Experiments with Biologically-Inspired Methods for Service Assignment in Wireless Sensor Networks

Tales Heimfarth and Peter Janacik

Abstract Given the scarcity of energy in wireless sensor networks (WSNs), in-network data processing by distributed, cooperating services is often used to reduce the amount of information that has to be routed to the base station and thereby to reduce communication and energy consumption. However, to minimize the amount of communication between services and their requesters, the locations of services in the network have to be selected carefully. Therefore, this paper proposes an efficient biologically-inspired heuristic for service assignment in WSNs. In order to reduce the amount of information exchange necessary for our heuristic, we use a concept observed in ant colonies that utilizes only local information. We model packets as ants (depositing pheromones at the visited nodes), services as food sources and requesters as formicaries. To optimize an objective function (reduction of communication distance between services and requesters), an explorer agent makes local service assignment decisions based on solely local information: the pheromones deposited by the ants. Furthermore, our paper presents the formal definition of the problem of service assignment and a thorough analysis and discussion of the results of our experiments, which show the efficiency of our approach.

1 Introduction

Wireless sensor networks (WSN) consist of a large number of embedded sensors connected via wireless links that are deployed in the monitored environment. Each node in such a network is equipped with a small processor, constrained memory, a set of sensors and, in some application examples, actuators. One key point of such

Tales Heimfarth
Federal University of Rio Grande do Sul, Brazil, e-mail: theimfarth@inf.ufrgs.br

Peter Janacik
University of Paderborn, Germany, e-mail: pjanacik@uni-paderborn.de

networks is the energy efficiency: since it is not feasible to replace batteries after deployment, the energy must be carefully managed in order to increase the life time of the system.

In-network data processing techniques have been successfully employed to improve considerably the energy efficiency of the network. Instead of letting each single node send its raw sensor data to the base station, which incurs a high amount of multi-hop traffic, the idea is to process the data locally in order to compute a higher level result that will be transmitted to the base station. Since nodes are only equipped with a very constrained hardware, this in-network processing, carried out by cooperating services, is distributed among neighboring nodes. Therefore, there is the need for an adequate abstraction implemented by the operating system (OS), which offers the functionality of dynamic service re-assignment to the application. This means that the OS should control the migration of the services.

We developed NanoOS [5], an OS for wireless sensor networks with the aim of supporting collaborative processing. In this work, we formalize the problem of allocating the mobile services to nodes of our network with the objective of reducing the communication overhead. Given a network topology graph and a task/service interaction graph, we aim to map the services to the nodes of the network targeting the minimization of the objective function that in our case is the communication cost. Due to the fact that our problem is NP-complete, we introduce a heuristic, responsible for the dynamic assignment of the system services within the sensor network.

This paper is organized as follows: Section 2 reviews the state-of-the-art in service assignment for WSNs, before Section 3 presents the problem definition. Section 4 introduces our ant-based service assignment heuristic, which consists of a basic and extended version. The results of the evaluation of our heuristic are then described in Section 5. Finally, Section 6 presents the conclusions.

2 Related Work

In this section, existing approaches dealing with migration of services in wireless sensor networks will be presented. Although there is a wide range of middleware and virtual machine approaches, at this moment, the majority of operating systems for WSNs do not provide service assignment mechanisms. Given the fact that most task/service assignment mechanisms used in WSNs are online (deciding during runtime), code mobility is necessary for such approaches.

In the Sensorware [3] virtual machine, the application consists of scripts deployed on a subset of network nodes. Scripts function like state machines, influenced by external events. Scripts may replicate, so that the application has the control of the service assignment, which enables agents to have individual strategies, implemented by the application programmer. In contrast, in our approach, the operating system controls the migration of services according to an OS location policy optimizing a given objective function. This disburdens the user from having to implement a migration policy in each application.

MagnetOS [1] uses the two online algorithms NetCenter and NetPull for decisions on system component assignment. NetPull monitors communication at the link level and migrates components one hop towards the neighbor with the greatest communication. NetCenter, on the other hand, relies on network-level information and migrates objects to the node hosting the component(s) they communicate the most with, possibly over multiple hops. Differently from our system, NetCenter transfers the system components directly to the node hosting the object with the highest interaction. This may lead to a non-optimal assignment and oscillations, since the sum of the communication coming from other objects at different nodes may exceed the communication traffic generated by the single component chosen as migration endpoint by MagnetOS.

