TAMPA: Tabu Search-based Multiple Queries Optimization for Wireless Sensor Networks

MingJian Tang, Jinli Cao and Naveen K Chilamkurti
Department of Computer Science and Computer Engineering
La Trobe University
Melbourne, Australia-3086
mj2tang@students.latrobe.edu.au, {j.cao,n.chilamkurti}@latrobe.edu.au

Abstract

In this paper, we address one of the wireless sensor network query processing issues posed due to the lack of support for multiple sensor network queries. The objective of the paper is to provide efficient and effective support to multiple queries so that the set of queries are pre-processed before disseminating them into the sensor network. It is very important that only necessary works will be assigned to the sensor network by virtue of strict energy constraint. The problem is modeled by Minimum Set Cover, which is one of the NP-complete problems. We propose an optimization scheme called TAMPA – a TAbu search-based Multiple queries oPtimizAtion to find an optimal merge order. The final set of queries to be sent into the network then can be derived from that merge order. We evaluate the proposed algorithm by conducting extensive simulation studies. The results show that energy can be significantly saved while the overall workload still satisfies the user requirements.

Keywords: Wireless Sensor Networks, Multi-query Optimization, Tabu Search

1. Introduction

With the technology advances, the wireless sensor network applications are growing. Applications such as environment monitoring, military surveillance, building structure monitoring, vehicle detection, information gathering are few of them [1]. The actual network consists of large number of nodes know as sensors. Each of them is equipped with low-cost radio transceiver, a sensor, and some simplified computer hardware allowing some local processing and storage capabilities. As sensor hardware is becoming cheap, it can be expected that with more sensor nodes scattered around, the WSN (Wireless Sensor Network) can be more powerful in terms of providing fine-grained environmental information. Inevitably one deployed sensor network will be able to provide necessary data to multiple applications simultaneously. However, this can result in dissemination of redundant tasks which may strain the energy-constrains in a WSN. Additionally, the unique characteristics of WSN such as densely deployed, multi-hop, and broadcast communication can make the situation worse. Therefore, it is critical to provide a pre-processing scheme at the base-station for efficiently merging similar queries.

In this paper, the problem is formulated as Minimum Set Cover problem (Section 2), which is NP-complete [2]. The problem implies a combinatorial optimization problem, which is finding a good query merging order. We propose an optimization scheme named TAMPA to solve this problem. The key concept of TAMPA is a tabu search based optimization algorithm. This algorithm iteratively seeks good merge orders until it converges to an optimal point. In the end, a processed query set will be produced, in which only necessary workloads are kept.

The rest of the paper is organized as follows: Section 2 lists some background works in multi-query optimization, and analyses are also given. Section 3 discusses the architecture of the WSN and the multiple-application WSN environment along with problem formulation and modeling. In Section 4, a tabu search-based optimization algorithm is proposed for pre-processing the set of queries. Section 5 presents the simulation evaluation and study on the algorithm. Section 6 concludes the paper.

2. Background Work

Our problem is essentially related to multiple-query optimization problems in traditional database system. Many researches were to find the common expressions or sub-expressions among a group of concurrent queries, and one such work is done in [3].

With the recent growth in emerging technologies, the MQO (Multiple Queries Optimization) problems are also found in new domains, such as in multicast environment [4], mobile database domain [5], data stream systems [6], and eventually in WSN environment [7,8, 9].

In [8], the authors proposed a result sharing and result encoding techniques for processing multiple queries. Their research focuses on reducing the in-network communication. A Fjords architecture [7] is another example of managing multiple queries, and the focus is also on in-network information processing and control. In [9], authors propose a two-tier multiple-query optimization scheme for processing multiple queries in WSN. The base-station tier and in-network tier are the two tiers of the scheme. Though their base-station works well for multiple

1-4244-1312-5/07/$25.00 © 2007 IEEE

sequential queries processing, it might not be a best solution for multiple concurrent queries optimization.

