Optimizing Multiple In-Network Aggregate Queries in Wireless Sensor Networks*

Huei-You Yang, Wen-Chih Peng and Chia-Hao Lo
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan, ROC
E-mail:{hyyang, wcpeng, chlo}@cs.nctu.edu.tw

Abstract. In this paper, we explore the feature of sharing partial results of multiple queries to reduce the total number of messages incurred. Those queries sharing their partial results are referred to as backbones. Given a set of queries, we shall determine backbones with the purpose of minimizing the total number of messages. Specifically, given a set of queries, we derive a graph, where each vertex represents one query and the corresponding weight edge denotes the number of messages reduced by sharing partial results. Then, we develop a heuristic algorithm SB (standing for Selecting Backbones) to derive a cut in which both backbones and non-backbones are determined. Simulation results show that by sharing partial results, algorithm SB is able to significantly reduce the total number of messages involved.

1 Introduction

In monitoring applications of wireless sensor networks, queries are typically long-running and executed over a specified period. Since each query is independently performed, wireless sensor networks consume a considerable amount of energy when the number of queries increases. Queries issued could be categorized according to their aggregate operator and monitored time period. Queries in the same group have the same aggregate operator and monitored time period. As such, queries in the same group are able to further share their query results to reduce the number of messages. Given a set of queries with the same aggregate operator and time duration, we elaborate on sharing query results to reduce the total number of messages without influencing final query results. Queries are divided into two sets: backbone and non-backbone sets. Queries in the backbone set are issued as usual and should share their partial results with those queries in the non-backbone set. The problem addressed in this paper is that given a set of query trees, we should determine which query tree should be put in the backbone set and non-backbone set so as to minimize the total number of messages involved in multiple queries.

* This paper is supported in part by the National Science Council, Project No. NSC 95-2221-E-009-061-MY3 and NSC 995-2221-E-009-026, Taiwan, Republic of China.

In order to determine which query tree should be put in the backbone set and the number of backbones, we first formulate the problem of selecting backbones and transform this problem into Max-Cut problem. Specifically, given a set of queries, we derive a graph, where each vertex represents one query and the corresponding weight edge denotes the number of messages reduced by sharing partial results. According to the graph derived, we develop a heuristic algorithm SB (standing for Selecting Backbones) to derive a cut in which both backbones and non-backbones are determined. Performance of algorithm SB is comparatively analyzed and simulation results show that by sharing partial results, algorithm SB is able to significantly reduce the total number of messages.

A significant amount of research efforts have been elaborated upon issues of in-network query processing for power saving in wireless sensor networks [3][5][6]. Prior works [2][6] explore the feature of in-network aggregation in which sensor nodes in a routing tree are able to perform aggregate operators. The authors in [1] proposed in-network materialized view that could be shared by multiple queries to reduce the number of messages. To the best of our knowledge, no prior works exploit the feature of sharing partial results of in-network aggregate queries, let alone formulating the problem of selecting backbones and devising algorithms to determine backbones for partial result sharing.

The rest of this paper is organized as follows. Preliminaries are presented in Section 2. In Section 3, we develop algorithm SB for backbone selection. Performance studies are conducted in Section 4. This paper concludes with Section 5.

2 Preliminaries

The goal of this study is to reduce the total number of messages spent for multiple queries. In order to share partial results, queries with the same aggregate operator and time duration are considered. Similar to prior works in [6], query Q_i is able to represent as a query tree, denoted as T_i. The leaf nodes of a query tree are data sources that will report sensing data and intermediate nodes of the query tree are used to aggregate sensing data from their child nodes. Hereafter, to facilitate the presentation of our paper, query Q_i is referred to as a query tree T_i. The number of messages spent for a query Q_i, expressed by $N(T_i)$, is the number of tree edges in T_i. For query tree T_i, $D_i(S_j)$ represents the partial result generated at sensor S_j and $N_i(S_j)$ is the number of messages spent for partial result $D_i(S_j)$, which is the number of tree edges of the subtree rooted at sensor S_j.

To facilitate the presentation of this paper, the backbone set (respectively, the non-backbone set) is expressed by B (respectively, NB). Clearly, by sharing partial results from backbones, non-backbones are able to reduce a considerable amount of messages. Denote the number of messages reduced for non-backbone T_j as $R(T_j, B)$. Thus, the total number of messages involved for a set of query trees can be formulated as follows:

$$\sum_{T_i \in B} N(T_i) + \sum_{T_j \in NB} (N(T_j) - R(T_j, B))$$

From the above formula, we could verify that minimizing the total number of messages is achieved by maximizing the number of messages reduced for queries in the non-backbone set. Intuitively, this problem is able to model as a Max-Cut problem. Explicitly, each query tree is viewed as a vertex and an edge represents the number of reduced messages achieved by sharing partial results.

