A Method for Incremental Data Fusion in Distributed Sensor Networks

Damianos Gavalas[1], Grammati Pantziou[2], Charalampos Konstantopoulos[3], Basilis Mamalis[2]
[1] Department of Cultural Technology and Communication, University of the Aegean, Mytilini, Lesvos Island, Greece
dgavalas@aegean.gr
[2] Department of Informatics, Technological Education Institute of Athens, Athens, Greece
{pantziou, vmamalis}@teiath.gr
[3] Research Academic Computer Technology Institute, Patras, Greece
konstant@cti.gr

Abstract. The use of mobile agents for data fusion in wireless sensor networks has been recently proposed in the literature to answer the scalability problem of client/server model. In this article, we consider the problem of calculating a near-optimal route for a mobile agent that incrementally fuses the data as it vis-its the nodes in a distributed sensor network. The order of visited nodes affects not only the quality but also the overall cost of data fusion. Our proposed heu-ristic algorithm adapts methods usually applied in network design problems in the specific requirements of sensor networks. It suggests the optimal number of MAs that minimizes the overall data fusion cost and constructs near-optimal itineraries for each of them. The performance gain of our algorithm over alter-native approaches is demonstrated by a quantitative evaluation..

1 Introduction

Multiple sensor data fusion is an evolving technology, concerning the problem of how to fuse data from multiple sensors in order to make a more accurate estimation of the environment [6]. It improves reliability while offering the opportunity to minimize the data retained. Applications of data fusion cross a wide spectrum, including environment monitoring, automatic target detection and tracking, battlefield surveillance, remote sensing, global awareness, etc [1]. They are usually time-critical, cover a large geographical area, and require reliable delivery of accurate information for their completion. Most energy-efficient proposals are based on the traditional client/server computing model to handle multisensor data fusion in

Please use the following format when citing this chapter:
Gavalas, Damianos, Pantziou, Grammati, Konstantopoulos, Charalampos, Mamalis, Basilis, 2006, in IFIP International Federation for Information Processing, Volume 204, Artificial Intelligence Applications and Innovations, eds. Maglogiannis, I., Karpouzis, K., Bramer, M., (Boston: Springer), pp. 635–642

Distributed Sensor Networks (DSNs); in that model, each sensor sends its sensory data to a back-end processing element (PE) or sink. However, as advances in sensor technology and computer networking allow the deployment of large amount of smaller and cheaper sensors, huge volumes of data need to be processed in real-time. In this paper, we propose the usage of mobile agents in DSNs for data fusion tasks as an alternative to the traditional client/server model.

The remainder of the paper is organized as follows: Section 2 reviews works related to our research. Section 3 discusses the design and functionality of our heuristic algorithm for designing near-optimal itineraries for mobile agents performing data fusion tasks in DSNs. A quantitative evaluation is presented in Section 4, while Section 5 concludes the paper and presents future directions of our work.

2. Related Work

Mobile agent (MA) technology has been proposed as an answer to the scalability problems of centralized models. The term MA refers to an autonomous program with the ability to move from host to host and act on behalf of users towards the completion of a given task [5]. MAs have been proposed in a variety of applications in traditional networks, including e-commerce, network management, information retrieval, etc [5]. DSN environments form a promising application area for MAs; yet, they pose new challenges as the link bandwidth is typically much lower than that of a wired network and sensory data traffic may even exceed the network capacity.

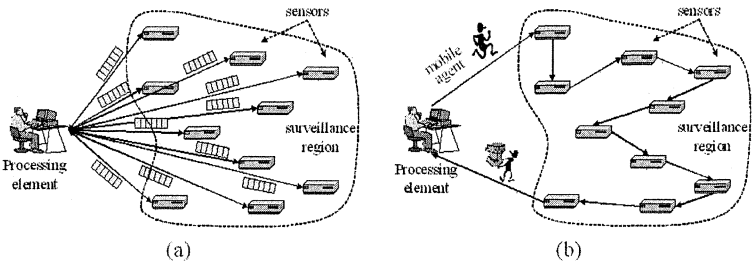

Fig. 1. Centralized vs. Mobile Agents-based data fusion in Distributed Sensor Networks.

To solve the problem of the overwhelming data traffic, [6] and [7] proposed the use of MAs for scalable and energy-efficient data aggregation. By transmitting the software code (MA) to sensor nodes, a large amount of sensory data may be filtered at the source by eliminating the redundancy. MAs may visit a number of sensors and progressively fuse retrieved sensory data, prior to returning to the PE to deliver the data. This scheme proves more efficient than traditional client/server model, wherein row sensory data are transmitted to the PE where data fusion takes place (see Fig. 1).

