Resource-Aware Clustering of Wireless Sensor Networks Based on Division of Labor in Social Insects

Tales Heimfarth, Dalimir Orfanus and Flávio Rech Wagner

Abstract In this work concepts of division of labor in social insects and emergent self-organization are used to design a very efficient heuristic for clustering wireless sensor networks. Differently from previous approaches, we aim at creating clusters with a minimum amount of resources and good intra-cluster connectivity. Our heuristic has two steps. First, we elect the most suitable clusterheads that have the extra responsibility of leading and representing the cluster. Afterwards, the heuristic selects the respective members of the clusters. These processes are guided by a response function that determines the suitability of each node to a given task (role). For example, nodes with good connectivity and high energy level are good candidates for being clusterheads. In addition to the division of labor, we are using a positive/negative feedback mechanism to control the stimulus for attracting new members. Until having enough resources, the positive feedback acts in order to recruit new members. After gathering enough resources, the negative feedback starts to play a major role. Simulations showed that for 80% of cases the proposed heuristic could find results which are below 2.3 times the theoretical optimal solution, define as the sum of the intracommunication cost of the clusters.

1 Introduction

Wireless sensor networks (WSN) are constantly gaining popularity and attracting more research over the years. One of the reasons is a myriad of novel applications that can be implemented with them. The applications range from human-embedded

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

Dalimir Orfanus
University of Paderborn, Germany, e-mail: orfanus@uni-paderborn.de

Flávio Rech Wagner
Federal University of Rio Grande do Sul, Brazil, e-mail: flavio@inf.ufrgs.br

Please use the following format when citing this chapter:

Heimfarth, T., Orfanus, D. and Wagner, F.R., 2008, in IFIP International Federation for Information Processing, Volume 268; *Biologically-Inspired Collaborative Computing*; Mike Hinchey, Anastasia Pagnoni, Franz J. Rammig, Hartmut Schmeck; (Boston: Springer), pp. 45–58.

sensing and ocean data monitoring to collaborative space exploration. Nevertheless, because of current hardware limitations of wireless nodes, e.g. commercial off-the-shelf sensor nodes, approaches for the management of WSN have to be designed to work using only a low amount of resources and low communication overhead.

In general, two heuristic design approaches for management of sensor networks at different levels (e.g. topology control, network layer, application) are prevalently used. The first method has in all nodes the knowledge of the (entire) network and let they manage themselves. This circumvents the need for a more advanced organization. Nevertheless, this generates overhead in terms of communication and memory at each node. Each node must, for example, maintain routes to the other nodes in the network. In large networks, the number of messages needed to maintain routing tables may cause congestion in the network and depletes the energy of the nodes. Ultimately, the need of individual self-management will generate a huge exchange of messages and overhead.

The second method identifies a subset of nodes within the network and vest them with the extra responsibility of being a leader (clusterhead) of certain nodes in their proximity. The clusterheads are normally responsible for managing communications between nodes in their own neighborhood as well as routing information to other clusterheads in other neighborhoods [1]. This creates a hierarchy in the network. Clustering in large-scale networks was proposed as a means of achieving scalability through a hierarchical approach [11]. Some examples of clustering benefits can be found at the medium access layer, where clustering helps to increase system capacity due to the promotion of the spatial reuse of the wireless channel, and at the network layer, where it helps to reduce the size of routing tables. Sensor networks and, more generally, wireless ad hoc networks largely benefit from clustering.

In this paper, we present a new heuristic that organizes a WSN into clusters. Differently from previous approaches, our proposal addresses the problem of partitioning the nodes of the network in multi-hop groups with a guaranteed minimum amount of resources q (or budget) in each one of them. This kind of clustering is useful in various scenarios. In our case, the clustering heuristic is used in the development of an efficient service distribution in our Operating System (OS).

In our OS for sensor networks, the application and OS services are distributed among different nodes and services are called remotely by the applications. Sharing the services in the network reduces the amount of resources required in each single node. An instance of the OS with all required services should be placed inside each cluster. This means that each cluster must have a minimum amount of resources.