3 Algorithm SB: Selecting Backbones

3.1 Determining Edges and Weights among Query Trees

When two query trees have some overlaps in their data sources, an edge will be added in the graph to represent the partial result sharing relationship. Specifically, suppose that the partial result on S_m of T_i is the same as the one on S_n of T_j. In other words, $D_i(S_m)$ is the same as $D_j(S_n)$ due to the same data sources. For query trees T_i, T_j, $w_{i,j}(S_m, S_n)$ denotes the number of messages reduced when sensor S_n in T_j obtains the partial result of sensor S_m in T_i. To formulate the value of $w_{i,j}(S_m, S_n)$, we should consider that S_n should access the partial result $D_i(S_m)$ from S_m. Therefore, an extra transmission cost is required and this extra transmission cost is therefore estimated as the minimal hop count between S_m and S_n, denoted as $d_S(S_m, S_n)$. Consequently, the value of $w_{i,j}(S_m, S_n)$ is formulated as $N_j(S_n) - d_S(S_m, S_n)$.

To facilitate the presentation of all possible ways for sharing partial results from T_i and T_j, $W_{i,j}$ is used to represent the set of weights for various partial result sharing scenarios between T_i and T_j. As mentioned above, the weight of each possible sharing scenario is in fact in the form of $w_{i,j}(S_m, S_n)$, meaning that T_i shares the partial result in S_m to sensor S_n in T_j.

3.2 Design of Algorithm SB

The objective of algorithm SB is to maximize $\sum_{T_j \in NB} R(T_j, B)$. Thus, algorithm SB is greedy in nature and selects the backbone with the maximal number of messages reduced for those non-backbones each time until no any message could be reduced when additional backbone is selected.

In essence, the value of $R(T_j, B)$ is related to how the sensor nodes of T_j access partial results from backbones. Since there are many possible scenarios for T_j to get partial results from backbones, we should avoid redundant message transmissions when formulating the value of $R(T_j, B)$. Thus, we have the following property.

Property 1: *If non-backbone T_j gets a partial result for a node S_x in T_j, it is unnecessary to further access partial results for the ancestors or descendants of S_x in T_j.*

Assume that nodes S_y and S_z are the child nodes of node S_x and both nodes S_y and S_z access partial results from backbones. Obviously, since the partial results of S_y and S_z is used to aggregate the result on S_x, node S_x should not access the partial result from backbones. For the same reason, it is also unnecessary to get partial results for the descendants of S_y or S_z.

In light of Property 1, we have developed a procedure to determine how many messages could be reduced through the partial result sharing. The algorithmic form of the proposed procedure is given below:

Procedure $R(T_j, B)$:

1. set $Y = \cup_{i \in B} W_{i,j}.To$, to determine the union set of sensors from the auxiliary table ;

2. Generate the power set of Y, denoted as $P(Y)$, is the set of all subsets of Y;

3. $\forall\ X \in P(Y)$, if there exists any ancestor or descendant relationship in X, prune X from $P(Y)$;

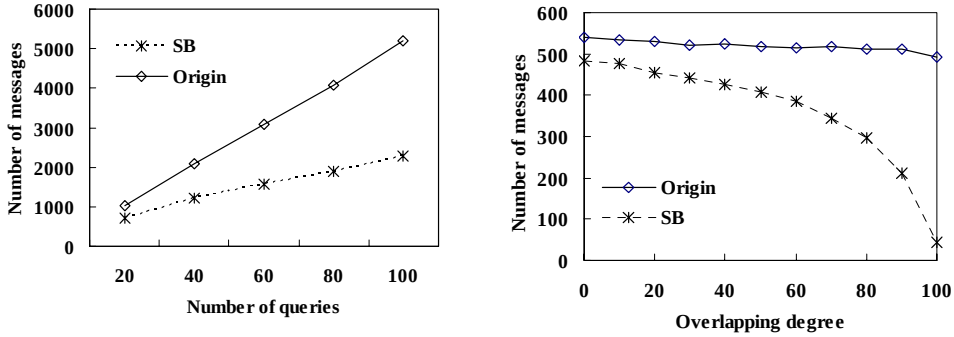
4. return $\mathbf{max}_{\forall X \in P(Y)}(\sum_{S_n \in X \text{ and } i \in B} w_{i,j}(S_m, S_n))$

In the beginning, we will determine the set of Y from the auxiliary table. As described above, the auxiliary table will contain all the detailed information related to the partial result sharing. Thus, given the backbone set, we could easily decide the set of Y. In fact, Y contains all the sensors in T_j that could access the partial results from backbones. In order to enumerate all the possible scenarios, we should generate the power set of Y, denoted as $P(Y)$. According to Property 1, we should avoid redundant message cost and thus, for each set in $P(Y)$, we should check whether there is any ancestor and descendant relationship or not. Note that one could refer to query tree T_j to verify any ancestor and descendant relationship. As such, the set of $P(Y)$ has all the possible scenarios of partial result sharing for T_j. Consequently, the number of messages reduced for T_j is able to be the maximal value among these possible scenarios.

