting point (see S_2' in Figure 1.b). If $parent$ is not the root, WCV goes to the next bigger block (see S_4'' in Figure1.c).

Having the seq_i and $nodes_i$ of the 4 sub-blocks, we attach them to $parent$ as its children in the order of how seq indicates and put them into Q. When Q becomes empty and the **while** loop breaks, we run Depth-First-Search (DFS) on $\mathcal{T}$ and output those $nodes$ having exactly one node in them.

Given the locations of $M = 10$ nodes in Figure 2, an example of Quadtree generated by Algorithm 1 is shown in Figure 3.

In Algorithm 1, the **while** loop takes $\mathcal{O}(N)$ and the DFS takes $\mathcal{O}(N)$. Finally, the time complexity of proposed algorithm HQuad is $\mathcal{O}(N)$.

IV. Analysis

We focus on an area of $r \times r$ and mark four sub-block as illustrated in Figure 1.a. Each of these blocks has a size of $r/2 \times r/2$. We name these blocks S_i, $i = 1, 2, 3, 4$, counterclockwise starting from top left.

We introduce a binary tuple $A = (b_1, b_2, b_3, b_4)$ to represent whether each sub-block contains *at least* one node. For example, $A = (1, 1, 0, 1)$ means each of the sub-blocks s_1, s_2, s_4 contains at least one node and the sub-block s_3 does not contain any node. Obviously there are 15 different settings for A, which we define as $\mathcal{A}$, ranging from $(0, 0, 0, 1)$ to $(1, 1, 1, 1)$.

Define $f(r, \rho, n)$ as PDF of number of nodes, n, within a block $r \times r$, given density ρ. We further assume Poisson point

Algorithm 1 $HQuad(nodes_0)$

```
seq ← (S1, S2, S3, S4)
root ← (seq, nodes)
Add root into a queue Q
while Q ≠ ∅ do
    parent ← Q.poll()
    (seq, nodes) ← parent
    if Size of nodes_0 is 1 then
        Continue

    (Sa, Sb, Sc, Sd) ← seq
    seq1 ← (Sa, Sd, Sc, Sb)
    seq2 ← seq3 ← seq
    if parent is root then seq4 ← (Sc, Sd, Sa, Sb)
    else seq4 ← (Sc, Sb, Sa, Sd)

    for i = 1 to 4 do
        nodes_i ← nodes within the sub-block seq[i]
        child_i ← (seq_i, nodes_i)
        Add child_i into Q
        Attach child_i as No.i child of parent
Run Depth-First Search (DFS) from root, meanwhile
output non-empty set nodes of every leaf.
```

distribution for ease of analysis (although we do not expect to see any difference in the operation of HQuad in other node distributions), we have

$$f(r, \rho, n) = \frac{(\rho r^2)^n}{n!} e^{-\rho r^2} \ . \tag{1}$$

Define $g(A, r, \rho)$ as the probability that, for an $r \times r$ block with node density ρ, the nodes are distributed as A indicates.

$$g(A, r, \rho) = \left(1 - f(\frac{r}{2}, \rho, 0)\right)^{|A|} \cdot \left(f(\frac{r}{2}, \rho, 0)\right)^{4-|A|}, \tag{2}$$

where $|A|$ is the number of 1's in tuple A.

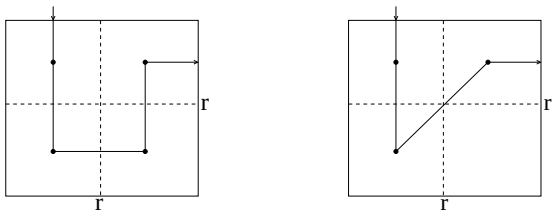
Assume that each node is at the center of its block. Define $h(A, r)$ as the Euclidean distance needed to visit all the nodes located at the center of sub-blocks and the path should enter the top-left sub-block and exit the top-right sub-block. For example, when $A = (1, 1, 1, 1)$, then $h((1, 1, 1, 1), r) = 2r$ as illustrated in Figure 4a. When $A = (1, 1, 0, 1)$, then $h((1, 1, 0, 1), r) = (1 + \frac{\sqrt{2}}{2})r$ as illustrated in Figure 4b. There are 16 mappings as listed in Table I.

