Toward Efficient Distributed Algorithms for In-Network Binary Operator Tree Placement in Wireless Sensor Networks

Zongqing Lu, *Student Member, IEEE,* Yonggang Wen, *Member, IEEE,* Rui Fan, *Member, IEEE,* Su-Lim Tan, *Member, IEEE,* and Jit Biswas, *Member, IEEE*

***Abstract*—In-network processing is touted as a key technology to eliminate data redundancy and minimize data transmission, which are crucial to saving energy in wireless sensor networks (WSNs). Specifically, operators participating in in-network processing are mapped to nodes in a sensor network. They receive data from downstream operators, process them and route the output to either the upstream operator or the sink node. The objective of *operator tree placement* is to minimize the total energy consumed in performing in-network processing. Two types of placement algorithms, centralized and distributed, have been proposed. A problem with the centralized algorithm is that it does not scale to large WSN's, because each sensor node is required to know the complete topology of the network. A problem with the distributed algorithm is their high message complexity. In this paper, we propose a heuristic algorithm to place a tree-structured operator graph, and present a distributed implementation to optimize in-network processing cost and reduce the communication overhead. We prove a tight upper bound on the minimum in-network processing cost, and show that the heuristic algorithm has better performance than a canonical greedy algorithm. Simulation-based evaluations demonstrate the superior performance of our heuristic algorithm. We also give an improved distributed implementation of our algorithm that has a message overhead of $O(M)$ per node, which is much less than the $O(\sqrt{NM}\log_2 M)$ and $O(\sqrt{NM})$ complexities for two previously proposed algorithms, Sync and MCFA, respectively. Here, N is the number of network nodes and M is the size of the operator tree.**

***Index Terms*—Sensor networks, operator tree placement, heuristic algorithm.**

I. Introduction

Wireless sensor networks (WSNs), owing to recent advances in wireless communication and microsystem technologies, have enabled a school of novel applications [1], [2], such as disaster management, factory automation, environment monitoring, offshore exploration, to name a few. In these applications, a large number of sensor nodes, each of which is capable of sensing, computing and communicating, are deployed to operate autonomously or collaboratively in remote environments. However, each sensor node has only a limited amount of system resource at its disposal. In particular, the limited energy resource for the sensor nodes is a major factor that determines the lifetime of WSNs. Therefore, energy optimization stands out as an imperative step in the design, operation and application of WSNs [3], [4].

One of the most prominent technologies for conserving energy in WSNs is in-network processing [5], [6]. Specifically, in-network processing leverages the computational capacity of sensor nodes to perform aggregation (or fusion) operations en route, eliminating data redundancy and minimizing data transmission to save energy for sensor nodes. Normally, the operators participating in in-network processing are placed in existing sensor nodes, which receive the data from downstream operators, process them and route the output along a particular path to either upstream operators or the sink node. The energy consumption in this scheme depends on the data transmission from one operator to its upstream counterpart, which in turn depends on the placement of operators over the sensor topology. It has been shown [7] that the placement of operators can greatly affect the transmission energy cost of in-network processing. In this research, we focus on developing algorithms for in-network operator placement in wireless sensor networks, with the objective of minimizing total in-networking processing cost and data transmission energy.

Previously proposed algorithms for optimal placement of in-network operator tree can be classified into two categories, centralized and distributed. In centralized algorithms, each node is required to know the topology of the entire network, limiting its scalability in large WSNs. Existing distributed algorithms can eliminate this constraint. However, they often incur a substantial message overhead: ($O(\sqrt{NM}\log_2 M)$ for Sync [8] and $O(\sqrt{NM})$ for MCFA [9], where N is the number of network nodes and M is the size of the operator tree). Other distributed algorithms, e.g. the greedy algorithm of [10], has a high cost to search for an optimal operator placement. These extra communication costs increase the energy cost of network operations, possibly negating the energy savings of in-network processing.

In this paper, we propose a distributed heuristic algorithm to place a tree-structured binary operator graph over the sensor topology, with an objective to reduce the message overhead and save energy. Inspired by the three-factory problem [11], we develop a heuristic that places each operator individually to minimize the total cost of routing data from its downstream

Manuscript received April 1, 2012; revised December 26, 2012

Z. Lu, Y.G. Wen and R. Fan are with the School of Computer Engineering, Nanyang Technological University, 639798 Singapore (e-mail: {luzo0002, ygwen, fanrui}@ntu.edu.sg).

S.L. Tan is with Singapore Institute of Technology, 179104 Singapore (e-mail: forest.tan@singaporetech.edu.sg).

J. Biswas is with Institute for Infocomm Research, 138632 Singapore (e-mail: biswas@i2r.a-star.edu.sg).

Digital Object Identifier 10.1109/JSAC.2013.1304xx.

operators to the sink node. We then develop an area-restricted flooding mechanism to solve the local minimization problem, which has a low message overhead. Distributed message passing protocols, based on the area-restricted flooding mechanism, are presented for numerical analysis. Our theoretical and numerical analyses show the performance advantage of our proposed algorithm in reducing communication and energy cost. Our technical contributions include the following:

- We propose a geometric heuristic, based on the three-factory problem, as a mathematical basis to place each operator individually.
- We establish a tight upper bound on the minimum energy cost for in-network operator tree placement and prove that our performance ratio is significantly better than that of the previously proposed greedy algorithms.
- We introduce a novel area-restricted flooding mechanism to reduce the message overheard of placing individual in-network operators.
- We develop a parameter-free distributed algorithm to place a tree of operators without requiring knowledge of the network topology. The message overhead of the algorithm is $O(M)$ per node.
- We provide both mathematical analyses and simulation-based evaluations that show our approach has better performance than the canonical greedy algorithm and has much smaller message overhead than the Sync and MCFA algorithms.

The rest of this paper is organized as follows. Section II reviews the existing work. Section III formulates the optimization problem for operator tree placement in a sensor network. We then characterize the optimal solution in Section IV. Section V presents our proposed heuristic algorithm and proves its approximation ratio. An improved distributed implementation of our heuristic algorithm is presented in Section VI. In Section VII, we give a simulation-based evaluation of our heuristic algorithm and compare it to other algorithms. We conclude the paper in Section VIII.

II. Related Work

The problem of optimal operator placement has been an important research subject. One school of prior works optimizes the throughput as in [12], [13], [14]. The other school of prior works optimizes the energy cost, including centralized algorithms [15], [16], [17], [18] that requires the global topology information, and distributed algorithms [19], [20], [21], [22], [8], [10].

Bonfils et. al. [19] proposed a decentralized algorithm for operator placement, which progressively refines the placement of operators by neighbor exploration and placement adaptation. The approach that an operator is gradually moved towards optimal placement is called in-network relaxation or operator migration. Other works in the same category include [20], [21]. As these algorithms of operator migration are based only on local information (information from neighbors), they suffer from oscillating change, which might force the placement of an operator to a different direction before reaching the optimal placement. They are also prone to local minima and they cannot guarantee the optimality of operator placement based on local information only.

In [22], 1-median point is considered as the optimal placement in network and a distributed search algorithm was proposed to find the optimal operator placement. However, this algorithm is designed to handle only one operator placement.

Abrams et. al. [10] proposed a greedy algorithm, which places each operator on the node with minimized input data transmission cost. Obviously, the greedy placement is not optimal and it can be much worse when the greedy placement is backward to the sink. Further, the distributed implementation of this greedy algorithm considered only the placement adaptation and the authors did not elaborate how to find the initial placement for each operator in a distributed manner.

In [8], Sync was proposed to achieve the optimal placement for a tree-structured operator graph with optimal in-network processing cost. However, Sync requires that the network should have full time synchronization, and all the nodes should know when other nodes finish information updating and when they finish broadcasting updated information. Furthermore, it incurs a huge message overhead for searching the optimal placement. To address this issue, Lu et. al. [9] proposed a minimum-cost forwarding based distributed algorithm (MCFA) to optimize the message overhead. Although MCFA has less message overhead than Sync, the message overhead is still very heavy.

III. System Model and Problem Formulation

In this section, we first present model assumptions for the in-network processing in WSNs, and then formulate the operator tree placement as a joint placement-routing optimization problem.

A. Model Assumptions

In this research, we assume that a WSN is deployed in a two-dimensional region, where each node has capabilities of sensing, communication and computing. For sensing, each node can observe a local variable and encapsulate the result as a data object. For communication, each node can communicate with other nodes that fall in with a radius (r_c) and the network is fully connected (no isolated nodes). For computing, each node can take all the data objects from downstream nodes and compute a summary of these objects into another date object, which will be forwarded to node in the upstream. Under these assumptions, we model the WSN as an undirected graph $G = (V, E)$, where V denotes the set of vertices representing sensor nodes and E denotes the set of edges representing communication links between nodes. The cardinality of the vertex set is N. For two nodes $p, q \in V$ within the communication radius, we denote the edge as $(p, q) \in E$. For any edge $(p, q) \in E$, we denote $C(p, q)$ as a chosen distance metric between the two incident nodes of p and q. Examples of the distance metric include hop counts, square of Euclidean distance between two nodes, and transmission energy, etc. Moreover, we assume the communication link is symmetric $C(p, q) = C(q, p)$.

