Pervasive and Mobile Computing 5 (2009) 135–147

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

JAID: An algorithm for data fusion and jamming avoidance on distributed sensor networks

Aristides Mpitziopoulos [a,*], Damianos Gavalas [a], Charalampos Konstantopoulos [b], Grammati Pantziou [c]

[a] Department of Cultural Technology and Communication, University of the Aegean, Lesvos, Greece

[b] Research Academic Computer Technology Institute, Patras, Greece

[c] Department of Informatics, Technological Educational Institution of Athens, Athens, Greece

ARTICLE INFO

Article history:
Received 8 October 2007
Received in revised form 6 May 2008
Accepted 15 June 2008
Available online 24 June 2008

Keywords:
Wireless sensor networks
Mobile agents
Jamming avoidance
Routing
Data fusion
Itineraries

ABSTRACT

Mobile Agent (MA) technology has been recently proposed in Wireless Sensor Networks (WSNs) literature to answer the scalability problem of client/server model in data fusion applications. In this paper, we describe the critical role MAs can play in the field of security and robustness of a WSN in addition to data fusion. The design objective of our Jamming Avoidance Itinerary Design (JAID) algorithm is twofold: (a) to calculate near-optimal routes for MAs that incrementally fuse the data as they visit the nodes; (b) in the face of jamming attacks against the WSN, to modify the itineraries of the MAs to bypass the jammed area(s) while not disrupting the efficient data dissemination from working sensors. If the number of jammed nodes is small, JAID only modifies the pre-jamming scheduled itineraries to increase the algorithm's promptness. Otherwise, JAID re-constructs the agent itineraries excluding the jammed area(s). Another important feature of JAID is the suppression of data taken from sensors when the associated successive readings do not vary significantly. Data suppression also occurs when sensors' readings are identical to those of their neighboring sensors. Simulation results confirm that JAID enables retrieval of information from the working sensors of partially jammed WSNs and verifies its performance gain over alternative approaches in data fusion tasks.

1. Introduction

Mobile Agent (MA) technology represents a relatively recent trend in distributed computing, which answers the flexibility and scalability problems of centralized schemes. The term MA [18] refers to an autonomous program with the ability to move from host to host and act on behalf of users towards the completion of an assigned task. A MA may be defined as an entity that comprises a data space to carry collected values and an itinerary which can either be fixed or dynamically determined based on the current network status and the application logic (or execution code) [19]. Lange and Oshima listed seven good reasons to use MAs [12]: reducing network load, overcoming network latency, robust and fault-tolerant performance, etc. The MAs-based computing model enables moving the code (processing) to the data rather than transferring raw data to the processing element.

Although the role of MAs in distributed computing is still being debated mainly due to security concerns [8], several applications have shown clear evidence of benefiting from the use of MAs [15], including e-commerce and m-commerce

* Corresponding author. Tel.: +30 22510 36643.
E-mail addresses: crmaris@aegean.gr (A. Mpitziopoulos), dgavalas@aegean.gr (D. Gavalas), konstant@cti.gr (C. Konstantopoulos), pantziou@teiath.gr (G. Pantziou).

1574-1192/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2008.06.001

trading [24], distributed information retrieval [10], network awareness [2], network & systems management [8,22], etc. Among others, MAs have found a natural fit in the field of Wireless Sensor Networks (WSNs); hence, a significant amount of research has been dedicated to proposing ways for the efficient use of MAs in the context of WSNs. In particular, MAs have been proposed for enabling dynamically reconfigurable WSNs through easy development of adaptive and application-specific software for sensor nodes [28], for separating sensor nodes in clusters [14], in multi-resolution data integration and fusion [4,19] and location tracking of moving objects [3,27]. These applications involve the use of multi-hop MAs visiting large numbers of sensors. The order in which those sensors are visited (i.e. MA itinerary) represents 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, this issue is not adequately addressed in these works.

WSNs applications often include monitoring and recording of sensitive information [1] (e.g. battlefield awareness, secure area monitoring and target detection). Hence, their critical importance raises major security concerns. Jamming is defined as the act of intentionally directing electromagnetic energy towards a communication system to disrupt or prevent signal transmission. In the context of WSNs, jamming is the type of attack which interferes with the radio frequencies used by network nodes [23]. In the event that an attacker uses a rather powerful jamming source, WSN communications will be disrupted. In effect, contemporary WSNs cannot take effective measures against jamming,[1] which raises a major security issue. A notable weakness of the above-mentioned MA-based data dissemination approaches in WSNs is that none takes into account the case that communication of a significant number of sensor nodes is disrupted due to a jamming attack.

The design objective of our Jamming Avoidance Itinerary Design (JAID) algorithm is twofold: (a) to calculate near-optimal routes for MAs that incrementally fuse the data as they visit the nodes; (b) in the face of jamming attacks against the WSN, to modify the itineraries of the MAs to avoid the jammed area(s) while not harming the efficient data dissemination from working sensors.

The first objective is met through the design of a novel algorithm that separates the sensor network into multiple groups of nodes, calculates near-optimal routes (itineraries) through the nodes of each group and assigns these itineraries to individual agent objects. To meet the second objective, the Processing Element (PE) uses the JAM algorithm [29] to map the jammed area(s) and identify the problematic nodes.[2] Next, it executes queries in specific time intervals so as to be informed as soon as they resume function. Assuming that not the entire WSN is affected, the MAs are scheduled so as to bypass the jammed nodes. Instead, they visit nodes close to the jammed area(s) that are not affected in order to avoid the security risk and thus the collapse of the WSN. If the number of jammed nodes is below a specific threshold, JAID only modifies the pre-jamming scheduled itineraries ('connects' the cut-off nodes to jam-free nodes) to increase the algorithm's promptness and minimize the itinerary scheduling cost. Otherwise, JAID re-constructs the agent itineraries excluding the jammed area(s).

Admittedly, jamming avoidance exhibits several common aspects with fault tolerance in the sense that jammed nodes may be perceived by the PE as – temporary – node failures. However, jamming involves specific symptoms and characteristics that differ from those of typical node failures. Jamming may be looked at as a situation that involves massive number of node faults or communication disruptions within a specific network area. On the other hand, sensor node failures due to energy depletion or communication disruptions due to RF unit failures follow a random temporal–spatial distribution pattern.

Another important feature of JAID is the suppression of data taken from sensors when the corresponding readings have spatial/temporal similarities. For instance, an MA may calculate and store the average temperature recorded from each sensor over a monitoring period (temporal data suppression) or the minimum humidity recorded by sensors deployed in an area of interest (spatial data suppression). Data suppression gives JAID the capability to conserve valuable energy resources since the MAs carry smaller amounts of data. Namely, the more load of data the MAs carry the more energy is required for agent migrations (transmission) [30], which seriously affects the overall network lifetime. JAID uses spatial–temporal suppression [25] to reduce communication cost and maximize the network lifetime.[3]

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 and security tasks in WSNs. Simulation results are presented in Section 4, while Section 5 concludes the paper and presents future directions of our work.

[1] A considerable percentage of the nodes deployed in contemporary WSNs are ZigBee [34] and IEEE 802.15.4 [9] compatible and use Direct Sequence Spread Spectrum (DSSS) modulation. However, these protocols have not been originally designed taking radio jamming into account. WSN nodes design also presents the same problem. Other types of widely utilized motes such as Mica-2 [5] are even more susceptible to jamming since they use a single frequency (433 MHz) for communication. The problem of avoiding or defending jamming attacks is very complicated and demands the use of complex and high-cost techniques (e.g. Frequency Hopping Spread Spectrum (FHSS), hybrid FHSS-DSSS, specialized antennas) [16]. The above reasons suggest these techniques as inappropriate for WSNs wherein the node manufacturing cost is a major issue.

[2] Simulation results in [29] in a WSN composed of 121 sensor nodes proved that the mapping activity varies from 1.5 s for moderately-connected networks to just over 5 s for the largest jammed region. This is fast enough to allow a reasonable prompt response to jamming.

