# Delay Efficient Data Gathering in Sensor Networks

Xianjin Zhu, Bin Tang, and Himanshu Gupta

Department of Computer Science,
State University of New York at Stony Brook, Stony Brook, NY 11794
{xjzhu, bintang, hgupta}@cs.sunysb.edu

**Abstract.** Data gathering is a very important functionality in sensor networks. Most of current data gathering researches have been emphasized on issues such as energy efficiency and network lifetime maximization; and the technique of data aggregation is usually used to reduce the number of radio transmissions. However, there are many emerging sensor network applications with different requirements and constraints. Rather, they are time critical, i.e., delivering sensed information of each individual sensor node back to a central base station quickly becomes most important. In this paper, we consider collision-free delay efficient data gathering problem in sensor networks, assuming that no data aggregation happens in intermediate nodes. We formally formulate this problem and propose optimal and near-optimal algorithms for different topologies. Particularly, in general topology, we present two approximation algorithms with performance ratio of 2 and $1+1/(k+1)$, respectively.

## 1 Introduction

### 1.1 Motivation

Recent advances in miniaturization of computing devices with advent of efficient short-range radios have given rise to strong interest in sensor networks [1],[2]. Sensor networks are ad hoc multi-hop wireless networks formed by a large number of low-cost sensor nodes with limited battery power and processing capacity. Wireless sensor networks are expected to be used in a wide range of applications, such as military surveillance, environmental monitoring, target tracking, etc. One of the most important communication primitives that has to be provided by sensor networks is data gathering, i.e., information collected at sensors has to be transmitted back to a sink node, which is responsible for further processing for end-user queries.

Since sensor nodes are usually deployed in adverse environments, it is often not feasible to replace or recharge their batteries. In order to prolong network lifetime, how to maintain energy efficient data gathering has attracted a lot of attention. Most of current data gathering researches have been conducted towards this goal [3],[4],[5],[6]. Some works also use data aggregation techniques [7],[8],[9] to reduce the number of radio transmissions, which is the main drain of the

X. Jia, J. Wu, and Y. He (Eds.): MSN 2005, LNCS 3794, pp. 380–389, 2005.

battery of sensor nodes. Data aggregation is especially appropriate when sensor networks have high density and as a result, data sensed at neighboring nodes are either highly correlated or simply redundant.

However, there are many emerging sensor network applications with different requirements and constraints. Rather, they are time critical, i.e., delivering sensed information of each individual sensor node back to a central base station as fast as possible becomes most important. Since the data is usually time varying in sensor networks, it is essential to guarantee that data can be successfully received by the sink the first time instead of being retransmitted due to collisions. In such scenarios, efficient data gathering scheduling is needed, which specifies how the data collected at each individual node is transmitted to the sink or base station without any collisions, within the shortest period of time. In this paper, we restrict our focus on collision-free delay efficient data gathering problem in sensor networks, assuming that no data aggregation happens in intermediate nodes. We first use a concrete example to motivate our research.

**Motivating Example.** For patient monitoring in hospital, a sensor network may consist of many sensor instruments, which are attached to patients to monitor their physical conditions. Medical data sensed at each sensor is transmitted back to a central platform for diagnosis. Due to the short radio transmission range, sensed data is transmitted in multi-hop manner. Such application has several characteristics. First, energy is not a concern as the battery of each sensor can be easily replaced. Secondly, the data collected from each individual patient is independent and not correlated in anyway. All the illness information of each patient has to be sent back to the central platform separately and no aggregation on the intermediate nodes is desired. Thirdly, the time criticality of such application is obvious without further explanation.

The delay efficient data gathering problem is not limited in sensor networks. It is suitable for any real time data gathering applications. For example, in network monitoring, wherein each individual computer in the network is required to send back to the server of its own source or traffic information. This monitoring happens in rounds and all the nodes are synchronized to send the data back. This way server can gather all the information and generate a complete picture of the network; and react quickly to coordinate for resource allocation and traffic congestion control, etc. In this scenario, data aggregation in intermediate nodes is obvious unwanted. Rather, the delay window (the period within which messages from all the nodes are received by the base station) is more critical since the quicker the central server gets all the necessary information, the more timely decisions it can make as to the actions each node has to take to prevent undesirable situations from happening.