An additional difference from our clustering heuristic to the existing ones is that we are trying to minimize the total communication cost inside the clusters. This communication cost is measured by means of a link metric that assigns a weight to each link, thus modeling the quality of the link. Moreover, the heuristic is in several aspects inspired by the behavior of various biological systems.

Our heuristic is very complex and was designed for a dynamically changing topology. In this paper we focus on the part of heuristic that deals with static topologies.

2 Related Work

In this section, a literature overview of clustering algorithms developed for sensor and ad hoc networks is presented. Some approaches were originally proposed for ad hoc networks but are also used in WSN (a subclass of ad hoc networks).

The idea of clustering is to decompose the nodes of a graph in subsets in a way that the union of the subsets contains all nodes of the graph. For each subset (or cluster), some conditions should hold.

Given a graph $G = (V, E)$ representing a communication network, where vertexes are the nodes and edges the communication links, the clustering process constructs subsets of nodes $V_i, i = 1, .., n$ where $\cup_{i=1,..,n} V_i = V$, such that each subset V_i induces a connected sub-graph of G. These vertex subsets are clusters. Ideally, the size of the clusters falls in a desired range. Moreover, for several approaches, a special vertex in each cluster is elected to represent the cluster and is called clusterhead [5].

There are several design factors concerning heuristics for cluster construction in ad hoc networks. A very important one is the maximal diameter of a cluster: when constructed as a maximum independent set (or minimum dominating set), clusters have the maximum diameter of 3. Nevertheless, we are interested in multi-hop clusters with higher diameter. Different objectives may be pursued in multi-hop clustering.

In [1], the issue of constructing the d-hop dominating set in an ad hoc network is addressed. Because of the NP-completeness of the problem for unit disk graphs, a heuristic called max-min d-cluster formation is presented. It can find good solutions with relative low communication ($O(d)$) and generalizes the dominating set problem. Nevertheless, differently from our approach, the link quality is not considered when selecting cluster members. Moreover, the size of the cluster is uncontrolled. Dense network areas result in larger clusters than sparse ones.

In [10], an algorithm for bounded size clustering based on an expanding ring search is presented. The algorithm relies on a sequence of rounds. In each round, new members are recruited for the cluster in the n-hop neighborhood. n is incremented in each round until the bound (number of nodes) of the cluster is reached. If more nodes than necessary are in the cluster, the clusterhead simply discards the excess. We compare our heuristic with a modified version of the expanding ring that guarantees clusters with a given amount of resources.

Two algorithms improving the expanding ring approach are presented in [7, 8]. They are called *Rapid* and *Persistent* clustering. As in the expanding ring, a maximum determined size (i.e. number of member nodes) for the cluster is desired. The algorithms are more efficient than the expanding ring. The cluster sizes produced should be as close as possible to the specified bound (which we will call here B) in order to limit the total number of clusters. Nevertheless, the bound should not be exceeded.

The *Rapid* heuristic uses less messages than the *Persistent* one. Nevertheless, it has a poor worst-case analytical performance. The *Persistent* heuristic persistently tries to produce a cluster of the specified bound if possible. The proposed algorithms do not violate the cluster size bound at any time. However, this bound is just given

in number of nodes and there is no way to differentiate nodes. In our approach, a weight is associated to each node (representing the amount of resources of the node), and the bound is related with this weight. Moreover, the clusters in the *Rapid* and *Persistent* heuristics are always smaller than or equal to the given bound. In our approach, all clusters have at least a specified amount of resources (as can be seen in the next section).

In the *Rapid* and the *Persistent* algorithms, the clusterheads are elected in a completely random fashion, which leads to the selection of nodes that are not very suitable for the role. In our solutions we use the opposite approach: strongly connected nodes plenty of energy have a higher probability to be selected as clusterheads. Another difference is related to the links: the *Rapid* and the *Persistent* heuristics do not attempt to rank the member candidates (concerning, for example, the links) in order to select the best connected nodes to form the cluster.