[3] Silberstein et al. [25] provide the definitions of spatial and temporal suppression and their possible combination: (a) spatial suppression: a node suppresses its reading if it is identical to those of its neighboring nodes; (b) temporal suppression: if a node's reading is not changed since the last transmission, it does not have to report to the sink (the sink can use its previous reading as the current reading); (c) spatial–temporal suppression: combination of spatial and temporal suppression (if readings do not change, they should not be reported, otherwise, if the relationship between neighboring nodes remains the same, some reports may still be suppressed).

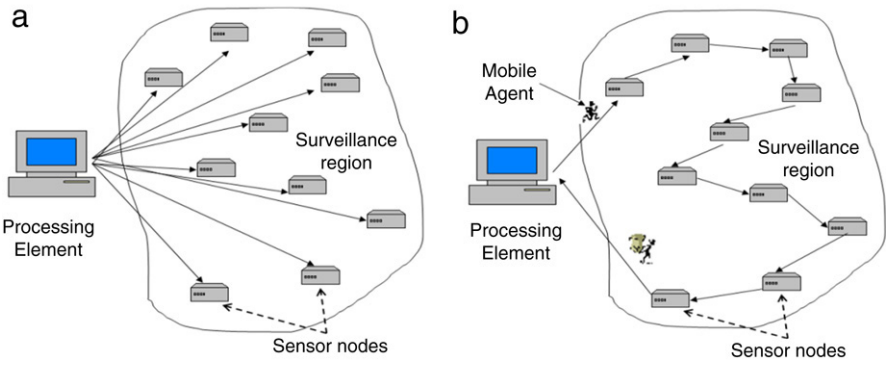


Fig. 1. Centralized vs. Mobile Agents-based data fusion in WSNs.

2. Related work

Data fusion applications in WSN environments often result in generating overwhelming sensory data traffic. To solve the problem of the overwhelming data traffic Qi et al. in [19,20] proposed the MA-based Distributed Sensor Network (MADSN) 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 through eliminating information redundancy. MAs may visit a number of sensors and progressively fuse retrieved sensory data, prior to returning to the PE to deliver their data. This scheme may be more efficient than the traditional client/server model; within the latter model, raw sensory data are transmitted to the PE where data fusion takes place (see Fig. 1). In [21,31] the higher performance of the MADSN over the client/server model is demonstrated through both analytical study and simulations. The shortcoming of these studies is that both assume a constant size for the MA, which is not realistic since MAs grow larger as they collect data from distributed nodes.

To the best of our knowledge, only [20,30] deal with the problem of designing optimal MAs itineraries in the context of WSN. In [20], Qi and Wang proposed two heuristics 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 centre of the surveillance region as the next-hop destination.

The output of LCF-like algorithms though highly depends on the MA original location, while the nodes left to be visited last are typically associated with high migration cost [11] (see, for instance, the last two hops in Fig. 5(a)); 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 centre, resulting in long route paths and unacceptably poor performance [20,30]. 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 WSN [30]. 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 detection and tracking.

Most importantly, both the approaches proposed in [20,30] involve the use of a *single* MA object launched from the PE that sequentially visits all sensors, regardless of their physical location on the plane. Their performance is satisfactory for small WSNs; however, it deteriorates as the network size grows and the sensor distributions become more complicated. This is because both the MAs' round-trip delay and the overall migration cost increase in polynomial order $O(n^2)$ with network size, as the traveling MA accumulates into its state data from visited sensors [7,22]. 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 approach for low-cost MAs itineraries should satisfy three goals. First, MAs itineraries should be derived as fast as possible and adapt quickly to changing networking conditions (hence, an efficient heuristic is needed). Then, 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 as well as the amount of data collected from each sensor. Finally, the order on which an MA visits its assigned nodes should be computed so as to minimize the overall migration cost.

In view of the three above-mentioned objectives, in a previous work we proposed the Near-Optimal Itinerary Design (NOID) algorithm [17]. This algorithm adapts a method usually applied in network design problems, namely the Esau–Williams (E–W) heuristic [6], in the specific requirements of sensor networks. Also, it not only suggests the number of MAs that minimizes the overall data fusion cost, but also constructs near-optimal itineraries for each of them. The key difference with LCF and GCF heuristics is that the NOID algorithm takes into account the amount of data accumulated by MAs at each visited sensor. Namely, NOID recognizes that traveling MAs become 'heavier' while visiting sensors without

returning to the PE to deliver their collected data [7]. Therefore, NOID restricts the number of migrations performed by individual MAs, thereby promoting the parallel employment of multiple cooperating MAs, each visiting a subset of sensors.

In this paper, we propose the Jamming Avoidance Itinerary Design (JAID) algorithm which improves upon the NOID algorithm by following a more direct approach to the problem of low-cost MA itineraries. NOID addresses the problem by reducing it to the problem of the Constrained Minimum Spanning Trees (CMST) problem [11] and then using an adaptation of the Esau–Williams (E–W) algorithm [6]; the cost function that precisely calculates the overall cost of derived itineraries is not used by the E–W algorithm, which instead uses a 'tradeoff' function 'hoping' that the end itinerary output provides a relatively low itinerary cost. In contrast, the JAID directly deals with the problem. Based on a formula that precisely computes the cost of an MA itinerary the JAID algorithm (cost function), it follows a greedy-like approach for choosing how to calculate near-optimal MAs itineraries. As the algorithm progresses, a number of trees of sensor nodes are growing and the traversal of these trees will determine the final MAs itineraries. Simulation results show that our method guarantees the formation of lower-cost itineraries compared to NOID and other heuristics proposed in the literature. Advancing one step further, JAID efficiently deals with small-scale jamming attacks and possible nodes failures omitting problematic nodes from its itineraries and allowing working sensor nodes to report their data to the PE. This gives JAID a great advantage in hostile environments where jamming attacks and interferences are likely to occur and the collection of data from the sensors is a challenging task. At this point we must note that in the case of a large-scale jamming attack where most of the nodes of the WSN are out of order, no algorithmic solution (including JAID) could efficiently defend the network. However, a combination of JAID with specialized sensor hardware (e.g. [16]), would represent a fairly resistant defensive mechanism, even against powerful jammers. Using contemporary sensor node hardware, JAID algorithm provides a viable solution in non-persistent jamming scenarios. When the jamming attacks are persistent though and affect large areas of the networks, the proposed solution may be of limited use since it cannot avoid data losses from the jammed node areas.

A strong aspect of JAID is that MAs possibly executing fusion tasks in a network area that undergoes a short duration jamming attack do not lose the data they carry; this is an advantage derived by the MAs' inherent suitability for disconnected operations [12]. When the jamming attack pauses the MAs can continue their routes back to the PE. Admittedly though, jamming typically does not last for a short time interval. Jamming can be energy efficient as pointed out in [13] and last for relatively prolonged time intervals. Furthermore, the duration of jamming and data collection rate may well exceed the capacity of the sensor on-board storage. Hence, in such a worst-case scenario enough storage will not be available and collected data will be discarded.

As far as security of WSNs is concerned to the best of our knowledge there is no proposed MA scheme that combines data fusion tasks and at the same time deals with possible node failures due to jamming attacks. Wood et al. in [29] propose a Jammed-Area Mapping (JAM) Service for Sensor Networks, yet, without employing MAs. Their basic assumption is that jammed nodes transmit a jammed signal to their neighbors, so that sensory data is forwarded through nodes residing in the perimeter of the jammed region. In our scheme we assume that PE is periodically informed about the jammed area(s) by utilizing the jamming detection and mapping service of JAM algorithm. Furthermore, in our proposed scheme the PE periodically pings the jammed nodes so as to be informed as soon as they resume proper communication.