### 1.2 Related Work

The work which is most related to the problem we consider in this paper is [10]. It studies the problems of data distribution and data collection in wireless sensor networks via simple discrete mathematical models. Our approach differs

with it in two aspects. First, instead of solving a converse distribution problem wherein a base station transmits data packets to nodes as proposed in [10], we design optimal or near optimal algorithms to directly solve the data gathering problem. Second, to further reduce delay while avoiding collisions, we introduce multiple channels in our work. In [10], a transmission from node $N_i$ to node $N_j$ is successful only if none of the neighbor nodes of receiver $N_j$ is transmitting simultaneously. With this model, they present a strategy for general graph networks within a factor of 3 of the optimal performance. In our model, each node is equipped with two channels. This way its two non-adjacent neighbors can transmit simultaneously as long as they choose different channels. As we will show later, such model will result in an approximation algorithm with performance ratio 2 in general graph networks. Multiple channels were shown to be more efficient for multi-hop wireless networks [11],[12].

Another closely related work is [13]. It studies the problem of minimum latency broadcast in ad hoc networks. It presents a collision-free broadcast algorithm which simultaneously produces provably good solutions in terms of latency and the number of retransmissions. Our proposed problem is not a simple reverse engineering of above problem. Unlike [13], in which one single message is broadcast to all the nodes, our work studies the condition whereby different messages are gathered and sent back to one node. In this sense, our problem is conjectured to be even harder than that in [13].

There are some research work [14],[15],[6] also realizing the importance to consider the delay incurred in gathering sensed data, in addition to minimizing energy. They capture this with the $engergy \times delay$ metric and present schemes that attempt to balance the energy and delay cost for data gathering from sensor networks. Our goal in this paper is solely delay minimization.

Paper Organization. The remainder of this paper is organized as follows. In section 2, we specify our network model and present the problem statement. In section 3, we design and analyze our data gathering algorithms for different special topologies. Two heuristics for general topology are then presented in section 4. We conclude our paper in section 5.

## 2 Network Model and Problem Statement

**Network Model and Assumptions.** We model the sensor network as a disk graph $G = (V, E)$, where $V = \{1, 2, \ldots, N\}$ is the set of sensor nodes and $E$ is the set of undirected edges. The edge exists between two nodes if their Euclidean distance is within the transmission range of each other (for simplicity, we assume all the nodes have the same transmission range). There is only one sink node, which is used to collect and analyze data for interest queries. All nodes are sensing data and trying to send/relay data back to the sink node. A *collision* happens at a node if it hears a message from more than two transmitters at the same time. Our goal is to intelligently schedule each node's receiving/transmitting time in order to guarantee collision-free data gathering with minimum delay.

We assume that each node has half duplex interface and is equipped with two channels. Therefore, it can not receive/transmit at the same time, but its non-adjacent neighboring nodes can transmit simultaneously as long as they choose different channels. Each node has one message waiting to be transmitted back to the sink node, but it has limited buffer size such that relay nodes only perform simple receive and forward type operations, i.e., packets received must be forwarded in the next time slot following its arrival. Such constraint is also used in [10]. Furthermore, in this work, we do not consider data aggregations, which may be explored in future research.

**Problem Statement.** Given a disk graph $G = (V, E)$ and a sink node $S$, node $S_i \in V$ is the source of data item $i$, our data gathering problem is to schedule for each data item $j$ and each node $S_i$, the time slot at which node $S_i$ transmits/receives data item $j$ collision-free, such that the time when the last data item received by $S$ is minimized.

## 3 Algorithms for Special Topologies

In this section, we address the delay efficient data gathering problem in some special topologies. We start with the simplest linear topology and prove the optimality of our solution. Then we propose the approach to solve our problem in star and tree topologies by exploring the relationship between these two topologies and the linear topology.

### 3.1 Linear Topology

In linear topology, all nodes are arranged in a line and the sink node $S$ is at one end of the line. The case wherein $S$ is at an arbitrary position instead of two ends of the line can be considered as a special case of star topology, which will be discussed later in Section 3.2. We label the nodes in sequence from 1 to $N$, with nodes closer to the sink assigned lower IDs (see Fig. 1). Before explaining the details of our algorithm, we present a method to assign channels for each node at each time slot, such that nodes two hops away from each other can send packets simultaneously without causing collisions.