Figure 1(a) illustrates a typical wireless sensor network, where each node can play three roles, including sensing node, relay node and sink node. The sensing node collects data object, the replay forwards information along a chosen routing

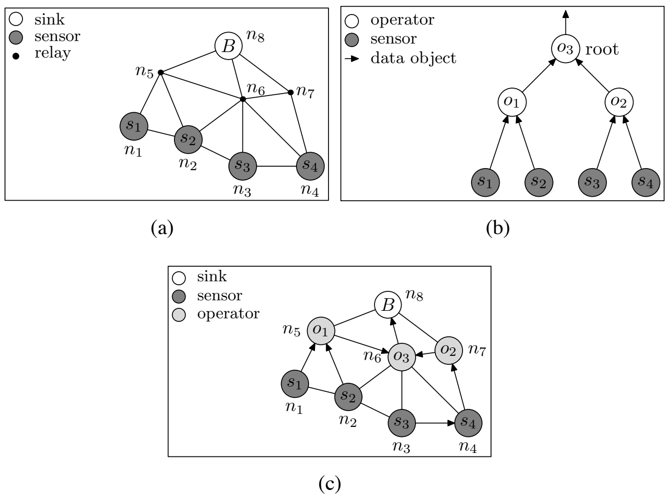


Fig. 1. Sensor network (a), operator tree (b) and placement (c). Noted that operator can be placed on any network node and (c) is an example of placement that might not be optimum.

path, and the sink node collects all the data objects. We assume that a set $X \in V$ of nodes are data sources of interests (sensors) and the sensed data needs to be gathered at sink node or base station $B \in V$ as shown in Fig. 1(a). Moreover, in order to reduce transmitted data size to save energy, in-network data aggregations or computations (defined as operators) are desired to process the sensed data along the routing paths. We assume that these sensors and operators formulate a routing tree, through which data objects are processed and routed toward the sink node. We call it an operator tree, which is a directed graph, denoted as $T = (X, O, A)$, where X denotes the set of sensors (data sources), O denotes the set of operators, and A denotes the set of directed edges. The cardinality of O is t and the size of operator tree (the number of sensors and operators) is M. We assume the operator tree is a complete *binary* tree as illustrated in Fig. 1(b). In the rest of paper, the complete binary operator tree is simplified as operator tree unless otherwise specified.

For each sensor $x \in X$, we denote d_x the size of data object generated by sensor x. For each operator $o \in O$, we denote Δ_o as the set of children of operator o, d_o the size of data object generated by operator o, r_o the data reduction ratio of operator o, which is defined as the ratio between the size of output data object and the sum size of input data objects. We assume $r_o \leq 1/2$, which is reasonable because output data of operator is much less than input data like query processing, stream processing, etc [23], [24]. Moreover, α_o is defined as the ratio of the larger input data object size to the smaller input data object size of operator o. Note that the final data object (the data object from root operator) will be transmitted to sink node, as shown in Fig. 1(b).

In this research, we focus on overlaying the operator tree atop of the WSN topology as shown in Fig. 1(c). Specifically, we identify a subgraph (i.e., a complete tree) in the WSN topology as the operator tree and route the data objects via the subgraph to the sink node. In this case, we define the *in-network processing cost* as the routing cost for the set of data objects forwarded to the sink node. Specifically, for a data object i of size d_i routed from node p to node q, its in-network processing cost is $d_i * C_q^p$, where $C_q^p = C(p,i) + C(i,\cdots) + C(\cdots,j) + C(j,q)$, $(p,i,\cdots,j,p)$ is routing path from node p to q. As a result, for a given operator tree T, the total in-network processing cost can be derived as

$$f(G,T) = \sum_{a \in A} d_a \times C_{a_t}^{a_h}, \tag{1}$$

where d_a is the size of the data object routed on arc $a \in A$, a_h and a_t denote the head and the tail node of arc a, respectively.

B. Problem Statement

In this research, we consider the problem of the operator tree placement. The operator tree placement problem consists of two steps. First, a subset of nodes in the WSN are chosen as the set of operators. Second, a set of paths are chosen to route the set of data objects from their sources to the sink node, via the set of chosen operators. Note that these two steps are not separated. The design objective for the operator tree placement is to minimize the total in-network processing cost.

Mathematically, the optimal operator tree placement problem can be stated as follows. Let P denotes the operator placement, such as $P(o,p) = 1$ if operator o is placed at node p, otherwise $P(o,p) = 0$. R denotes the routing scheme that routes data from the nodes generating the data to the node requiring the data. The problem is to identify an optimal scheme of (P^*, R^*) that minimizes the total in-network processing cost, denoted as the following:

$$(P^*, R^*) = \arg\min_{(P,R)} f(G,T;P,R), \tag{2}$$

where $f(G,T;P,R)$ denotes the total in-network processing cost under a chosen placement scheme of (P,R).

In this paper, we focus on developing efficient distributed algorithms to solve this joint placement-routing problem. In addition, we also aim to reduce the message overhead, defined as the number of messages exchanged in network for distributed operator placement.

IV. Dynamic Programming Approach for Optimal Solution

The optimal placement of an operator tree, mathematically, can be formed as a dynamic programming problem, which can be solved in a polynomial time. Under the dynamic programming framework, the iterative equation for the optimal in-network operator tree placement problem is given as follows,

$$cost_{p^*}^{o} = \begin{cases} \min_{\nu \in V} \sum_{o_c \in \Delta_o} (d_{o_c} C_{\nu}^{p^*_{o_c}} \\ + cost_{p^*_{o_c}}^{o_c}) & \text{if } o \neq \mathcal{R} \\ \min_{\nu \in V} (d_o C_{\nu}^{sink} + \sum_{o_c \in \Delta_o} \\ (d_{o_c} C_{\nu}^{p^*_{o_c}} + cost_{p^*_{o_c}}^{o_c})) & \text{if } o = \mathcal{R} \end{cases} \tag{3}$$

where $\mathcal{R}$ denotes the root operator, p^* denotes the optimal placement of o , $cost_{p^*}^{o}$ denotes the optimal cost of routing all data objects in subtree of operator o to node p^*, and $cost_{p^*_{o_c}}^{o_c}$ denotes the minimum cost of routing all the data to the subtree root node at operator o_c.

This dynamic programming problem can be solved via a *bottom-up* approach to calculate the minimum cost to route all

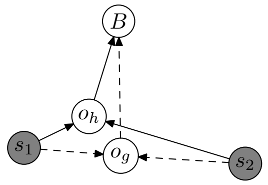


Fig. 2. Comparison between heuristic and greedy algorithm, o_h chosen by heuristic algorithm and o_g chosen by greedy algorithm, respectively.

the data to the root node from a subtree. Define $cost_p^o$ operator $o \in O$ and $p \in V$ to be the minimum possible communication cost of routing all data objects in subtree of o to node p (for o is root operator, $cost_p^o$ is the minimum in-network processing cost of operator tree). For $x \in X$ anchored at node q, define $cost_q^x = 0$; and $cost_p^x = \infty$ if $p \neq q$. The cost of $cost_p^o$ can be calculated recursively, using the following equation,

$$cost_p^o = \begin{cases} \sum_{o_c \in \Delta_o} \min_{\nu \in V}(d_{o_c} C_\nu^p + cost_\nu^{o_c}) & \text{if } o \neq \mathcal{R} \\ \sum_{o_c \in \Delta_o} \min_{\nu \in V}(d_{o_c} C_\nu^p + cost_\nu^{o_c}) & \\ + d_o C_{sink}^p & \text{if } o = \mathcal{R} \end{cases}$$

After computing $cost_p^o$ recursively, $\forall o \in O, p \in V$, map P^* with the set of pairs $o \in O$ and $p \in V$ is used in recursive unfolding of $cost_{p_\mathcal{R}}^\mathcal{R}$, where $\mathcal{R}$ is anchored at node $p_\mathcal{R}$. Then P^* is the optimal placement of operator tree.

This off-line iterative algorithm is centralized and its computational complexity can be calculated as $O(N^2 M)$. Although the algorithm can find an optimal placement in a polynomial time, it requires the precise knowledge of the shortest path between all pairs of nodes, and any change in network topology or communication link cost will incur re-execution of the searching algorithm. Moreover, although polynomial, the cost of running the algorithm may be prohibitively large for wireless sensor network of sizable dimension.

In fact, the dynamic programming algorithm for the optimal placement can be also solved in distributed way, such as the *Sync* algorithm proposed in [8] and the *MCFA* algorithm proposed in [9], respectively. Sync has a message overhead per node of $O(\sqrt{N} M \log_2 M)$, and MCFA has messages overhead per node of $O(\sqrt{N} M)$. Although Sync and MCFA would find the optimal placement in a distributed manner, their message overhead at each node is very high. Therefore, the cost of running these distributed algorithms would be also extremely large in WSNs.