Other related research works related with jamming attacks in WSNs, can be classified into software-based proposals [13,32,33,35] and into combined hardware/software-based proposals [16]. The former are compatible with existing nodes' hardware, which implies decreased implementation cost, however, defence against jamming attacks is inadequately addressed; the latter proposes new design requirements for sensor nodes along with software countermeasures; it guarantees a satisfactory packet success delivery rate even in heavily jammed environments, yet, it requires high implementation cost. It should be stressed that none of the above-mentioned works involves the use of MAs in jammed environments.

3. The Jamming Avoidance Itinerary Design (JAID) algorithm

3.1. Problem statement

A WSN is represented by a complete graph $G = (V, E)$, $|V| = n$, where each node i, $i = 0, \ldots, n-1$, in V corresponds to a sensor node S_i, $i = 0, \ldots, n-1$, and each edge (i, j) in E corresponds to a communication link between the sensors S_i and S_j. The sensor S_0 corresponds to the PE. Each link (i, j) is associated with a cost $c_{i,j}$, which is a function of the path loss of the link (i, j), (defined as the difference – in dB – between the effective power transmitted by S_i and the power received by S_j and is a function of the physical distance between S_i and S_j) as well as the transmitting power and the signal energy detected by the node S_i. The *Mobile Agent Routing* (MAR) *problem* asks for a path (itinerary) in a WSN, that optimizes a certain routing objective. The overall routing objective is to minimize the overall agent itinerary cost, that is to minimize the energy expenditure (power required for agent transfers) and the path losses [30]. The MAR problem is NP-complete [30] while approximate solutions to the problem are given by heuristic approaches [20,30], as discussed in the previous section.

Let us consider an extension of the problem where given a WSN, with $S = \{S_0, S_1, \ldots, S_{n-1}\}$ its set of sensors and $C = \{c_{i,j} | (i, j) \in E\}$ its cost matrix, instead of one itinerary, we ask for a set of itineraries $I = \{I_0, \ldots, I_k\}$, all originated

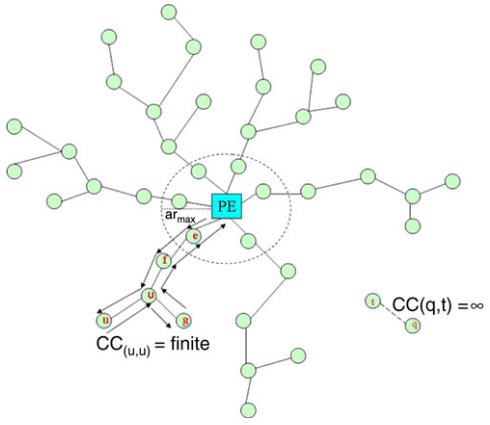


Fig. 2. Itineraries constructed by JAID.

and terminated at the PE (node S_0), such that the sum of costs of the itineraries in I is minimized. The total cost per polling interval over all itineraries is defined as:

$$c_{\text{total}} = \sum_{i=1}^{|I|} \sum_{j=0}^{|I_i|-1} (d_j + s) \cdot c_j \tag{1}$$

where d_j is the amount of data collected by the ith MA on the first j visited sensors, s the MA initial size and c_j the cost of utilizing the link (k, l) traversed by the MA on its jth hop, i.e., the wireless link connecting sensors S_k and S_l ($c_j = c_{k,l}$ is given by the network cost matrix). Therefore, the *extended* MAR *problem* asks for a set of itineraries I minimizing the cost function of Eq. (1).

In our study, we assume that each sensor stores d bytes that should be fused with the data already carried by the MAs. We also assume that this initial amount d is reduced by percent, f, due to data fusion. So, the amount of data collected on the first j visited sensors can be written as $d_j = j \cdot df$, that is $(j-1) \cdot df$ bytes already carried by the agent plus the df bytes collected from the jth sensor after performing data fusion on this sensor. Accordingly, Eq. (1) can be written as:

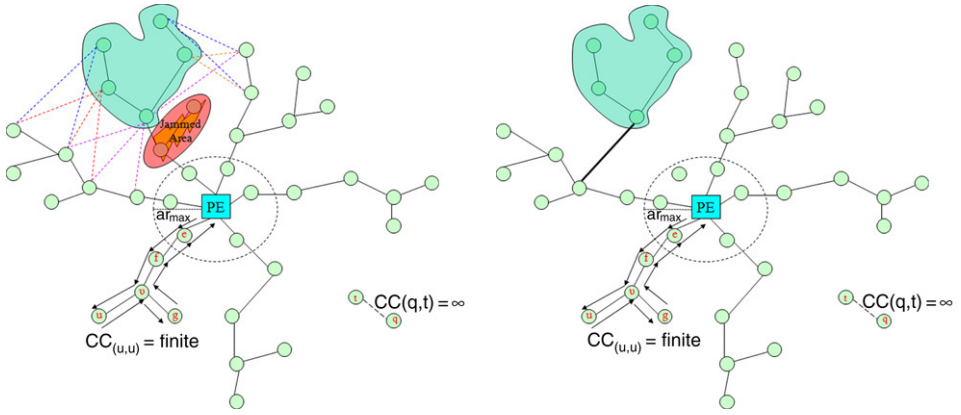
$$c_{\text{total}} = \sum_{i=1}^{|I|} \sum_{j=0}^{|I_i|-1} (j \cdot df + s) \cdot c_j. \tag{2}$$

3.2. The JAID algorithm

Herein, we present the Jamming Avoidance Itinerary Design (JAID) algorithm for determining the number of MAs that should be employed and scheduling their corresponding near-optimal itineraries both in jam-free (in the absence of interference) and jammed WSN environments. JAID's execution comprises three phases. In the initialization phase (see Algorithm 1) we 'connect' the PE with all sensor nodes located within the PE's transmission range Fig. 2. These nodes represent the starting point of the corresponding MA itineraries; namely, the number of these nodes equals the number of MA itineraries.

To dynamically adjust the number of itineraries, we use the parameter $a \cdot r_{\max}$ where a is an input parameter in the range (0, 1] and $r_{\max}$ is the maximum transmission range of the PE. In JAID_Build phase (Algorithm 2), we attach new nodes to the initially formed trees, so as to maintain low itinerary cost post the attachment. The connection cost is calculated in an efficient way, analyzed in Definition 1. The two basic rules that should be taken into account to establish each connection are: (a) the candidate for attachment node must not be already attached to another tree; (b) the candidate node is attached to an in-range node that provides a connecting path back to the PE. The above-mentioned rules guarantee that the itinerary construction begins from the nodes in vicinity to the PE and advances to more distant nodes.

JAID_Repair phase (Algorithm 3) executes in the event of radio interference or jamming attack. First, the mapping of jammed areas (using the JAM algorithm [29]) is completed and cut-off trees with in-range nodes located in the perimeter of the jammed area are identified (Fig. 3(a)). JAID chooses to connection the 'cut-off trees' to in-range nodes so as to minimize the overall cost of the formed itineraries (Fig. 3(b)). In the case that the number of jammed nodes exceeds a threshold (20% of the total number in our simulations) then JAID resumes the JAID_Build phase re-constructing the MA itineraries, ensuring that migration paths do not cross jammed area(s). This threshold can be adjusted according to the WSN deployment area characteristics and the overall number of sensors.



(a) Jammed area and connecting paths for cut-off tree. (b) Selected path for cut-off tree.

Fig. 3. JAID algorithm path selection.

JAID_Repair phase (Algorithm 3) is of low complexity in comparison to the itinerary construction phase (JAID_Build). Hence, it ensures a prompt WSN topology reconfiguration in the face of jamming attacks.

Algorithm 1 (*Initialization phase of JAID*).