Channel Assignment. Suppose every node can use two channels, viz. channel 1 and channel 2. The sink always listens to and receives packets at channel 1. For any other node $i$ ($1 \leq i \leq N$), the channel that it should use at time slot $t$ is chosen based on the values of $i$ and $t$. Basically, nodes with even IDs switch between two channels alternatively while nodes with odd IDs always stay at one assigned channel, so that a node is always assigned a different channel with its 2-hop neighbors and a pair of transmitter and receiver stay on a same channel. Such assignment guarantees that node $i$ and node $i+2$ can send packets collision free at the same time.

Algorithm for Data Gathering. As in channel assignment algorithm, we divide nodes into two sets based on their IDs, viz. odd set and even set. Similarly,

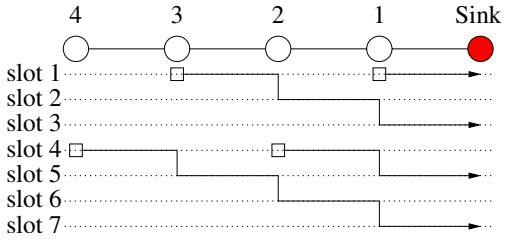



**Fig. 1.** Scheduling in linear topology: nodes are divided into odd set and even set; nodes in the same set transmit packets simultaneously

packets are also divided into two sets, viz. packets from nodes in odd set $P_{odd}$ and from nodes in even set $P_{even}$. With the channel assignment presented previously, for any $i > 0$, node $i$ and node $i+2$ use different channels to send packets at any time slot. Therefore, nodes in the same set can transmit packets collision-free simultaneously. Since in our network model, every node has a packet to be sent back to the sink, we let all packets in $P_{odd}$ be transmitted first. During this process, nodes both in odd set and in even set are involved in relaying those packets. After the sink gets all packets in $P_{odd}$, nodes in the even set then start to transmit their own packets in following time slots. Such scheduling guarantees that every sender-receiver pair uses the same channel so that each packet can be forwarded one hop per slot once the transmission starts.

A simple example is illustrated in Fig. 1. Initially, node 1 and 3 simultaneously send their packets to the sink. At the end of time slot 3, the sink has received packets both from 1 and 3. Then node 2 and 4 start their transmissions at time slot 4. The total delay in this example is 7.

Proof of Optimality. In the following, we show the above algorithm is optimal.

**Lemma 1.** $2N-1$ *is the lower bound for delay of data gathering in linear topology, i.e., for any algorithm, the resulting delay is at least* $2N-1$*, where* $N$ *is the number of nodes in the network excluding the sink.*

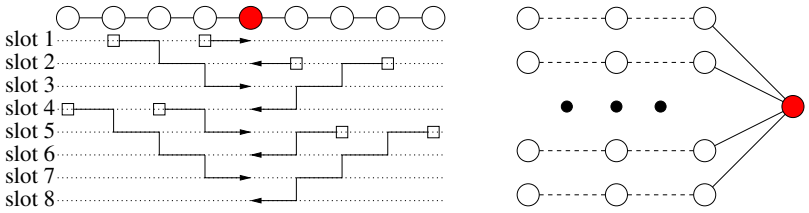
*Proof.* In the linear topology, before packets are received by the sink, they must pass through node 1 (the closest node to the sink). For packets from node $2, 3, \ldots, N$, each of them needs two slots to be received and forwarded by node 1. For the packet of node 1 itself, it only needs one slot to be sent to the sink. Thus, the total delay is at least $2(N-1)+1 = 2N-1$.

**Theorem 1.** *The above data gathering algorithm in linear topology is optimal.*

*Proof.* First, we prove the delay produced by our algorithm is exactly $2N-1$. The total delay equals to the time taken by packets in both $P_{odd}$ and $P_{even}$ to be received by the sink. For each set of packets, since packets are forwarded one hop per slot in our scheduling, the delay is bounded by the largest hop count away from the sink. Thus, the delay is $N$ for one set and $N-1$ for the other set. Straightforwardly, total delay is $2N-1$. Since the result delay of our algorithm exactly matches with the lower bound, our algorithm is optimal.

