Telecommun Syst (2016) 63:531–545
DOI 10.1007/s11235-016-0140-z

Energy-efficient multiple itinerary planning for mobile agents-based data aggregation in WSNs

Damianos Gavalas[1] · **Ioannis E. Venetis**[2] · **Charalampos Konstantopoulos**[3] · **Grammati Pantziou**[2]

Published online: 3 February 2016

Abstract Data aggregation is recognized as a key method for reducing the amount of network traffic and the energy consumption on wireless sensor network nodes. Mobile agent (MA) technology represents a distributed computing paradigm which has been proposed as a means for increasing the energy efficiency of data aggregation tasks and addressing the scalability problems of centralized methods. Nevertheless, the itineraries followed by travelling MAs largely determine the overall performance of the data aggregation applications. Along this line, this article introduces a novel algorithmic approach for energy-efficient itinerary planning of MAs engaged in data aggregation tasks. Our algorithm adopts an iterated local search approach in deriving the hop sequence of multiple travelling MAs over the deployed source nodes. Simulation results demonstrate the performance gain of our method against existing multiple MA itinerary planning methods.

✉ Damianos Gavalas
dgavalas@aegean.gr

Ioannis E. Venetis
ivenetis@teiath.gr

Charalampos Konstantopoulos
konstant@unipi.gr

Grammati Pantziou
pantziou@teiath.gr

[1] Department of Cultural Technology and Communication, University of the Aegean, Mytilene, Greece

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

[3] Department of Informatics, University of Piraeus, Piraeus, Greece

Keywords Wireless sensor networks · Mobile agents · Itinerary planning · Routing · Optimization · Iterated local search · Data aggregation · Simulation

1 Introduction

Wireless sensor networks (WSN) typically consist of a large number of sensor nodes (SNs) with limited energy availability, which monitor a geographical area and collect sensory information for specific environmental parameters. Sensory data are communicated to a remote base station (sink) through multihop wireless transmissions. To conserve energy, raw data may be aggregated at intermediate sensor nodes by applying an appropriate fusion function [1]. Data fusion combines several sources of sensory data to produce a consistent, accurate and useful knowledge representation [16]. Thus, data fusion reduces the amount of network traffic resulting in considerable energy savings in WSN environments [30].

Mobile agents (MAs) [5] have been proposed as a middleware solution for implementing efficient data aggregation schemes [3,10,21]. A MA is an autonomous program moving from node to node and acting on behalf of the users toward the completion of an assigned task. The software logic is carried with the MA to each SN and determines the processing to be performed on each SN. The resulting data after the local data processing is then embedded within the MA's state and carried to the next SN, where the MA resumes execution and fuses data retrieved thereon. Admittedly, MA-based data aggregation schemes are likely to perform poorly (in terms of network latency) in event-driven WSNs wherein the sink should collect data on demand around an event area and respond fast. However, the MA-based approach suits repetitive data aggregation tasks involving collection of mea-

surements characterized by high spatiotemporal redundancy from a list of SNs known a priori.

The use of MAs for data aggregation requires the determination of SNs visiting order, i.e., an itinerary has to be scheduled. The chosen itinerary largely affects the overall energy consumption and aggregation cost. Long itineraries may lead to significant inefficiencies because built-up data inflate the agent size [22], while their prolonged overall travelling time may be unacceptable for time-critical applications. Besides, careless itinerary planning may require the MA to traverse several intermediate nodes in the process of its migration among pairs of source SNs. Although not collecting sensory data from intermediate nodes, energy resources are consumed only to receive and transmit the passing-by MAs. The criticality of itinerary planning has motivated the design of several algorithms in order to minimize these costs. The literature includes algorithms that either dynamically determine the route of the MA by deciding on the fly the next SN to be visited [10,15] or approximate statically the optimal MA route through heuristics [6–8,14,17,28,29]. A dynamic approach is more suited for target tracking applications, wherein the trajectory of a moving object is initially unknown. Static itineraries are more suitable for data monitoring applications, where measurements of physical quantities (e.g., humidity, temperature, etc.) are periodically gathered and sent to the sink.

Static itinerary planning algorithms are classified into the following categories:

- Single itinerary planning (SIP), wherein only one MA is used for autonomic data aggregation in the sensor network [8,9,29].
- Multiple itinerary planning (MIP) wherein more than one MAs are sent in parallel, each assigned a subset of the nodes of the WSN [6,7,14,17,28].

The performance of SIP algorithms is satisfactory for small-to-medium scale WSNs but it does not scale for networks comprising hundreds or thousands of SNs. This is because both the MA's roundtrip delay and the energy consumption increase fast with network size, as the traveling MA accumulates data from visited sensors [26,27]. On the other hand, MIP algorithms assign shorter itineraries to MAs, mitigating the effects of the SIP algorithms [20]. However, they are more complex to design and execute, as they essentially require to first group SNs into disjoint subsets and then order the nodes of each subset to derive the actual itinerary of employed MAs. Notably, most itinerary planning approaches assume constant MA size throughout the itinerary, disregarding the fact that the data accumulated from visited nodes increase the load carried by the MA on every node. Furthermore, most studies assume the energy cost of each MA transfer to be proportional to the physical distance among its

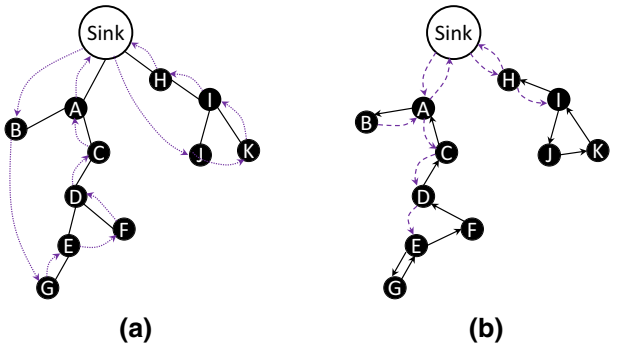

Fig. 1 Example output of a tree-based algorithm: **a** Two trees branches rooted at the sink (*solid lines* denote the three edges while *dotted lines* the post-order traversal of the branches); **b** final itineraries derived from the post-order traversal (*solid lines* denote transfers towards nodes to aggregate sensory data while *dashed lines* denote transfers towards intermediate nodes or returning back to the sink)

end points. However, this assumption does not hold, especially in relatively sparse network topologies, wherein the communication among a pair of nodes (u, v) may require more transmission hops than that for the pair (u', v') even though the in-between distance of the former pair is shorter than that of the latter pair, i.e., $d(u, v) < d(u', v')$.

Among existing MIP algorithms, tree-based approaches [6,14,17] appear to perform better with respect to the most important performance metrics, such as overall response (service) time, total energy expenditure and network lifetime [26]. Those algorithms create tree structures which either expand from the sink towards the periphery of the sensor field [6,17] or vice versa [14]. In all cases, the end result is a tree (rooted at the sink) which includes a path between each source node and the sink. Each branch (rooted at the sink) of the derived tree structure is then transformed to an itinerary (i.e. an ordered list of source nodes) through some sort of tree traversal method (usually post-order traversal). However, the tree traversal negates the benefits of the tree creation process as the ordering of SNs along the itinerary may distort the actual tree construction logic. Thus, SNs placed in successive positions are likely to be far from each other, hence, the corresponding MA transfer will involve multiple hops over intermediate nodes. For instance, Fig. 1 illustrates an example output of a tree-based algorithm, which separates source nodes in two different sets (tree branches). The itineraries associated with the two branches are derived from a post-order tree traversal: {Sink, B, G, E, F, D, C, A, Sink} and {Sink, J, K, I, H, Sink}, respectively. Evidently, nodes B and G are placed successively along the first itinerary, although they are located 5 hops away (they communicate via A, C, D, E). Namely, the actual hop sequence for the MA collecting data from the first 'branch' becomes {Sink, **B**, A, C, D, E, **G**, **E**, **F**, **D**, **C**, **A**, Sink} (bold labels denote visits to source nodes to collect data). Thus, the data collected from node B

are carried along many hops (resulting in unnecessary energy spending), while it would be clearly more appropriate if the visit at B was scheduled in the last itinerary hops.

Herein, we propose a novel MIP algorithmic approach. Our algorithm is based on an Iterated Local Search (ILS), a metaheuristic method commonly used for solving discrete optimization problems [19]. ILS methods iteratively apply a simple modification to a local search routine, each time starting from a different initial configuration, in search of an improved solution.[1] In the sequel, we will refer to our approach with the acronym ILS. Our ILS method addresses all the aforementioned weaknesses of existing MIP algorithms. Firstly, ILS takes into account the growing MA state size along its itinerary. Secondly, it considers the actual energy cost of potential MA transfers taking into account the energy spent for data forwarding on intermediate nodes. Thirdly, ILS does not disassociate the separation of nodes in disjoint sets from their ordering along an itinerary list; rather, it involves a single (main) execution phase wherein nodes are suitably positioned along the itineraries so as to minimize the overall itinerary cost.