V. Heuristic Placement Algorithm

In this section, we will present our proposed heuristic operator placement algorithm, investigate its approximate ratio to the optimal solution and compare the performance of our heuristic algorithm and a canonical greedy algorithm [10].

A. Heuristic Algorithm

Given an operator tree as shown in Fig.1(b), the operators will be assigned to network nodes hierarchically from bottom to top. In our research, we propose a heuristic algorithm that assigns each operator o to the node according to (4):

$$p = \arg\min_q \{d_o \times |(q, sink)| + \sum_{i \in \Delta_o} d_i \times |(i, q)|\}, \quad (4)$$

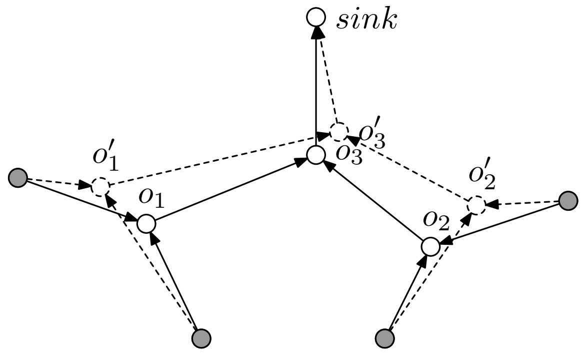


Fig. 3. Operator Placement of Heuristic and Optimal Algorithm

where $|(q, sink)|$ denotes the minimum cost between two nodes. The crucial part of (4) is $d_o \times |(q, sink)|$, which enables the heuristic algorithm to be equipped with the sense of direction. In this case, the heuristic algorithm will place each operator toward the sink as shown in Fig. 2. The key observation is that the heuristic placement is nearer to the optimal placement than the greedy placement, where the operator will be hosted by the node with minimized $\sum_{i \in \Delta_o} d_i \times |(i, q)|$. As the final data object of the operator tree will be transmitted to the sink, the difference between the optimum solution and the greedy algorithm is the orientation of each operator placement. Our heuristic solution gradually places the operators towards the sink, while the greedy algorithm constantly chooses the node with minimized data input communication cost. That is the reason why we add $d_o \times |(q, sink)|$ rather than just $\sum_{i \in \Delta_o} d_i \times |(i, q)|$ in (4), which actually benefits the latter operator placement and communication cost, and makes a further step of approximation to the optimal solution. $d_o \times |(q, sink)| + \sum_{i \in \Delta_o} d_i \times |(i, q)|$ is called *placement cost* for operator o and node q, and denoted by $cost_o'^q$.

As discussed in Section IV, the placement of operators is only determined after finding minimum in-network processing cost. As such, the placement of each operator is correlated and globally determined. Unlike the optimal solution, our heuristic algorithm separates each operator placement and makes it locally determined so as to reduce the computational complexity and message overhead of finding an efficient solution. The placement of operators in our heuristic algorithm depends only on the placement of children of the operator and the sink position. It follows that it is relatively easier to find the placement for each operator and to implement the heuristic algorithm in a distributed manner. When the operator tree has only one operator, the heuristic algorithm finds the same placement for operator as the optimal solution. When there are more than one operator, our heuristic algorithm may incur slightly more in-network processing cost than the optimum solution. The penalty will be investigated in next subsection.

B. Optimality Performance

An immediate task would be to characterize the performance of our proposed heuristic algorithm. In this section, we adopt an Euclidean cost metric to show that our heuristic algorithm is bounded and its approximate ratio is lower than that of the greedy algorithm.

As shown in Fig. 3, o_1, o_2 and o_3 are the operator placements of optimal solution, meanwhile o_1', o_2' and o_3' are the operator placements of heuristic algorithm. The difference between the optimal placement and the heuristic placement varies according to different data object size and data reduction ratio at each operator. In order to evaluate our heuristic algorithm, we give its approximate ratio to the optimal solution.

It is known that (4) will be the three factory problem[1] [25] when the operator has two children as shown in Fig. 2. Noted that the three-factory problem is applicable for any distance metric. Before proving the approximate ratio, first we introduce two lemmas about three factory problem, which help us conclude the further theorems.

Lemma 1. *When $r \le \frac{1}{2}$, the solution point of the three-factory problem is located in area BOC enclosed by arc $\widehat{BOC}$ and line BC as shown in Fig. 4, where O is the Fermat point[2] of triangle ABC.*

Proof. To locate the Fermat point: construct two equilateral triangles on any of the three sides of the given triangle; construct the circumscribed circle for each equilateral triangle; the two circles intersect at the Fermat point. As shown in Fig. 4, arc $\widehat{BOC}$ and arc $\widehat{AOB}$ are part of circumscribed circle of constructed equilateral triangle for BC and AB, respectively. For point O and $O'' \in$ arc $\widehat{BOC}$, $\angle BOC = 120^o$ and $\angle BO''C = 120^o$.

We use d_a, d_b and d_c to denote the weight at point A, B and C as shown in Fig. 4, and $r = \frac{d_a}{d_b+d_c}$ (noted that in our real scenario, d_a is the size of data object generated by operator and r is the data reduction ratio of operator). For the three factory problem, we have the following [26]:

$$\cos(\angle B'BC) = \frac{d_a^2 - d_b^2 - d_c^2}{2d_bd_c}.$$

To solve the three-factory problem, we can construct a circle by using point B, C and BB' as a tangent line. Similarly, we can construct another circle for A, B and $A'A$. Then the intersection of two circles inside the triangle ABC is the solution [26], as O' in Fig. 4. Noted that when the weight of one point is no less than the sum of other two, the solution is located at the point with the largest weight [26]. When $d_a = d_b = d_c$, $\angle B'BC = 120^o$, the solution is point O as shown in Fig. 4, which is also known as the Fermat point. We call the area inclosed by arc $\widehat{BOC}$ and line BC *Fermat area*. When $r \le 1/2$, we have

$$\cos(\angle B'BC) \le \frac{(\frac{d_b+d_c}{2})^2 - d_b^2 - d_c^2}{2d_bd_c} \le -\frac{1}{2}.$$

As $\angle B'BC \ge 120^o$ thus $\angle BO'C \ge 120^o$ ($\angle B'BC = \angle BO'C$), arc $\widehat{BC}$, which is a part of circle constructed using B, C and BB' as a tangent line, will be in the Fermat area. So the three factory solution point $O' \in \widehat{BC}$ is also in the

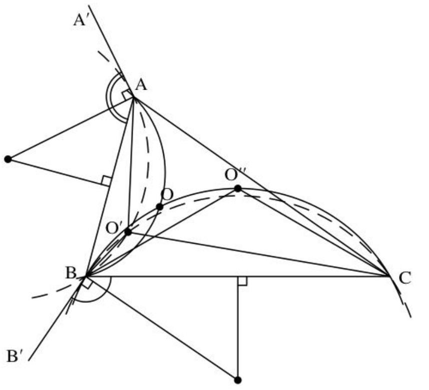


Fig. 4. The Three Factory Problem

Fermat area. For example O' in Fig. 4 is the solution point for $d_a = 1, d_b = 1.2$ and $d_c = 0.8$.

□

One implication of Lemma 1 is that the optimal operator placement always locates inside the *Fermat area* when $r \le 1/2$.

Lemma 2. *As shown in Fig. 4, the point O'' inside the Fermat area, which maximizes $d_b \cdot BO'' + d_c \cdot CO''$, is located on arc $\widehat{BOC}$ and deterministic, where $\alpha = d_b/d_c$ (assume $d_b \ge d_c$).*

Proof. As $d_b \cdot BO'' + d_c \cdot CO''$ includes distances BO'' and CO'', we can quickly infer O'' must be on arc $\widehat{BOC}$.

As $O'' \in$ arc $\widehat{BOC}$ and θ denotes $\angle O''BC$, we have

$$\frac{sin120^o}{BC} = \frac{sin\theta}{O''C} = \frac{sin(60^o - \theta)}{O''B}. \quad (5)$$

From (5), we get

$$\begin{aligned} d_b \cdot BO'' + d_c \cdot CO'' = \ & BC \cdot [d_b(\cos\theta - \tfrac{1}{\sqrt{3}}\sin\theta) \\ & + \tfrac{2}{\sqrt{3}} d_c \sin\theta]. \end{aligned}$$

To maximizes $d_b \cdot BO'' + d_c \cdot CO''$, we need to maximize:

$$\max_\theta\{d_b(\cos\theta - \frac{1}{\sqrt{3}}\sin\theta) + \frac{2}{\sqrt{3}}d_c\sin\theta\}, \quad (6)$$

$$\max_\theta\{\alpha\cos\theta - \frac{\alpha}{\sqrt{3}}\sin\theta + \frac{2}{\sqrt{3}}\sin\theta\}, \quad (7)$$

and

$$\max_\theta\{\sin(\theta + \phi)\}, \quad \tan\phi = \frac{\sqrt{3}\alpha}{2-\alpha}. \quad (8)$$

As $0 \le \theta < 60^o$, when $1 \le \alpha \le 2$, from (8) we have $\theta = \frac{\pi}{2} - \phi$. So point O'' can be determined by α. For example, when $\alpha = 1$, $\theta = 30^o$, and when $\alpha = 2$, $\theta = 0$, in other words point O'' locates at point C. When $\alpha > 2$, $\tan\phi = \frac{\sqrt{3}\alpha}{2-\alpha} > \sqrt{3}$, so $\phi < -60^o$. As $0 \le \theta < 60^o$, $d_b \cdot BO'' + d_c \cdot CO''$ will be a negative value. So (8) is not applicable for $\alpha > 2$. However, since o'' is located at C when $\alpha = 2$, we can conclude that O'' is always located at C when $\alpha > 2$.