A number of research articles propose ways for the efficient usage of MAs in the context of DSNs. In particular, MAs have been proposed for enabling dynamically reconfigurable DSNs [8], in multi-resolution data integration and fusion [6], etc. These applications involve the usage of multi-hop MAs visiting large numbers of sensors. The order in which those sensors are visited (i.e. MAs itinerary) is a critical issue, seriously affecting the overall performance. Randomly selected routes may even result in performance worse than that of the conventional client/server model; yet, that issue is not addressed in these works.

To the best of our knowledge, only [7] and [9] deal with the problem of designing optimal MA itineraries in the context of DSNs. In [7], Qi and Wang proposed two heuristic algorithms to optimize the itinerary of MAs performing data fusion tasks. In Local Closest First (LCF) algorithm, each MA starts its route from the PE and searches for the next destination with the shortest distance to its current location. In Global Closest First (GCF) algorithm, MAs also start their itinerary from the PE node and select the node closest to the center of the surveillance region as the next-hop destination.

The output of LCF-like algorithms highly depends on the MAs original location, while the nodes left to be visited last are associated with high migration cost [4] (see, for instance, the last two hops in Fig. 2a); the reason for this is that they search for the next destination among the nodes adjacent to the MA's current location, instead of looking at the 'global' network distance matrix. On the other hand, GCF produces in most cases messier routes than LCF and repetitive MA oscillations around the region center, resulting in long route paths and undesirable performance [7][9].

Wu et al proposed a genetic algorithm-based solution for computing routes for an MA that incrementally fuses the data as it visits the nodes in a DSN [9]. Although providing superior performance (lower cost) than LCF and GCF algorithms, this approach implies a time-expensive optimal itinerary calculation (genetic algorithms typically start their execution with a random solution 'vector' which is improved as the execution progresses), which is unacceptable for time-critical applications, e.g. in target location and tracking. Also, in such applications, the group of visited sensor nodes (i.e. those with maximum detected signal level) is frequently changed over time depending on target's movement; hence, a method that guarantees fast adaptation of MAs itinerary is needed.

Most importantly, both the approaches proposed in [7] and [9] involve the use of a single MA object launched from the PE station that sequentially visits all sensors, regardless of their physical location on the plane. Their performance is satisfactory for small DSNs; however, it deteriorates as the network size grows and the sensor distributions become more complicated. This is because the MA's roundtrip delay increases linearly with network size, while the overall migration cost increases exponentially as the traveling MA accumulates into its state data from visited sensors [3]. The growing MA's state size not only results in increased consumption of the limited wireless bandwidth, but also consumes the limited energy supplies of sensor nodes.

Our algorithm has been designed on the basis of thee objectives: (a) MA itineraries should be derived as fast as possible and adapt quickly to changing networking conditions (hence, an efficient heuristic is needed), (b) MA itineraries should include only sensors with sufficient energy availability and exclude those

with low energy level, (c) The number of MAs involved in the data fusion process should depend on the number and the physical location of the sensors to be visited; the order an MA visits its assigned nodes should be computed in such a way as to minimize the overall migration cost.

3. The Near – Optimal Itinerary Design (NOID) Algorithm

The problem of designing optimal itineraries is similar to traditional network design problems such as the Constrained Minimum Spanning Trees (CMST) problems [4]. In such problems, the objective is the optimal selection of the links connecting terminals to concentrators or directly to the network center, resulting in the minimum possible total cost. The output of CMST algorithms typically comprises topologies partitioned on several multi-point lines (or tree branches), where groups of terminals share a sub-tree to a specific node (center). Since the objective of itinerary planning in DSN environments is to connect groups of sensors with multi-lines (itineraries) all originated at the PE node (center), the similarity with CMST problems becomes evident. Hence, it is reasonable to use algorithms originally devised for CMST problems in the application area of MA itinerary planning. Our NOID (Near – Optimal Itinerary Design) algorithm adapts some basic ideas of Esau-Williams (E-W) algorithm [2] in the requirements of itinerary planning problem.