Define $D(r, \rho)$ as the mean of length of Hamiltonian cycle needed for an $r \times r$ block with node density ρ. It can be calculated recursively.

$$D(r, \rho) = \sum_{A \in \mathcal{A}} \Big[h(A, r)(1 - \frac{1}{|A|}) + |A| \cdot D(\frac{r}{2}, \rho) \Big] \cdot g(A, r, \rho) \ , \tag{3}$$

TABLE I: Function $h(A, r)$

A	$h(A,1)$	A	$h(A,1)$
0 0 0 1	0.809	0 0 1 0	1.4604
0 0 1 1	1.6514	0 1 0 0	1.6514
0 1 0 1	1.7071	0 1 1 0	1.809
0 1 1 1	2	1 0 0 0	1
1 0 0 1	1	1 0 1 0	1.5161
1 0 1 1	1.7071	1 1 0 0	1.6514
1 1 0 1	1.7071	1 1 1 0	1.809
0 0 0 0	0	1 1 1 1	2



(a) $A = (1, 1, 1, 1)$ (b) $A = (1, 1, 0, 1)$

Fig. 4: Illustrations of $h(A, r)$ with different A.

in which the term $1-\frac{1}{|A|}$ reduces the overlap of length between different recursive levels. The $|A| \cdot D(\frac{r}{2}, \rho)$ represents the length sum of sub-blocks.

Define $\Theta_L(r, \rho, \ell)$ as CDF of the length, ℓ, of Hamiltonian cycle for an $r \times r$ block with node density ρ. It can be calculated recursively as well. Denote $\mathcal{I}(A)$ the set of index of terms as 1 in A, e.g., for $A = (1, 1, 0, 1)$, $\mathcal{I}(A) = \{1, 2, 4\}$.

$$\Theta_L(r,\rho,\ell) = Pr(L \le \ell \mid r,\rho) = \sum_{A\in\mathcal{A}} \Bigg\{ \int \cdots \int_{\ell_i=0, i\in\mathcal{I}(A)}^{\mu} Pr\Big[\sum_{i\in\mathcal{I}(A)} \ell_i + h(A,r)(1-\frac{1}{|A|}) \le \ell\Big] \cdot \prod_{i\in\mathcal{I}(A)} \Theta_L(\frac{r}{2},\rho,\ell_i)d\ell_i \Bigg\} \cdot g(A,r,\rho) \tag{4}$$

Recursion continues until $f(r, \rho, 2) < \epsilon$ (an accuracy control parameter), in which case

$$\Theta_L(r,\rho,\ell) = \int_{v=0}^{\ell} \lambda(r,v), \tag{5}$$

where $\lambda(r, v)$ is defined as the PDF of the distance, v, between two random points in an $r \times r$ square [15]. Let $s = v^2$, define

$$\Lambda(r,s) = \begin{cases} \frac{\pi}{r^2} - 4\frac{\sqrt{s}}{r^3} + \frac{s}{r^4}, & 0 < s \le r^2 \\ -\frac{2}{r^2} + \frac{4}{r^2} sin^{-1}(\frac{r}{\sqrt{s}}) & \\ +\frac{4}{r^3}\sqrt{s-r^2} - \frac{\pi}{r^2} - \frac{s}{r^4}, & r^2 < s \le 2r^2. \end{cases}$$

Then $\lambda(r, v) = \Lambda(r, s)\frac{ds}{dv} = 2v\Lambda(r, v^2)$.

In (4), the integral cap μ is supposed to be ∞. In practice, $\mu = 2D(\frac{r}{2}, \rho)$.

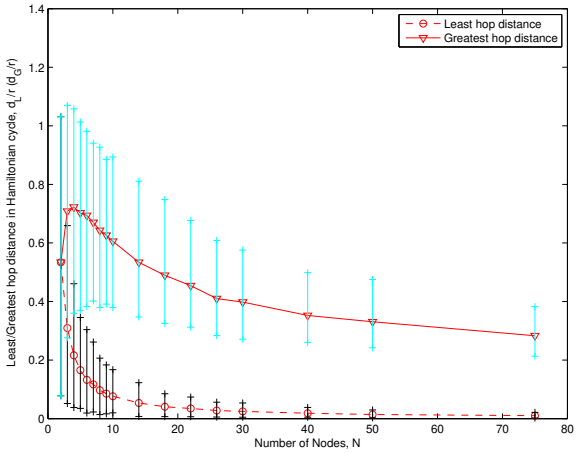


Fig. 5: Least and Greatest hop distance in each Hamiltonian cycle.