□

The term of $d_b \cdot BO'' + d_c \cdot CO''$ can be seen as the communication cost in the operator placement context. It follows that the point in the *Fermat area* that maximizes the communication cost can be derived as:

[1]Three factory problem can be stated as follows: Given three points, a, b, c. Each one is connected with a fourth point, u. Suppose the length of $ua = r_1$, $ub = r_2$, and $uc = r_3$. Choose the point u so that $k_1r_1 + k_2r_2 + k_3r_3$ is minimized. k_1, k_2, k_3, are arbitrary positive weighing factors.

[2]Fermat point is a point such that the total distance from the three vertices of the triangle to the point is the minimum possible.

- It locates at the midpoint of the arc of the *Fermat area* when $\alpha = 1$;
- It locates on the arc between the midpoint and the point with the smaller data output when $1 < \alpha < 2$;
- It locates at the point with the smaller data output when $\alpha \geq 2$.

We denote $C^*(T)$, $C(T)$ and $\hat{C}(T)$ as the in-network processing cost of the optimal solution, the heuristic algorithm and the greedy algorithm, respectively, for the operator tree T, where each operator is compatible with the constraint that $r \leq 1/2$. The following theorem summarize the approximate ratio for our heuristic algorithm.

Theorem 1. *For an operator tree T and an data reduction ratio at each operator is no more than $1/2$, the performance of our proposed heuristic algorithm is bounded above:*

$$\begin{cases} C(T) \leq \frac{2}{\sqrt{3}}\sqrt{\alpha^2-\alpha+1}C^*(T) & \text{if } 1 \leq \alpha \leq 2 \\ C(T) \leq \alpha C^*(T) & \text{if } 2 < \alpha < 3 \\ C(T) = C^*(T) & \text{if } \alpha \geq 3 \end{cases}$$

Proof.
Case 1: if $1 \leq \alpha \leq 2$

From Lemma 1, it is known that for an operator with $r \leq 1/2$, the optimal operator placement (the solution of the three-factory problem) is located inside the *Fermat area*. The operator placement with the minimum communication cost for the input data is B, where the communication cost $C'_\beta = d_c \times BC$. At the same time, when $1 \leq \alpha \leq 2$, from Lemma 2, we have the worst case with the maximum communication cost of input data $C'_\gamma = d_b \times BO'' + d_c \times CO''$, where

$$\angle O''BC = \frac{\pi}{2} - \arctan\frac{\sqrt{3}\alpha}{2-\alpha}. \quad (9)$$

It follows that

$$\frac{C'_\gamma}{C'_\beta} = \frac{d_b \times BO'' + d_c \times CO''}{d_c \times BC} = \frac{\alpha \times BO'' + CO''}{BC}.$$

From (5) and (9), we can obtain

$$\begin{aligned} \frac{C'_\gamma}{C'_\beta} &= \frac{2\alpha}{\sqrt{3}}\sin(60^\circ - \theta) + \frac{2}{\sqrt{3}}\sin\theta \\ &= \frac{(\sqrt{3}\cos\theta - \sin\theta)\alpha + 2\sin\theta}{\sqrt{3}} \\ &= \frac{2}{\sqrt{3}}\sqrt{\alpha^2-\alpha+1}. \end{aligned}$$

Assume that optimal solution always has the best case of communication cost of input data for each operator and the heuristic solution always has the worst case. From (3), we can obtain

$$\frac{C(T)}{C^*(T)} \leq \frac{2}{\sqrt{3}}\sqrt{\alpha^2-\alpha+1}.$$

Case 2: if $2 < \alpha < 3$

When $\alpha > 2$, from Lemma 2 we have O'', which maximizes $d_b \times BO'' + d_c \times CO''$, is always located at point C. And the best case for the communication cost of input data is still B. So we have

$$\frac{C'_\gamma}{C'_\beta} = \frac{d_b \times BC}{d_c \times BC} = \alpha.$$

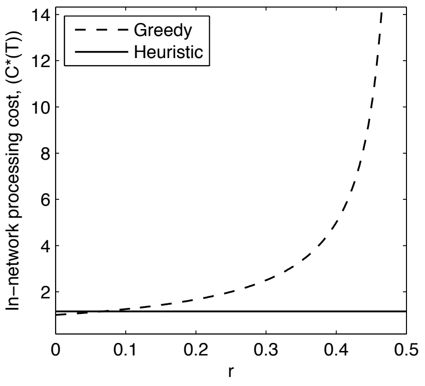


Fig. 5. Comparison of in-network processing cost between heuristic algorithm and greedy algorithm for T, where all the operators have $r \leq 1/2$, when $\alpha = 1$.

Similarly, $\frac{C(T)}{C^*(T)} \leq \alpha$.

Case 3: if $\alpha \geq 3$

As $r = \frac{d_a}{d_b+d_c} \leq \frac{1}{2}$ and $\alpha \geq 3$, by summing up we can obtain

$$d_b \geq d_a + d_c.$$

In the three-factory problem, when $d_b \geq d_a + d_c$, the solution is always located at point B, no matter where point A is located [25] as shown in Fig. 4. Although our heuristic algorithm uses the sink node as the parent for each operator placement decision, in this case, the resulted operator placement is always the child that has larger data output, which is the optimal placement for the operator.

If the placement of an operator tree is globally optimal, the placement of each operator is also locally optimal with respect to its children and the parent operator [16]. In this case, the optimal solution and the heuristic algorithm have the same placement for each operator, $C(T) = C^*(T)$.

□

From Theorem 1, we can see the in-network processing cost of the heuristic algorithm is at worst three times of that of the optimum solution ($C(T) < 3C^*(T)$). For the greedy algorithm, $\hat{C}(T) \leq \frac{1}{1-2r}C^*(T)$, when $r < 1/2$ and $\alpha = 1$ (in [10], only the results about $\alpha = 1$ are provided). As shown in Fig. 5, the heuristic algorithm performs much better than the greedy algorithm for $\alpha = 1$. The processing cost resulted from the greedy algorithm increases dramatically with the increase of r.

It follows, from the mathematical analysis above, that our heuristic algorithm has much better approximate ratio to the optimal solution than the placement resulted from the greedy algorithm.

VI. Low-Overhead Distributed Implementation

As shown in Section V, when the data reduction ratio of an operator is no more than $1/2$, the solution of the three-factory problem is located in the *Fermat area*, no matter where the parent of the operator is. It follows that we only need to find the heuristic operator placement within the *Fermat area*. In this section, we describes an efficient way to find the placement for each operator in the network, using our

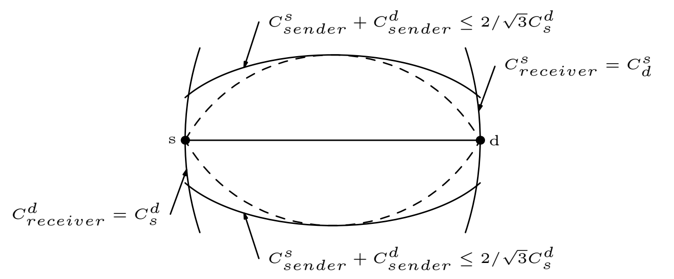

$C^s_{sender} + C^d_{sender} \leq 2/\sqrt{3}C^d_s$
$C^s_{receiver} = C^s_d$
s
d
$C^d_{receiver} = C^d_s$
$C^s_{sender} + C^d_{sender} \leq 2/\sqrt{3}C^d_s$

Fig. 6. An illustration of the search region involved in the area-restricted Flooding

heuristic algorithm in a distributed manner. This is achieved by proposing an area-restricted flooding mechanism to locate the operator placement within a desired area, minimizing the number of nodes involved in flooding and reducing the message overhead.

A. Area-restricted Flooding

In order to minimize the messages overhead of the distributed implementation of our heuristic algorithm, we propose an area-restricted flooding approach to locate each operator placement. As shown in Fig. 6, s and d denote the two children of an operator that needs to be placed, the area enclosed by the dashed line is called the *search region*, two times of the *Fermat area*. We use the *search region* instead of the *Fermat area*, because the nodes do not have the location information of the sink and the desired placement can be at either side of line sd. Thus, using the *search region* can guarantee to find the desired heuristic operator placement. Notice that the flooding area is set up based on the cost between nodes, instead of geographic information. For example, Fig. 1 can be seen as a coordinate, where the location of a node is based on its minimum costs to node s and to node d. In this case, our proposed area-restricted flooding mechanism does not require any location information.