It is noted that the advantages of MA-based middleware systems in WSNs have motivated their incorporation on real deployments as evidenced by the proliferation of available prototypes in the recent years. MAPS [2], Agilla [12] and ActorNet [18] are among those MA-based middleware systems tested on a variety of application scenarios such as data collection, gradient search, event tracking, forest fire detection, cargo monitoring, etc. All the above mentioned systems exclusively consider static itineraries. This highlights the significance and applicability of our proposed ILS approach in practical settings, especially when considering repetitive data aggregation tasks.

The remainder of this article is organized as follows: Sect. 2 reviews research related to the work presented herein. Section 3 discusses modeling and implementation details for our proposed ILS algorithm. Section 4 presents simulation results, while Sect. 5 concludes our work and draws directions for future research.

2 Related work

In this section we review existing static MA MIP approaches. It is noted that all the algorithms presented herein are executed centrally, on the sink; the itinerary assigned to each MA is known upon MA instantiation.

The visiting central location (VCL) algorithm [7] adopts a cluster-based approach. VCL uses the notion of the impact factor to designate a set of SNs to act as central points in clusters. The impact factor is analogous to the gravity force measured in real gravity fields, where the gravity impact is quantized by hop count between two nodes (i.e. the impact factor between two source nodes reduces with the number of communication hops among them). In principle, VCLs are located at the center of areas with a high source node density. At each iteration, the SN with the highest impact factor is selected as the next VCL. All the source nodes located within the circular area centered at VCL with a radius of R (which equals the SNs' transmission range) are grouped in a cluster and assigned to an MA. The procedure repeats with the remaining SNs, until all SNs are assigned in a cluster; thereafter, a SIP algorithm is executed to decide the visiting sequence of SNs within each cluster. A tuning parameter σ is used to determine how strong an SN impacts other SNs.[2] As in [7], we set $\sigma = 8$ in this paper.

Another MIP approach is the balanced minimum spanning tree (BST) algorithm [6]. Prim's algorithm is executed to determine a spanning tree, rooted at the sink. Nodes of a single tree branch are considered to belong to the same cluster. A SIP algorithm is then executed to derive the nodes' visiting order within each cluster (in our implementation, we used IEMA [9]). Furthermore, a parameter α is used in BST to balance the weights calculated during Prim's algorithm (large α values tend to assign larger weights to edges among nodes located far from the sink, hence, they disfavor the creation of large tree branches). The idea is to create more balanced branches, i.e., each branch should contain as many SNs as all other branches. In [6] it is shown that $\alpha = 0.6$ is a good choice; therefore, this value is used in our simulation tests.

The EMIP algorithm [28] iteratively partitions a directional sector zone, wherein SNs lying within each sector are grouped together in a separate itinerary. The nodes located within a circular zone around the sink are chosen as the starting points for each itinerary (i.e. the centers of the sector zones); the adjustment of the circular zone's radius determines the number of itineraries. The length of an itinerary is controlled by the angle of the directional sector zone; those angles depend on the nodes density (estimated through a VCL-like technique) and may be different for each itinerary. Upon determining the sectors, a SIP algorithm is used to derive the visiting order within each sector.

The near-optimal itinerary design (NOID) algorithm [14] adapts a method originally designed for network design problems (the Esau–Williams heuristic [11] for the Constrained Minimum Spanning Tree problem), in the specific requirements of WSNs. NOID recognizes that MAs aggregate data

[1] Iterated Local Search is based on building a sequence of locally optimal solutions by: (a) perturbing the current local minimum; (b) applying local search after starting from the modified solution.

[2] The impact factor G_{ij} between two nodes i and j is given by the following equation, with H_j^i denoting the estimated hop count between nodes: $G_{ij} = e^{-\frac{\left(H_j^i - 1\right)^2}{2 \cdot \sigma^2}}$

and grow heavier while visiting SNs. Therefore, it restricts the number of migrations performed by individual MAs. In particular, NOID iteratively adds SNs to separate subtrees, which progressively converge towards the sink. On each iteration, the subtrees which include the pair of nodes (including the sink) that minimize a 'tradeoff' function are merged into one (a subtree structure is finalized when 'merged' with the sink node). Intuitively, the itineraries grow as long as the continuation of an MA's trip is found preferable than the return back to the sink to unload its data; that is, when the amount of data collected from each SN grows or the filtering ratio decreases, NOID derives shorter tree branches. Each subtree is then assigned to a separate MA, which performs a post-order traversal of its subtree so as to visit first (when the MA has not yet accumulated any data) the SNs residing furthest from the sink.

The tree-based itinerary design (TBID) algorithm [17] takes a more direct tree-based approach to the problem of determining low-cost MA itineraries. Essentially, the algorithm determines a spanning forest of trees in the network, calculates efficient tree traversal orders (itineraries) and, eventually, assigns these itineraries to individual MAs. Like NOID, TBID assumes a general aggregation model, wherein data after aggregation does not necessarily have constant size. TBID is built around the idea of co-centric zones with the sink as the center; the first concentric zone's width equals the maximum transmission range $r_{\max}$ of any SN, whereas the subsequent zones' width equals $r_{\max}/2$ (namely, each SN in any zone can only communicate with nodes belonging to the previous, current, and the next outer zones). Similarly to EMIP, the number of itineraries is determined by the number of SNs residing within the first zone and an MA is assigned to each itinerary rooted at those SNs. On each iteration, itineraries grow from the inner towards the outer zones linking the pair of adjacent nodes u and v (where u belongs to an itinerary and v is not yet included) which minimize a 'potential cost' (PC) value; the PC value of an edge equals the cost of the itinerary derived in the case that the edge will be incorporated in the itinerary.

3 The Ils multiple itinerary planning algorithm

3.1 Preliminaries and problem statement

A WSN is represented by a complete graph $G = (V, E)$, $|V| = N$, where each node i in V, $i = 0, ..N-1$, corresponds to an SN SN_i, $(i = 0, ..N-1)$, and each edge $(i, j) \in E$ corresponds to a communication path between the sensors SN_i and SN_j, which can either be direct or multi-hop. The node SN_0 corresponds to the sink. Each link (i, j) is associated with a cost $c_{i,j}$, which is analogous to the overall energy cost required to forward a byte of information among the nodes SN_i and SN_j. The basic Mobile Agent Routing (MAR) problem asks for a single MA itinerary, which includes all the source nodes of a WSN[3] and optimizes a certain routing objective. The routing objective is to minimize the overall energy expenditure associated with the transfers of traveling MAs [29]. The MAR problem is NP-complete [29] while approximate solutions to the problem are given by heuristics [26], as it is discussed in the previous sections.

Now, the problem can be extended in two aspects. First, a more general than the full aggregation model is considered, where the data collected at each SN are reduced only by a certain ratio f $(0 < f < 1)$. Thus, if data of size d is retrieved from each SN, the size of data collected by an MA on the first j visited SNs can be written as $d_j = jdf$, i.e., $(j-1)df$ data elements already carried by the MA plus df data elements derived from processing the raw d elements of the jth SN. A second extension to the MAR problem is to consider multiple MAs and the objective is a set of near-optimal disjoint itineraries (i.e. ordered sets of SNs) $I = I^1, \ldots I^k$, all originated and terminated at the sink (SN_0), such that the total energy cost c_{total} of the itineraries in I per 'polling' round is minimized. The total energy consumption per polling round is defined as follows:

$$c_{total} = \sum_{i=1}^{|I|} IC^i \tag{1}$$

where IC^i is the overall energy cost associated with MA transfers across itinerary I^i:

$$IC^i = \sum_{j=0}^{|I^i|} (jdf + s)c_j \tag{2}$$

where $c_j = c_{k,l}$, i.e., the energy cost of the path (SN_k, SN_l) traversed by the MA on its jth hop and s is the MA's initial size, namely the size of MA code plus the total addressing bits required to store the IDs of SNs included in the itinerary of the MA. It is noted that an MA needs to carry its code only on the first tour; if some functions are routinely executed at each SN, the corresponding code parts are stored at each SN locally on the first visit, and in the next rounds, the MA carries only functionality that may alter between successive rounds [13]. So, s in (2) can be considered constant and relatively small; in fact, in most cases, it can be null [13].