V. Performance Evaluation

We present performance evaluation in this section. We first show results from simulations, in which N nodes are randomly placed on an $r \times r$ region and we used a Genetic Algorithm [7] to identify the near optimum Hamiltonian cycles. Then the results of our HQuad algorithm and computation are shown and compared to the simulation results. Finally, we use an example of network planning to further demonstrate how to use these PDF results.

In Figure 5, we show the least and the greatest hop distance in each Hamiltonian cycle as a function of N. These represent the shortest and longest hop in the entire Hamiltonian cycle when N nodes are randomly deployed in the $r \times r$ region. The 95% confidence intervals are also shown in the same figure. In general, a decreasing trend can be observed in both greatest and least hop distance as N increases after 4, caused by the increasing number of nodes in the network. The initial increase of greatest hop distance when $N = 2, 3$ is because of the small number of hops in the entire Hamiltonian cycle. For example, there are only two hops between $N = 2$ nodes a region and three hops between $N = 3$ nodes.

In Figure 6, we compare simulation and numerical results on Hamiltonian cycle lengths as a function of N for different r (the region size). The 95% confidence intervals are also shown for the simulation results. It can be seen that these two sets of results match quite well with each other. Under large n and large r, our numerical results are lower than simulation results with larger gaps (due to recursive approximations).

We demonstrate how to use these PDF results in a hypothetical network planning. Suppose N nodes will be deployed to a $1,000 \times 1,000$ region and we have only one WCV to charge these devices trying to keep all of these active. The WCV has a battery capacity of 120 KJ, consumes 50 Joules per meter on traveling and 1 J/s in general, charges sensors at the rate of 2 J/s, and moves at a speed of 1 m/s [5].

We will use our numerical results to compute the probability of successful charging support of N nodes with the above

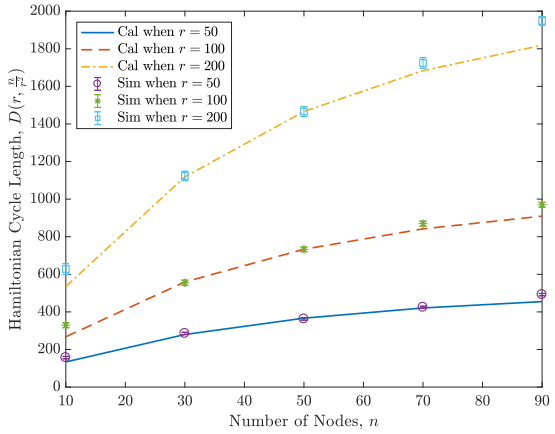


Fig. 6: Comparison between simulation and numerical results (3) on Hamiltonian cycle lengths, D.

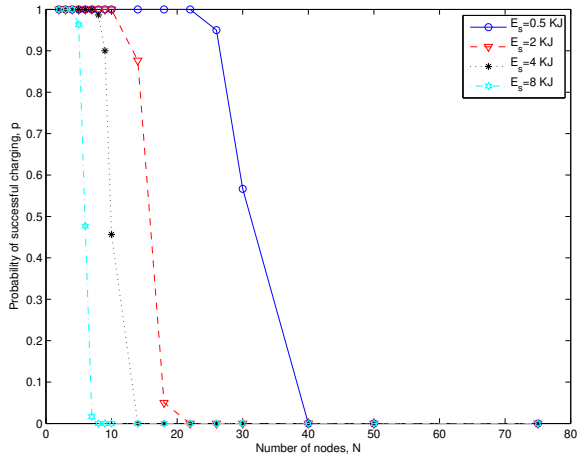


Fig. 7: Probability of successful charging with different sensor battery capacities as a function of N.

network parameters for different sensor battery capacities, $E_s = 0.5, 2, 4, 8$ KJ, respectively. Such probabilities are shown in Figure 7. When E_s is relatively small (assuming that energy consumption rate of the sensors is also small enough to wait for next round of charging), roughly 25 to 35 sensors can be supported. As E_s increases, fewer and fewer sensors can be supported. This is because of the heavier load to charge sensors and the longer Hamiltonian cycles for the WCV to traverse before it can return and replace/charge its own battery. For example, when $E_s = 8$ KJ, the network can support only 7 sensors.