PE = the Processing Element; $\mathbf{r}_{max}$= the maximum transmission range of PE;
a = constant between 0 and 1; **N**= the number of sensor nodes;
$\mathbf{SN_i}$ = the i_{st} sensor node ($i - 1 \ldots N$); $\mathbf{c}_{(\mathbf{u},\upsilon)}$ = the edge cost for edges (u, υ);
$\mathbf{Jamd_{SNi}}$ = property of sensor SN_i (**0** if sensor SN_i is not jammed and **1** if it is jammed)
$\mathbf{Attchd_{SNi}}$ = property of sensor SN_i (**0** if sensor SN_i is not connected to a tree with a path leading to the PE and **1** if it is connected)
$\mathbf{Sptemp\_sup_{SNi}}$ = property of sensor i (**0** if the readings of sensor SN_i exhibit no spatial–temporal similarities and **1** if it does)
Nbrnodes = the total number of sensor nodes that exist in the WSN
Total_cost = total cost output of all itineraries of the MAs
{Definition of itineraries}
Z1 = $\{SN_j : d(\text{PE}, SN_j) \leq a \cdot r_{\max}\}$
k = $|Z_1|$ //the total number of itineraries
d = the load given to the agent by each *SN*
s = the size of agent code
f = data volume reduction percent due to data fusion at each *SN*
//set the initial values of the Potential Cost of edges
for all edges (u, υ)**do**
if $SN_u \in Z_1$ and υ = PE **then** {u the node to be connected}$CC(u, \upsilon) = (df + s) \cdot c_{(u,\upsilon)}$
else $CC(u, \upsilon) = \infty$; attchd_u = FALSE
end //for

Algorithm 2 (*JAID_Build*).

for $i = 1$ to Nbrnodes **do** // for all nodes of the WSN
while there exists SN_u with attchd_d = FALSE **do**
$(u_{\min}, \upsilon_{\min})$ = the edge with minimum $CC_{(u,\upsilon)}$ value among all (u, υ) with attchd_u = FALSE and attchd_υ = TRUE
attach $SN_{u_{\min}}$ to the tree of $SN_{\upsilon_{\min}}$; $\text{attchd}_{u_{\min}}$ = TRUE
for all edges (u, w) with attchd_u = FALSE **and** attchd_w = TRUE and SN_w in the tree of $S_{u_{\min}}$**do**
Update $CC_{(u,w)}$
end {for}
end {while}
end {for}
for $i = 1$ **to** k {k = the number of itineraries}**inc** (Total_cost, *cost*(itinerary$_k$)) // calc. the total cost output

Algorithm 3 (*JAID_Repair // Executed in the event of jamming attack*).

Cut-off_nodes $= \phi$
V = the set of all non-jammed sensor nodes
For each Sensor i in V **do**
  if node i is not jammed and its parent in the tree which it belongs to is jammed then
    remove the node i and all the nodes in its subtree from V
    insert these nodes into Cut_off_nodes
  end {if}
end {for}
while |Cut_off_nodes| > 0 **do**
  Find the minimum $CC(i, j)$ where node i in Cutt_off_nodes and node j in V-Cut_off_nodes and let $C(i_{\min}, j_{\min})$ be that minimum
  Attach node $i_{\min}$ and its tree to the node $j_{\min}$
  Remove $i_{\min}$ and all nodes in its tree from the Cut_off_nodes
end {while}
  Total_cost := 0 // initialization of Total_cost
for $i = 1$ **to** k {k = the number of itineraries}**inc** (Total_cost, cost(itinerary$_k$)) //calc. the total cost output

Algorithm 4 (*Update* $CC_{(u,v)}$ *// Connection Cost* $CC_{(u,v)}$ *of edge* (u, v)).

Let $SN_{i_0}, SN_{i_1}, SN_{i_2}, SN_{i_{m-2}}, SN_{i_{m-1}}$ be the post-order traversal of the tree of v after attaching u to v\{$SN_{i_0} = SN_{i_{m-1}}$ = PE} and shortcutting wherever possible; Note that in the first lap SN_{i_0}, SN_{i_1} of the traversal the agent goes down the tree without gathering data till meeting the leaf SN_{i_1}.
$CC_{(u,v)} = \sum_{j=0}^{m-1} s \cdot c_{i_j \cdot i_{j+1}} + \sum_{j=1}^{m-1} (jdf + s) \cdot c_{i_j \cdot i_{j+1}}$

Algorithm 5 (*Cost*(itinerary$_k$)).

Let $SN_{i_0}, SN_{i_1}, SN_{i_2}, \ldots, SN_{i_{m-2}}, SN_{i_{m-1}}$ be the post-traversal of the tree of itinerary k
\{$SN_{i_0} = SN_{i_{m-1}}$ = PE} with shortcutting wherever possible and including only the nodes SN_{ik} with $\mathbf{Sptemp\_sup}_{SN_{ik}} = 0$.
Note that in the first lap SN_{i_0}, SN_{i_2} of the traversal the agent goes down the tree without gathering data till meeting the leaf SN_{i_1}.
Cost(itinerary$_k$) $= \sum_{j=0}^{m-1} s \cdot c_{i_j \cdot i_{j+1}} + \sum_{j=1}^{m-1} (jdf + s) \cdot c_{i_j \cdot i_{j+1}}$

3.3. JAID implementation details

The initialization phase (Algorithm 1) finds the number k of nodes within the radius $a \cdot r_{\max}$ away from the PE. The initialization phase completes with the PE connected to all k nodes. These nodes represent the starting points of the MA itineraries. That also implies that JAID instantiates k MA objects ($k = 6$ in Fig. 3). With the inclusion of parameter α, the number of parallelly employed MAs may be adjusted.

The end result of our method is k trees each rooted at one of the k nodes located within the radius of PE. Then, the itineraries of the MAs are easily derived, by following a post-order traversal of these trees,[4] shortcutting the route whenever possible. Fig. 4(c) illustrates the post-order traversal that complies with the actual structure of the itinerary tree along with our approach (a variant of the post-order traversal that enables route 'shortcutting'). Circular sequence numbers indicate intermediate MA hops on their way to a destination node. Square sequence numbers indicate arrival to a destination node accompanied with retrieval of sensor readings.

Let us now assume that all the WSN nodes are connected to a subtree. We consider all the edges (u, v) that connect node u with node v that has a connecting path back to the PE. We also provide the following definition:

[4] Post-order traversal (for each node v, visit the subtrees rooted at v, then visit v) is more efficient than pre-order (for each node v, visit v, then the subtrees rooted at v) or in-order (for each node v, visit a number of subtrees rooted at v, then visit v, then the rest of the subtrees rooted at v) traversal, as it is shown to derive better total itinerary cost. Specifically, post-order traversal enables the MA to visit distant sensors first and leave sensors located close to the processing element for the end of the itinerary. Hence, the relatively 'expensive' migrations are performed when the MA has not yet collected many data; in the end of their itinerary, when MAs have already accumulated data from the previously visited sensors, they only have to perform short migrations. The cost efficiency of post-order against pre-order or in-order traversals has been experimentally verified through the simulations presented in Section 4.

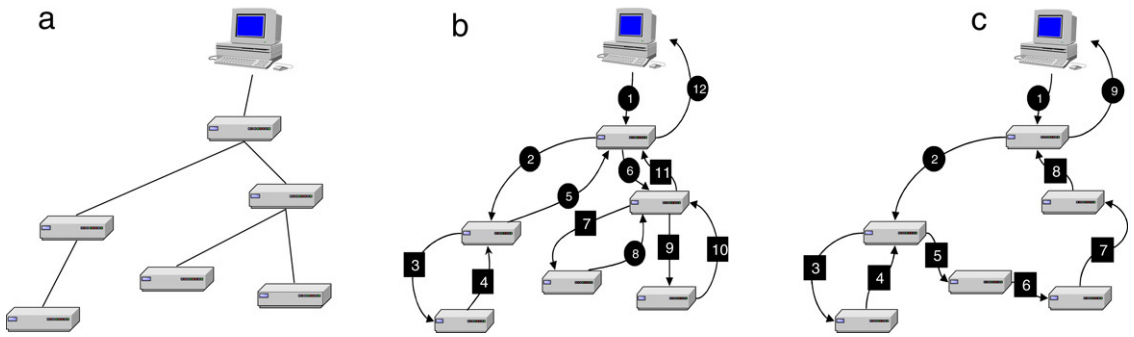


Fig. 4. (a) Itinerary tree constructed by JAID; (b) actual agent itinerary: post-order tree traversal 'complying' with the tree's structure; (c) actual itinerary: post-order tree traversal enabling route 'shortcutting'.