3. Problem Formulation and Modeling

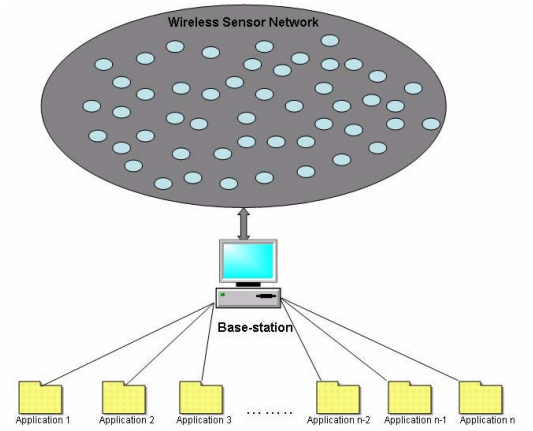

Figure. 1. Multiple-application wireless sensor network scenario

In this paper, we consider a flat WSN with a large number of sensor nodes, where the sensor nodes are uniformly deployed in the sensing field and they are self-organized into a connected network. From the architecture point of view, we consider a typical WSN is two-tiered, which are in-network tier and out-network tier. The in-network tier consists of the deployed sensor nodes. The base-station forms the out-network tier. The in-network resources are normally considered to be highly constrained in terms of energy and computation capabilities. We assume high network resources for out-network compared to the in-network ones.

Figure 1 depicts a typical relationship between a sensor network and its applications. There will be multiple applications interacting with a centralized base-station through high-speed network facilities. Queries will be generated by those applications according to their needs. These queries then will be sent to the base-station, where they will be queued and pre-processed before they can be disseminated into the sensor network via certain communication means.

3.1 Problem Formulation

The multi-query optimization problem is formally formulated as follows:

- Let $U = \{U_i \mid 1 \le i \le n,\ i \in N\}$ be a set of queries.
- Let *Cost:* $U \rightarrow R$ be a cost function, which assigns a real number to each query to reflect its incurred energy cost.
- Let W be a workload cover function.
- Let $Cost(U) = \sum_{i=1}^{n} Cost(Ui)$ be the overall cost of set U.
- Let $W(U) = \sum_{i=1}^{n} W(Ui)$ be the overall workload cover of set U.
- Let S be the collection of all possible query sets derived from U that cover $W(U)$.

Our goal is to minimize $Cost(U)$ by rewriting query set U into a new set C, where $C = \{C_j \mid 1 \le j \le m \le n,\ j \in N\}$. The new set C should also keep the following constraints:

- $W(U) = W(C)$
- $W(C_i) \cap W(C_j) = \varnothing \quad 1 \le i \le j \le m \ \&\ C_i, C_j \in C$
- $Cost(C) \le Cost(U)$, *if* $\exists C'$ *such that* $Cost(C') \le Cost(U) \Rightarrow \forall C$ *that* $Cost(C) < Cost(C')$
- $C \subseteq S$

To find an optimal transferring strategy (with the minimum cost) for U in general is equivalent to the Minimum Set Cover (MSC) problem. A formulation of MSC is described in [10] as follows:

- Instance: Given a universe U and a collection S of subsets of U, a set cover is a sub-collection $C \subseteq S$ of sets whose union is U. In the set cover optimization problem, the input is a pair (U, S).
- Find: A set cover $C \subseteq S$ which contains the fewest sets to cover U.

MSC is proved to be NP-complete. This means, we can not find an optimal rewriting plan for the query set within a polynomial time complexity.

3.2 Cost Model

A typical WSN query consists of the following key elements:

<Attribute[i]> <Range> <Time Window> <Interval>

where *Attribute*[*i*] defines a number of interests expressed by an application. The Range attribute is used for defining the size of the queried area. *Time Window* is in minutes, which rules how long a query will be valid for. The last element is *Interval*, which sets the temporal granularity of the query, which is expressed in minutes. An example of a query is shown below:

<N.Light, N.Sound><(10, 100), (20, 30)> <200><20>

The above example shows the application is interested in investigating light and sound within the range of *X: 10 → 100m* and *Y: 20 → 30m*, and the query will be valid for *200* minutes and the reporting intensity is *20* minutes per report.

According to the above query structure, a cost model is provided as follows:

$$Cost = \frac{\sqrt{(X_u - X_l)^2 + (Y_u - Y_l)^2} * \mid Attributes \mid * TimeWindow}{Interval} \quad (1)$$

The square root in formula *(1)* is the length of the diagonal of coverage size of the query. The longer the diagonal

length, the larger the coverage is. |*Attributes*| is the number of monitored attributes involved in a query. We use a uniform cost value for modeling all types of attribute collections. They can easily be modeled differently by assigning a ratio to each attribute, and then multiply with a pre-defined base value. We divide the *Time Window* by the Interval, which is used to quantify number of information units that need to be transferred back to the base-station. These factors are multiplied together to form the cost model for a query in the final stage.