Suppose our goal is to ensure that the WCV should have a battery that is large enough to provide successful charging to all sensors in the network with 80% and 95% probability, respectively. We can use the numerical results to compute the required capacity of the WCV battery, based on a fixed E_s value, e.g., 4 KJ. We show the results on Table II. The increase in required WCV battery capacity is due to both of the additional energy needed to charge the sensors as N increases.

TABLE II: Battery Capacity Requirement

N	2	4	8	22	50
$E_{WCV}(80\%)$	43.9	71.3	109.0	221.5	428.2
$E_{WCV}(95\%)$	52.6	77.6	115.2	225.3	432.7

VI. Conclusion

In this paper, we have investigated the charging trip problem of the WCV in wireless recharging networks. We have introduced a fast heuristic algorithm, HQuad, that generates Hamiltonian cycle in $\mathcal{O}(N)$, based on but not restricted to which we have analyzed and modeled the length of Hamiltonian cycle by giving a cumulative distribution function (CDF) $\Theta_L(r, \rho, \ell)$, as well as a mean function $D(r, \rho)$.

Simulation results and Calculation results not only show that HQuad works well, but also demonstrates that the probabilistic models are precise.

References

[1] L. Xie, Y. Shi, Y. T. Hou, and H. D. Sherali, "Making sensor networks immortal: An energy-renewal approach with wireless power transfer," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1748–1761, Dec. 2012.

[2] P. Cheng, S. He, F. Jiang, Y. Gu, and J. Chen, "Optimal scheduling for quality of monitoring in wireless rechargeable sensor networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 3072–3084, June 2013.

[3] C. Wang, J. Li, F. Ye, and Y. Yang, "A mobile data gathering framework for wireless rechargeable sensor networks with vehicle movement costs and capacity constraints," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2411–2427, Aug 2016.

[4] S. Guo, C. Wang, and Y. Yang, "Joint mobile data gathering and energy provisioning in wireless rechargeable sensor networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 12, pp. 2836–2852, Dec 2014.

[5] G. Jiang, S. K. Lam, Y. Sun, L. Tu, and J. Wu, "Joint charging tour planning and depot positioning for wireless sensor networks using mobile chargers," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–17, 2017.

[6] G. Reinelt, "Fast heuristics for large geometric traveling salesman problems," *ORSA Journal on Computing*, vol. 4, no. 2, pp. 206–217, 1992.

[7] A. M. Frieze and J. E. Yukich, *Probabilistic Analysis of the TSP*. Boston, MA: Springer US, 2007, pp. 257–307.

[8] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The traveling salesman problem: a computational study*. Princeton university press, 2011.

[9] S. Arora, "Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems," *Journal of the ACM (JACM)*, vol. 45, no. 5, pp. 753–782, 1998.

[10] L. Xie, Y. Shi, Y. T. Hou, W. Lou, H. D. Sherali, and S. F. Midkiff, "Multi-node wireless energy charging in sensor networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 23, no. 2, pp. 437–450, 2015.

[11] S. Martello, "Algorithm 595: An enumerative algorithm for finding hamiltonian circuits in a directed graph," *ACM Transactions on Mathematical Software (TOMS)*, vol. 9, no. 1, pp. 131–138, 1983.

[12] M. Krivelevich, C. Lee, and B. Sudakov, "Compatible hamilton cycles in random graphs," *Random Structures & Algorithms*, vol. 49, no. 3, pp. 533–557, 2016.

[13] V. Vig and U. S. Palekar, "On estimating the distribution of optimal traveling salesman tour lengths using heuristics," *European Journal of Operational Research*, vol. 186, no. 1, pp. 111 – 119, 2008.

[14] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 187–260, 1984.

[15] J. Philip, *The probability distribution of the distance between two random points in a box*, ser. TRITA / MAT / MA: TRITA. KTH mathematics, Royal Institute of Technology, 2007.