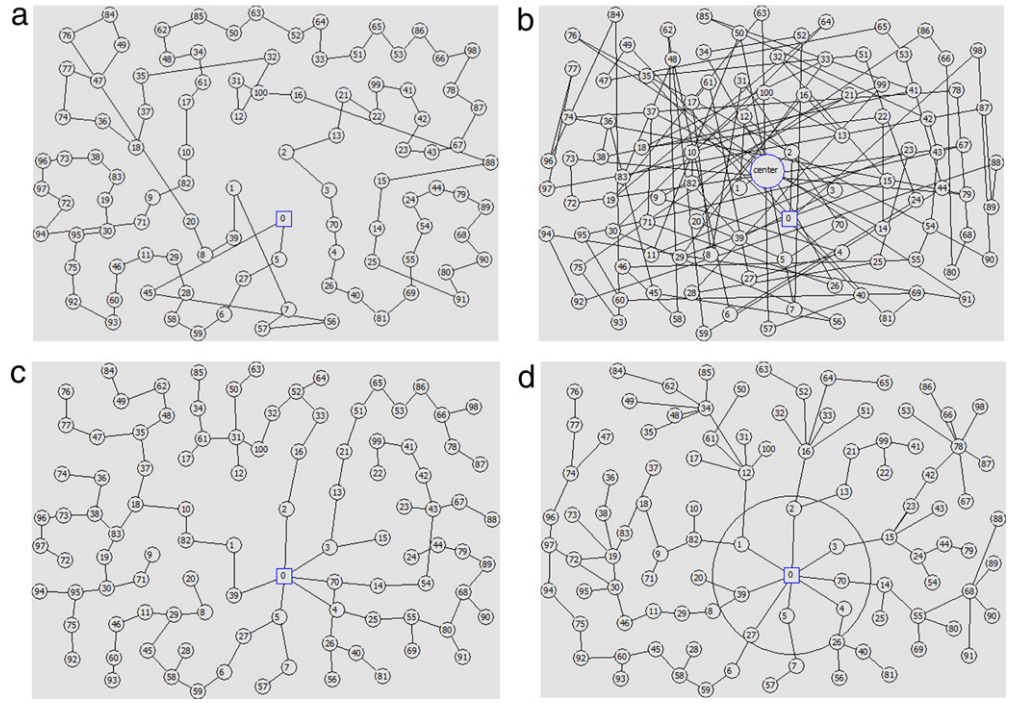


Fig. 5. Delphi-based simulation of MA-based distributed data fusion algorithms: (a) LCF; (b) GCF; (c) NOID; (d) JAID.

Definition 1. The Connection Cost CC of edge (u, υ) is the ensuing itinerary cost after connecting the node u to the tree of node υ provided that υ is in some tree. Otherwise, $CC = \infty$.

In the example of Fig. 2, the CC value of the edge (u, υ) is finite. The itinerary followed by a MA, if the node u is attached to the tree of node υ, is also illustrated. The CC value of the edge is:

$$CC_{(u,\upsilon)} = s \cdot (c_{\mathrm{PE},e} + c_{e,f} + c_{f,\upsilon} + c_{\upsilon,u}) + (df + s) \cdot c_{u,\upsilon} + (df + s) \cdot c_{\upsilon,g} + (2df + s) \cdot c_{g,\upsilon} + (3df + s) \cdot c_{\upsilon,f} + (4df + s) \cdot c_{f,e} + (5df + s) \cdot c_{e,\mathrm{PE}}. \quad (3)$$

$CC_{(u,\upsilon)}$ is equal to the total itinerary cost of nodes e, f, υ, g, u assuming no use of spatial-temporal suppression. As a result, the cost value $C_{(\mathrm{itinerary}_{e,f,\upsilon,g,u})}$ is calculated according to the above formula. The MA visits first the more distant leaf node (u in our example) without collecting any data from intermediate nodes visited throughout its route. When the MA reaches the leaf node, it collects the reported data and on its way back to the PE it collects data from the other nodes. In the last itinerary node (the one closest to the PE) the MA has already grown heavy, since it carries data retrieved from the previously visited nodes; however, the last migration is associated with low cost since its load is only multiplied with the connection cost $(5df + s) \cdot c_{e,\mathrm{PE}}$. This approach drastically minimizes the total cost output.

Note that in the derivation of this expression we have not considered shortcutting the MA's route. Let us also consider the case of edge (q, t) which is associated with an infinite CC value since node t is not yet connected to a tree.

Table 1
Simulation parameters

Parameter	Value
Simulated plane (m^2)	700×450
# sensors	100
Sensor transmission power (dBm)	4
Sensor transmission range (m), assuming clear terrain	10
Network transfer rate (Kbps)	250
Initial sensors' battery lifetime	100 energy units
MA code size (s, in bytes)	1000
Bytes accumulated by the MA at each sensor (d, in bytes)	200[a]
Parameter a (JAID)	1
Data fusion coefficient (f)	1

[a] The 200 bytes might appear as an unrealistic data size since the packet size of contemporary sensor nodes is typically smaller (e.g. 35 bytes in Mica-2 motes [5], 128 bytes in SunSPOTs [26]). It should be noted though that the sensor's reading recording frequency does not necessarily coincide with the MAs polling frequency (in fact the former is typically a multiple of the latter). Besides, a strong asset of the MA paradigm is that MAs may accumulate multiple recordings (data packets) from visited sensors.

Next, the edges are sorted in increasing CC values. If $(u_{\min}, \upsilon_{\min})$ is the edge currently examined, node $u_{\min}$ is connected to the tree of node $\upsilon_{\min}$, with node $\upsilon_{\min}$ being the parent of $u_{\min}$ in the tree. Then, all edges $(u_{\min}, w)$ with neighboring nodes w already connected to a tree are no longer considered. Note that for each edge which connects a node currently not attached to a tree with some node on the tree of node $\upsilon_{\min}$, the CC value is then recalculated, since this cost is modified post the attachment of node $u_{\min}$ to the tree. Furthermore, post the connection of the node to a tree, all edges pointing to this node may change their CC from infinite to a finite value, so they should be reconsidered in the subsequent steps.

After adjusting the CC values, JAID examines the remaining edges until their start vertices are connected with trees that have the minimum itinerary cost among other candidate trees. An edge is removed from the list of the edges to be visited when it connects nodes already connected to trees. JAID continues until all sensor nodes have been attached to a tree. At this point, JAID_Build phase (Algorithm 2) is completed.

JAID_Repair (Algorithm 3) is executed only in the event of a jamming attack. In JAID_Repair we assume that the PE has already identified the non-jammed nodes using the JAM algorithm [29]. The nodes that are not jammed but their parents in the tree build in the JAID_Build phase are jammed nodes, are identified and added to Cut_Off_nodes list.

While Cut_Off_nodes list contains at least one node, we examine the CC values of edges that connect cut-off nodes with non-jammed nodes that have a path to PE (nodes in V-Cut_off_nodes). The cut-off node u with the lowest $CC(u, v)$ value as well as all nodes in its cut-off tree is connected under the node v. Apparently, the node u and all nodes in its cut-off tree have now a path to PE and thus they should be removed from the Cut_Off_nodes list. This procedure is repeated till the Cut_off_nodes list ends up empty.

4. Simulation results

Our simulation tests compare the performance of JAID against NOID, LCF and GCF algorithms in terms of the overall itinerary length and data fusion cost. A second set of simulations demonstrate the responsiveness and effectiveness of JAID algorithm in the face of jamming attacks against a simulated WSN environment. Unless otherwise specified, the parameters used throughout the simulation tests are those shown in Table 1. The simulation results presented herein have been averaged over ten simulation runs (i.e. ten different network topologies).