In Cougar [8], queries are broadcasted to all nodes of the network and results are aggregated and forwarded to a given leader node. The query optimizer, located on the gateway node, is responsible for analyzing the queries and generating a good query execution plan, which contains the data flow inside the node and network. As this query optimizer-based approach relies on a centralized node, this and our approach are not comparable.

3 Problem Definition

In our approach we are optimizing the position of the services of the system through *migration*. Our heuristic dynamically re-assigns the services to nodes in the system in order to reduce the communication overhead. To enable the evaluation of our heuristic, we define the problem to be solved in each steady state as a formal optimization problem.

The system is represented by two graphs. The first is the network (resource) graph and the second one is the processing thread (task/service) graph (similar to the task interaction graph, TIG). The ad hoc network is modeled by an undirected graph $G = (V, E)$, where V is the set of wireless nodes and an edge $\{u, v\} \in E$ if and only if a communication link is established between nodes $u \in V$ and $v \in V$. The two nodes in this case are neighbors.

For each link, a weighting function attributes a positive weight. $w : E \rightarrow \mathbb{R}^+$. This weight measures the quality (or goodness) of a wireless link. We define for each edge not in the graph ($\{u, v\} \notin V$), $w(u, v) = \infty$. The quality of the link is calculated combining the following parameters: transmission success rate, received signal strength and history of the link. The statistic-based observation of transmission success is a good indication of the future success rate, nevertheless it reacts slowly to changes and at beginning has no data to be calculated. The received signal strength indication makes quick indications possible, but it is not very precise. Therefore, we combine these two parameters. Moreover, in order to prioritize stable links, the history is also used. We use normalized link metrics, where 0 means very good link and 1 poor one. We call the link metric *virtual distance*.

For each node, the weighting function r describes the amount of resources available at a node. $r : E \rightarrow \mathbb{R}^+$. This models the resource capacity of the node.

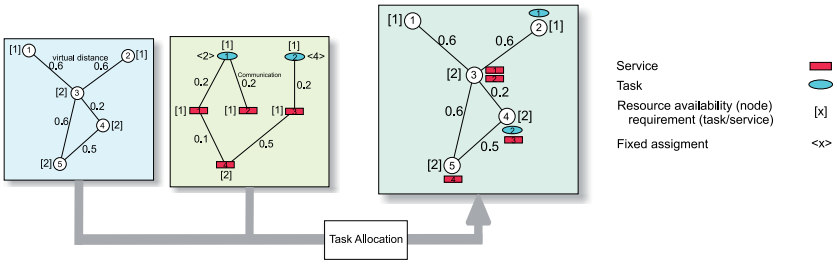


Fig. 1 Example of an instance of service assignment problem.

The processing thread (task/service) graph $T = (M,C)$ models the communication requirements between the diverse processing threads of the OS and application. M is the set of tasks and services (processing threads) running at the moment in the system and an edge $\{m_1,m_2\} \in C$ when there exist an interaction (with communication) between the executable units m_1 and m_2. For each interaction $c \in C$, a function b attributes a positive weight that measures average of traffic between the tasks/services. $b : C \rightarrow \mathbb{R}^+$. This function defines the amount of interaction between two modules of the system. Moreover, the function $e : M \rightarrow \mathbb{R}^+$ attributes the amount of resources necessary for the execution of each task/service. Finally, the function $f : M \rightarrow V$ defines the fixed assignment, i.e., the tasks that are statically assigned to a determined node and should not be moved.

The *service assignment in wireless sensor network problem* consists of allocating the tasks and services of the task graph T to the nodes of the network graph G, minimizing the amount of communication. The amount of communication is measured by the sum of all products of the amount of communication times the distance of the communicating entities. This distance is measured in terms of our link metric. A schematic diagram of the input and result of the assignment is shown in the Figure 1.

The figure 2 presents the formal definition of the optimization problem.

The problem is NP-complete (for a similar NP-complete allocation problem, see [4]), since it generalizes the well-known NP-complete quadratic assignment Problem (QAP) [7]. The QAP is a special case of our problem when the services are in the same number as the processors and just a single service (anyone) may be assigned to each processor.