A clustering algorithm where a lower and a upper bound are used to control the size of the clusters is presented in [2]. The algorithm is based on the idea of finding a rooted spanning tree of the graph (using Breadth-First-Search) and to form clusters from the subtrees that match the clustering constraints. The upper and lower bound approach tries to keep the amount of nodes in the clusters inside a specified interval. But differently from our approach, overlaps are allowed. Moreover, the link quality is also not relevant to the heuristic.

Another very important difference between all existing approaches and the one presented in this work is the fact that we try to minimize the communication overhead among all nodes inside a cluster. For that, as it will be presented in the next section, we use the smallest distance between each pair of nodes inside the clusters for the objective function. This distance is calculated by means of our combined link metric.

3 Problem Definition

In this section, a formal definition of our exact clustering problem is described.

We call our problem *minimum intracommunication-cost clustering*.

The ad hoc network is modeled as 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 node $u \in V$ and $v \in V$. The two nodes in this case are neighbors. Each node $v \in V$ has a unique identifier (ID_v).

For each link, a weighing function assigns a positive weight. $w : E \rightarrow \mathbb{R}^+$. This weight measures the quality (or goodness) of a wireless link. We define for each edge that is not in the graph ($\{u,v\} \notin V$), that $w(u,v) = \infty$.

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 there is no data to calculate its value. The received signal strength indication makes possibly quick indications, but it is not very precise. Therefore, the combined metric uses these two

parameters. Moreover, in order to prioritize stable links, the history is also used. We use normalized link metrics, where 0 means a very good link and 1 a very poor one. We call the link metric *virtual distance*.

For each node, an additional weighing function r is responsible for characterizing the amount of resources available in the node. $r : E \rightarrow \mathbb{R}^+$. This models the resource capacity of the node.

The clustering process partitions the nodes into *clusters*, each one with a *clusterhead* and possibly some *ordinary nodes*. As presented in the related work section, there are several different types of clustering strategies pursuing different objectives.

In our problem, the objective is to get multihop clusters with enough resources for the OS and application processing. Moreover, the minimization of the intra-cluster communication cost is also desired.

This optimization problem is modeled as follows:

Input: A graph with weighted nodes and links (G,w,r) and a resource requirement $q \in \mathbb{R}^+$, where the sum of all node weights in each cluster must be higher or equal to q.

Constraints: For every input instance (G,w,r,q), $\mathscr{M}(G,w,r,q) = \{C_1, C_2, .., C_k | C_k$ is the k^{th} cluster configuration$\}$, where the following properties hold
$C_k = \{c_{k1}, c_{k2}, .., c_{k(nk)}\}$ is the k^{th} possible cluster configuration of the graph, where $k = \{1,2,..,n\}$ (n is the number of possible configurations, nk is the number of clusters in the k^{th} configuration, $nk = |C_k|$)
$c_{ki} = \left\{ v_{ki}^1, v_{ki}^2, .., v_{ki}^{|c_{ki}|} \right\} \in Pot(V)$ is the i^{th} cluster of the k^{th} configuration, where v_{ki}^j is the j^{th} element of the cluster c_{ki}
For each configuration C_k, $k = 1,2,..,n$, the following properties must hold:

1. $\bigcup_{i=1,2,..,nk} c_{ki} = V$ (cluster definition constraint)
2. $\bigcap_{i=1,2,..,nk} c_{ki} = \emptyset$ (no overlapping constraint)
3. Let $P(u,v) = \left\{ p_1^{(u,v)}, p_2^{(u,v)}, .., p_m^{(u,v)} \right\}$ be the set of all possible paths between nodes u and v. $p_h^{(u,v)} \in Pot(E)$ is the h^{th} possible path where:
$p_h^{(u,v)} = \left\{ \{u, x_1^h\}, \{x_1^h, x_2^h\}, .., \{x_{g-1}^h, x_g^h\}, \{x_g^h, v\} \right\}, x_f^h \in V, f = 1,2,..,g, g \in \mathbb{N}$
For each $\{u,v\} \in E \wedge u,v \in c_{ki}$, $i = 1,2,...,nk$, $\exists p_h^{(u,v)} \in P(u,v) | x_f^h \in c_{ki}$ for $f = 1,2,..,g$. (Connectivity constraint)
4. $\sum_{j=1}^{|c_{ki}|} r(v_{k_i}^j) \geq q$, for each $i = 1,2,...,nk$ (minimum amount of resources per cluster)