Assume that every node knows the minimum communication costs to both the sink and node d (we will show how to obtain these information in next subsection), node s will initialize an area-restricted flooding algorithm to find a node as the operator placement within the search region, which minimizes (4). We call the node that starts the flooding an *initiator*, like node s in Fig. 6, and call the node that finishes the flooding a *terminator*, like node d in Fig. 6.

To find the heuristic operator placement and minimize the message overheads, we need to address several issues, including

1) How to control the flooding within the search region?
2) In order to find the operator placement accurately, the minimum cost field of an initiator must be set up for the nodes in the search region. In other word, every node in the search region needs to know the minimum cost to the initiator. Then the nodes can properly calculate the placement cost according to (4).
3) In order to get the minimum placement cost of nodes in the search region, a straightforward way is to collect all the placement costs at one node. However, the process for each node to sends its to one particular node will incur huge amount of message overhead, one needs to develop an intelligent strategy to efficiently collect all the costs and find out the node with the minimum placement cost in the search region?
4) How to combine the procedures of setting up the minimum cost field of the initiator and finding operator placement into one flooding round?

Area-restricted flood for searching each heuristic operator placement is initiated by broadcasting an advertisement (ADV) packet at the initiator. It floods from the initiator to the terminator. ADV is transmitted during area-restricted flooding, which carries the following information:

- C^s_d: the minimum cost between the initiator and the terminator.
- C^s_{sender}: the cost between the initiator and the current sender that broadcasts the packet.
- $cost^{'}_o$: the current minimum placement cost obtained at the sender.

In order to restrict the flooding area, when the node receives an ADV packet, it will re-broadcast the packet only if the following two conditions are satisfied:

- $C^s_{sender} + C^d_{sender} \leq 2/\sqrt{3}C^d_s$: From Lemma 2, we know that all the points in the search region satisfy this condition as shown in Fig. 1. Although some points outside the search region may also satisfy this condition, we intend to use it to simplify the computation of guaranteeing the flooding covering the search region.
- $C^s_{sender} < C^s_{receiver} \leq C^s_d$ and $C^d_{receiver} < C^d_{sender} \leq C^d_s$: To make the flooding forwarded from the initiator to the terminator and avoid devious path and loop that the packets may travel, these constraints will straighten the traveling paths from the initiator to the terminator. In other word, the receivers are supposed to have a smaller minimum cost to the terminator than the sender, and a higher minimum cost to the initiator than the sender.

The aforementioned procedure addresses the first issue, which restricts the flooding in the area enclosed by the solid line as shown in Fig. 6 in a simple and efficient way.

To tackle the second issue, although a straightforward flooding will eventually find the minimum cost path to the initiator for each node, it will incur significant amount of message overhead. Here we adapt the cost field establishment algorithm [27] to set up the cost field for the initiator. A backoff timer is enabled after the node receives ab ADV packet, the expire time of which is set to be proportional to the cost between the sender and the receiver (time coefficient denoted by λ) as follows:

$$T_{delay} = \lambda C(p, q).$$

During T_{delay}, the node might receive more than one ADV packet. However, the node will only broadcast the ADV with the least cost to the initiator and discard all other ADV packets received after the timer expires. We use Fig. 7 as an example to illustrate how the cost field establishment algorithm works to reduce the ADV broadcast, explained as follows:

- At time t, node a broadcasts an ADV packet that includes C^a_s. After node b, c and d receive the ADV packet from node a, they set $C(s, b) = C^a_s + 4$, $C(s, c) = C^a_s + 2$, and

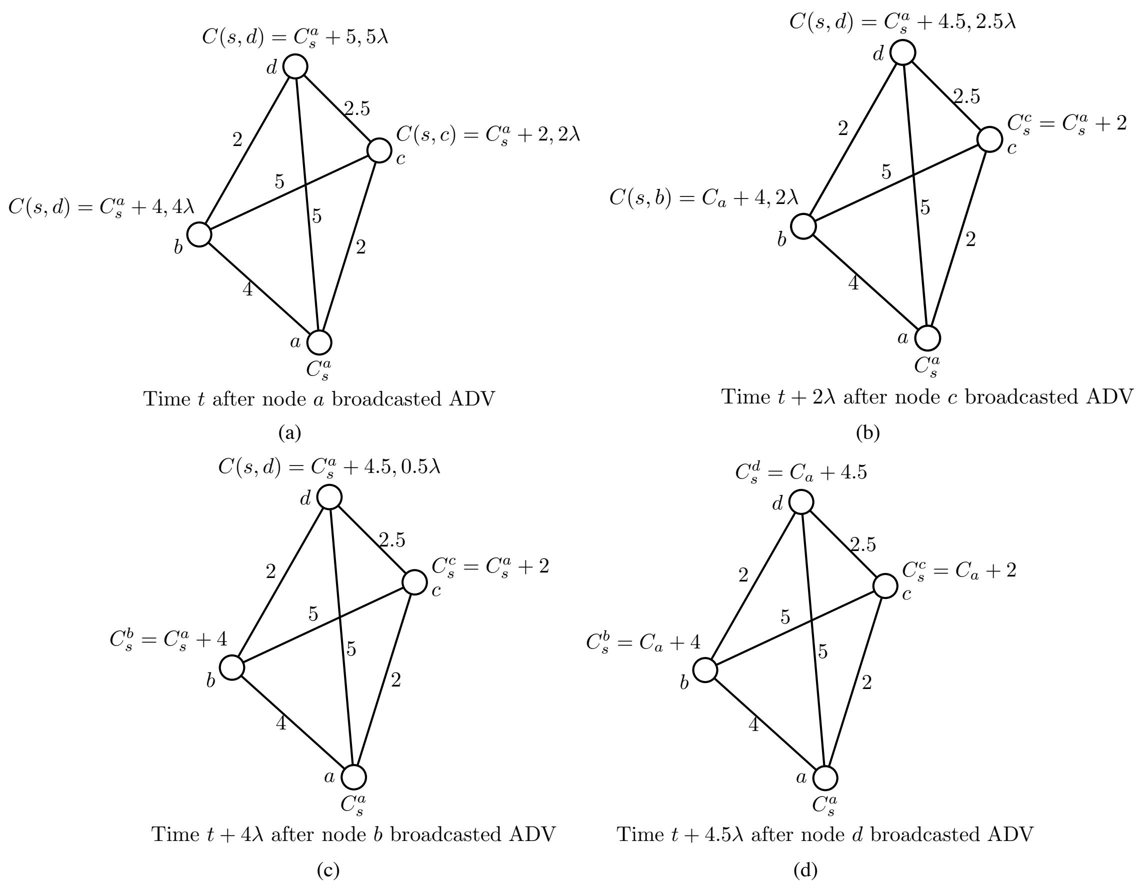

$C(s,d) = C_s^a + 5, 5\lambda$
d
2.5
$C(s,c) = C_s^a + 2, 2\lambda$
c
2
5
$C(s,d) = C_s^a + 4, 4\lambda$
b
5
2
4
a
C_s^a
Time t after node a broadcasted ADV
(a)
$C(s,d) = C_s^a + 4.5, 2.5\lambda$
d
2.5
$C_s^c = C_s^a + 2$
c
2
5
$C(s,b) = C_a + 4, 2\lambda$
b
5
2
4
a
C_s^a
Time $t + 2\lambda$ after node c broadcasted ADV
(b)
$C(s,d) = C_s^a + 4.5, 0.5\lambda$
d
2.5
$C_s^c = C_s^a + 2$
c
2
5
$C_s^b = C_s^a + 4$
b
5
2
4
a
C_s^a
Time $t + 4\lambda$ after node b broadcasted ADV
(c)
$C_s^d = C_a + 4.5$
d
2.5
$C_s^c = C_a + 2$
c
2
5
$C_s^b = C_a + 4$
b
5
2
4
a
C_s^a
Time $t + 4.5\lambda$ after node d broadcasted ADV
(d)

Fig. 7. Example of cost field establishment algorithm.

$C(s,d) = C_s^a + 5$, respectively (assuming the initial C_s^b, C_s^b and C_s^b are ∞ as they do not have cost information to the initiator). Then each of them sets a timer for re-broadcasting ADV. The expiring period is proportional to the transmission cost between the sender and the receiver. For node b, c and d, the expiring periods are 4λ, 2λ and 5λ, respectively.

- At time $t + 2\lambda$, the timer of node c expires. Node c finalizes $C_s^c = C(s,c)$ and broadcasts an ADV message including C_s^c. When node d receives this ADV packet, as $C(s,d) > C_s^c + 2.5 = C_s^a + 4.5$, node d updates $C(s,d)$ to $C_s^a + 4.5$ and resets its timer to 2.5λ (noted that previous timer does not expire by the time $t + 2\lambda$). For node a and b, as $C_s^a < C_s^c + 2$ and $C(s,b) < C_s^c + 3$, they simply discard this ADV message.
- At time $t + 4\lambda$, the timer of node b expires. Node b finalizes $C_s^b = C(s,b)$ and broadcasts an ADV message containing C_s^b. All other nodes will discard this ADV message because they have already had a lower cost.
- At time $t + 4.5\lambda$, the timer of node d expires. Node d finalizes $C_s^d = C(s,d)$ and broadcasts an ADV message with C_s^d.