4 Ant Based Service Assignment

In this section our heuristic to distribute the services in the sensor (or ad hoc) network will be presented.

Input: A processing thread (task and service) graph with weighted nodes, weighted links, and fixed assignment function (T,b,e,f) and a network graph with weighted nodes and links (G,w,r)

Constraints: For every input instance (G,w,r,T,b,e,f), Let $S = \{s_1,s_2,..,s_n\} = \{s \in M | f(s) = \emptyset\}$ be the set of mobile services (without a fixed assignment). The valid solution space is given by:

$\mathscr{M}(G,w,r,T,b,e,f) =$

$= \{(g_1,g_2,..,g_n) \in V^n | \forall v \in V, \sum_{\{i \in \mathbb{N} | g_i = v\}} e(s_i) + \sum_{\{m \in M | f(m) = v\}} e(m) \leq r(v)\}$

The tuple $(g_1,g_2,..,g_n)$ is an assignment and has the following meaning: service s_i is assigned to node g_i. The constraint assures that the services and tasks assigned to the node v do not request more resource than the availability on the node.

Costs: For every assignment $(g_1,g_2,..,g_n) \in \mathscr{M}(G,w,r,T,b,e,f)$, the cost is calculated as follows: Let the function $q : M \rightarrow V$ be: $q(m \in M) = \begin{cases} f(m) & \text{if } f(m) \neq \emptyset \\ g_i | s_i = m & \text{otherwise} \end{cases}$

$$cost((g_1,g_2,..,g_n),(G,w,r,T,b,e,f)) = \sum_{\{m_1,m_2\} \in C} b(\{m_1,m_2\}) \cdot D(q(m_1),q(m_2)) \quad (1)$$

Where $D(u,v)$ is the cost of the multi-hop shortest path employing the virtual distance between nodes $u,v \in V$.

Goal: *Minimum*

Fig. 2 Formal Definition of the Optimization Problem

4.1 Basic Heuristic

In our approach, we are optimizing the position of the services through *migration*, i.e., we try to find the optimal configuration where the communication overhead caused by the remote requests is minimized and to react to demand and topology changes adequately. In order to solve this online discrete optimization problem, we decide to use an ant-inspired algorithm. We assume, in our heuristic, that an initial distribution of the services in the network already exists. In order to describe our heuristic, some additional definitions are necessary.

The set P contains the types of all possible services of the system. Each service s is an instance of same type $p \in P$. Every task $a \in \{M - S\}$ has no type. Let $r \in M$ be the requester (a service or a task) of some service $s \in S$. The service state S_r^i represents the connection between the requester r to the service s (a flow of communication, generated by the requests and responses). The set of all flows of the system we will call W. In our system, each node $v \in V$ has a pheromone table $P_v = [p_{S_r^s}^v]_{r \in M, s \in S}$, where $p_{S_r^s}^v \in [0,1]$. This pheromone level represents the request rate (and traffic) made by the requester r to the service i that is crossing the node v. In our approach, all nodes are responsible for the service assignment, since each node's evaluation is based on its *local* view, in order to reduce communication costs. Moreover, the needed information is constantly changing, due to frequent pheromone updates.

Using an analogy with the ant foraging behavior [2], the services in our approach are the equivalent of the food source. The calls made by the requesters are the agents (or ants) and the requesters are the formicaries. The wireless links form the pathway used by the ants. While the requests are being routed to the destination service, they

leave pheromone on the nodes. The pheromone tables in each node are updated according to the following equation: $p_{S_r^i}(t+1) = \frac{p_{S_r^i}(t)+\delta p(h)}{1+\delta p(h)}$ where the $\delta p(h)$ is the variation of the pheromone and it is a function of the size of the packet. After the introduction of some basic concepts of our heuristic, we will present here the component policy of our migration mechanism:

Transfer policy: In our heuristic, each service is independent and may decide itself about starting a migration. The target of a service migration is every node with sufficient resources.

Selection policy: The selection policy is based on a threshold θ that is compared to the measured current communication overhead of the service s. If it is above θ, the service s is selected to migrate.

Location policy: The location policy decides about which node should receive a migrating service. We will describe it in the next section.