### 3.2 Star Topology

In star topology, the sink $S$ has $m$ linear branches. A branch $i$ has arbitrary number of nodes $N_i$ and the total number of nodes is $N = \sum_{i=1}^{m} N_i$. Each branch can be assigned channel separately with the channel assignment algorithm for linear topology[1].




(a) Special case of star topology: sink has two equal size branches

(b) General star topology

**Fig. 2.** Data Gathering in Star Topology

Obviously, the data gathering algorithm for linear topology can also be applied directly on each branch, and the result delay is $2N - m$, since the sink can receive one packet every two slots on average. However, an ideal scheduling to get minimum delay in star topology must fill every idle slot so that the sink can get one packet one slot. In such ideal case, the total delay is $N$. In a special case of star topology, wherein the sink only has two equal size branches (see Fig. 2(a)), transmissions of each branch can be pipelined in a proper way to meet the target of one packet one slot. Now we generalize the scenario to $m$ branches (see Fig. 2(b)). If we can divide all branches into two groups which contain same number of nodes, then transmissions of these two groups can be pipelined in a similar way to minimize the delay. Thus, the key becomes how to make such division.

NP-Completeness of Group Division Problem. We prove the NP-completeness of our group division problem by reduction from the well-known Integer Partition problem. We first give the decision version of these two problems separately, and then show the proof.

**Definition 1.** *Group Division problem is defined as: given a star topology with $m$ branches, is it possible to divide $m$ branches into two groups so that each group has equal number of nodes?*

[1] This channel assignment algorithm can also be easily generalized to tree and general topology. For general topology which can be decomposed into several independent components, each component can be assigned channel separately. Otherwise, we may need 3 channels.

**Definition 2.** *Integer Partition problem $IP = (X, y)$ is defined as: given a set of integers $X = \{x_1, x_2, \ldots, x_n\}$ and a target number $y$. Is there a subset $X' \subseteq X$, such that the sum of all the elements in $X'$ is equal to $y$?*

**Theorem 2.** *Group division problem is NP-complete.*

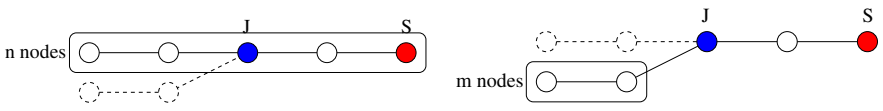
*Proof.* Given a group division, since there are only polynomial number of nodes in total, we can decide whether this division is valid or not in polynomial time. Thus, the group division problem is NP-hard.

To prove it is NP-complete, we show a polynomial reduction from Integer Partition problem $IP = (X, y)$. For each $x_i \in X$, we create a branch with $x_i$ nodes. After doing this, we also create an extra branch with $(\sum_{x_i \in X} x_i - 2y)$ nodes. All these branches are connected to the sink. If $IP = (X, y)$ has a solution, i.e., we can find a subset $X'$ such that $\sum_{x_i \in X'} x_i = y$, there is also a solution to the group division problem. Since $\sum_{x_i \in X'} x_i + (\sum_{x_i \in X} x_i - 2y) = (\sum_{x_i \in X} x_i + (\sum_{x_i \in X} x_i - 2y))/2$, the corresponding branches of all $x_i \in X'$ plus the extra branch contain exact half of the total number of nodes. On the other hand, if we can divide all branches into two equal sets, the extra branch with $(\sum_{x_i \in X} x_i - 2y)$ nodes must be in one set. Then, the integers corresponding to the rest branches in this set make up of the solution to the integer partition problem, since $(\sum_{x_i \in X} x_i + (\sum_{x_i \in X} x_i - 2y))/2 - (\sum_{x_i \in X} x_i - 2y) = y$.

### 3.3 Tree Topology

In a tree topology with the sink $S$ as the root, all $N$ other nodes are arranged as an arbitrary tree. First, let us look at a simple situation when the root has only one child. The basic idea here is to view it as conjunction of multiple lines, which can be processed one by one.

