Brief Announcement: k-Selection and Sorting in the SINR Model

Preliminary version of a brief announcement to appear at DISC'14

Stephan Holzer[*]
holzer@mit.edu
MIT

Sebastian Kohler
sekohler@student.ethz.ch
ETH Zurich

Roger Wattenhofer
wattenhofer@ethz.ch
ETH Zurich

Abstract

We study algorithms and lower bounds for k-selection and sorting in the SINR model. For the problem of finding the k-th smallest value in the network, we provide a $\mathcal{O}(\log^2 n)$ algorithm based on the aggregation trees presented in [3]. We argue that any algorithm using this approach has runtime $\Omega(\log^2 n/\log\log n)$. We show that sorting can be done in $\Theta(n)$ time slots.

1 Introduction

Data aggregation is one of the most basic tasks in wireless networks. The goal of data aggregation in a network is to compute some aggregation function such as the sum, maximum or the median of (the input values of) the nodes. In the past 8 years the signal-to-interference-plus-noise-ratio (SINR) model (or *physical model*, [2]) which is known to model interference more accurately than graph-based models of wireless networks (see, e.g., [5]) has gained significant attention from an algorithmic viewpoint. In [3], aggregation trees were presented that can directly be used to compute *distributive*[1] aggregation functions such as sum, maximum and minimum in time $\mathcal{O}(\log n)$. We study these aggregation trees with respect to a *holistic* aggregation function, namely finding the k-largest value in a network (k-selection). For this generalization of computing the median we provide an almost optimal algorithm using this approach. As a side-effect this demonstrates that any speedup (which can be at most quadratic) can only be obtained using different techniques.
Finally we study sorting, another basic task when working with any kind of data, which is a problem related to k-selection. We provide matching upper and lower bounds for this task in the SINR model.
Note that our algorithm for k-selection does not explicitly use the SINR model and is based on the aggregation trees obtained within the SINR model in [3], while our results on sorting are directly obtained in the SINR model.

2 Model and Preliminaries

We consider a set $V := \{v_1, v_2, \ldots, v_n\}$ of $n := |V|$ nodes. Each node $v \in V$ has an arbitrary position $\text{pos}_v \in \mathbb{R}^2$ in the Euclidean plane. The Euclidean distance between two nodes $u, v \in V$ is denoted by $d(u,v) := \|\text{pos}_v - \text{pos}_u\|_2$. Each node $v \in V$ has a unique ID $\text{id}_v \in [n]$ (We use the notation $[m] := \{1, 2, \ldots, m\}$ for $m \in \mathbb{N}$) and is given an arbitrary input value $x_v \in [W]$ for some

[*]Work has been done at ETH Zurich. At MIT the author is supported by the following grants: AFOSR Contract Number FA9550-13-1-0042, NSF Award 0939370-CCF, NSF Award CCF-1217506, NSF Award number CCF-AF-0937274.

[1]An aggregation function $f\colon A^m \mapsto B$ for an integer m and sets A and B is *distributive* if for all $(a_1, a_2, \ldots, a_m) \in A^m$ and for every arbitrary partition $I_1 \dot\cup I_2 \dot\cup \ldots \dot\cup I_k = [m]$ for some $k \leq m$ there exists a function $g\colon B^k \mapsto B$ such that $f(a_1, a_2, \ldots, a_m) = g(f(\{a_i\}_{i\in I_1}), f(\{a_i\}_{i\in I_2}), \ldots, f(\{a_i\}_{i\in I_k}))$. For some $j \in [k]$, we call $f(\{a_i\}_{i\in I_j})$ a *subaggregate* of $f(a_1, a_2, \ldots, a_m)$. An aggregation function is *holistic* if there is no constant bound on the storage size required to describe a subaggregate.

$W \in \mathbb{N}$. Time is slotted into discrete *time steps* of equal length and every node wakes up at the same time. Local computation does not count towards the complexity-measure as we are interested in communication complexity. Communication bandwidth is limited to only one message containing $\Theta(1)$ values from $[n]$ and $\Theta(1)$ values from $[W]$ can be sent/received by a node in a single time step. In each time step, each node $v \in V$ can choose an arbitrary *transmission power* $P_v \geq 0$. A message sent by a node s is received by node r if $P_r = 0$ and the perceived SINR at r exceeds a constant threshold $\beta > 1$, i.e., the SINR condition $\frac{P_s/d(s,r)^\alpha}{\sum_{s' \in V \setminus \{s\}} P_{s'}/d(s',r)^\alpha + N} \geq \beta$ is satisfied. Here, $\alpha > 2$ is the constant *path-loss exponent*. The positions of the nodes, their IDs, n and W are known to all nodes. We say a probabilistic event A happens *with high probability* (w.h.p.) if $\Pr[A] \geq 1 - 1/n$.

3 k-selection Algorithm

We use the construction of a minimum-latency aggregation schedule (MLAS) presented in Section 7.1 in [3], which is based on the fact that in our model $\Omega(n)$ links of a minimum spanning tree of V can be scheduled in a single time step. While the tree in [3] is stated to be directed towards the root, we can also obtain such a tree with *bidirectional links* using the bidirectional version of the *amenability* in [3]. Hence, we construct a tree spanning all nodes in the network on which we can execute a *convergecast* (i.e., compute an aggregated value at the root of the tree) and a *broadcast* (i.e., an inverse convergecast) with runtime $\mathcal{O}(\log n)$ each. This is a prerequisite for most aggregation functions (except distributive functions). We call this tree *aggregation tree* and a schedule consisting only of convergecasts and broadcasts a *level schedule*. Let w_k be the node with the k-smallest input value (we may assume that no two input values are the same since we can break ties by comparing the node IDs). The k-selection algorithm maintains a set C of *candidates* for w_k, that is initially equal to V, and iteratively removes candidates from C until it is a singleton containing w_k. One iteration consists of the following three steps with runtime $\mathcal{O}(\log n)$ each.