Information policy: Our heuristic uses almost just passive information gathering by means of pheromone tables. We avoid any broadcasting or proactive information dissemination to save the scarce energy resources.

The general idea is to migrate the service to some node that rely in some requests flow (path) or near to it, in the direction of a requester. Each service has several flows coming from the diverse requesters. In order to determine which node should receive the service s, an explorer packet will be used. Its next hop is defined based on the pheromone value of the neighborhood and its final location will eventually be the target node for the migration of s.

We will describe the two main phases (exploration and settlement) of the selection of the new target node for the service s through the migration of the exploration packet.

Exploration Phase

In this phase, the exploration packet will migrate along the nodes of the wireless sensor network in order to find a new target position for the service s. The exploration phase ends and the settlement phase starts when the exploration packet has migrated a determined number of hops (allowed_h) or a loop occurred (detected using a history list *history*).

After the deployment of the exploration packet, its migration is controlled by means of attraction forces. Let $u \in V$ be the actual location (node) of the explorer packet. Ngh_u is the set of neighbors of u, and $d \in Ngh_u$ is a neighbor of u.

$$b_{u,d}^s = \begin{cases} \frac{\sum_{x \in M} P_{S_x^s}^d}{\sum_{y \in (Ngh_u - l)} \sum_{x \in M} P_{S_x^s}^y + pot\_pher} & \text{if} \quad d \neq l \\ \frac{pot\_pher}{\sum_{y \in (Ngh_u - l)} \sum_{x \in M} P_{S_x^s}^y + pot\_pher} & \text{otherwise} \end{cases} \tag{2}$$

$b_{u,d}^s$ represents the sum of the pheromone of all flows coming through node d to the service s normalized over the total amount of pheromone related to requests to the service s in the neighborhood. It represents relatively how much of the traffic directed to the service s is using the node d as path (proportional use of d for

the requests). The $b^s_{u,d}$, in the exploration phase, will act as a force attracting the exploration packet to the corresponding node.

The potential pheromone (*pot_pher*) is the sum of all other pheromones related to the service *s*, coming from the neighbors not selected as next hop for the exploration packet *pak*, when leaving the node hosting *s*. It is used to estimate the level of pheromone potentially caused by those flows if the service would migrate to the node being evaluated. An example can be seen in the Figure 3.

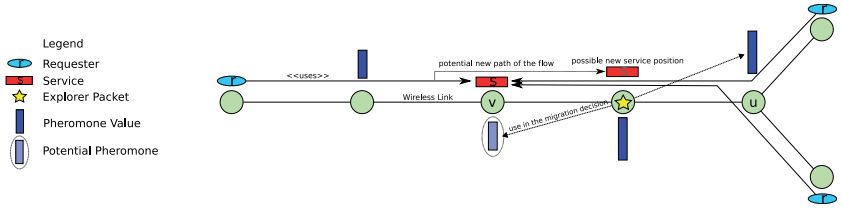


Fig. 3 Example showing the new potential path of a flow when service would migrate to the next hop.

The main idea is to predict which situation would occur if the service would migrate to the current exploration packet position and which would be the next hop for a possible migration. The assumption made here is that the request flows not attended by the first migration decision would have their path size increased exactly by the pathway executed by the exploration packet. This means, although the pheromone level from these flows would not appear to the exploration packet when far away from the node (*v*) hosting *s*, they should be considered when deciding the next exploration packet hop. This is shown in Figure 3, where the exploration packet is in the node *u*. It uses the real pheromone of the node j and, in the case of node *v*, the potential pheromone level measured by the first migration of the exploration packet. The potential pheromone level is the sum of all pheromone levels related to the service *s* that are in all other nodes than *u* because *u* was selected as target for the first exploration packet migration. In this example, the potential pheromone level is exactly the same level of the pheromone on node *h*. It will be formally defined later on.

The next hop of the explorer packet is selected using the equation 3. We call j the selected node.

$$e_i = max_{\{d \in Ngh_u\}}(b^i_{v \to d}), d \in Ngh_u \tag{3}$$

Settlement Phase

After the exploration of possible candidates to host the service *s*, this phase is responsible to find the appropriated node with enough resources to host the service. We call *u* the actual node of the exploration packet.