Algorithm for Single Child Tree. For the topology in Fig. 3, it can be viewed as conjunction of two lines, viz. the top line (Fig. 3(a)) and the bottom line (Fig. 3(b)), which are jointed at node $J$. Suppose there are $n$ nodes in the top line, and there are $m$ nodes in the bottom line except the joint part. In this case, $N = n + m$. The first step is to process the top line, where the data gathering algorithm for linear topology can be applied. Supposing that node $J$ is $k$ hops away from the sink $S$, it is easy to know that the last packet of the top line will leave node $J$ at time slot $2n - 1 - (k - 1)$. Otherwise, all the packets of the top line cannot be collected by the sink in $2n - 1$ time slots. Until this time, all the




(a) Processing of the top line. (b) Processing of the bottom line.

**Fig. 3.** Data gathering for the single child tree topology

nodes in the top line after node $J$ have been processed, and we can continue to the bottom line. For the bottom line without the joint part, it can be handled as if node $J$ is the sink. Beginning from slot $2n-k+1$, the nodes in the bottom line can begin to send one packet to node $J$ every two time slots. As soon as node $J$ receives the first packet from the bottom line, it immediately forwards the packet to the sink $S$. As a result, the first packet from the bottom line arrives at $S$ at slot $2n-k+1+k=2n+1$. Thereafter, $S$ can continue to receive one packet from $J$ every two time slots until all the remaining $m-1$ packets have been collected. Then, the total gathering delay is $2n+1+2(m-1)=2N-1$.

Generalizing the above algorithm, for any given single child tree, it can be eventually decomposed as a bunch of lines, which can be processed one-by-one in a similar way according to certain order.

**Lemma 2.** *For a single child tree whose root has only one child, the minimum gathering delay is* $2N-1$. *$N$ is the total number of nodes except the root.*

*Proof.* Similar as analysis in linear topology, for the single direct child of the sink, it takes two time slots to transmit a packet from any of the rest $N-1$ nodes and one slot to send its own packet to the sink. Thus, the total gathering delay is at least $2(N-1)+1=2N-1$.

**Theorem 3.** *The proposed data gathering algorithm for the single child tree topology achieves the optimal delay.*

After finding the optimal solution for the single child tree, we continue to the general tree topology, as illustrated in Fig. 4. Based on the above discussion, it can be seen that, from the point view of the sink, a single child subtree is identical to a line. Hence, a general tree can be processed as a star with multiple branches. We define a direct subtree as a subtree whose root is the direct child of the sink. Then, if all direct subtrees can be divided into two equal size groups, the two groups can be fully pipelined so that the sink can receive one packet each time slot alternately from the two groups, and an optimal delay of $N$ can be achieved. As discussed earlier, the group division problem is NP-complete, and no efficient polynomial solution exists. However, in the worst case, the direct subtrees can be processed one by one, and the upper bound of the delay is $2N-1$.

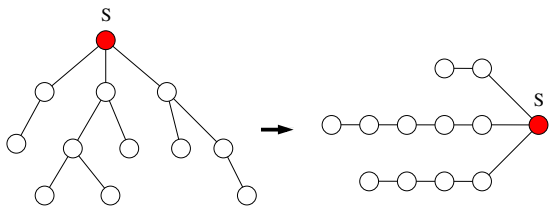



**Fig. 4.** A tree whose root has multiple children can be viewed as a star

## 4 Heuristics for General Topology

In this section, we propose two heuristics for general topology based on the techniques designed in the previous section. The first heuristic is simple. We first generate a BFS tree for the original graph [16]. Then, the data gathering algorithm for tree topology can be applied. We have known that the upper bound of the gathering delay for any tree is $2N - 1$, and the lower bound of the gathering delay is $N$. Thus, the tree based solution for general topology guarantees 2-approximation.

In the following, we further propose a more efficient heuristic to divide the set of direct subtrees into two equal size groups and then pipeline transmissions of these two groups to fill as many idle slots as possible at the sink.

Group-based Heuristic Algorithm. For Integer Partition problem, there exists a $(1+1/k)$ approximation algorithm [17], for a constant integer $k$. Such algorithm can be used to get an approximate group division. The basic idea is as follows. We consider each of the subsets with at most $k$ direct subtrees for a given constant $k$. If the number of nodes contained in the subset has not exceeded half of the total number, we add the remaining direct subtrees as many as possible. A subset with maximal number of nodes is selected at last. Obviously, the algorithm will run more times and get more accurate answer with a larger $k$. After dividing all direct subtrees into two groups, we pipeline the transmissions of these two groups so that the sink can receive packets from each group alternatively.