The fixed aggregation ratio f in (2) is a reasonable assumption since detailed information about the correlation

[3] In our implementation we assume that every node i in V is a source node. Nevertheless, our modeling also suits scenarios wherein the source nodes comprise a subset of V. Note that a source node may be visited multiple times. Once to actually retrieve sensory data and the remainder times in the process if migrating from one node to another (in the latter case, the node acts as intermediate node).

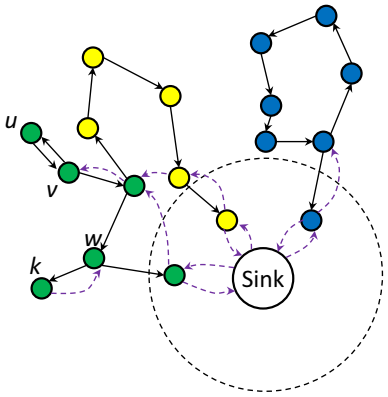

Fig. 2 Illustration of the ILS output; *color* codes are used to indicate the separation of nodes in disjoint itineraries; *black solid lines* denote transfers towards nodes to aggregate sensory data while *blue dashed lines* denote transfers towards intermediate nodes (or returning back to the sink) (Color figure online)

existing in the sensory data for each pair of SNs is usually difficult to obtain. Similar correlation models have been used in [3,8,23]. However, it is important to note that the ILS algorithm can also be fast implemented under a more general correlation model, where f varies among SNs.

3.2 The ILS algorithm

The ILS algorithm is executed centrally at the sink which statically determines the number of MAs that should be used and the itineraries these MAs should follow. Similarly to EMIP and TBID approaches, ILS first determines a zone around the sink, which includes nodes directly reachable (i.e. in single-hop distance) from the sink; those nodes comprise a set Z. The radius of the zone is equal to $a \cdot r_{max}$, where a is an input parameter in the range (0, 1] and r_{max} is the maximum transmission range of any SN (and the sink). The number of nodes in Z determine the number of itineraries k to be derived by ILS ($k = |Z|$). Each SN in Z is the first node to be attached on each of the k itineraries. The initialization phase of ILS is detailed in Algorithm 1.

Following initialization, ILS performs an iterative process which executes until all SNs have been attached to an itinerary (see Fig. 2), considering the k itineraries alternately (to ensure that the constructed itineraries are of equal length). At each step, the algorithm examines the potential attachment of each candidate node SN_u (among those that remain unconnected) at any position along the examined itinerary I^i, namely between any pair of—already attached—subsequent nodes. This is in contrast with the existing itinerary planning algorithms which only consider attaching unconnected nodes at the end of so-far created itineraries (or trees). Among all the candidate (currently unconnected) nodes SN_u and all alternative attachment positions (next to any node $SN_v \in I^i$), ILS chooses to insert the candidate node $SN_{u_{min}}$ next to the node $SN_{v_{min}}$ (executing the *attach* (I^i, $SN_{v_{min}}$, $SN_{u_{min}}$) function), provided that this attachment minimizes the potential overall energy cost of the resulting itinerary $IC^i_{(v,u)}$ (among all alternative attachment options). The energy cost of the itinerary I^i is then updated (executing the $update(I^i, IC^i_{(v_{min},u_{min})})$ function). The above described process is detailed in Algorithm 2.

Algorithm 1. Initialization phase of ILS

1: r_{max} : the maximum transmission range of SNs
2: $a \in (0,1]$
3: $Z = \{SN_j : d(sink, SN_j) \leq a \cdot r_{max}\}$
4: $k = |Z|$ {the total number of itineraries}
5: $i \leftarrow 0$ // auxiliary counter
6: **for each** node $SN_u \in V$ **do**
7: $\quad attached_{SN_u} = FALSE$
8: end for
9: **for each** $SN_u \in Z$ **do**
10: $\quad IC^i_{(v,u)} = (df + s) \cdot c_{v,u}$, where $v \equiv sink$
11: $\quad attach\ (I^i, SN_v, SN_u)$
12: $\quad attached_{SN_u} = TRUE$
13: $\quad i \leftarrow i + 1$
14: end for

Algorithm 2. Main execution phase of ILS
1: $i \leftarrow 0$ // auxiliary counter indicating the itinerary index
2: **while** there exists $SN_u \in V$ with $attached_u = FALSE$ **do**
3: (v_{min}, u_{min}): the edge with minimum $IC^i_{(v,u)}$ value with $SN_{v_{min}} \in I^i$ and $attached_{SN_u} = FALSE$
4: $attach\ (I^i, SN_{v_{min}}, SN_{u_{min}})$
5: $attached_{SN_u} = TRUE$
6: $update(I^i, IC^i_{(v_{min},u_{min})})$
7: $i \leftarrow (i+1)\ mod\ k$
8: end while

An example of the iterative process performed in the main execution phase of ILS is illustrated in Fig. 3. At some step of the process an itinerary I^i includes the nodes $\{sink, u, v\}$ (see Fig. 3a). Based on the formula (2) and assuming that $f = 1$ the current cost of the itinerary is $IC^i = s{\cdot}c_{sink,u}+(s+d)c_{u,v}+(s+2d)c_{v,sink}$. Let k and w denote the sensor nodes being still unconnected (i.e. candidate nodes). On that step, the algorithm considers the attachment of each one of the two candidate nodes at all possible positions along the current itinerary, i.e. between the pairs $(sink, u)$, (u, v) and $(v, sink)$. Figure 3b–d illustrate the solutions derived when inserting node k on each of the potential itinerary positions. Thus, the potential itinerary cost when inserting node k between the *sink* and node u, is $IC^i_{(sink,k)} = s \cdot c_{sink,k} + (s + d)c_{k,u} + (s + 2d)c_{u,v} + (s + 3d)c_{v,sink}$. Likewise, the potential itinerary cost when inserting node k between the nodes u and v, is $IC^i_{(u,k)} = s \cdot c_{sink,u} + (s + d)c_{u,k} + (s + 2d)c_{k,v} + (s + 3d)c_{v,sink}$. Last, the potential itinerary cost when inserting node k between the node v and the *sink*, is $IC^i_{(v,k)} = s \cdot c_{sink,u} + (s + d)c_{u,v} + (s + 2d)c_{v,k} + (s + 3d)c_{k,sink}$. The algorithm could, for example, finally choose to insert the candidate node w between the *sink* and node u iff $IC^i_{(sink,w)} = \min\{IC^i_{(sink,k)}, IC^i_{(u,k)}, IC^i_{(v,k)}, IC^i_{(sink,w)}, IC^i_{(u,w)}, IC^i_{(v,w)}\}$.

ILS initializes the derived itineraries by connecting the sink with each of the nodes located within the zone determined by the sink's transmission range; thereafter, the itineraries progressively grow towards the outer zones of the sensor field iteratively attaching the remaining source nodes. Typically, ILS constructs radial-like itineraries, while its itinerary construction strategy clearly favors the incorporation of adjacent nodes into the same itinerary. Moreover, the consideration of the overall itinerary energy cost in selecting proper node attachments motivates the ordering of itineraries nodes starting from the node furthest form the sink and proceeding towards the node nearest to the sink (note the visiting order in the itineraries of Fig. 2). That ensures that the return transfer back to sink (i.e. the most energy-expensive as the MA is highly loaded) is relatively short.

Upon the conclusion of the ILS main phase, a number of local search perturbations may be applied to better explore the solution space and further improve the originally derived itineraries (intuitively, the fact that candidate nodes are attached to itineraries in a round-robin fashion often allows room for further improvements). For instance, the 2-opt procedure removes two arcs within an itinerary and tries to replace them with two new arcs not previously included in the path (see Fig. 4). If this procedure reduces the itinerary cost, the new solution is accepted, otherwise the original solution is retained. Similarly, the swap perturbation exchanges two nodes included in different itineraries in the hope that the new solution will incur lower overall itinerary cost (see Fig. 5).