For this cost field establishment algorithm, we can observe from Fig. 7 that each node only broadcasts an ADV message once with the optimal cost and eliminates the broadcast of non-optimal ADV messages. The algorithm has the following two properties [27]:

- Each node only broadcasts the optimal cost to the initiator, and discards all redundant or non-optimal ADV messages.
- Node can get the minimum cost to the initiator by only one ADV message broadcast at each node.

For the third issue, we use the terminator to collect the minimum placement cost in the restricted area. Instead of forwarding the placement cost at each node individually, we propose to aggregate the placement cost information en route and only forward aggregated information to the terminator.

Once a node gets the minimum cost to the initiator, it can calculate its placement cost. So it is efficient to include the placement cost in the ADV packet rather than send individual message to the terminator. However, as discussed above in the cost field establishment algorithm, since the node will abandon all the messages received after the backoff timer expires, ADV packets are eliminated during flooding. In order to merge the establishment of the cost field for the initiator and the process of locating the operator placement together so as to reduce the message overhead collaboratively, we introduce the following process into the area-restricted flooding:

- The node broadcasts an ADV with the minimum C_{sender}^s and mthe inimum placement cost received during the backoff period.
- After the backoff timer expired, the node only re-broadcasts its ADV packet, if the placement cost of which is less than the current obtained one.
- During the flooding, the node will cache the least placement cost and its sender. Or if the least placement cost is

generated by itself, it would cache itself as the generator of the least cost. This will be used to trace back the operator placement later.

For completeness, the pseudo code of the area-restricted flooding is shown in Algorithm 1.

Since the flood area is set up based on the cost, at least two nodes (initiator and terminator) are included in the flood area. Although the minimum cost to the initiator and the placement cost are both included in the ADV message that can be eliminated after the timer expires, we give the priority to the ADV message with less placement cost and make it forwarded to the terminator. After receiving the minimum placement cost, the terminator performs a trace-back process to find the operator placement.

Proposition 1. *For a pair of initiator and terminator, area-restricted flooding can locate the heuristic operator placement within time* $\lambda\frac{2}{\sqrt{3}}C_s^d$.

Proof. The proof can be divided into three sequential steps. First, we will show that the node that is operator placement is included in the flood area. Second, the placement cost of a node can be received at the terminator. Finally, the delay is no more than $\lambda\frac{2}{\sqrt{3}}C_s^d$.

For the first part, any node $p \in V$ with $C_p^s + C_d^p \leq \frac{2}{\sqrt{3}}C_s^d$ (the point in the *Fermat area*) will be included in the flood area as shown in Fig. 6. Therefore, the node that is operator placement is certainly included in the flooding area as proved in Lemma 1.

As discussed above, the node can get its minimum cost to the initiator with broadcasting once and only once. Since before broadcasting the ADV message the node has received the minimum cost to the initiator, the placement cost of the node is accurately computed. Assuming node q is the placement of operator o, $cost_o^{'q}$ is minimum. $cost_o^{'q}$ is included in ADV message and broadcasted. When the neighbor node p in the flood area, which is complied with $C_q^d > C_p^d$ (recall that is used to avoid devious path and loop), receives this ADV message, it will replace the $cost_o'$ with $cost_o^{'q}$ and forward it toward the terminator no matter whether the time expires or not. Therefore, the minimum placement cost will be eventually received at the terminator.

As $C_q^s + C_q^d \leq \frac{2}{\sqrt{3}}C_s^d$, the delay of the message including $cost_o^{'q}$ will arrive at the terminator no later than $\lambda\frac{2}{\sqrt{3}}C_s^d$ after the area-restricted flooding is initialized, according to the scheme of the cost field establishment algorithm.

□

Algorithm 1: Area-restricted Flooding Algorithm

```
Event: node $p$ receives ADV from node $q$
begin
    if $C_q^d > C_p^d$ && $C_q^s + C_p^d + C(p,q) \leq \frac{2}{\sqrt{3}}C_s^d$ then
        if $C_q^s \leq C_d^s$ && $C_q^s + C(p,q) < C_p^s$ then
            reset timer to expire after $\lambda \cdot C(p,q)$
            $C_p^s = C_q^s + C(p,q)$
            calculate $cost_o^{'p}$
            if $cost_o^{'} > cost_o^{'p}$ then
                cache $p$
            else
                $cost_o^{'p} = cost_o^{'}$
                cache $q$
            end
        else if $C_q^s \leq C_d^s$ then
            if $cost_o^{'} < cost_o^{'p}$ then
                $cost_o^{'p} = cost_o^{'}$
                cache $q$
            end
        end
    end
end
Event: node $p$'s timer expires
begin
    replace $C_{sender}^s$ with $C_p^s$
    replace $cost_o^{'}$ with $cost_o^{'p}$
    broadcast ADV
end
Event: node $p$ receives ADV from $q$ after $p$'s timer
       expired
begin
    if $C_q^s \leq C_d^s$ && $C_q^d > C_p^d$ && $C_p^s + C_p^d \leq \frac{2}{\sqrt{3}}C_s^d$
    then
        if $cost_o^{'} < cost_o^{'p}$ then
            replace $cost_o^{'p}$ with $cost_o^{'}$
            cache $q$
            broadcast ADV
        end
    end
end
```

B. Distributed Operator Tree Placement

As discussed in Section VI-A, before the initiator initializes the area-restricted flooding to find the operator placement, the prerequisite is that all the nodes within the search region know the minimum costs to the sink and the terminator. Unlike in [10], where the authors assumes the nodes are knowledgable about the distance between nodes, we do not assume the nodes have any location information so as to make our distributed algorithm applicable. We also use the cost field establishment algorithm to obtain these minimum cost information.

Another issue about the area-restricted flooding is when the initiator should initiate the flood and when the terminator should finalize it. As the delay at each node is proportional to the communication cost between the sender and the receiver in the cost field establishment algorithm, the longest delay of the cost field establishment is λC_{max}, where C_{max} is the largest minimum cost path in the network. In order to avoid redundant message and to keep it from the effect of transmission, propagation and processing delay, λ is a constant value for certain network as shown in [27]; in turn λC_{max} is also determined. Therefore, for initializing area-restricted flooding, the initiator waits for a λC_{max} delay after obtaining the minimum cost to the terminator. For finalizing area-restricted flooding, the terminator will receive the first

ADV message λC_s^d time later after the initiator starts the flooding and it will wait for a delay $(\frac{2}{\sqrt{3}}-1)\lambda C_{source}^d$ before finalization.

To prevent the description from becoming cumbersome we have omitted the details that can either be inferred or implemented using standard techniques. These details include the precise content of the messages, the manner of calculating the hosting cost, the delay before performing the initialization and finalization of area-restricted flooding and node's behaviors during cost field establishment flooding and area-restricted flooding.

The distributed implementation of our heuristic algorithm is shown as follows according to different node types. It will find the placement of an operator tree in a bottom-up manner. Note that node type is not exclusive, that is, the node can be more than one type at the same time. For example, when a node is both the initiator and the terminator, it will select itself as the operator placement.

SINK NODE:
Event: When there is a in-network processing with operator tree (T) needs to be performed.
Action:
- For each operator of T, assign its two children as *initiator* and *terminator* of area-restricted flooding, respectively.
- Broadcast T into network using cost field establishment flooding. Thus, every node will receive T and be aware of the minimum cost to sink.

SENSOR NODES and OPERATOR PLACEMENT:
Event: When sensor node receives T or the node is selected as operator placement.
Action: If the placed operator is root operator, inform sink that the placement of operator tree T is complete, else perform the cost field establishment flooding if it is terminator.

INITIATOR:
Event: After being aware about the minimum cost to terminator.
Action: Perform area-restricted flooding.

TERMINATOR:
Event: After obtaining the minimum placement cost for current operator.
Action: Perform trace-back process to inform the node with minimum placement cost and assign it as operator placement.

As the cost field establishment flooding is used to obtain the minimum cost information in the distributed implementation, which establishes the minimum cost filed with only one message broadcast at each node in one flooding round, the message overhead at each node for the minimum cost information is the number of floods. The message overhead of one area-restricted flooding round is much smaller than one round of the cost field establishment flooding, because the search region is much smaller than the entire network field and the message overhead of the area-restricted flooding is no more than $3/2$ that of the cost field flooding for the same area as shown in [9]. As the number of the cost field establishment flooding round and the number of the area-restricted flood rounds are $(M+1)/2$ and $(M-1)/2$, respectively. It follows that the message overhead per node is $O(M)$. As a comparison, for Sync and MCFA, the message overhead is of $O(\sqrt{N}M\log_2 M)$ and $O(\sqrt{N}M)$, respectively.