Theoretical Analysis. With the group-based heuristic, all direct subtrees can be divided into two groups $B'$ and $B - B'$. The size of $B'$ and $B - B'$ must satisfies that $kN/2(k+1) \leq |B'| \leq N/2$ and $N/2 \leq |B - B'| \leq N - kN/2(k+1)$. Thus, in the worst case, the difference of the number of nodes in these two sets is equal to $N - kN/2(k+1) - kN/2(k+1) = N/(k+1)$. From the observations we get in the previous section, the total delay is at most $2kN/2(k+1) + 2N/(k+1) - 1 \leq N + N/(k+1)$. Therefore, with any $1 + 1/k$ approximation algorithm for group division, we can get a $1 + 1/(k+1)$ approximation algorithm for data gathering.

**Theorem 4.** *For any $1+1/k$ approximation algorithm for group division, there exists a corresponding $1+1/(k+1)$ approximation algorithm for data gathering.*

## 5 Conclusions

In this paper, we have studied the collision-free delay efficient data gathering problem in sensor networks by tackling with different topologies. In linear topology, we prove that an optimal delay of $2N-1$ can be achieved with two channels for each node, where $N$ is the number of nodes except the sink in the graph. In star topology, we conclude that if all branches can be divided into two equal size groups, an optimal delay of $N$ can be achieved. This group division problem is proved to be NP-complete. In tree topology, we show that a general tree can be processed as a star topology and a 2-approximation algorithm also applies. Finally, for the general topology, we propose the group based heuristic algorithm, which is able to achieve $1 + 1/(k + 1)$ approximation.

## References


1. Estrin, D., Govindan, R., Heidemann, J., eds.: Special Issue on Embedding the Internet, Communications of the ACM. Volume 43. (2000)
2. Badrinath, B., Srivastava, M., Mills, K., Scholtz, J., Sollins, K., eds.: Special Issue on Smart Spaces and Environments, IEEE Personal Communications. (2000)
3. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proceedings of the International Conference on Mobile Computing and Networking (MobiCom). (2000)
4. Heinzelman, W., Kulik, J., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks. In: Proceedings of the International Conference on Mobile Computing and Networking (MobiCom). (1999)
5. Chang, J., Tassiulas, L.: Energy conserving routing in wireless ad hoc networks. In: Proceedings of the IEEE INFOCOM. (2000)
6. Lindsey, S., Raghavendra, C., Sivalingam, K.M.: Data gathering algorithms in sensor networks using energy metrics. IEEE Transactions on parallel and distributed systems (2002)
7. Solis, I., Obraczka, K.: The impact of timing in data aggregation for sensor networks. In: Proceedings of the International Conference on Communications (ICC). (2004)
8. von Rickenbach, P., Wattenhofer, R.: Gathering correlated data in sensor networks. In: Proceedings of the 2004 joint workshop on foundations of mobile computing. (2004)
9. Cristescu, R., Beferull-Lozano, B., Vetterli, M.: On network correlated data gathering. In: Proceedings of the IEEE INFOCOM. (2004)
10. Florens, C., McEliece, R.: Packets distribution algorithms for sensor networks. In: Proceedings of the IEEE INFOCOM. (2003)
11. Nasipuri, A., Zhuang, J., Das, S.: A multichannel csma mac protocol for mobile multihop neworks. In: Proceedings IEEE Wireless Communications and Networking Conference. (1999)
12. So, J., Vaidya, N.: Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transciver. In: Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc). (2004)
13. Gandhi, R., Parthasarathy, S., Mishra, A.: Minimizing broadcast latency and redundancy in ad hoc networks. In: Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc). (2003)
14. Yu, Y., Krishnamachari, B., Prasanna, V.: Energy-latency tradeoffs for data gathering in wireless sensor networks. In: Proceedings of the IEEE INFOCOM. (2004)
15. Raghavendra, C., Sivalingam, K., Lindsey, S.: Data gathering in sensor networks using the energy*delay metric. In: Proceedings of IPDPS Workshop on Issues in Wireless Networks and Mobile Computing. (2001)
16. Skiena, S.S., Revilla, M.: Programming Challenges. (2003)
17. Baase, S., Gelder, A.V.: Computer Algorithms: Introduction to Design and Analysis. (2001)