The cost function used in E-W algorithm considers selected links cost as the only contributing factor to the total itinerary cost. This is certainly not adequate metric to evaluate the cost of agents itineraries c_{total}. A key factor also affecting c_{total} is the agent size; more importantly, the agent size increment rate [3], which depends on the amount of data collected by the MA on every sensor. Let us assume that a set of itineraries $I = \{I_0, I_1, \ldots, I_{k-1}\}$ is constructed, each assigned to an individual MA object i. Each itinerary I_m includes a set of sensors to be sequentially visited by a single MA: $I_m = \{S_0, S_1, .. , S_n, S_0\}$. Note that all itineraries originate and terminate at the PE node S_0. The total cost per polling interval over all itineraries $|I|$ becomes:

$$c_{total} = \sum_{i=0}^{|I|-1} \sum_{j=0}^{|I_i|-1} \left(d_{ij} + s_i \right) \cdot c_{ij} \qquad (1)$$

where d_{ij} is the amount of data collected by the i^{th} MA on the first j visited sensors, s_i the MA initial size and c_{ij} the cost of utilizing the link traversed by the MA i on its j^{th} hop, i.e. the wireless link connecting sensors S_j and S_{j+1} (c_{ij} is given by the network cost matrix). In principle, NOID algorithm aims at constructing a set of itineraries I minimizing the cost function of equation (1).

A comparison among NOID and LCF, GCF heuristics is illustrated in Fig. 2. The algorithms' outputs for the particular DSN configuration of Fig. 2 are based on the cost matrix presented in Table 1. In our prototype implementation, the calculation of the DSN cost matrix entries is only based on the spatial distance between sensors. This decision approach has been taken because the transmission power (hence, energy) required to transmit data between pairs of sensors increases linearly with

their physical distance [1][9]. However, as a future extension, we intend to incorporate sensor energy availability metric in the calculation of cost matrix values.

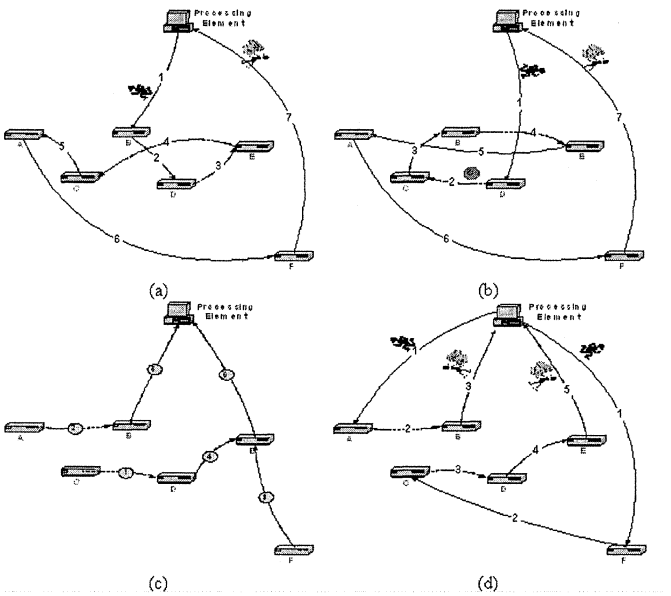

Fig. 2. (a) Output of LCF, (b) Output of GCF, (c) Output of NOID (the sequence numbers indicate the order in which the corresponding MA migrations are accepted, i.e. the algorithm's iteration sequence numbers), (d) MA itineraries derived from the NOID algorithm's output.

The itinerary design algorithm is executed at the PE node; this is a reasonable choice since an MA always starts its data collection journey from the PE node, which can usually be equipped with more powerful computing resources than regular sensor nodes. The PE node has the predetermined knowledge necessary for performing the global optimization, such as the geographical locations (through GPS interfaces) and transmitting/receiving parameters of sensor nodes.

Unlike LCF and GCF algorithms, NOID takes into account the amount of data accumulated by MAs at each visited sensor (without loss of generality, we shall assume this is a constant d). Namely, it recognizes that traveling MAs become 'heavier' while visiting sensors without returning back to the PE to 'unload' their collected data [3]. Therefore, NOID promotes small itineraries enabling the parallel employment of multiple cooperating MAs, each visiting a subset of sensors.

Specifically, the aim of NOID algorithm is, given a set of sensors $S = \{S_0, S_1, \ldots, S_{n-1}\}$, the PE node S_0 and the cost matrix C, to return a set of near-optimal itineraries $I = \{I_0, .., I_k\}$, all originated and terminated at the PE. Initially, we assume $|S|$ $(= n)$ itineraries $I_0, .., I_{n-1}$, as many as the network nodes, each containing a single host (S_0, $S_1, \ldots, S_{n-1}$, respectively). On each algorithm step, two nodes i and j are 'connected' and, as a result, the itineraries $I(i)$ and $I(j)$ including these hosts respectively are merged into a single itinerary.