4. TAMPA

Tabu search is a widely used search algorithm, which is invented by Glover (1986) [11]. It is a local search algorithm in nature, but it differs from others by not only accepting cost-decrease transitions but also cost-increase transaction. Hence, it has the ability to escape from local optima. The ability is due to the utilization of tabu.

The detailed implementation is described as follows:

TAMPA

```
INPUT: U = {U1... Ui}, itr = 0, imax = 20, Tenure = 5
OUTPUT: C
START:
1: WHILE U is not empty DO
2:   Select a random query Uran from U
3:   Find all the adjacent merging solutions S for Uran
4:   IF |S| > 1 THEN
5:     Select a random solution s from S
6:     S = S - s
7:     Compute G (s)
8:     Set S* = s and Set G* = G(s)
9:     Initialize tabu list Tb
10:    WHILE itr < imax && |S| != 0 DO
11:      Compute G(S)
12:      Find the best neighbor solution s'
13:      IF G(s') ≤ G(s*) THEN
14:        Set S* to s' and G* to G(s)
15:        Add s' to Tb
16:        itr++
17:      ELSE
18:        IF s' is in Tb && Aspiration condition fails THEN
19:          Exclude solution s' from S
20:        ELSE
21:          Set current solution s to s'
22:          itr++
23:    END WHILE
24:  ELSE
25:    C = C + S
26:    U = U - S
27: END WHILE
END
```

The algorithm starts by picking a random query U_{ran} from U, and then all the adjacent merging orders for U_{ran} will be derived based on certain merging rules and constraints. If there exists one or no merging order for U_{ran} (≤ 1), then the Tabu process is skipped so that processing time can be saved and that merging order or just U_{ran} will be included in the final solution. If there are multiple merging orders (>1), then those merging orders will go through the Tabu process inside which a good merging order will be chosen. When the algorithm converges, an optimal solution will be generated with a new set of queries C that minimizes the workload. We map the set of adjacent merging orders to the set of neighbors, and gain function $G()$ to the objective function, which is derived from the cost difference between two queries that before merge and after merge. The tenure of tabu is set to be *5*, and the aspiration condition is set to be the ratio of the gain function exceeds *10%*.

5. Performance Evaluation

In this section, we examine the performance of the proposed TAMPA using several performance metrics. We implemented a program using C# for this evaluation. The program mimics the procedure of collecting queries from different sources by generating a set of queries. Normal Distribution is used as the probability distribution model for query set generation. Then these generated queries will be processed by the TAMPA, and an optimized set will be produced. Meanwhile, evaluation statistics are collected during the query processing procedure, and it will be examined using different metrics.

5.1 Experimental Setup

We have implemented the application program in C#. The hardware available for the experiments was a desktop computer with Pentium IV (*3.2GHz* CPU with *2* MB on-chip cache) and *2* GB ram. This is a typical computer nowadays with moderate hardware configuration that most likely a base-station computer will be equipped with. All simulation tests were run on this machine. We used Microsoft's Visual Studio 2005 on Windows XP as the application run-time environment for developing the application.

Sample queries were synthetically generated for providing inputs to examine TAMPA under a range of application scenarios. We ran each scenario repeatedly for *20* times, and the final evaluation result will be derived by averaging these results.

5.2 Reduction Ratio

Reduction ratio further consists of two metrics, namely cost reduction and query number reduction. The percentage of the cost reduction is expressed as the ratio of the cost before query set optimization and after query set optimization. The percentage of query number reduction is expressed as the ratio of the number of queries before the optimization and the number of queries after the optimization. We also observe the impact of varying number of queries on them.

Figure 2 shows two reduction ratios, and it proves TAMPA is very effectively in terms of eliminating query

redundancies. The averaged cost reduction is about *27%*, whereas the averaged query number reduction is about *55%*.