VII. Numerical Evaluation

In this section, we evaluate the performance of our heuristic algorithm through simulations, and compare their performance among the heuristic algorithm, the greedy algorithm, Sync and MCFA.

Two basic network topologies are used in simulation, which are the Manhattan Graph (MG) and Controlled Random Graph (CRG). These graphs represent both abstract and realistic sensor network topologies. In order to make the network topologies more realistic, we choose to degrade the connectivity. This can be done for MG by randomly removing a percentage of the nodes and for CRG by increasing the area in which the network is deployed so as to reduce the number of reachable neighbors. The CRG is generated by randomly distributing network nodes in a square area. When placing a node, we adopt the approach in [19] and we require the placement of the node is at least half of radio range to any other node so as to avoid unrealistically close nodes. The length of a side of the square area can be calculated by $\sqrt{N}r_c f$, where N is the number of network nodes, r_c is radio range (50m in simulation) and f is a factor to control the connectivity. In the simulation, three particular network topologies each with 200 nodes are used in simulation as shown in Fig. 8.

In simulation, we use d_{pq}^2 as the communication cost between two adjoining nodes p and q, where d_{pq} is the distance between node p and q and $d_{pq} \leq 50m$, i.e., signals attenuate inversely proportional to the square of distance. For real applications, the sender includes its broadcasting power in its ADV message. When a receiver hears the message, it can calculate the minimum energy needed and the minimum transmission power needed for the sender to just reach the receiver by measuring the signal strength and employ one chosen signal attenuation model, then acknowledge the sender with these information. As mentioned before, we do not have any constraint for the communication cost, which can take any common form, such as hop count or consumed energy.

We firstly evaluate the impact of parameters including α and network topology on the operator placement for both heuristic and greedy algorithms, then we compare message overheads of distributed implementation among heuristic, Sync and MCFA.

A. In-network Processing Cost

As analyzed mathematically in V-B, α impacts the approximate ratio of the heuristic algorithm. To verify our mathematic analysis, we simulated all three algorithms in three network topologies and varied α from 1 to 3.5. Notice that we use Sync as the optimal solution to obtain the minimum in-network processing cost.

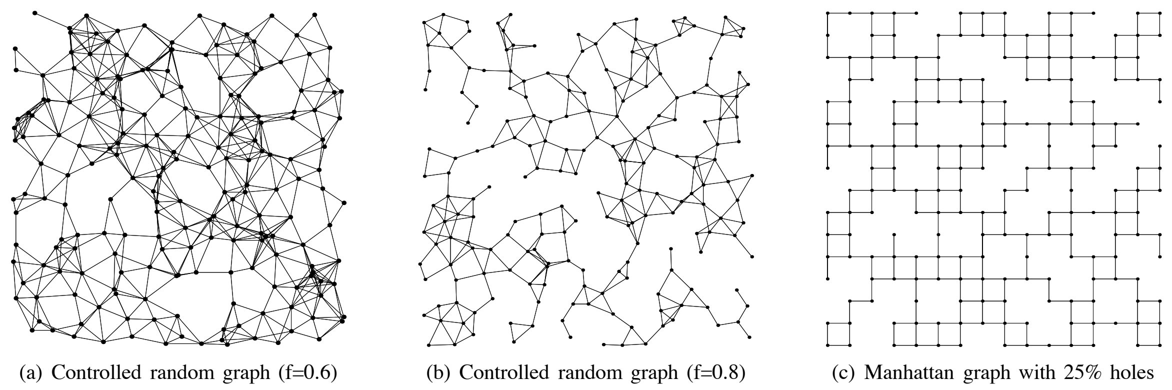

(a) Controlled random graph (f=0.6) (b) Controlled random graph (f=0.8) (c) Manhattan graph with 25% holes

Fig. 8. Network Topologies

As shown in Fig. 9, we can observe that, the ratio between the processing cost resulted from both heuristic and greedy algorithms and the resulted from the optimal algorithm, decreases overall as α increases from 1 to 3. It is expected from our mathematical analysis in Section V-A. Moreover, the heuristic algorithm performs better than the greedy algorithm(at least no worse than the greedy algorithm) in all these three topologies. In CRG (f=0.6) and CRG (f=0.8) shown in Fig. 9(a) and 9(b), the heuristic algorithm and the greedy algorithm achieve the same optimal cost when α reaches a threshold, $\alpha = 2.4$ and $\alpha = 2.6$, respectively. In Fig. 9(c), the heuristic algorithm obtains the optimum cost much earlier (at $\alpha = 2.2$) than the greedy algorithm (at $\alpha = 3.0$).

As analyzed in Section V, when $a \geq 3$ and the data reduction ratio of operator is no more than $1/2$, the optimal operator placement is the node with the largest data output, as verified by our simulation results. The intersection point of these three algorithms is that when $a \geq 3$, all of them choose the node with the largest data output as the operator placement, so both heuristic and greedy algorithms can obtain the optimal cost at most when α is 3.

From Fig. 9, we can also notice that the worst-case in-network process cost for the greedy algorithm varies according to different network topologies, while the worst case of the heuristic algorithm in each topology is when $\alpha = 1$, which is not compliant with the mathematical analysis in Section V. That is because we always choose the worst case of the operator placement as the placement of our heuristic algorithm in mathematical analysis, which would never occur in real network scenarios.

B. Message Overhead

The message overhead is the metric to measure the cost of performing distributed implementation of operator placement algorithms. In this subsection, we compare the message overheads for our heuristic algorithm, Sync and MCFA.

Fig. 10(a) illustrates the message overheads per node for these three algorithms, as a function of operator tree size for a given network size (i.e., $N = 200$). From Fig. 10(a), we can see the message overhead increase as the operator tree size increases. We can also see that the heuristic algorithm has much less message overhead than that of Sync and at least 50% less than that of MCFA. Finally, the gap increases as the operator tree increases.

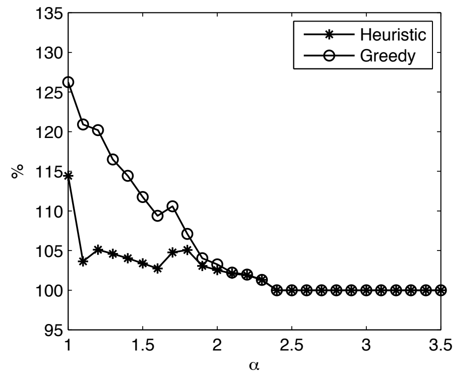
In Fig. 10(b), we compared the message overhead per node for the heuristic algorithm, MCFA and Sync, as a function of the network size with a fixed-size operator tree (i.e., $M = 7$). As analyzed mathematically, the message overhead of Sync and MCFA increases with the network size, while for the heuristic algorithm, the message overhead stays flat for increasing network size.

Notice that, although the heuristic algorithm causes slightly more in-network processing cost for the operator tree than that of the optimal solution, the distributed implementation of our heuristic algorithm has much less message overhead than that of distributed optimal solutions (i.e., Sync and MCFA). The complexity reduction in managing the message overhead would be translated into energy saving in WSNs, which in turn, would justify the rationale of deploying our heuristic algorithm.

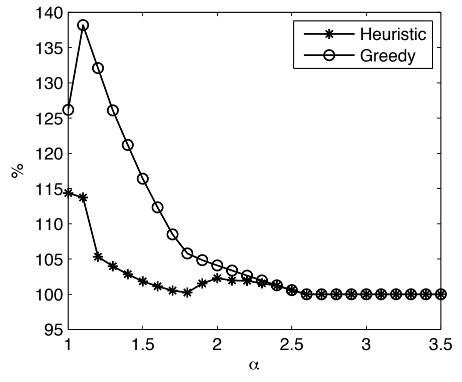
VIII. Conclusion

The problem of operator tree placement is crucial for in-network processing in wireless sensor networks. In the paper, we designed a heuristic algorithm to place the operator tree on network nodes. First we presented the mathematical analysis of our proposed heuristic algorithm compared with the optimal solution. We then followed with a low-cost distributed implementation of our distributed heuristic algorithm and analyzed its message overhead, which is of $O(M)$ and is much less than that of other distributed algorithms. Finally, the simulation results verified that our heuristic algorithm has better performance than the greedy algorithm and incurs slightly more cost than the optimum solution. The results also indicated that the heuristic algorithm has much less message overhead than that of other distributed placement algorithms (i.e., Sync and MCFA). Our research, with its practical algorithm and fundamental analysis, shed new lights onto the in-network processing paradigm, which are pervasive in modern communication and network infrastructure.