The idea of this phase is to evaluate whether there are enough resources at the candidate node to host the service *s*. In the positive case, the service will migrate to

the node. In the negative one, the neighborhood will be checked and, according to the actual situation of the neighborhood, a neighbor may be selected or the exploration packet may migrate to the last visited potential candidate (retrieved from the history field), to search there for the final destination of the service *s*.

The following procedure is executed in the settlement phase: The current node *u* is tested whether it may host the service *s*. The test consists of checking whether node *u* has enough free resources. The formalization of the test can be seen in the following equation: $e(s) \leq r(u) - \sum_{\{m \in M | q(m)=u\}} e(m)$ If the resources are enough, the settlement phase is terminated and the node *u* sends a message to the service *s* to trigger the migration process. Otherwise, the same test is made in all the nodes of the direct neighborhood of *u*. The virtual distance is used for ordering the test process. Nodes within smallest virtual distance are tested first. The process ends when a suitable node is found, i.e., the node with enough resources and the smallest virtual distance to *u* is selected. We denote this node as f. If $w(u, f) < w(u, last(history))$, i.e., the virtual distance between *u* and f is smaller than the virtual distance between *u* and the last visited node by the exploration package (before reaching *u*), the node *g* is selected definitively to be the new host of *s*. A message is sent to *s* in order to start the migration. Otherwise, the exploration package is sent back to the $last(history)$ node. The node *u* is deleted from the history field and the settlement procedure starts again.

The procedure described above repeats until an appropriate node is found. In the rather improbable case of not finding any new node to host the service, the migration is canceled.

4.2 Extended Heuristic

This section identifies a problem caused by the greedy nature of the basic heuristic and presents an improvement to overcome possible adversarial situations. For the sake of simplicity, we assume in the following example that allowed_h=1, i.e., just one hop migrations are allowed. Nevertheless, the problem occurs for arbitrary values of this parameter when more than one nearby located requesters use the same service, but due to the employed routing algorithm, the requests are routed through different paths. An example of such situation is depicted in Fig. 4, where requesters r_1, r_2 and r_3 are accessing the service *s* in the node *u*. For a straightforward communication cost calculation, we assume that the average bandwidth utilization is proportional to the pheromone deposited at a node inside the flow path. Thus, the total communication cost is 1.135 (using equation 1).

We analyze the migration that would be decided by the basic heuristic. As the pheromone value of node *h* is higher than the values deposited at nodes *j* and *k* (separately), the exploration packet is sent to node *h*. Suppose that allowed_h=1, the service would migrate to node *h*. The total communication cost of the system changes to 1.22. This result shows that the heuristic, in such adverse situation, selects the wrong node to migrate to, increasing the total communication cost of the

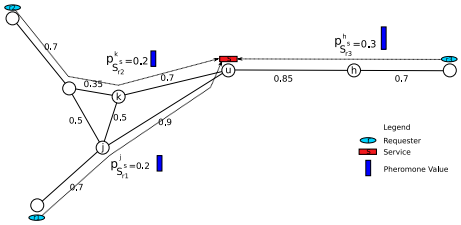


Fig. 4 Instance of the problem that will result in a wrong migration decision due to greedy behavior

system. This happens because of the lack of information over not directly-connected parts of the network (each node has just the *local* view of the system).

The main idea of the improvement is not to migrate the service to the neighbor with the highest amount of requests (highest flow) as in the basic heuristic, but to the neighbor whose flow, in some part, is crossing nodes near to other flows requesting the same service. If the defined metric (virtual distance) has (geographical) norm properties, this will be equivalent to migrating the service to the geographical *direction* from where the highest amount of requests is coming. Two flows related to the requesters r_1 and r_2 (see Fig. 4) are transversing neighboring nodes in their path to s, thus, they should attract the service instead of r_3. We define that such flows transversing neighboring nodes are called correlated flows.