Costs: For every cluster configuration $C_k = \{c_{k1}, c_{k2}, .., c_{k(nk)}\} \in \mathscr{M}(G,w,r,q)$, the cost is given by:

$$cost(C_k, (G,w,r,q)) = \sum_{i=1}^{nk} \sum_{u,v \in c_{ki}} \frac{1}{2} \cdot D_{c_{ki}}(u,v) \cdot \left(\alpha \cdot r(u) + (1-\alpha) \right) \qquad (1)$$

Where $D(u,v)$ is the virtual distance between $u,v \in V$. $D_{c_{ki}}(u,v)$ is the virtual distance between u,v using just edges that are inside the cluster c_{ki}. Note that

$\forall v,u \in c_{ki}, D_{c_{ki}}(u,v) = D(u,v)$ iff the cluster c_{ki} is a convex cluster, i.e., the global shortest path between any two nodes in the clustering must use only links inside the cluster. $\alpha \in [0,1]$ controls how much the amount of resources influences the distance metric. For $\alpha = 0$, just the distances between cluster members enter into the metric; $\alpha = 1$ means that nodes with n times more resources have an n times stronger influence.

Now, we define how the virtual distance is calculated. Firstly, we introduce the cost of a path as $PCost(p_h^{(u,v)}) = w(u,x_1^h) + \sum_{f=1}^{g-1} w(x_f^h, x_{f+1}^h) + w(x_g^h, v)$. The virtual distance between u and v is the cost of the shortest path, gived by: $D(u,v) = PCost(p_h^{(u,v)}) = \min_b \left(PCost(p_b^{(u,v)}) \right)$, for $b = 1,2,..,m$.

The virtual distance using only nodes inside the cluster is defined by:

$D_{c_{ki}}(u,v) = PCost(p_h^{(u,v)})$, where $p_h^{(u,v)} \in P(u,v) | x_f^h \in c_{ki}$ and $PCost(p_h^{(u,v)}) = \min_b \left(PCost(p_b^{(u,v)}) \right)$, for $b = 1,2,..,m$

Goal: *Minimum*, i.e. $min_k \{cost(C_k, (G,w,r)), \text{ for } k = 1,2,..,n\}$

The *minimum intracommunication-cost clustering* is an NP-complete problem. The proof can be performed by means of reducing the *partition problem* to our clustering problem (*partition problem* $\leq_p$ *minimum intracommunication-cost clustering*). The complete proof can be seen in [6].

4 The Emergent Clustering Heuristic

Our heuristic cluster construction process consists of two subparts: (1) The clusterhead election, responsible for selecting a subset of nodes and vesting them with the extra responsibility of leading and representing the cluster; (2) The membership selection, responsible for selecting the members of a cluster. Both subparts use behaviors and principles observed in the nature.

Clusterhead election (Section 4.2) is inspired by division of labor and task allocation in swarms of social insects, described in detail by Bonabeau et al. [3]. The possible tasks (or roles) that a node can assume are:

Clusterhead (CH): The clusterhead nodes are the representatives of the clusters. The identification of the cluster is given by the clusterhead. Moreover, special tasks are assigned to the clusterhead. Once the clusterhead is not present in a cluster anymore, the cluster ends its existence.

Member (Me): The members of the cluster are the nodes that have decided which cluster they belong to.