The MA itineraries derived by the proposed ILS approach are not modified over consecutive data aggregation rounds, unless a node fails or relocates elsewhere. Such incidents (i.e. node failures due to energy depletion or hardware faults) are detected by travelling MAs which fail to move to failed (or relocated) nodes, hence, interrupting their visit schedule and returning back to the sink to report the node

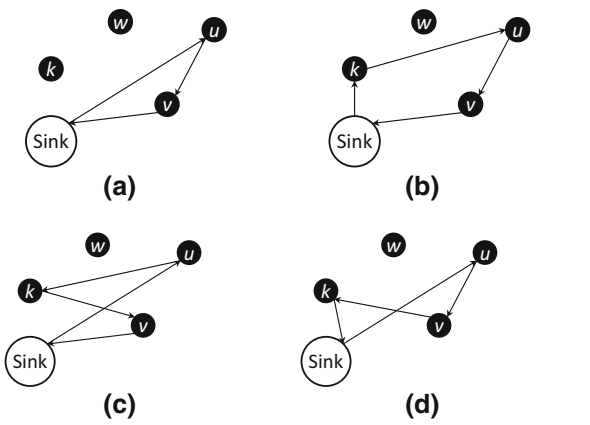

Fig. 3 Illustration of a step of the iterative process performed in the main execution phase of ILS

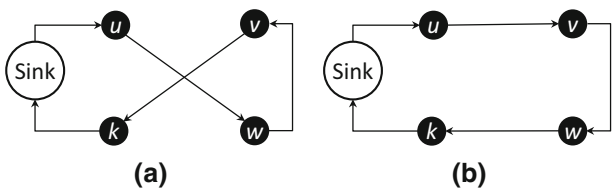

Fig. 4 2-opt perturbation

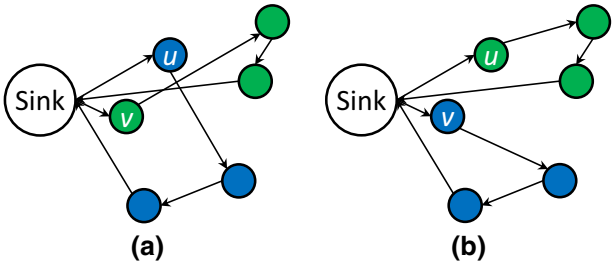

Fig. 5 Swap perturbation; color codes are used to indicate the separation of nodes in disjoint itineraries (Color figure online)

failure. The sink then updates the topology graph removing the failed/relocated node (as well as the edges incident upon it) and re-executes the ILS algorithm deriving new MA itineraries. This process for detecting topology changes causes no extra power consumption and ensures instant adjustment of MA itineraries. Note that similar approaches for addressing fault-tolerance issues in MA transfers have been proposed in the context of IP networks [13].

Figure 6 provides a visual illustration of ILS output (i.e. derived MA itineraries). The figures (generated by a custom Java-based tool, based on output files of the Castalia simulator [4]) present relatively small-scale topologies (200 × 200 m^2, with the sink positioned at the center and 50 SNs randomly deployed in the area), so as to provide a clear overview of the derived output. The blue node (0) represents the sink, whereas black nodes represent the source SNs. Black arrows represent the transitions of MAs from one node to another, whereas green arrows represent the return path (from the last source node) followed by the MA to deliver the acquired data to the sink.

Figure 6a depicts the itineraries calculated by the corresponding algorithm, namely, the order in which source nodes are visited by the MAs. As discussed above, this does not coincide with the actual hop sequence of the MAs since long-distance transfers among source nodes are realized via several intermediate hops (taking into account the maximum RF transceiver transmission range of each intermediate node supported in Castalia). The actual hop sequence is illustrated in Fig. 6b (red arrows indicate migrations towards intermediate nodes while black arrows migrations towards source nodes). The method used to determine the intermediate nodes associated with each transfer among source nodes will be discussed in Sect. 3.3. Figure 6c illustrates an improved solution yield from the 2-opt perturbation. The effect of 2-opt appears,

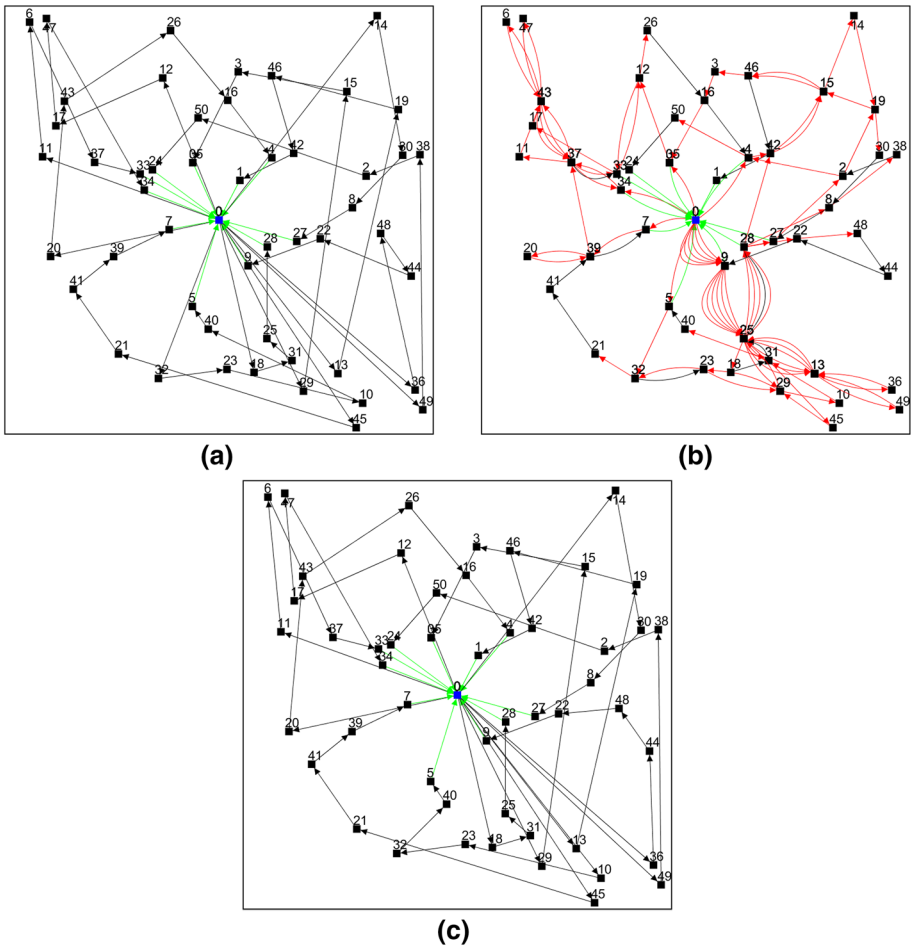

Fig. 6 Example itinerary calculated by the ILS algorithm: **a** without intermediate nodes; **b** with intermediate nodes; **c** improved solution yield from the 2-opt perturbation (Color figure online)

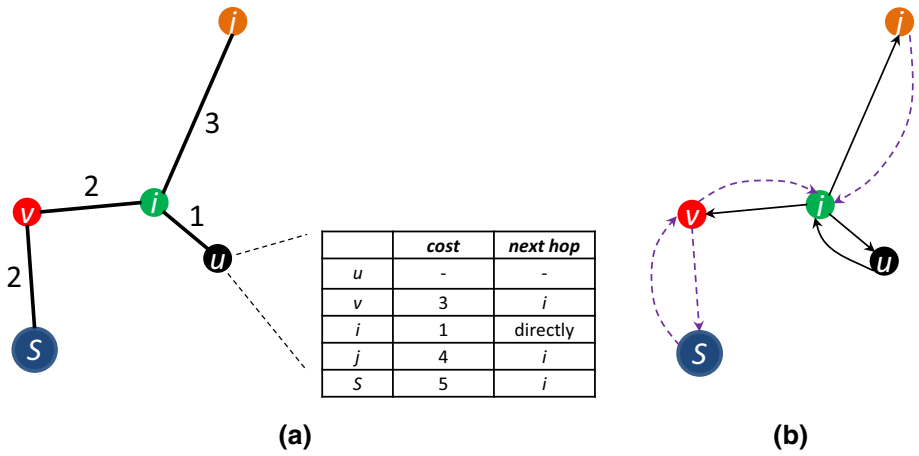

Fig. 7 **a** 'Distance' matrix of node u; **b** itinerary followed by the MA

for instance, in the replacement of the original itinerary {0, 36, 48, 44, 22, 9, 0} (see Fig. 6a) which crosses over itself by {0, 36, 44, 48, 22, 9, 0} which does not (see Fig. 6c).

3.3 Calculation of edge cost

With the exception of TBID, all existing MIP methods make the simplified assumption that the cost of an MA transfer among a pair of nodes is proportional to their physical (Euclidian) distance. However, this assumption does not hold, especially in sparse topologies. In real settings, when considering MA transfers among distant (i.e. out-of-range) nodes, the MA follows a path including several intermediate nodes, ensuring that individual hops are undertaken among in-range nodes. In our implementations, the MA is assigned—upon instantiation—a full path which clearly designates source and intermediate nodes; the former are visited to retrieve sensory data while the latter serve as in-between stops while migrating from a source node to another.