The concept of the correlated flows is used in the exploration phase in order to guide the migration of the exploration packet. Instead of counting solely the pheromone deposited at each neighbor when analyzing the amount of pheromone of a neighbor, the sum of the pheromone deposited at the node with all correlated pheromone is used to guide the migration. Therefore the equation 2 is modified as follows:

$$b^s_{u,d} = \frac{\overbrace{\sum_{x \in M} p^d_{S^s_x}}^{\text{flows using d}} + \overbrace{\sum_{x \in M} \sum_{z \in M} \sum_{g \in Ngh_v - \{d,l\}} p^g_{S^s_z} \cdot \lceil p^d_{S^s_x} \rceil \cdot F(S^s_z, S^s_x)}^{\text{correlated flows}}}{normalizer} \quad (\text{if } d \neq l) \tag{4}$$

The first term of the equation is the same as in eq. 2, i.e., the sum of all requests coming to service s through node d. The second term of the numerator is the sum of the pheromone generated by correlated flows of the flows present at node d. The function F tests whether S^s_z and S^s_x are correlated flows, and the ceiling $\lceil p^d_{S^s_x} \rceil$ checks whether the connection S^s_x exists in the node d (i.e. $p^d_{S^s_x} > 0$). The denominator normalizes $b^s_{u,d}$ ($0 \leq b^s_{u,d} \leq 1$). The next hop of the exploration packet is selected using equation 3.

5 Results

In this section, we present the simulation results of our basic and extended service assignment heuristics. The simulations were performed using the Shox [6] simulator, an event-based wireless network simulator. For our simulation, we assume fixed transmission power, bidirectional links (which is achieved in reality by ignoring unidirectional links) and Friis Free Space propagation model for isotropic point source in an ideal propagation medium for RSSI calculations. The link metric used is based on the RSSI and each node only offers enough resources for running a single service and task. Tasks request different, randomly selected services. The bandwidths needed in the different communications were randomly selected.

5.1 Simulation Scenarios and Evaluation

Table 1 provides an overview of the simulated scenarios. The small scenarios were selected since they also allow the calculation of the optimal solution. For large scenarios, it is not possible to calculate the optimal (reference) solution of our discrete optimization problem due to its computational complexity. Nevertheless, we decided to make an example simulation of a large scenario to show that its behavior is similar to small instances.

For the generation of the task/service graph for each task, a random number of services was selected. The tasks request those services with a random bandwidth requirement (normalized). Dijkstra's shortest path algorithm was used for finding routes between requesters and the services.

Scenario Name	Field Size (m^2)	Number of Nodes	Radio Range	Connection Probability	Node density	Average Degree (Theoretic)	Num. Services, Requesters
Small Scenarios							
small-sparse-sd	80x60	10	28	0.9	0.002	5.13	8, 6
small-dense-sd	80x60	10	43	1	0.002	12.1	8, 6
Large Scenarios							
large-sparse-sd	102x77	100	13	0.9	0.013	6.7	20, 40

Table 1 Overview of the different simulation scenarios.

The presented scenarios were evaluated using different algorithms. For the small scenario, the optimum solution was calculated using a branch-and-bound algorithm. For all scenarios, our basic and extended ant-based service assignment heuristics were simulated. Moreover, we decided to calculate the cost of a completely random assignment, i.e., the tasks and services are randomly distributed among the nodes of the network.

5.2 Experiments

We executed 40 experiments for each scenario presented in Table 1. In the next sections, we will present and analyze the results of the experiments.

Optimal Assignment Cost

In this section, we will analyze the results achieved with the optimal cost assignment. Figure 5(a) presents the optimal service assignment cost for the small-sparse-sd and small-large-sd scenarios[1]. As expected, denser scenarios exhibit a smaller assignment cost. This can be explained by the fact that better links (lower cost) are available for the communication between tasks and services, reducing the total cost. Moreover, due to the higher amount of neighbors (higher node degree), assignments that yield a high amount of costs in sparse environments may be attractive in dense environments because of the existence of multiple new links.

Experiment Results

This section presents the outcome of our 40 experiments for the presented scenarios. In Figure 5, the results for our three scenarios are depicted. For the small scenarios, each result is normalized against the optimal assignment. For the large scenario, we present the nominal result.

As we can see in Figures 5(b) and 5(c), our heuristic found the optimal solution in several cases. Moreover, for the vast majority of cases, the heuristic has a much better performance than the random initial assignment. The extended heuristic and the basic one have also a very similar behavior, nevertheless, for some experiments, the extended one has a much better performance than the basic. The reasons for these outcomes will be discussed further below.