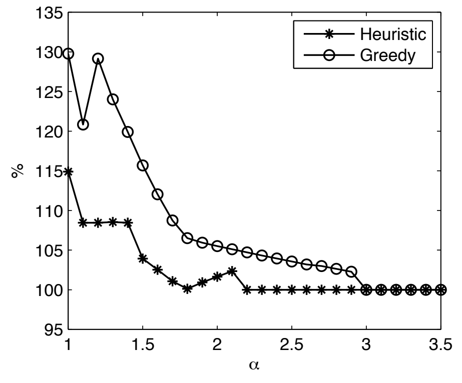
As future works, three alternative venues are under considerations. First, we would like to extend our binary tree structure into a M-ary tree structure, which provides an elevated capability for data fusion at operators. Second, we plan to explore the possibility of joint optimization of energy efficiency and load balancing in large wireless sensor networks. Finally, we are also exploiting this fundamental research into



(a)

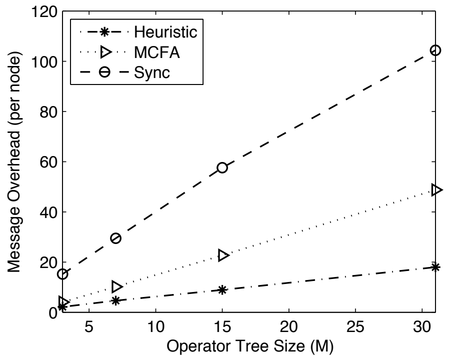


(b)

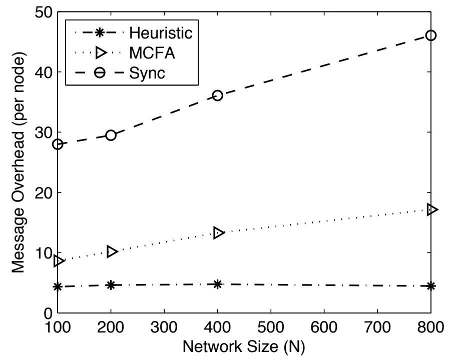


(c)

Fig. 9. In-network processing cost of heuristic and greedy algorithms compared with that of optimum solution according to α in network topology CRG (f=0.6) (a), CRG (f=0.8) (b), and Manhattan with 25% holes (c), respectively.



(a)



(b)

Fig. 10. Message overheads per node of heuristic algorithm, Sync and MCFA vs operator tree size in (a) and network size in (b).

real applications, such as, smartgrid communications, data-center management, to name a few.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[3] S. Liu, K. Fan, and P. Sinha, "Cmac: an energy-efficient mac layer protocol using convergent packet forwarding for wireless sensor networks," *ACM Trans. Sensor Netw. (TOSN)*, vol. 5, no. 4, p. 29, 2009.

[4] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009.

[5] J. Al-Karaki, R. Ul-Mustafa, and A. Kamal, "Data aggregation and routing in wireless sensor networks: Optimal and heuristic algorithms," *Computer networks*, vol. 53, no. 7, pp. 945–960, 2009.

[6] C. Hua and T. Yum, "Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 892–903, 2008.

[7] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Commun.*, vol. 14, no. 2, pp. 70–87, 2007.

[8] L. Ying, Z. Liu, D. Towsley, and C. Xia, "Distributed operator placement and data caching in large-scale sensor networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, april 2008, pp. 977 –985.

[9] Z. Lu and Y. Wen, "Distributed and asynchronous solution to operator placement in large wireless sensor networks," in *Mobile Ad-hoc and Sensor Networks, IEEE International Conference on.* IEEE, 2012.

[10] Z. Abrams and J. Liu, "Greedy is good: On service tree placement for in-network stream processing," in *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, 2006, p. 72.

[11] I. Greenberg and R. Robertello, "The three factory problem," *Mathematics Mag.*, vol. 38, no. 2, pp. 67–72, 1965.

[12] V. Shah, B. Dey, and D. Manjunath, "Network flows for functions," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on.* IEEE, 2011, pp. 234–238.

[13] ——, "Efficient flow allocation algorithms for in-network function computation," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE.* IEEE, 2011, pp. 1–6.

[14] R. Sappidi, A. Girard, and C. Rosenberg, "Maximum achievable throughout in a wireless sensor networks using in-network computation," *IEEE/ACM Trans. Netw.*, 2012.

[15] U. Srivastava, K. Munagala, and J. Widom, "Operator placement for in-network stream query processing," in *Proc. twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* ACM, 2005, pp. 250–258.

[16] N. Jain, R. Biswas, N. Nandiraju, and D. Agrawal, "Energy aware routing for spatio-temporal queries in sensor networks," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 3. IEEE, 2005, pp. 1860–1866.

[17] G. Chatzimilioudis, H. Hakkoymaz, N. Mamoulis, and D. Gunopulos, "Operator placement for snapshot multi-predicate queries in wireless sensor networks," in *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*. IEEE, 2009, pp. 21–30.
[18] A. Pathak and V. Prasanna, "Energy-efficient task mapping for data-driven sensor network macroprogramming," *IEEE Trans. Computers*, vol. 59, no. 7, pp. 955–968, 2010.
[19] B. Bonfils and P. Bonnet, "Adaptive and decentralized operator placement for in-network query processing," in *Proc. 2nd international conference on Information processing in sensor networks*. Springer-Verlag, 2003, pp. 47–62.
[20] K. Oikonomou, I. Stavrakakis, and A. Xydias, "Scalable service migration in general topologies," in *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*. IEEE, 2008, pp. 1–6.
[21] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer, "Network-aware operator placement for stream-processing systems," in *Data Engineering, 2006. ICDE'06. Proc. 22nd International Conference on*. IEEE, 2006, p. 49.
[22] G. Chatzimilioudis, N. Mamoulis, and D. Gunopulos, "A Distributed Technique for Dynamic Operator Placement in Wireless Sensor Networks," in *Eleventh International Conference on Mobile Data Management*. IEEE, 2010, pp. 167–176.
[23] M. Ye, X. Liu, W. Lee, and D. Lee, "Probabilistic top-k query processing in distributed sensor networks," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*. IEEE, 2010, pp. 585–588.
[24] S. Misra, M. Reisslein, and G. Xue, "A survey of multimedia streaming in wireless sensor networks," *IEEE Commun. Surveys & Tutorials*, vol. 10, no. 4, pp. 18–39, 2008.
[25] I. Greenberg and R. Robertello, "The three factory problem," *Mathematics Mag.*, vol. 38, no. 2, pp. 67–72, 1965.
[26] W. Van De Lindt, "A geometrical solution of the three factory problem," *Mathematics Mag.*, vol. 39, no. 3, pp. 162–165, 1966.
[27] F. Ye, A. Chen, S. Lu, and L. Zhang, "A scalable solution to minimum cost forwarding in large sensor networks," in *Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on*. IEEE, 2001, pp. 304–309.

Zongqing Lu is currently a Ph.D. candidate with the School of Computer Engineering in Nanyang Technological University, Singapore. He received bachelor and master degree both from Southeast University, China. His research interests include wireless sensor networks, mobile ad hoc networks, social networks, delay tolerant networks, mobile computing, network privacy and security. He is a student member of IEEE.

Yonggang Wen (S99-M08) was born in Nanchang, Jiangxi, China in 1977. He received his PhD degree in Electrical Engineering and Computer Science (with minor in Western Literature) from Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2008; his MPhil degree (with honor) in Information Engineering from Chinese University of Hong Kong (CUHK), Hong Kong, China, in 2001; and his BEng degree (with honor) in Electronic Engineering and Information Science from University of Science and Technology of China (USTC), Hefei, Anhui, China, in 1999. His major field of study focuses on information and communication technologies (ICT).

He is currently an Assistant Professor with School of Computer Engineering at Nanyang Technological University, Singapore. Previously, he has worked in Cisco as a Senior Software Engineer for content networking products. He has also worked as a Research Intern at Bell Laboratories, Sycamore Networks and Mitsubishi Electric Research Laboratory (MERL). He has published more than 50 papers in top journals and prestigious conferences. His system research on Cloud Social TV has been featured by international media (e.g., The Straits Times, The Business Times, Lianhe Zaobao, Channel News Asia, ZDNet, CNet, United Press International, ACM Tech News, Times of India, Yahoo News, etc). His research interests include cloud computing, mobile computing, multimedia network, cyber security and green ICT.

Dr. Wen is a member of Sigma Xi (the Scientific Research Society), IEEE and SIAM.

Rui Fan is currently an assistant professor with School of Computer Engineering at Nanyang Technological University. He received BSc from California Institute of Technology, MSc and Ph.D. both from Massachusetts Institute of Technology.

Su-Lim Tan is currently an assitant professor with Singapore Institute of Technology. He received bachelor degree from computer engineering in Nanyang Technological University and Ph.D. degree from University of Warwick, UK.

Jit Biswas is a Senior Scientist in the Neural and Biomedical Technology (NBT) department at the Institute of Infocomm Research, A*STAR (Agency for Science, Technology and Research), Singapore. He is currently working on vital signs monitoring using ambient sensing approaches and wireless sensor networks. Dr. Biswas has an undergraduate degree in Electrical and Electronics Engineering from Birla Institute of Technology and Science, Pilani (India) and a Ph.D. degree in Computer Science from the University of Texas at Austin (USA).