Ordinary Node (Not member, Nm): Nodes that neither decide to enter a cluster nor become clusterhead.

In the case of membership selection (Section 4.3), we are combining division of labor with the concept of emergence of self-organization. Self-organizing systems

acquire structure by themselves and are normally composed by a large number of locally interacting components. [4] presents two basic modes of interaction among the components: positive and negative feedback. Our emergent clustering heuristic is specifically inspired on the behavior of the male bluegill sunfish (*Lepomis macrochirus*), which uses for nesting these two modes of interaction.

Positive feedback can be simplified as the behavioral rule "I nest where others nest". The nesting pattern appears in a large lake with an initial homogeneous structure due to the amplification of fluctuations: if the density of bluegills is sufficient, through a random process, several nesting sites will be occasionally close enough to provide a sufficient attraction that stimulates even more bluegills to nest nearby. This random pattern of nest sites now becomes unstable and a cluster of nest sites will grow. A process like this, with positive feedback, is also called an autocatalytic process.

The negative feedback is responsible for controlling and shaping the system in a particular pattern. Without it, a potential destructive explosion may be easily reached. The feedback can be rephrased as "I nest where others nest, unless the area is overcrowded". Physical constraints like depletion of the building blocks can be also included in the negative feedback.

As the result of the interplay of these modes of interaction, a nice-shaped cluster of nests emerges at the bottom of a lake. This happens without any central control or blueprint, exactly like in our heuristic.

4.1 Overview of the Approach

The first task of the heuristic is to elect the clusterheads of the network using the response function $T_{\theta_{ch_v}}$:

Nonmember → Clusterhead: The response threshold function $T_{\theta_{ch_v}}$ returns the probability of a nonmember v to become a clusterhead. The function is responsible for modeling the emergence of clusterheads in areas of the WSN where no clustering is already taking place.

A clusterhead is now a unitary cluster with some resource ($R_i = r(v)$, v is the clusterhead of cluster i). When a clusterhead is elected in some part of the network, as a consequence of missing resources it starts to "attract" new members with help of the response function $T_{\theta_{recr_{v,i}}}$:

Nonmember → Member of x: The response function (recruitment function, $T_{\theta_{recr_{v,i}}}$) models the recruiting of new cluster members through a positive feedback process. It provides the probability that node v will enter into the cluster $ID = i$.

The idea is that a cluster incrementally grows until it achieves at least the requirement q of resources. The intensity of the attraction force (and consequently the stimulus to enter into the cluster) is regulated by the amount of resources already in the cluster. A growing cluster exercises an attraction force to the nodes

that are in the vicinity. This attraction force is expressed by a higher stimulus s in the $T_{\theta_{recr_{v,i}}}$ response function (positive feedback). Then, when a cluster attracts nodes that bring enough resources, the attraction force becomes much smaller (negative feedback).

4.2 Clusterhead Election

As we mentioned before, the clusterhead has an extra responsibility of representing the cluster and leading the selection of members. Nodes have different predispositions to be a clusterhead, i.e. they have distinct connectivity and distinct amounts of energy. It is obvious that the clusterhead should have good connectivity to other nodes and enough energy to cover the extra activity due to the leadership (build-up and maintenance of a cluster). The opposite is also true: nodes with poor connectivity and an almost depleted source of energy are not good candidates. This concept is derived from the division of labor of social insects. Instead of having just a certain number of fixed morphology agents (like the *majors* and *minors* in the *Pheidole* genus), we have here the complete spectrum of nodes: from nodes very capable of assuming the clusterhead role to nodes not suitable at all for this task. We model the probability of node v to become a clusterhead with the response function $T_{\theta_{CH_v}}(s_{CH_v}) = \frac{s^{\beta}_{CH_v}}{s^{\beta}_{CH_v}+\theta^{\beta}_{CH_v}}$.

The fitness of the node to the role of clusterhead is modeled in the response function with the threshold (θ_{CH}). A small threshold means that the node is very suitable to be a clusterhead. Parameter s_{CH} models the stimulus to become a clusterhead. For a given threshold, a high stimulus increases the probability of the node to become a clusterhead.