Simulations have been conducted using a Delphi-based tool, implemented for this purpose. The simulator allows the user-friendly configuration of various simulation parameters and graphically illustrates the output of JAID, NOID, LCF and GCF, while also recording their respective overall itinerary length and data fusion cost.

Fig. 5 illustrates representative screens of our Delphi-based simulator that draw the output of the four MA-based distributed data fusion algorithms. The circle in Fig. 5(b) denotes the network centre. Notably, GCF typically suggests wireless hops among distant sensor nodes which are not within mutual transmission range and, hence, should be routed through intermediate nodes, thereby frequently requiring complex routing decisions and increasing the overall latency and energy consumption. The same usually applies for the last hops of MA itineraries suggested by LCF (which suggests the smaller total itinerary length).

The output of NOID (shown in Fig. 5(c)) involves considerably shorter overall itinerary length than GCF (shown in Fig. 5(b)), yet, larger than that of LCF (shown in Fig. 5(a)). However, the six itineraries of NOID result in smaller overall cost due to smaller amount of data carried by the MAs, since the MAs visit smaller number of nodes. The output of JAID (Fig. 5(d)) involves smaller overall itinerary length compared to NOID, almost equal to that of LCF. However JAID's cost is considerably smaller. This results in reduced energy consumption, which is the most wanted feature in WSN environments.

A first set of simulation experiments compares the performance of LCF, GCF, NOID and JAID algorithms in terms of their total itinerary length. As illustrated in Fig. 6(a), JAID and LCF have the smaller itinerary length, followed by NOID. GCF suggests the longest itinerary because it suggests wireless hops among distant sensor nodes. In Fig. 6(b) we can see that

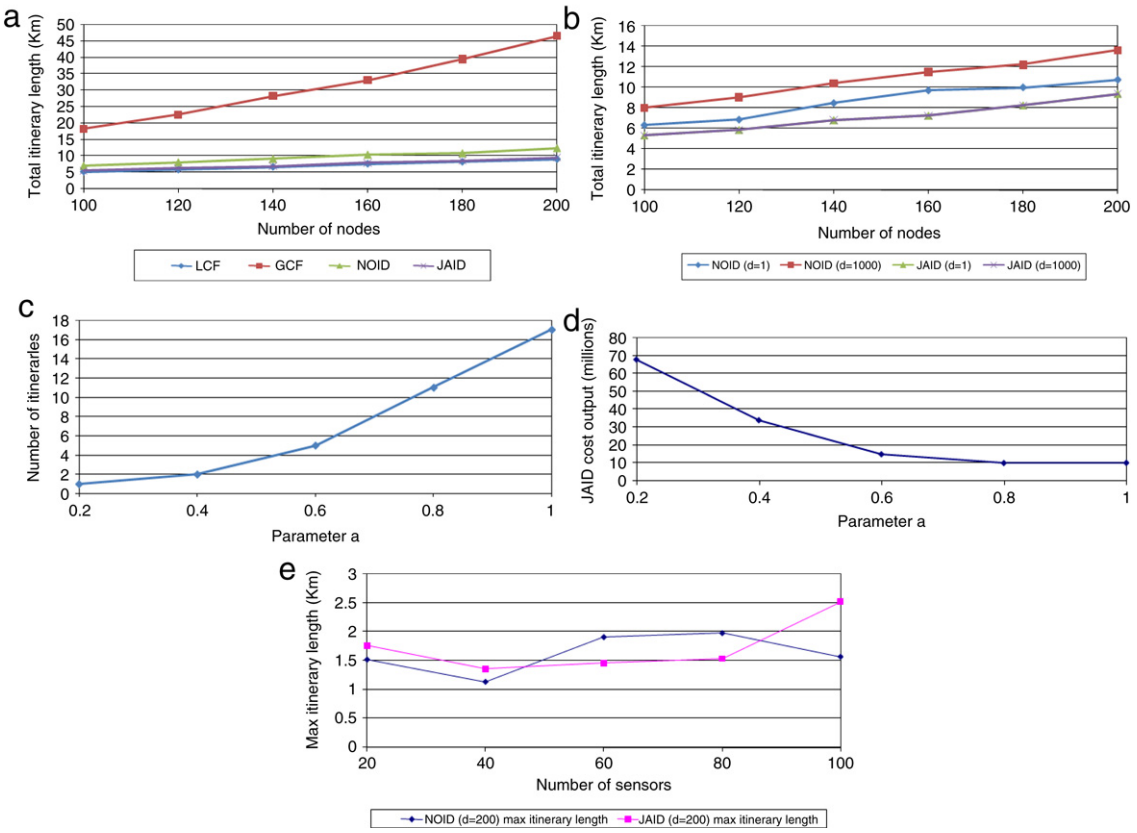


Fig. 6. (a) Total itinerary length of NOID, JAID, LCF, GCF $d = 200$; (b) total itinerary length of NOID, JAID for $d = 1$ and $d = 1000$; (c) number of itineraries suggested by JAID as a function of parameter a; (d) overall data fusion cost of JAID as a function of parameter a; (e) maximum suggested itinerary length of NOID and JAID, where $s = 1000$ bytes, $d = 200$, $a = 1$ (for JAID).

d/s ratio appears to have no effect in the total JAID itinerary length, in contrast to NOID wherein as d/s ratio increases, the total length of NOID itineraries increases remarkably (larger number of MAs cooperate in the data fusion task). The number of itineraries in JAID depends on the parameter a and the PE's transmission range (Fig. 6(c)). In Fig. 6(d) we can see that as parameter a increases, the total cost output of JAID decreases. This is happening because along with the increase of parameter a, the number of MAs increases too, resulting in smaller total cost output (more MAs are working in parallel). Finally Fig. 6(e) illustrates the total length of the longest suggested itinerary of NOID and JAID in simulated WSNs with variable scale. The smaller itinerary length suggested by JAID compared to that of NOID is attributed to the 'articulate' optimization criteria (objective function) of the former. That is, NOID's output includes hops between nodes that do not necessarily lie within mutual transmission range; such hops are performed via intermediate nodes chosen through a simplistic heuristic approach. JAID corrects this drawback, proposing near-optimal itineraries comprising sequential in-range MA hops (shortcutting the post-order tree traversal, wherever possible).

Fig. 7(a) compares LCF, GCF, NOID and JAID algorithms in terms of their respective overall data fusion cost. GCF is shown to suggest the larger overall cost, followed by LCF. NOID performs better than LCF and GCF, while JAID has the smaller cost output among all alternative algorithms. As the number of sensors increases, the cost output difference between JAID, NOID and the other two algorithms (LCF, GCF) increases proportionally. In Fig. 7(b) we compare NOID and JAID for $d = 200$, $d = 500$ for networks up to 100 sensors. As d and the number of sensors increase, the cost output difference increases in favour of JAID. Fig. 7(c) verifies that for a given network size the cost benefit of JAID over NOID increases drastically with the d/s ratio. As d increases the output cost of JAID increases at a slower pace, mainly due to the more sophisticated method that JAID uses to collect data from sensors (as exemplified in Notation 3, the MA visits first the more distant leaf of the itinerary and then on its way to the PE starts the collection of data from the other sensors). As expected, NOID outperforms JAID since the former does not directly derive agent itineraries based on the cost function shown in Eq. (1) (this issue is discussed in depth in Section 3). Finally, Fig. 7(d) compares the total cost output of JAID and NOID with variable number of nodes in the $a \cdot r_{\max}$ area (we set $a = 1$ since NOID does not support the a parameter) from 20% to 100% (in the latter case, all sensor nodes are located within the $a \cdot r_{\max}$ area). From the results we can see that JAID shows an increase in cost output as the number of nodes within the $a \cdot r_{\max}$ space increases. On the contrary the output cost of NOID decreases since this variable does not affect its output.

