國立交通大學

資訊科學與工程研究所

碩 士 論 文

在感測網路中針對多個網路中彙整的查詢
做最佳化來節省電量

Optimizing Multiple In-network Aggregate Queries in
Wireless Sensor Networks for Power Saving

研 究 生：楊慧友

指導教授：彭文志　教授

中 華 民 國　九 十 五　年 六 月

Chapter 2

Preliminaries

The goal of this study is to reduce the total number of messages spent for multiple queries. In order to share query results, queries with the same aggregate operator and time duration are considered. Similar to prior works in [16], 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 sensed data (referred to as data streams) and intermediate nodes of the query tree are used to aggregate the input data streams 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, $D_i(j)$ represents the partial result generated at sensor S_j and $N_i(j)$ is the number of messages spent for partial result $D_i(j)$, which is the number of tree edges of the subtree rooted at sensor S_j.

Description	Symbol
The query tree of Q_i	T_i
The number of messages involved for aggregation tree T_i	$N(T_i)$
The partial query result on sensor j of T_i	$D_i(j)$
The number of messages involved for $D_i(j)$	$N_i(j)$
The set of backbones	B
The set of non-backbones	NB
The number of messages reduced for aggregation tree T_i by accessing partial results	$R(T_i, B)$

Table 2.1: Description of symbols

Given a set of query trees, we shall select some query trees as backbones to share partial query results. Those query trees not selected as backbones are referred to as non-backbones. Query trees selected as backbones should be propagated the whole query trees in sensor networks, whereas those non-backbones could obtain some partial results from backbones. 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 can be formulated as follows:

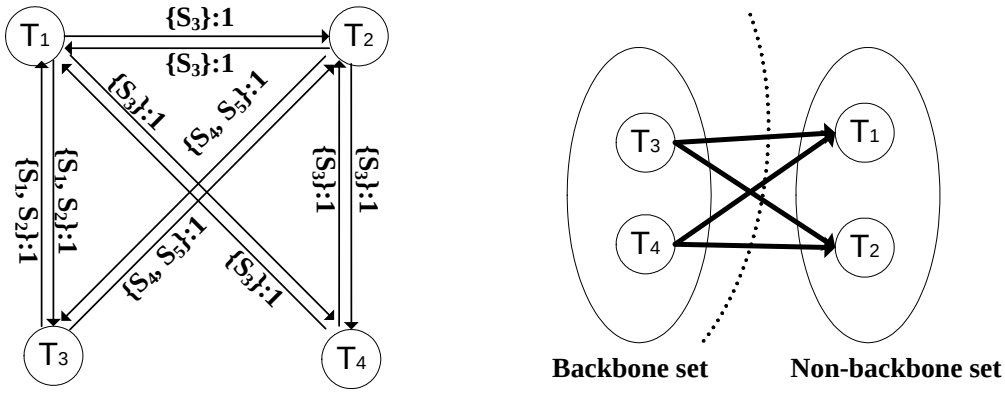
$$\sum_{T_i \in B} N(T_i) + \sum_{T_j \in NB} (N(T_j) - R(T_j, B))$$

Problem: Given a set of query trees, we should determine which query trees and how many query tress should be included in the backbone set with the purpose of minimizing the total number of messages. When a larger number of query trees in the backbone set, the number of messages reduced for each query tree in the non-backbone set will increase due to that query trees in non-backbone have more opportunities to get partial results from the backbone set. Clearly, though reducing the number of messages involved in queries tree belonging to the non-backbone set, the number of messages for query trees in the backbone set will increase. Thus, it has been recognized as an important issue to strike a compromise between the number of backbones and that of non-backbones.

Problem transformation: As mentioned before, given a set of query trees, we shall divide these query trees into two set: the backbone set and the non-backbone set so as to minimize the total number of messages. According to the formulation of the total number of messages, we have the following derivations.

$$
\begin{aligned}
& \sum_{T_i \in B} N(T_i) + \sum_{T_j \in NB} (N(T_j) - R(T_j, B)) \\
= \; & (\sum_{T_i \in B} N(T_i) + \sum_{T_j \in NB} N(T_j)) - \sum_{T_j \in NB} R(T_j, B) \\
= \; & \sum_{T_i \in (B \cup NB)} N(T_i) - \sum_{T_j \in NB} R(T_j, B)
\end{aligned}
$$

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. The input of a max-cut problem is a graph $G(V, E)$, where V is the set of vertices and E is the set of edges with their corresponding weights. In the max-cut problem, one would divide vertices into two sets with the purpose of maximizing the summation of edge weights between these two sets. In other words, a cut is defined as a summation of edge weights, where the two vertices of each edge belong to two sets (e.g., S and A), $S \cup A = V$ and $S \cap A = \phi$. The goal of the max-cut problem is to generate two proper sets so as to obtain a cut of G with the maximal sum of edge weights. Consequently, the problem of selecting backbones is able to transform to a max-cut like problem. Explicitly, each query tree is viewed as a vertex and an edge represents the number of reduced messages achieved by sharing partial query results. Consider an example shown in Figure 2.1, where vertices represent the four example queries in Figure 1.2 and edge $(T_i -> T_j)$ denotes the number of message reduced through the partial result sharing of T_i to T_j. For example, edge $(T_1 -> T_2)$ with its weight $\{S_3\} : 1$ indicates that by sharing the partial result of S_3 of T_1, the number of messages reduced for T_2 is 1. The formal graph definition will be described later. Note that Figure 2.1b depicts the cut among four queries, which refers to the case 5 in Table 1.1. In this case, T_3 and T_4 are backbones, whereas T_1 and T_2 are non-backbones. The sum of weights on these edges is 4, showing that the total number of messages reduced is 4 through the partial results of T_3 and T_4.



(a) Graph representation **(b) Max-Cut among query trees**

Figure 2.1: Transforming the problem of selecting backbones into Max-Cut problem

As such, the problem of selecting query trees in the backbone set can be transformed into the Max-Cut problem. Before developing an efficient algorithm for the Max-Cut problem, given a set of query trees, we should generate a graph $G = (V, E)$, where each vertex denotes one query tree and each edge represents the reduced of messages between two vertices. However, since these two query trees may have many same partial results, we should judiciously determine edges and the corresponding weights. Furthermore, given a graph G derived, we shall develop an efficient algorithm to derive two proper sets (i.e., backbone and non-backbone sets) with the maximal cut.