At bootstrap, the sink collects connectivity information; namely, for each node it yields its position and calculates its physical distance from every other node, hence, the power level required to reach each one of its reachable neighbors. Using this information we construct the network topology graph and assign costs to the edges (edges only connect nodes lying within transmission range), where the cost of each edge equals the transmission power required for direct communication. Thereafter, the sink executes Dijkstra's algorithm to construct a cost matrix for each node; for each pair of nodes (i, j), the respective cost matrix entry $c_{i,j}$ indicates the minimum overall (energy) cost associated with transferring one byte among those nodes (assuming symmetric links) as well as the actual communication path, i.e. the intermediate nodes (if any) that should be traversed along the MA transfer. Intuitively, this approach allows ILS to act as if any two nodes in the network can communicate directly, provided that there is a multi-hop path among them. In particular, we execute the ILS steps detailed in Sect. 3.2 where the edge cost taken into account to calculate itineraries' cost is the energy cost given by Dijkstra. Thus, ILS derives MA itineraries (i.e. ordered sets of source nodes) paying no attention to whether each MA transfer is single-hop or multi-hop; then, we yield (from the cost matrix) the path to be followed to realize each transfer among consecutive source nodes. The whole ordered set of source and intermediate nodes is carried by the MA along its journey.

In the example topology of Fig. 7, edges connect nodes which lie within mutual communication range and the edge costs denote the energy units required for transmitting one byte of information among the connected nodes (it depends on the power level required to transmit data). Figure 7a illustrates a representative cost matrix for node u; for instance, the total energy cost for transferring an MA of size d from u to v is $3d$ and the transfer is realized via node i (d energy units are spent by u, while $2d$ energy units are spent by i). An example MA itinerary planned for this particular topology is $\{S, j, u, i, v, S\}$, while the actual hop sequence is $\{S, v, i, j, i, u, i, v, S\}$ (see Fig. 7b).

Notably, in most MIP algorithm implementations, it is assumed that the sink is aware of the absolute geographic location of all source nodes, in order to calculate inter-node physical distances. In real settings, that would imply the integration of a GPS receiver on each SN, thereby significantly increasing the overall deployment cost. In our current implementation the sink becomes aware of the deployed sensors' locations through the information provided by Castalia. To relax this requirement, an alternative implementation could take advantage of the fact that itinerary planning approaches practically require only the distances among nodes, rather than the exact nodes' locations. Thus, SNs could broadcast (at bootstrap) 'hello' messages using their maximum power level. Then, each SN would estimate its physical distance from its neighbor nodes through the RSSI values of received 'hello' messages and communicate this information to the sink.

Table 1 Available transmission levels of the CC2420 radio module, along with their respective power and transmission range

Transmission level (dBm)	0	−1	−3	−5	−7	−10	−15	−25
Transmission power (mW)	57.42	55.18	50.69	46.20	42.24	36.30	32.67	29.04
Transmission distance (m)	46.42	42.17	34.81	28.73	23.71	17.78	11.01	4.22

Table 2 The number of nodes randomly deployed on the simulated area and the number of sensors that report their position to the sink (not isolated)

Nodes deployed	100	200	300	400	500
Reachable nodes	20 (20 %)	160 (80 %)	298 (99.33 %)	399 (99.75)	499 (99.8 %)

4 Experimental results

4.1 Simulation settings

In order to evaluate our algorithm, we have implemented ILS as well as its variants incorporating the 2-opt, 3-opt and swap perturbations. For the rest of this section, we will use the 'plain' ILS algorithm as a reference point along with the ILS followed by the 2-opt heuristic, which gave the best results with respect to the most important metrics, i.e., service time, energy consumption and network lifetime. The above two solutions have been compared against five MIP algorithms, namely BST [6], VCL [7], EMIP [28], TBID [17] and NOID [14], which we have also implemented.

The implementations have been tested on the Castalia [4] simulation platform and the source code is publicly available from github.[4] The size of the simulated terrain has been set to 500 × 500 m^2. We consider network sizes of 100, 200, 300, 400 and 500 nodes. For each network size we have generated 10 random deployments; to ensure fairness, each algorithm has been tested on the same set of deployments. In the results discussed later on, we present the averages over all deployments. The sink is positioned at the center of the simulated terrain.

Each sensor node is assumed to be equipped with a CC2420 chip, i.e. a 2.4 GHz IEEE 802.15.4-compliant RF transceiver. Table 1 presents the eight transmission levels supported by CC2420 along with their respective transmission range (assuming clear terrain). While transferring an MA from one node to another the lowest possible transmission level is used. Further parameters related to energy consumption per node (e.g. receiving data, switching states, sensing, processing, etc) are handled by Castalia.

Table 2 presents the percentage of deployed nodes which are successful in establishing a path towards the sink (to report their connectivity information). Evidently, for topologies of 100 or 200 nodes, a large percentage of nodes appear disconnected, hence, they are not considered in the itinerary planning process.

Table 3 presents the parameters used in our simulations. The power consumed for receiving data and the supported data rate stem from the CC2420 RF module's specifications. The values referring to MA-specific parameters have been based on simulation parameters published in the relevant literature. The filtering coefficient has been set to 1.0; namely, sensory data are assumed to be accumulated without any processing.

4.2 Performance tests

Figure 8a shows the aggregate energy consumption (for sensing, processing and receiving/transmitting data) of all network nodes. This includes energy consumption of intermediate nodes for MA forwarding. Isolated nodes (i.e. those with no path towards the sink) are excluded. Both variants of the ILS (i.e. the 'plain' ILS and the ILS followed by the 2-opt perturbation) perform better than their counterparts, with the one incorporating the 2-opt improvement exhibiting the best results. The differences against TBID are in the range from 8.5 up to 30 %, whereas the differences between the two ILS variants vary from 1 to 4 %. The prevalence of ILS is mainly due to (a) incorporating the real energy cost associated with MA transfers in the cost function used to identify more energy-efficient solutions, (b) using a single execution phase to separate the nodes in disjoint sets and schedule appropriate ordering along the itineraries, unlike tree-based solutions which involve two separate phases, and (c) considering node attachments at any itinerary position, unlike tree-based solutions which only consider connecting candidate nodes with leaf nodes (i.e. at the edge of the so-far created tree). As expected, the 2-opt perturbation improves ILS solutions as the improvement upon the overall itinerary cost is the sole criterion for accepting its perturbed solutions. Nevertheless, this improvement is achieved at the expense of higher computational cost.

Evidently, the performance gain of our approach over TBID increases in topologies of 200 nodes. Notice that ILS pre-calculates the shortest paths between each pair of nodes in a pre-processing phase, therefore, it always considers all nodes in the network for deciding the next node to be included

[4] https://github.com/ivenetis/Castalia-3.2-MIP.

Table 3 Simulation parameters used throughout all conducted experiments

Simulation parameter	Value
Simulated terrain (m^2)	500 × 500
Power consumed while receiving data (per sec, staying in Rx mode)	62 mW
Power consumed while in standby mode	1.4 mW
Network transfer rate	250 Kbps
Initial node energy	18720 J (2 × AA batteries)
Energy consumed for MA execution (data aggregation)	5 nJ
Energy consumed for sensing	0.02 mJ (per sample)
Mobile agent instantiation delay	10 ms
Mobile agent processing delay	50 ms
Code size of MA	1024 bytes
Size of data collected at each node	200 bytes
Filtering coefficient	1.0