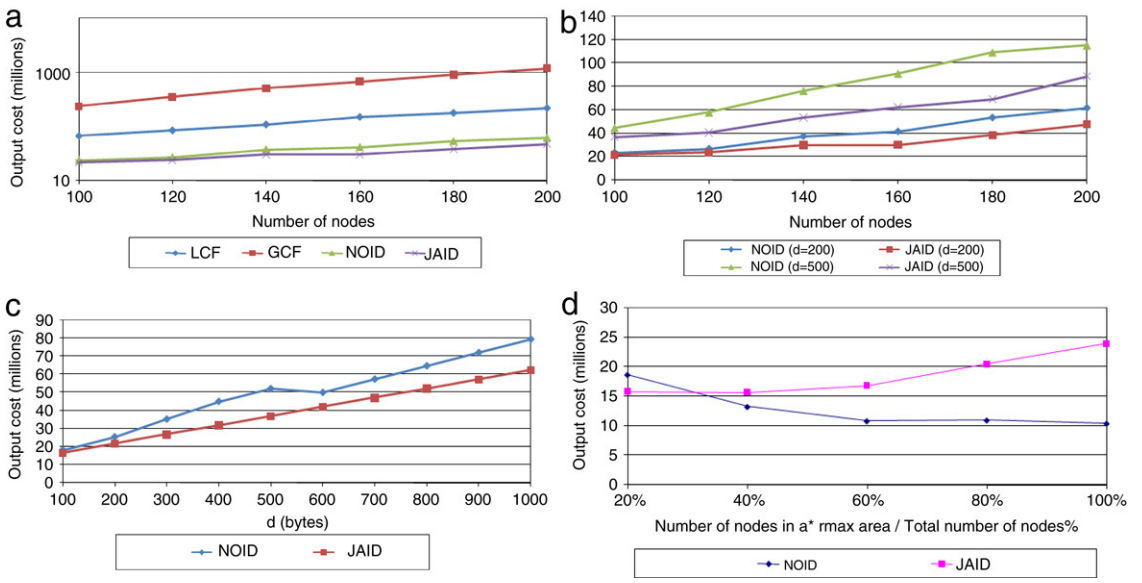


Fig. 7. Comparison of LCF, GCF, NOID and JAID algorithms in terms of their overall data fusion cost for: (a) $s = 1000$, $d = 200$ bytes; (b) comparison of JAID, NOID ($s = 1000$, $d = 200$ bytes or $d = 500$ bytes; (c) $s = 1000$ bytes, network size of 100 sensors; (d) $s = 1000$ bytes, $d = 200$ bytes, 20%–100% of overall number of nodes in $a \cdot r_{\max}$ area.

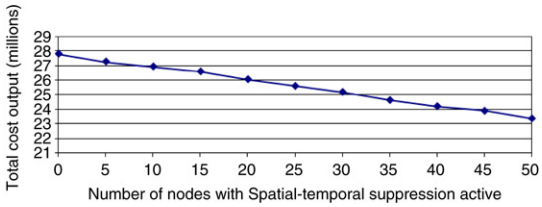


Fig. 8. JAID output cost utilizing spatial–temporal suppression.

The next set of experiments evaluates the overall performance of JAID when utilizing the spatial–temporal suppression scheme. In this case, we examine a WSN of 150 nodes, randomly deployed in the sensor field. As shown in Fig. 8, as the number of nodes that suppress their readings increases the cost output linearly decreases. In the case of 50 nodes with suppressed readings, the cost output is reduced by 16%, implying considerably smaller energy consumption. For example, in environmental monitoring applications (e.g. temperature or moisture monitoring), spatial–temporal suppression represents an asset since the readings of most nodes are likely to remain unchanged for long periods or to be equal to those of neighboring nodes.

Finally, a last set of experiments evaluates the performance of JAID under jamming attacks. We assume 100 nodes randomly deployed in the sensor field. Although we allow multiple jammers, we assume their jamming range is limited enough and the adversaries cannot jam the entire network (no algorithmic solution could effectively defend such attacks). The threshold for the execution of JAID_Repair phase (Algorithm 3) of JAID is set to 20%[5] (in this test, if more than 20 nodes are jammed, JAID re-executes). Fig. 9(a, b) illustrates the itineraries derived by JAID if the number of jammed nodes is under the 20% threshold, while Fig. 9(c, d) presents the itineraries derived when the jammed nodes exceed that threshold.

5. Conclusion and future work

In this article we presented JAID, an efficient heuristic algorithm that derives near-optimal itineraries for MAs performing incremental data fusion in WSN environment. JAID is also able to respond to jamming attacks modifying MA itineraries through a low-complexity update, so as to exclude jammed nodes from the data fusion process. Extensive simulation results have demonstrated the performance gain of JAID against alternative approaches. Furthermore, we provided evidence of JAID's effectiveness and responsiveness even under heavy jamming attacks from multiple jammers.

[5] This threshold has been arbitrarily chosen to support experimentation purposes. Our simulation tests though have demonstrated that a 20% threshold serves as a good balance between avoiding frequent costly re-constructions while preserving the near-optimality of the MA routes.

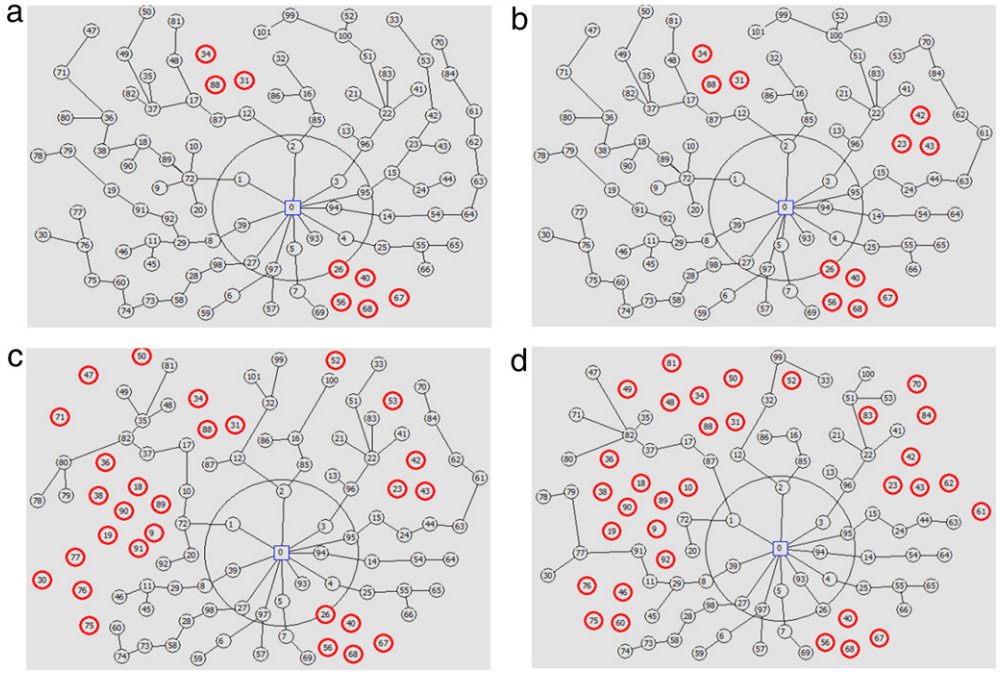


Fig. 9. Delphi-based simulation of JAID algorithm in case of jamming (the thick red circles represent the jammed nodes): (a) 8 jammed nodes; (b) 11 jammed nodes; (c) 28 jammed nodes; (d) 33 jammed nodes.

As a future work, we intend to implement JAID in real WSN environments and test its performance under jamming attacks. A set of sensor nodes capable of hosting and providing an execution environment for MAs programmed in Java [26] will form the basis of our field trials.