1. Each candidate is assigned a unique temporary ID as follows in a bottom-up fashion with respect to the tree. Each node $v \in V$ counts the number c_v of candidates in its subtree and sends a request for c_v temporary IDs to its parent. The root r eventually knows $|C|$. If $r \in C$, r assigns itself a temporary ID from $[|C|]$ and sends each child u an element of a partition of $[|C|]$ of size c_u. Each node in $V \setminus \{r\}$ proceeds similarly once it receives a set of temporary IDs from its parent.

2. The root sends a value i chosen uniformly at random from $[|C|]$ to every node in V. The node with temporary ID i then returns its input value x to the root.

3. Let $C_< := \{v \in C \mid x_v < x\}$. First, $|C_<|$ is calculated and then sent to every node in V. If $|C_<| \geq k$, then $w_k \in C_<$, otherwise, $w_k \in C \setminus C_<$. In either case, a constant fraction of the candidates can be discarded.

Theorem 3.1. *The k-selection problem can be solved in $\mathcal{O}(\log^2 n)$ time steps on an aggregation tree with a level schedule.*

The proof will be presented in the full version of this brief announcement.

4 Lower Bound for k-Selection

It has been shown in [3] that any distributive aggregation function can be computed in $\mathcal{O}(\log n)$ time steps in the SINR model with an MLAS. A matching lower bound is also mentioned in [3], which extends to k-selection, as finding the minimum is a special case of k-selection (i.e., k-selection with $k = 1$). Thus we could still hope for a quadratic speedup. However, this (if it is possible) requires new techniques since we can show that by using a level schedule this can not be achieved.

Theorem 4.1. *The number of time steps required to solve the k-selection problem w.h.p. in an aggregation tree with a level schedule is in $\Omega(\log^2 n/\log\log n)$.*

We only present an outline of the proof. The formal proof can be found in the full version of this brief announcement. The theorem is proved with two reductions. First, solving the k-selection problem cannot be harder than solving the k-selection problem w.r.t. a subset of V. Second, it can be shown that in every MLAS aggregation tree as constructed in [3], there exist two disjoint subsets of V of size $\Omega(\sqrt{n})$ with the property that sending a message from one set to the other requires $\Omega(\log n/\log\log n)$ time steps, as pipelining (e.g. sending $\log n$ elements over a path of lenght $\log n$ in time $\log n$) is not possible in this construction. Thus any algorithm that solves the k-selection problem on an aggregation tree with a level schedule can therefore be used to build an algorithm that is $\Omega(\log n/\log\log n)$ times faster in the setting of the two-party k-selection problem (see [4]). Those results together with a lower bound of $\Omega(\log n)$ for the two-party k-selection problem shown in [4] imply the lower bound stated in 4.1.

5 Sorting

We say that data in a network is sorted, when each node $v \in V$ knows the id_v-th smallest input value in the network. For the sorting algorithm and the lower bound we only require $\alpha > 0$. We can sort the values in the network in n time steps as follows. In each time step $t \in [n]$, node v with ID t sends its value x_v and every node $u \in V \setminus \{v_t\}$ sets $P_u = 0$. The value of P_{v_t} is chosen such that we have a single-hop network, that is the SINR exceeds β at the positions of every node in $V \setminus \{v_t\}$. Thus every node knows every input value in the network after n time steps and can locally determine the specified output. Note that there are faster sorting algorithms for special networks, e.g., if the nodes are placed in a grid [1].

Theorem 5.1. *Every (possibly randomized) algorithm in the SINR model for sorting has runtime $\Omega(n)$ in the worst case.*

The proof of 5.1 is based on the following lemma and can be found in the full version of this paper.

Lemma 5.2. *Let C_A and C_B be two non-overlapping discs in the Euclidean plane with radius r each and let d be the minimum distance between any two points $c_A \in C_A$ and $c_B \in C_B$. Let $A = \{v \in V \mid \mathrm{pos}_v \in C_A\}$ and $B = \{v \in V \mid \mathrm{pos}_v \in C_B\}$. If $d > 4r/(\beta^{1/\alpha} - 1)$, then at most one message can be sent from a node in A to a node in B (or vice versa) in a single time step.*

Given a parition $A \dot\cup B = V$ as described in the lemma with $|A|, |B| \in \Theta(n)$, we can assign the input values such that $\min\{|A|, |B|\} \in \Theta(n)$ input values have to be interchanged bet ween A and B in order to solve the sorting problem. This requires $\Omega(n)$ time steps by the lemma.

Acknowledgements: We thank Magnus Halldórsson for answering questions on his work [3].

References

[1] J.L. Bordim, K. Nakano, and H. Shen. Sorting on single-channel wireless sensor networks. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, pages 133–138, 2002.

[2] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[3] M.M. Halldórsson and P. Mitra. Wireless connectivity and capacity. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 516–526, 2012.

[4] F. Kuhn, T. Locher, and R. Wattenhofer. Tight bounds for distributed selection. In *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures (SPAA)*, pages 145–153, 2007.

[5] T. Moscibroda, R. Wattenhofer, and Y. Weber. Protocol Design Beyond Graph-Based Models. In *5th Workshop on Hot Topics in Networks (HotNets)*, 2006.