Table 1. Cost matrix of the DSN shown in Fig. 2.

	S_0	A	B	C	D	E	F
S_0	-	50	40	62	56	42	88
A		-	22	24	58	73	177
B			-	22	21	27	130
C				-	19	39	131
D					-	18	80
E						-	73
F							-

As mentioned in Section 2, LCF and GCF algorithms usually fail as they tend to leave hosts located far from the center stranded since they prioritize the inclusion of hosts closed to last selected host or the center. As a result, relatively expensive links are left last to be included in the solution, significantly increasing the overall cost. A way of dealing with this problem is to pay more attention to nodes far from the center, giving preference to links incident upon them. NOID algorithm accomplishes this by using the concept of 'tradeoff function' ti ,j associated with each link (i, j), defined by:

$$t_{i,j} = c_{i,j} + \sum_{k=1}^{|I(i)|+|I(j)|} d - C_{i,S_0} \tag{2}$$

where $c_{i,j}$ is the cost of link connecting nodes i and j.

The concept of the tradeoff function is introduced in E-W algorithm, defined as follows: $t_{i,j} = c_{i,j} - C_{i,S_0}$. Equation (2) extends and adapts this function in the specific requirements of agent itinerary planning problem. In particular, the inclusion of a parameter representing the amount of data collected from each host (d) and also the number of hosts already included in the itineraries considered for merging, i.e. $|I(i)|$ and $|I(j)|$, obstructs the construction of large itineraries, thereby promoting the formation of multiple itineraries, assigned to separate MAs. Equation (2) implies that the more nodes an itinerary already includes, the more difficult for a new host to become part of that itinerary, especially when d is large.

In equation (2), C_{i,S_0} is the cost of connecting $I(i)$ to the PE S_0. Initially, this is simply the cost of connecting node i directly to the PE. As i becomes part of an itinerary containing other sensors, however, this changes to:

$$C_{i,S_0} = \min_{k \in I(i)} c_{k,S_0} \tag{3}$$

On each algorithm's step, tradeoff function values t_{ij} are evaluated for all pairs (i,j), except of those where nodes i and j are already part of the same itinerary; the 'itineraries' including the nodes that produce the minimum t_{ij} value are merged. For instance, if the tradeoff function is minimized for the pair of nodes m and n, then $I(m)$ and $I(n)$ are merged into one itinerary. When NOID's execution finishes, one or more 'sub-trees' (groups of nodes) rooted at the PE node have been constructed; this is shown on Fig. 2c, where the sequence numbers enclosed within circles indicate the order in which individual links (or migrations) become accepted in the corresponding algorithm steps. It is then a trivial task to produce the itineraries

(started and terminated at the PE node) for traversing the nodes of each sub-tree; these itineraries correspond to a post-order traversal of the sub-trees (shown in Fig. 2d).

4. Quantitative Evaluation

The total costs associated with LCF, GCF and NOID proposed solutions (shown in Fig. 2a, Fig. 2b and Fig. 2d, respectively) are calculated using the generic cost function of equation (1):

$C_{LCF} = s * C_{0,B} + (s + d) * C_{B,D} + (s + 2d) * C_{D,E} + (s + 3d) * C_{E,C} + (s + 4d) * C_{C,A} + (s + 5d) * C_{A,F} + (s + 6d) * C_{F,0}$

$C_{GCF} = s * C_{0,D} + (s + d) * C_{D,C} + (s + 2d) * C_{C,B} + (s + 3d) * C_{B,E} + (s + 4d) * C_{E,A} + (s + 5d) * C_{A,F} + (s + 6d) * C_{F,0}$

$C_{NOID} = [s * C_{0,A} + (s + d) * C_{A,B} + (s + 2d) * C_{B,0}] + [s * C_{0,E} + (s + d) * C_{E,C} + (s + 2d) * C_{C,D} + (s + 3d) * C_{D,F} + (s + 4d) * C_{F,0}]$