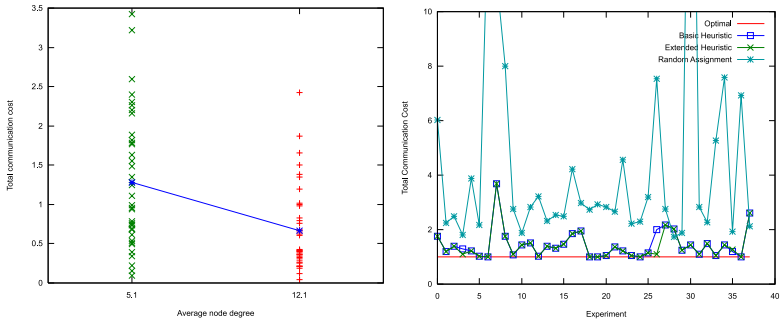
Figure 5(d) shows the results for a large scenario. Due to the fact that we do not have, for large scenarios, a reference approach, it is not possible to make statements about the absolute performance of the algorithms. Nevertheless, it is possible to notice that the heuristics could find a much better cost than the initial random assignment. Moreover, the behavior of the extended and basic heuristics are similar to the one observed in the small experiments.

Heuristics' Assignment Costs

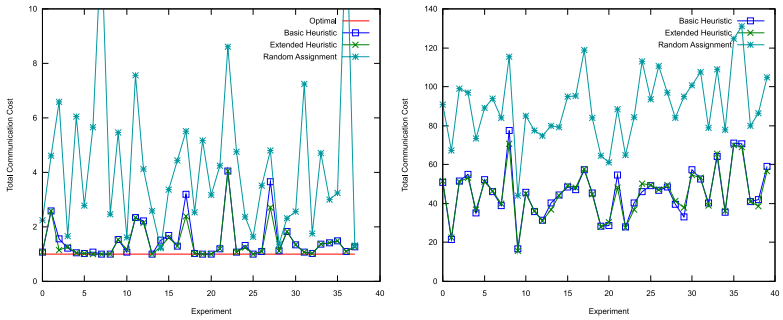
In this section, the mean value of the achieved costs for each heuristic for all scenarios will be presented. The cost for the absolute assignment of our test scenario is shown in Figure 6. The random assignments and basic and extended heuristic assignments have the same tendency of the optimum: for sparse networks, they deliver always an assignment with higher cost. This is expected due to the relation between the assignment cost and the link costs, and for sparse environments, the average link cost increases.

In Figure 6(a), the different costs are shown together for the small scenarios. As it can be seen in the figure, our basic and extended heuristics have a good performance,

[1] For the service assignments, the terms total communication cost (presented in the figure) and assignment cost have the same meaning.

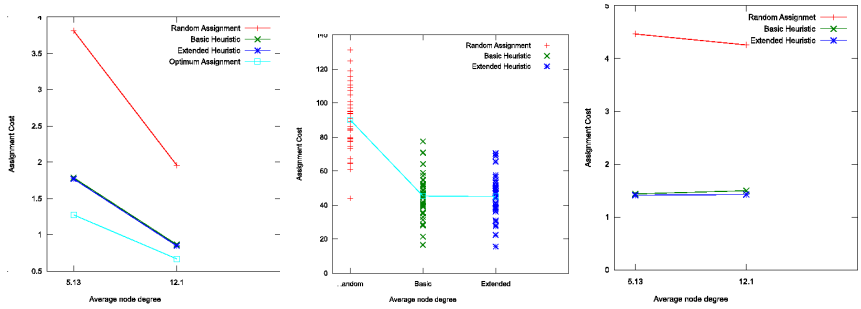


(a) Optimal assignment cost (b) Communication cost small-sparse, normalize



(c) Comm. cost small-dense, norm. (d) Comm. cost large-sparse

Fig. 5 (a) Optimal assignment cost of sparse and dense scenarios and (b–d) communication cost results of the realized experiments



(a) Absolute, small (b) Absolute, large (c) Normalized, small

Fig. 6 Assignment costs for the different heuristics with small and large problem size.

not far from the optimal solution. The basic and extended heuristics have a very similar performance. We discuss the reasons and the performance difference further below.

In Figure 6(b), the results of our large scenario are depicted. It is possible to see that they are very similar to the small scenario, improving our confidence that the heuristics could find good solutions for small as well as for large scenarios.