References

[1] F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine (August) (2002) 102–114.
[2] A. Al-Hammouri, W. Zhang, R. Buchheit, V. Liberatore, P. Chrysanthis, K. Pruhs, Network awareness and application adaptability, Information Systems and E-Business Management 4 (4) (2006) 399–419.
[3] A. Boulis, Programming sensor networks with mobile agents, in: Proceedings of the 6th International Conference on Mobile Data Management, MDM'2005, May 2005, pp. 252–256.
[4] M. Chen, T. Kwon, Y. Choi, Data dissemination based on mobile agent in wireless sensor networks, in: Proceedings of the 30th IEEE Conference on Local Computer Networks, LCN'05, November 2005, pp. 527–529.
[5] Crossbow Technology Inc. http://www.xbow.com/.
[6] L.R. Esau, K.C. Williams, On teleprocessing system design, part II — A method for approximating the optimal network, IBM Systems Journal 5 (1966) 142–147.
[7] A. Fuggeta, G.P. Picco, G. Vigna, Understanding code mobility, IEEE Transactions on Software Engineering 24 (5) (1998) 346–361.
[8] D. Gavalas, Mobile software agents for network monitoring and performance management, Ph.D. Thesis, University of Essex, UK, July 2001.
[9] J.A. Gutierrez, E.H. Callaway, R. Barrett, IEEE 802.15.4 Low-Rate Wireless Personal Area Networks, ISBN 0-7381-3677-5 SS95127, October 2003.
[10] Y. Jiao, A.R. Hurson, Adaptive power management for mobile agent-based information retrieval, in: Proceedings of the 19th International Conference on Advanced Information Networking and Applications, AINA'05, March 2005, pp. 675–680.
[11] A. Kershenbaum, Telecommunications Network Design Algorithms, McGraw-Hill, 1993.
[12] D.B. Lange, M. Oshima, Seven good reasons for mobile agents, Communications of the ACM 42 (3) (1999) 88–89.
[13] Y.W. Law, L. van Hoesel, J. Doumen, P.H. Hartel, P.J.M. Havinga, Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols, in: Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks, SASN'2005, 2005, pp. 76–88.
[14] M. Lotfinezhad, B. Liang, Energy efficient clustering in sensor networks with mobile agents, in: Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC'05, March 2005.
[15] D. Milojicic, Mobile Agent Applications, IEEE Concurrency 7 (3) (1999).
[16] A. Mpitziopoulos, D. Gavalas, G. Pantziou, C. Konstantopoulos, Defending wireless sensor networks from jamming attacks, in: Proceedings of the 18th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC'2007, September 2007.
[17] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, G. Pantziou, Deriving efficient mobile agent routes in wireless sensor networks with NOID algorithm, in: Proceedings of the 18th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC'2007, September 2007.
[18] V. Pham, A. Karmouch, Mobile software agents: An overview, IEEE Communications Magazine 36 (7) (1998) 26–37.
[19] 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 31 (3) (2001) 383–391.
[20] H. Qi, F. Wang, Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks, in: Proceedings of the13th International Conference on Wireless Communications, Wireless'2001, 2001, pp. 147–153.
[21] H. Qi, Y. Xu, X. Wang, Mobile-agent-based collaborative signal and information processing in sensor networks, Proceedings of the IEEE 91 (8) (2003) 1172–1182.
[22] M.G. Rubinstein, O.C. Duarte, G. Pujolle, Scalability of a mobile agents based network management application, Journal of Communications and Networks 5 (3) (2003).
[23] E. Shi, A. Perrig, Designing secure sensor networks, Wireless Communications Magazine 11 (6) (2004) 38–43.

[24] D.H. Shih, S.Y. Huang, D.C. Yen, A new reverse auction agent system for m-commerce using mobile agents, Computer Standards & Interfaces 27 (4) (2005) 383–395.
[25] A. Silberstein, R. Braynard, J. Yang, Constraint chaining: On energy efficient continuous monitoring in sensor networks, in: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, 2006, pp. 157–168.
[26] Sun Microsystems, Sun Spot project home page. http://www.sunspotworld.com/.
[27] Y.C. Tseng, S.P. Kuo, H.W. Lee, C.F. Huang, Location tracking in a wireless sensor network by mobile agents and its data fusion strategies, Computer Journal 47 (4) (2004) 448–460.
[28] T. Umezawa, I. Satoh, Y. Anzai, A mobile agent-based framework for configurable sensor networks, in: Proceedings of the 4th International Workshop on Mobile Agents for Telecommunications Applications, MATA'02, October 2002, pp. 128–140.
[29] A.D. Wood, J.A. Stankovic, S.H. Son, JAM: A jammed-area mapping service for sensor networks, in: 24th IEEE Real-Time Systems Symposium, RTSS'2003, 2003, pp. 286–297.
[30] 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) (2004) 740–753.
[31] Y. Xu, H. Qi, Distributed computing paradigms for collaborative signal and information processing in sensor networks, Journal of Parallel and Distributed Computing 64 (2004) 945–959.
[32] W. Xu, T. Wood, W. Trappe, Y. Zhang, Channel surfing and spatial retreats: Defenses against wireless denial of service, in: Proceedings of the 2004 ACM Workshop on Wireless Security, WiSe'04, 2004, pp. 80–89.
[33] W. Xu, W. Trappe, Y. Zhang, T. Wood, The feasibility of launching and detecting jamming attacks in wireless networks, in: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2005, 2005, pp. 46–57.
[34] ZigBee Alliance. http://www.zigbee.org.
[35] G. Zhou, T. He, J.A. Stankovic, T.F. Abdelzaher, RID: Radio interference detection in wireless sensor networks, in: Proceedings of IEEE INFOCOM'2005, 2005.

Aristides Mpitziopoulos graduated from Hellenic Air Force Technical NCO Academy in 1996 as a telecommunications engineer. He received his B.E. degree in Culture Technology and Communication in 2006 from the University of Aegean, Mytilene, Hellas. He is an officer in Hellenic Air Force enrolled in Electronic Warfare. He is pursuing his Ph.D. degree at the department of Culture Technology and Communication, Aegean University, Mytilene, Hellas. His main research interests are in electronic warfare, network design, mobile agents, data fusion and security in wireless sensor networks and in wireless networking in general. He is a student member of IEEE.

Damianos Gavalas received his B.Sc. degree in Informatics (Computer Science) from the University of Athens, Greece, in 1995 and his M.Sc. and Ph.D. degrees in electronic engineering from University of Essex, UK in 1997 and 2001, respectively. Currently, he is an Assistant Professor in the Department of Cultural Technology and Communication, University of the Aegean, Greece. He has served as a TPC member in several leading conferences in the field of mobile and wireless communications. He has co-authored over 50 papers published in international journals and conference proceedings. His research interests include distributed computing, mobile code, network and systems management, mobile computing, m-commerce, mobile ad hoc & sensor networks.

Charalampos Konstantopoulos received his Diploma in computer engineering from the Department of Computer Engineering and Informatics at University of Patras, Greece (1993). He also received his Ph.D. degree in Computer Science from the same department in 2000. Currently, he is a Ph.D. researcher at the Research Academic Computer Technology Institute, Patras, Greece. His research interests include parallel and distributed algorithms/architectures, mobile computing and multimedia systems.

Grammati Pantziou received the Diploma in Mathematics and her Ph.D. Degree in Computer Science from the University of Patras, Greece, in 1984 and 1991, respectively. She was a Post-Doctoral Research and Teaching Fellow at the University of Patras (1991–1992), a Research Assistant Professor at Dartmouth College, Hanover, NH, USA (1992–1994), an Assistant Professor at the University of Central Florida, Orlando, FL, USA (1994–1995) and a Senior Researcher at the Computer Technology Institute, Patras (1995–1998). Since September 1998, she has been a Professor at the Department of Informatics of the Technological Educational Institution of Athens, Greece. Her current research interests are in the areas of parallel computing, design and analysis of algorithms, distributed and mobile computing and multimedia systems.