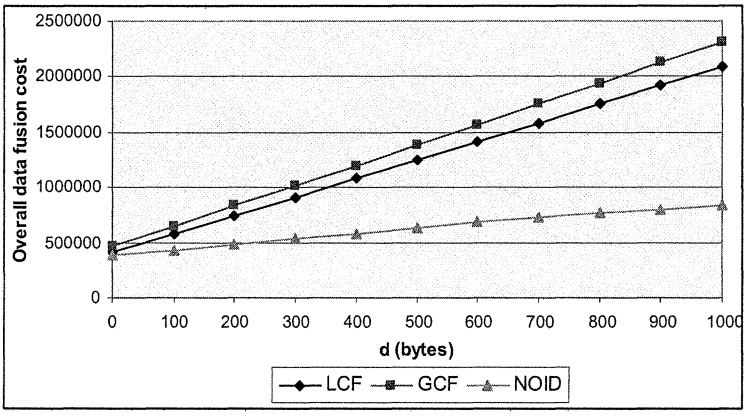

Fig. 3. The overall cost of performing data fusion tasks in the DSN of Fig. 2 by MAs with code size s = 1000 bytes and itineraries derived by LCF, GCF and NOID algorithms.

Assuming an MA of initial size s=1000 bytes that collects an amount of d=100 bytes from each sensor visited and after substituting various costs with the corresponding values found in the cost matrix of Table 1, we get: C_{LCF} = 575300, C_{GCF} = 646900 and C_{NOID} = 430500 cost units (see Fig. 3), i.e. NOID offers cost saving of 25.2% over LCF and 33.5% over GCF algorithm.

Note that unlike LCF and GCF algorithms, NOID proposes different number of itineraries (MAs employed in parallel) depending on the s/d ratio. In particular, as the s/d ratio decreases (the MA accumulates larger amounts of data), NOID algorithm proposes a large number of small itineraries (so that the corresponding MAs do not become too 'heavy') and its performance gain over LCF and GCF improves further. For instance, for s=1000 bytes and d=700 bytes, NOID proposes three itineraries: I_1 = {0,1,2,0}, I_2 = {0,3,4,0}, I_3 = {0,6,5,0}; in that scenario, the

cost saving offered by NOID over LCF and GCF becomes 54,3% and 58,7% respectively.

5. Conclusions and Future Work

In this article we presented NOID, an efficient heuristic algorithm that derives near-optimal itineraries for MAs performing incremental data fusion in DSN environments. Our algorithm considers spatial distance among sensor nodes for constructing MA itineraries and is shown to outperform alternative existing approaches.

At the time these lines were written, NOID algorithm was under evaluation through simulation tests through a Java-based implementation. LCF and GCF will also be implemented for demonstration and comparison purposes. Simulation results will be analyzed to compare the performance of these algorithms in target tracking applications, in large-scale DSNs, in terms of: (a) the overall agents itinerary cost (total itinerary length), (b) mean energy level of the sensors visited by MAs, (c) overall time for completing data fusion tasks.

Acknowledgement

This work has been co-funded by 75% from EU and 25% from the Greek government under the framework of the Education and Initial Vocational Training II, Programme Archimedes.

References

1. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "A survey on sensor networks", IEEE Communications Magazine, pp. 102-114, August 2002.
2. L.R. Esau, K.C. Williams, "On teleprocessing system design. Part II- A method for approximating the optimal network, IBM Systems Journal, 5, 142–147, 1966.
3. A. Fuggeta, G.P. Picco, G. Vigna, "Understanding Code Mobility", IEEE Transactions on Software Engineering 24(5), pp. 346–361, 1998.
4. A.Kershenbaum, "Telecommunications Network Design Algorithms", McGraw-Hill, 1993.
5. Milojicic D., "Mobile agent applications", IEEE Concurrency, 7(3), July-Sep. 1999.
6. H. Qi, S.S. Iyengar, K. Chakrabarty, "Multi-Resolution Data Integration Using Mobile Agents in Distributed Sensor Networks", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Rev., 31(3), pp. 383-391, August 2001.
7. H. Qi, F. Wang, "Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks", Proceedings of the International Conference on Wireless Communications, pp.147-153, 2001.
8. T. Umezawa, I. Satoh, Y. Anzai, "A Mobile Agent-Based Framework for Configurable Sensor Networks", Proceedings of the 4th International Workshop on Mobile Agents for Telecommunications Applications (MATA'02), pp. 128-140, October 2002.
9. Q. Wu, N. Rao, J. Barhen, S. Iyengar, V. Vaishnavi, H. Qi, K. Chakrabarty, "On Computing Mobile Agent Routes for Data Fusion in Distributed Sensor Networks", IEEE Transactions on Knowledge and Data Engineering, 16(6), pp. 740-753, June 2004.