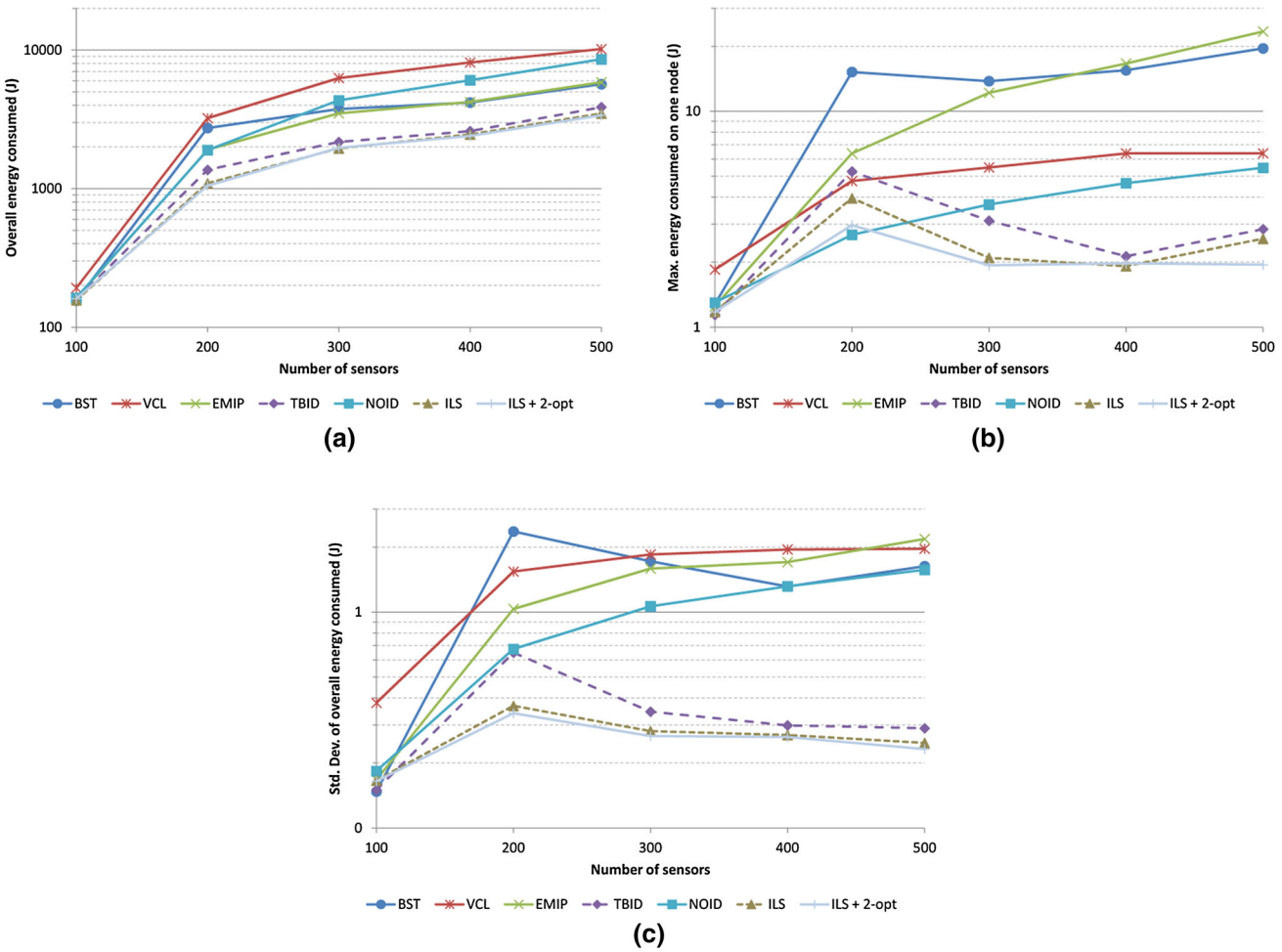

Fig. 8 **a** Overall energy consumed by network nodes throughout a single data aggregation round; **b** energy consumed by the most heavily utilized node among all network nodes included in itineraries; **c** standard deviation of energy consumed among the network nodes

in an existing itinerary. In contrast, on each step TBID only considers the nodes which are adjacent to the current tree structure. However, in the case of 100 nodes, the network is very sparse, hence, the options that TBID and ILS have for extending their itineraries are similar and rather few. For 200 nodes, the network is still sparse but ILS has more possi-

bilities due to the pre-calculated all-to-all shortest paths. For higher number of nodes, TBID starts having more options for the next node to insert in the itineraries; thus, the difference in the performance among ILS and TBID decreases.

Figure 8b illustrates the performance of benchmarked algorithms with regard to network lifetime, i.e., the time that the first node fails due to energy depletion. Since the path derived by the examined algorithms is not modified over consecutive data aggregation rounds (unless a node fails), the same node will always spend the maximum amount of energy (among its peers) and it will be the first to deplete. Again, the ILS variants maintain a clear advantage. This is because ILS attaches new nodes considering the k itineraries in a round-robin fashion, thereby ensuring that the constructed itineraries are of—almost—equal length (i.e. hop count). On the other hand, in the case of unbalanced itineraries, the nodes incorporated in the—relatively—longer itineraries and being closer to the sink suffer from extra data forwarding burden. Inevitably, the energy resources of those nodes deplete sooner (than in the case of balanced itineraries), thus limiting the network lifetime.

Figure 8c presents the degree of dispersion (standard deviation, SD) among the energy consumption values of network nodes. Itineraries associated with high SD energy spending values are those in which subsets of nodes are heavily utilized as intermediate nodes (thereby paying additional energy cost for MA relays) in comparison with the remaining nodes. Both variants of ILS perform better with respect to that metric as they tend to derive itineraries wherein successive nodes are typically within transmission range. Therefore the MAs make little use of intermediate nodes or utilize a low transmission power level, thus ensuring more balanced energy consumption (the intermediate nodes consume more energy for data forwarding than the remaining nodes which are only visited once to retrieve sensory data).

Figure 9 illustrates the time required for travelling MAs to visit all source nodes and deliver the collected data to the sink. As with the energy consumption, this includes the time required to hop over intermediate nodes. The service time is determined by the time elapsed when the first MA is sent to the network, until the last MA returns to the sink. The results indicate that ILS outperforms the rest of the MIP algorithms, again, due to constructing balanced itineraries. Compared to TBID, the differences range from 4 up to 37 %. It is worth noting that these algorithms (i.e. ILS and TBID) maintain a rather stable service time as the network size increases. As the size increases, more sensors lie within the circular zone around the sink. Hence, more itineraries are creating thereby keeping the itinerary length short.

Figure 10a presents the number of itineraries derived by MIP algorithms. VCL and NOID generate the largest number of itineraries. NOID is the only algorithm that explicitly attempts to optimize the length (hence, the number) of itineraries based on the volume of aggregated data and the filtering ratio. In contrast, BST tends to create longer tree branches (the initially created branches tend to 'absorb' all network nodes, not allowing the development of other branches), hence, reducing the number of resulting itineraries. ILS, TBID and EMIP results coincide as they all determine the number of itineraries based on the number of sensors located within the sink's transmission range.

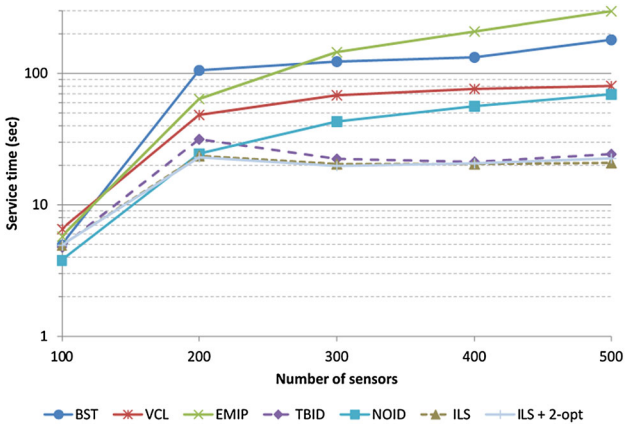

Fig. 9 Time required for MAs to visit the whole set of source nodes

Figure 10b presents the aggregate distance travelled by all travelling MAs, until they return to the sink. Visits to intermediate nodes are taken into account. It is interesting to notice that the distance travelled does not necessarily correlates with the energy consumed. BST and EMIP optimize the travelled distance of MAs, assuming that distance corresponds to the required energy to transfer the MA from one node to another. However, their energy consumption is higher than ILS and TBID (as shown in Fig. 8a). The latter also take into account the size of the MA in estimating the energy cost for each transfer. Further considering that MAs have to be transferred over intermediate nodes, this strategy provides better results.

Figure 11a presents the maximum distance travelled among all MAs (i.e. it considers the longest among the itineraries yield from each algorithm), which is inversely proportional to the total number of itineraries calculated for these algorithms (see Fig. 10).

Figure 11b presents the total number of nodes visited by all MAs, namely the aggregate number of visits on source and intermediate nodes. This metric follows the same trend as the total distance travelled by all MAs (see Fig. 10b). The fact that ILS and TBID require more transfers for the MAs, but consume less energy overall, shows that the transfers are between nearby nodes, hence requiring lower power transmission levels. The selection of neighboring nodes for each transmission can be attributed, again, to the fact that they take into account the size of the MA as well as the energy cost for each transfer.