To evaluate the benefits of selecting query tree T_i as a backbone, we have the following definition.

Definition 1: The *backbone gain* achieved by selecting T_i as a backbone, denoted by $\delta(T_i)$, can be formulated as $\delta(T_i) = \sum_{T_j \in (NB - T_i)} R(T_j, B \cup T_i) - \sum_{T_j \in NB} R(T_j, B)$.

In light of Definition 1, we propose a heuristic algorithm SB that iteratively select backbones according to *backbone gains* of query trees. Initially, the backbone set is empty and the non-backbone set is the set of query trees given. For each query tree in the non-backbone set, we will calculate the corresponding backbone gain. Then, the query tree with the maximal backbone gain is included in the backbone set. Once one query tree is selected as a backbone, we should update backbone gains for query trees in the non-backbone set. Similarly, according to the backbone gains of query trees in the non-backbone set, we will select the one with the maximal backbone gain as a backbone. Algorithm SB selects the query trees in the non-backbone set iteratively until no additional query tree is selected in the backbone set. When query trees in the non-backbone set



(a) (b)

Fig. 1. Performance comparison of Origin and SB (a) with number of queries varid. (b) with overlapping degree varied

have their corresponding backbone gains smaller than zero, no query tree will be selected in the backbone set since no more benefit will be earned. As such, a set of query trees is divided into two sets: the backbone set and the non-backbone set, which is akin to Max-Cut problem with the objective of maximizing the cut, meaning that the number of messages reduced is maximized.

4 Performance Evaluation

4.1 Simulation Model

A wireless sensor network is simulated, where there are 500 sensors randomly deployed in a 500*500 m^2 region. The sink is at the left-top corner of the region. The transmission range of sensors is set to 50 m. Users submit queries to the sink and each query utilizes scheme TAG [2] to form a query tree, where the root node is the sink. Query range is referred to those sensors whose sensing data are the data sources of one query tree. A query region set of a query tree is referred to the set of nodes in the query trees except the root node (i.e., sink). Assume that two query trees with their query region sets as R_1 and R_2. Then, we define the overlapping degrees of these two query trees as $\frac{R_1 \cap R_2}{R_1 \cup R_2}$. Note that with higher value of overlapping degree, query trees have more sensors in their overlap area, which means that more partial results are sharable among query trees. For the comparison purpose, scheme *Origin* is referred to the scenario that queries are performed as usual without any partial result sharing.

4.2 Experimental Results

First, we investigate the impact of sharing partial query results, where the overlapping degree is set to 50% and we set the query range to 100 x 100 m^2. As

can be seen in Fig. 1(a), the numbers of messages of scheme Origin, algorithm SB increase as the number of queries increases. Note that through the partial result sharing, algorithm SB has smaller numbers of messages involved. Note that when query trees have more overlapping area of query regions, these query trees are likely to have more opportunities to share partial results. Now, we examine the impact of overlapping degree, where the number of queries is set to 10 and the query range of each query is set to 100 x 100 m^2. The performance of Origin and SB with the overlapping degree varied is shown in Fig. 1(b). The number of messages is reduced in SB as the overlapping degree increases. This phenomenon agrees with our above statement that with a larger value of the overlapping degree, query trees have more changes to share partial results. As a result, the performance of SB is better than that of Origin.

5 Conclusion

In this paper, we exploited the feature of sharing partial results to reduce the total number of messages. Specifically, given a set of queries, we derived a graph, where each vertex represents one query and the corresponding weight edge denotes the number of messages reduced by sharing the partial results. According to the graph derived, we developed heuristic algorithm SB to derive a cut in which both backbones and non-backbones are determined. Performance of algorithm SB was comparatively analyzed and experimental results show that by sharing the partial results, algorithm SB is able to significantly reduce the total number of messages.

References

1. K. C. K. Lee, W.-C. Lee, B. Zheng, and J. Winter. Processing multiple aggregation queries in geo-sensor networks. In *Proceeding of the 11th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 20–34, 2006.
2. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.
3. S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on Data Base Systems (TODS)*, 30(1):122–173, 2005.
4. A. Sharaf, J. Beaver, A. Labrinidis, and K. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks. *The VLDB Journal*, 13(4):384–403, 2004.
5. N. Trigoni, Y. Yao, A. J. Demers, J. Gehrke, and R. Rajaraman. Multi-query optimization for sensor networks. In *Proceeding of the first IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 307–321, 2005.
6. Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Record*, 31(3):9–18, 2002.