Figure 6(c) shows the normalized results for the small scenarios. The optimal assignment is used as reference. It is possible to notice that, for all cases, a very small difference could be verified for sparse and dense scenarios. The basic heuristic has an average cost of 1.44 times the optimal cost for sparse environments and 1.5 for dense ones. The extended heuristic shows a small improvement: 1.41 for sparse and 1.43 for dense scenarios. This means that the cost of the basic heuristic was about 2% higher in sparse scenarios and 5% in dense scenarios. A similar behavior has been found in the large scenario.

As it can be observed in Figures 5(b) and 5(c), the basic and extended heuristic, for several experiments, could find solutions with very similar costs and for some experiments, the extended overcame the basic one. For the experiments where the results were similar, we suppose that there are not flow correlations that help the heuristic behavior. For the experiments where the extended heuristic has a much better performance, correlations could be found and a better service migration was realized.

The question that arises from the results is why correlations were not so common. We suppose that the reason was the selected routing algorithm together with the influence of the Friis Free Space Model in our link metric. Because of the exponential path loss, nodes near to each other have a greater advantage in the signal strength than others with a small higher physical distance. Due to the fact that we are, for our simulations, relying strongly on the signal strength to calculate the link metric, it reflects very much this exponential path loss. The Dijkstra's shortest path algorithm always selects the shortest path between any two nodes and does not try to divide the load among the existing link channels. Further we are also not taking into account the link utilization (and possible congestion). Together, such facts act in a way that effectively just a small subset of links is used for all communications. A kind of backbone emerges in the network. This leaves less space for our flow correlations. We suppose that, in real scenarios, where the link metric has a more irregular nature and where there are routing mechanisms that divide the transmission effort among different routes, the extended heuristic will increase its performance in relation to the basic one.

6 Conclusion

In this paper, we study the problem of automatic assignment of mobile services on wireless sensor networks. We model our problem as an optimization problem and present an efficient biologically-inspired heuristic to solve it. Given an initial assignment, the heuristic is responsible to drive the migration of the services based on the

actual network/service configuration targeting the reduction of the communication cost.

We chose to use concepts observed in ant colonies, since they exhibit several properties that are desirable in wireless sensor networks. Hence, we propose an extension to the heuristic: neighboring pheromone trails act together to attract the service to the direction with higher request rate.

Simulations done using the wireless network simulator Shox showed that the heuristic perform well. The basic heuristic has an average cost of 1.44 times the optimal cost for sparse environments and 1.5 for dense ones. The extended heuristic produces a slighter better result than the basic one: 1.41 for sparse and 1.43 for dense scenarios. We suppose that this occurs due to the fact that just a small subset of links is used to route almost all packets in the network (a backbone is formed).

For a large number of real applications, the basic heuristic yields adequate results with very small computational cost. The extra effort necessary for the extended heuristic may not be compensated. However, in real environments, where the link metric does not follow in a regular way the Friis path loss and different routing mechanisms may be used, the extended heuristic may bring better results.

Concluding, our work provides an additional piece of evidence that concepts inspired by biology can be successfully transferred to computer systems.

References

1. Rimon Barr, John C. Bicket, Daniel S. Dantas, Bowei Du, T. W. Danny Kim, Bing Zhou, and Emin Sirer. On the need for system-level support for ad hoc and sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(2):1–5, 2002.
2. Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NY, 1999.
3. A. Boulis and M. B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *Proc. of the First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003), San Francisco, CA, USA*, San Francisco, CA, USA, May 2003.
4. David Fernandez-Baca. Allocating modules to processors in a distributed system. *IEEE Transactions on Software Engineering*, 15(11):1427–1436, November 1989.
5. Tales Heimfarth and Achim Rettberg. Nanoos - reconfigurable os for embedded mobile devices. In *In Proceedings of the International Workshop on Dependable Embedded Systems (WDES)*, Florianopolis, Brazil, 2004.
6. Johannes Lessmann, Tales Heimfarth, and Peter Janacik. Shox: An easy to use simulation platform for wireless networks. In *Proceedings of the 10th International Conference on Computer Modeling and Simulation*, Cambridge, England, Apr. 2008.
7. Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, 1976.
8. Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD*, 31(3), September 2002.