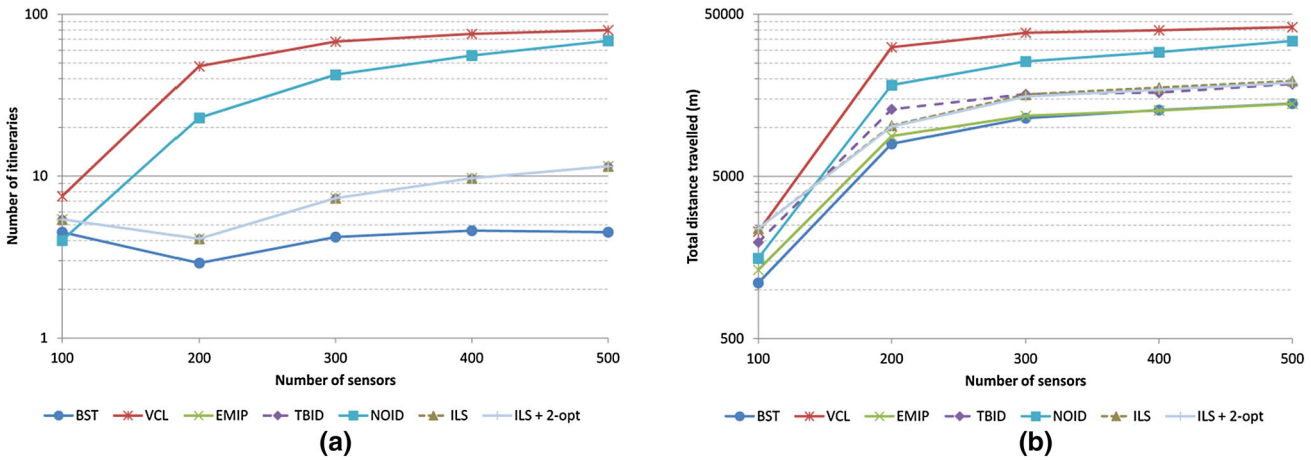

Fig. 10 **a** Total number of itineraries derived by MIP algorithms; **b** total distance travelled by all MAs sent to the network

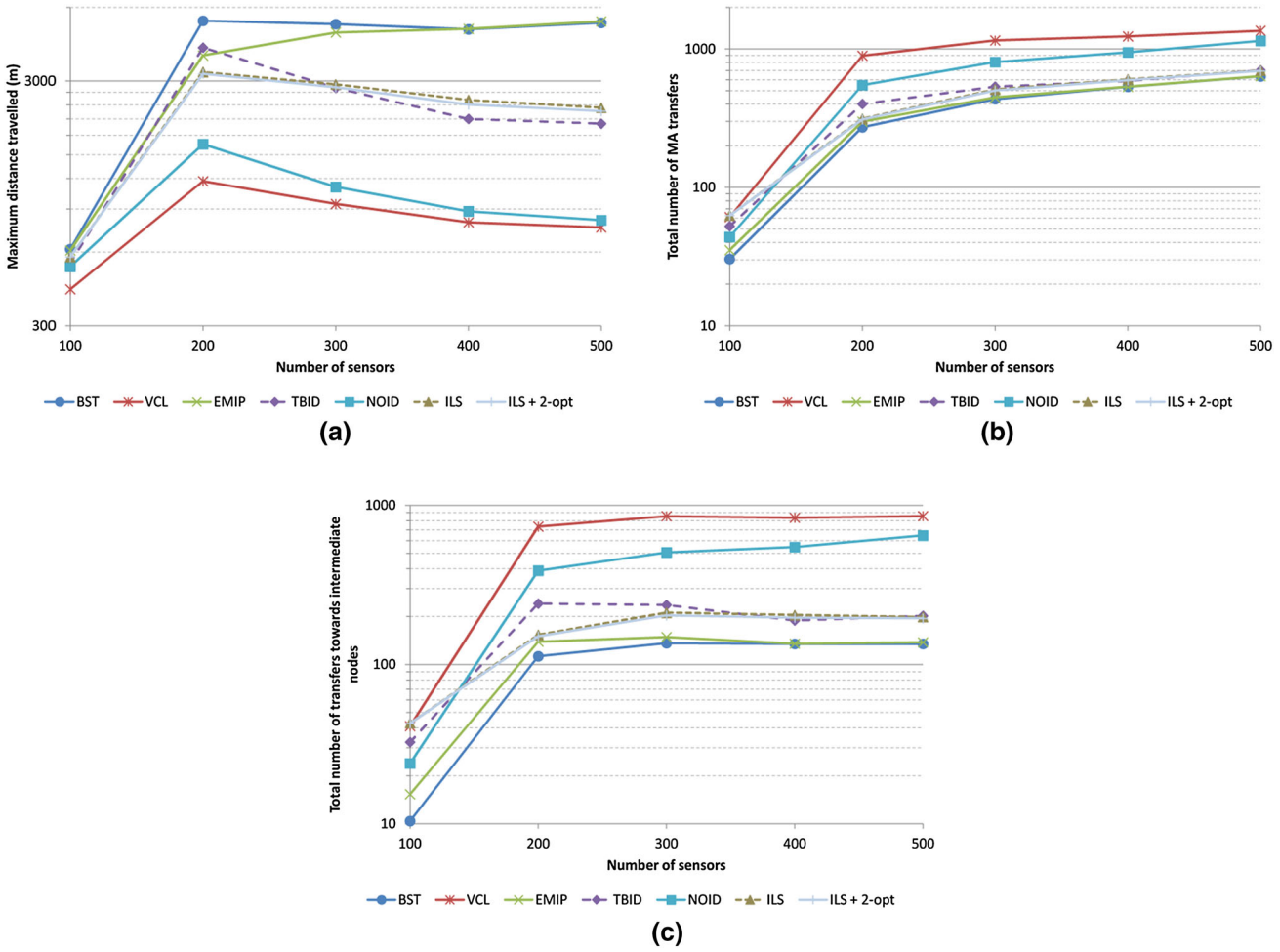

Fig. 11 **a** Maximum distance travelled by a single MA; **b** total number of nodes visited by all MAs sent to the sensor network; **c** total number of hops via intermediate nodes over all itineraries

Figure 11c presents the total number of intermediate nodes visited for all MAs. Although no data is collected at these nodes, energy is consumed to receive the MA and retransmit it to the next node. This metric follows the same trend as the previous one (see Fig. 11b), further strengthening the statement that nearby nodes are selected even when hopping

over intermediate nodes. This result has been expected, since Dijkstra's algorithm calculates the hopping sequence for each MA transfer based on a minimum end-to-end energy cost criterion.

5 Conclusion and future work

In this article we presented ILS, a heuristic that derives multiple near-optimal itineraries for MAs that incrementally aggregate sensory data as they visit SNs in a WSN. ILS suggests a simple, yet effective, approach in MA itinerary planning adopting the principles of iterated local search metaheuristics commonly employed to tackle discrete optimization problems.

ILS possesses some properties, which appear to have positive impact on its performance against alternative itinerary planning strategies:

- It explicitly incorporates the effect of MA size inflation along the itinerary into the utilized cost function.
- It employs an accurate method for estimating the actual energy cost of potential MA transfers taking into account the energy spent for data forwarding on intermediate nodes.
- Candidate nodes may be inserted at any position within an itinerary (i.e. not only on the edge of the itinerary).
- The separation of nodes in disjoint sets and their ordering along an itinerary list are treated together, in a single execution round.
- MAs initiate data dissemination from a node located far from the sink and progressively converge towards the sink ensuring lower overhead for the last itinerary transfers (when the MA is more heavily loaded).

Simulation tests executed in the Castalia simulator demonstrate the performance gain of our method against existing multiple MA itinerary planning methods with respect to the most important performance indicators (overall energy spending, network lifetime and service time).

Our future research plans include the investigation of optimization problems closely related to the MIP. For instance, a promising research direction could be to design methods for the dynamic selection of subsets of SNs to be considered in the itinerary planning process, under strict service time (i.e. itinerary length) constraints. The selection of those SNs could be based on different criteria, e.g., the location and moving direction of a target, the residual energy of SNs, etc. The Orienteering Problem (OP)[5] [25] could serve as a starting point to tackle this challenge. Another research opportunity could be to investigate the use of multiple sinks [24] in data aggregation tasks. The consideration of multiple sinks would involve an additional network partitioning phase which would result in assigning each source node to a particular sink. Thereafter, each sink could perform our proposed ILS approach upon its assigned source nodes.

[5] In the OP the objective is to determine a path, limited in length that visits some vertices and maximizes the sum of the collected scores.

Acknowledgments This research has been co-financed by the European Union (European Social Fund–ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF)—Research Funding Program: Archimedes III. Investing in knowledge society through the European Social Fund.