5.3 Execution Time

We also used execution time to evaluate the temporal efficiency of the TAMPA. We evaluate the scheme by providing varying number of queries, and execution times, that are derived from different experimental settings. Execution time is an important evaluation factor since

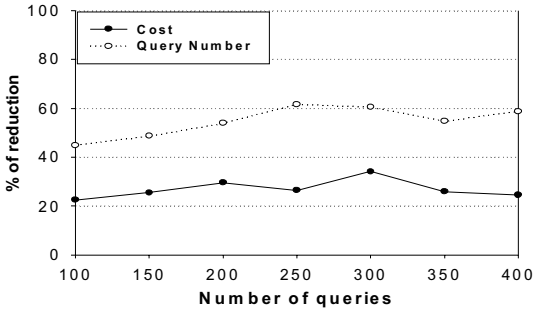

Figure. 2. Reduction ratio (Cost and Query Number)

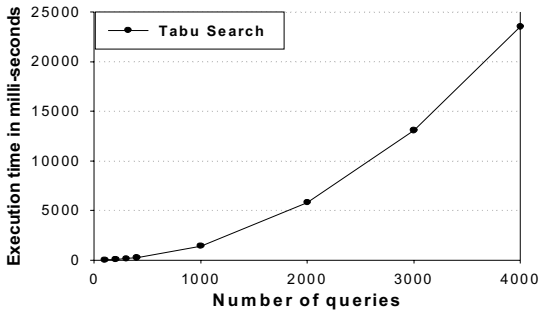

Figure. 3. Execution time (Tabu)

typically WSN environment is very dynamic in nature. So we assume that user acceptable delay will be less than *20* seconds, and we will stop experimenting after this limit is reached by the algorithm. In the end, the result can be used to demonstrate the processing capacity of the scheme in terms of number of queries.

Figure 3 shows the results of TAMPA execution times that are observed through a number of experiments. It can be seen that the scheme is extremely fast while processing up to *1000* queries at a time. With the increasing number of queries to be processed, the algorithm execution time does grow asymptotically, and finally reaches its limit which is close to *4000* queries.

6. Conclusion

In this paper, we proposed an optimization scheme, TAMPA, for pre-processing a set of queries simultaneously. The scheme can be run on the central base-station. Our extensive simulation showed that by implementing TAMPA we proved that cost and query numbers are reduced dramatically. On the other hand, the efficiency of TAMPA is proven to be very promising. The conclusion is that the redundancies among different similar queries can be efficiently and effectively eliminated. The results show that energy can be significantly saved while the overall workload still satisfies the user requirements.

7. References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on sensor networks," IEEE Communications Magazine, vol. 40(8), pp. 102-114, 2002.

[2] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of Np-Completeness: W H Freeman & Co, 1990.

[3] S. Chakravarthy, "Divide and conquer: A basis for augmenting a conventional query optimizer with multiple query-processing capabilities," presented at Proceedings of the Seventh International Conference on Data Engineering, Kobe, Japan, 1991.

[4] A. Crespo, O. Buyukkokten, and H. Garcia-Molina, "Query merging: improving query subscription processing in a multicast environment," IEEE Transactions on Knowledge and Data Engineering, vol. 15, pp. 174- 191, 2003.

[5] R. Malladi and K. C. Davis, "Applying multiple query optimization in mobile databases," presented at Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003.

[6] S. Seshadri, V. Kumar, and B. F. Cooper, "Optimizing Multiple Queries in Distributed Data Stream Systems," presented at Proceedings. 22nd International Conference on Data Engineering Workshops, 2006.

[7] S. Madden and M. Franklin, "Fjording the Stream: An Architecture for Queries over Streaming Sensor Data," presented at the 18th International Conference on Data Engineering (ICDE), 2002.

[8] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman, "Multi-query Optimization for Sensor Networks," in International Conference on Distributed Computing in Sensor Systems (DCOSS), 2005.

[9] S. l. Xiang, H. B. Lim, and K. L. Tan, "Multiple Query Optimization for Wireless Sensor Networks (Poster)," in Proc. of the 23rd International Conference on Data Engineering (ICDE'07 Poster). Istanbul, Turkey, 2007

[10] Wikipedia, "Set cover problem," http://en.wikipedia.org/wiki/Set_cover_problem

[11] E. Aarts and J. K. Lensta, Local Search in Combinatorial Optimization: New York: Wiley, 1997