References

1. Abdul-Salaam, G., Abdullah, A. H., Anisi, M. H., Gani, A., & Alelaiwi, A. (2016). A comparative analysis of energy conservation approaches in hybrid wireless sensor networks data collection protocols. *Telecommunication Systems*, *61*(1), 159–179.
2. Aiello, F., Fortino, G., Gravina, R., & Guerrieri, A. (2011). A Java-based agent platform for programming wireless sensor networks. *The Computer Journal*, *54*(3), 439–454.
3. Biswas, P. K., Qi, H., & Xu, Y. (2008). Mobile-agent-based collaborative sensor fusion. *Information Fusion*, *9*(3), 399–411.
4. Castalia, http://castalia.research.nicta.com.
5. Cao, J., & Das, S. K. (2012). *Mobile agents in networking and distributed computing*. New York: Wiley.
6. Chen, M., Cai, W., González, S. & Leung V.C.M. (2010). Balanced itinerary planning for multiple mobile agents in wireless sensor networks. *Proceedings of the second international conference in ad hoc networks (ADHOCNETS'2010)* (pp. 416–428). Springer, Berlin.
7. Chen, M., González, S., Zhang, Y. & Leung, V.C.M. (2009). Multi-agent itinerary planning for wireless sensor networks. *Proceedings of the IEEE 2009 international conference on heterogeneous networking for quality, reliability, security and robustness (QShine'2009)* (pp. 584–597).
8. Chen, M., Kwon, T., Yuan, Y., Choi, Y., & Leung, V. C. M. (2007). Mobile agent-based directed diffusion in wireless sensor networks. *EURASIP Journal on Applied Signal Processing*, *2007*(1), 219–219.
9. Chen, M., Yang, L. T., Kwon, T., Zhou, L., & Jo, M. (2011). Itinerary planning for energy-efficient agent communications in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, *60*(7), 3290–3299.
10. Dong, M., Ota, K., Yang, L. T., Chang, S., Zhu, H., & Zhou, Z. (2014). Mobile agent-based energy-aware and user-centric data collection in wireless sensor networks. *Computer Networks*, *74*, 58–70.
11. Esau, L., & Williams, K. (1966). On teleprocessing system design, Part II—a method for approximating the optimal network. *IBM Systems Journal*, *5*(3), 142–147.
12. Fok, C. L., Roman, G. C., & Lu, C. (2009). Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, *4*(3), 16.
13. Gavalas, D. (2001). *Mobile software agents for network monitoring and performance management*. PhD Thesis, University of Essex.
14. Gavalas, D., Mpitziopoulos, A., Pantziou, G., & Konstantopoulos, C. (2010). An approach for near-optimal distributed data fusion in wireless sensor networks. *Wireless Networks*, *16*(5), 1407–1425.

15. Gupta, G. P., Misra, M., & Garg, K. (2014). Energy and trust aware mobile agent migration protocol for data aggregation in wireless sensor networks. *Journal of Network and Computer Applications*, *41*, 300–311.
16. Khaleghi, B., Khamis, A., Karray, F. O., & Razavi, S. N. (2013). Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, *14*(1), 28–44.
17. Konstantopoulos, C., Mpitziopoulos, A., Gavalas, D., & Pantziou, G. (2010). Effective determination of mobile agent itineraries for data fusion tasks on sensor networks. *IEEE Transactions on Knowledge and Data Engineering*, *22*(12), 1679–1693.
18. Kwon, Y., Mechitov, K. & Agha, G. (2014). Design and implementation of a mobile actor platform for wireless sensor networks. In: *Concurrent objects and beyond* (pp. 276–316). Springer, Berlin.
19. Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. *Handbook of Metaheuristics* (pp. 320–353). Springer, Berlin.
20. Mercadal, E., Vidueira, C., Sreenan, C. J., & Borrell, J. (2013). Improving the dynamism of mobile agent applications in wireless sensor networks through separate itineraries. *Computer Communications*, *36*(9), 1011–1023.
21. Mpitziopoulos, A., Gavalas, D., Konstantopoulos C. & Pantziou, G. (2009). Mobile agent middleware for autonomic data fusion in wireless sensor networks. In: Denko, M.K., Yang, L.T., & Zhang, Y. (eds.), *Autonomic computing and networking*. Chapter 3 (pp. 57–81). Springer, Berlin.
22. Paul, T. & Stanley, K. G. (2014). Data collection from wireless sensor networks using a hybrid mobile agent-based approach. *Proceedings of the 2014 IEEE 39th conference on local computer networks (LCN'2014)* (pp. 288–295).
23. Shakshuki, E., Malik, H., & Denko, M. K. (2008). Software agent-based directed diffusion in wireless sensor network. *Telecommunication Systems*, *38*(3–4), 161–174.
24. Shim, Y. & Kim, Y. (2014). Data aggregation with multiple sinks in information-centric wireless sensor network. *Proceedings of the 2014 international conference on information networking (ICOIN'2014)* (pp. 13–17).
25. Vansteenwegen, P., Souffriau, W., & Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, *209*(1), 1–10.
26. Venetis, I. E., Pantziou, G., Gavalas, D. & Konstantopoulos, C. (2014). Benchmarking mobile agent itinerary planning algorithms for data aggregation on WSNs. *Proceedings of the 6th international conference on ubiquitous and future networks (ICUFN'2014)* (pp. 105–110).
27. Wang, X., Chen, M., Kwon, T., & Chao, H. C. (2011). Multiple mobile agents' itinerary planning in wireless sensor networks: Survey and evaluation. *IET Communications*, *5*(12), 1769–1776.
28. Wang, J., Zhang, Y., Cheng, Z., & Zhu, X. (2015). EMIP: Energy-efficient itinerary planning for multiple mobile agents in wireless sensor network. *Telecommunication Systems*, in press.
29. Wu, Q., Rao, N. S., Barhen, J., Iyenger, S. S., Vaishnavi, V. K., Qi, H., et al. (2004). On computing mobile agent routes for data fusion in distributed sensor networks. *IEEE Transactions on Knowledge and Data Engineering*, *16*(6), 740–753.
30. Yadav, S. S., Chitra, A., & Deepika, C. L. (2015). Reviewing the process of data fusion in wireless sensor network: A brief survey. *International Journal of Wireless and Mobile Computing*, *8*(2), 130–140.
31. Zaslavsky, A. (2004). Mobile agents: Can they assist with context awareness?. *Proceedings of the 2013 IEEE 14th international conference on mobile data management (MDM'13)* (pp. 304–304).

Damianos Gavalas received his BSc degree in Informatics from the University of Athens, Greece, in 1995 and his MSc and PhD degree in Electronic Engineering from University of Essex, U.K., in 1997 and 2001, respectively. Currently, he is an Associate Professor in the Department of Cultural Technology and Communication, University of the Aegean, Greece. He has served as TPC member in several leading conferences and as an editorial board member of Journal of Network and Computer Applications and Personal and Ubiquitous Computing. He has co-authored over 130 papers published in international journals and conference proceedings. His has participated in numerous research projects funded by Greek national and European funds. His research interests currently include mobile & pervasive computing, wireless sensor networks and optimization algorithms.

Ioannis E. Venetis received his diploma degree in Computer Engineering from the Department of Computer Engineering and Informatics at the University of Patras, Greece, in 1998, and his PhD degree from the same department in 2006. Currently, he is Adjunct Professor at the above department, as well as at the Department of Informatics of the University of Piraeus, Greece. He participates in European and National research projects as an external Post-Doctoral Researcher at the Department of Computer Engineering and Informatics at the University of Patras, Greece and at the Department of Informatics of the Technical Educational Institute (TEI) of Athens, Greece. His research interests include programming models and associated run-time systems for parallel architectures, accelerator based computing, parallelization of applications and optimization for diverse parallel architectures, mobile computing and sensor networks.

Charalampos Konstantopoulos received the diploma degree in Computer Engineering from the Department of Computer Engineering and Informatics at the University of Patras, Greece, in 1993, and the Ph.D. degree in Computer Science from the same department in 2000. Currently, he is an Assistant Professor at the Department of Informatics of the University of Piraeus, Greece. His research interests include parallel and distributed algorithms, mobile computing, sensor networks, as well as algorithmic techniques for optimization problems. He is also currently a Senior Research Associate of the

Research Academic Computer Technology Institute (RA-CTI). He has published more than 70 articles in major international computer science journals and conference proceedings. He has served as a technical program committee member in several international conferences, and as a guest editor of three journal special issues. His research was funded by the European Community and the Greek State. He has participated as a researcher in several research and development projects.

Grammati Pantziou received her Ph.D. degree in Computer Science from the University of Patras, Greece, in 1991. She was a researcher with the Computer Technology Institute, Patras, Greece (1985-1992), a Research Assistant Professor at Dartmouth College, NH, USA (1992-1994), an Assistant Professor at the University of Central Florida, USA (1994-1995), and a Senior Researcher at the Computer Technology Institute (1995-1998). From 1998, she is a Professor at the Department of Informatics of the Technological Educational Institute of Athens, Greece. She served as chair of the Department from 2003 until 2008. Her research interests include parallel and distributed computing, optimization algorithms, and wireless sensor networks. She has co-authored over 130 research articles and has participated in more than 20 research projects funded by the European Community, the Greek State, and the National Science Foundation, USA.