The definition of threshold is in Equation 2.

$$\theta_{CH_v} = k_1\left(\frac{\sum_{u\in Ngb_{Nm}(v)} w(u,v)}{|Ngb_{Nm}(v)|}\right) + k_2(1-E_v) + k_3\left(1 - min\left(1, \frac{|Ngb_{Nm}(v)|}{Max\_Neighb}\right)\right) \quad (2)$$

Where $Ngb_{Nm}(v)$ is the set of all neighbor nodes which are in nonmember state, $w(u,v)$ measures the quality of the link between two nodes, and E_v describes the energy level of the node.

As we said before, factors that influence the threshold are good connectivity (the first and the third term) and amount of energy (the second term). Each factor has a different importance for the overall threshold, which is captured with weights (k_1, k_2, k_3). Weights range from 0.0 to 1.0, and the sum of them is 1.0.

The stimulus function is given by $s_{CH_v} = k_1\frac{t_{elapsed}}{t_{required}} + k_2\left(1 - \frac{|Ngb_{Me}(v)|+|Ngb_{CH}(v)|}{|Ngb(v)|}\right)$.

Where $t_{elapsed}$ is the elapsed time since the clustering heuristic has started and $t_{required}$ is the maximum running time of the algorithm. $Ngb(v)$ is the set of all neigh-

bors of the node v, $Ngb_Me(v)$ is the set of nodes in member state and is subset of $Ngb(v)$, the same for $Ngb_{CH}(v)$.

As we can see, there are two factors that stimulates a node to become a clusterhead: (1) nodes that for a long time did not belong to any cluster (first term); and (2) nodes without clusters in the vicinity (second term).

Based on the response function presented, each node periodically tests whether it should become a clusterhead. Initially, all nodes are nonmembers. With time, clusterheads will emerge and attract other nodes to be a member of their clusters.

If a clusterhead, after a certain number of attempts, could not keep the requirement q of resources per cluster, then the (incomplete) cluster will cease its existence and the current members will be free to join other existing clusters.

4.3 Member Selection

Once clusterheads emerge, they start to send messages to attract new members. Each nonmember that receives this message will evaluate its probability of assuming the task of member of the cluster using the response function: $T_{\theta_{recr_{v,i}}} = \frac{s_{recr_{v,i}}}{s_{recr_{v,i}} + \theta_{recr_{v,i}}}$.

Where the threshold and the stimulus have the following meaning:

Threshold $\theta_{recr_{v,i}}$: measures how connected the node v is to the cluster i. A small value means high suitability to be a member.

Stimulus $s_{recr_{v,i}}$: represents the volition of a cluster to attract new members. Here the positive and negative feedback act.

The threshold function for node v is defined by:

$$\theta_{recr_{v,i}} = k_1 \cdot D_i^v + k_2 \cdot \; min\left\{\frac{D(v,Clusterhead_i)}{Max\_dist}, 1\right\} + k_3 \cdot \min\left\{\frac{Cn_i^v}{Max\_connect}, 1\right\} + k_4 \cdot \frac{r(v)}{q} \quad (3)$$

Where D_i^v is the distance to the nearest member of the cluster i and $D(v,Clusterhead_i)$ is the distance to the clusterhead. $Cn_i^b = \sum_{e \in \{Ngb(b) \cap c_i\}} (1 - w(b,e))$ measures the connectivity to the neighbors that are already in the cluster using the link metric w.

The first factor that influences the threshold (first term) reduces the distance among members of the cluster. The factor that influences the shape of the cluster is captured in the second term. Advantage is given to flat configurations (small cluster diameter).

The selection of nodes that are well connected to members of the cluster increases the probability of reducing the cluster cost. This idea is reflected in the third term.

The fourth term covers the idea that nodes with higher resource availability will potentially reduce the cost of the cluster because they reduce the necessity of taking additional nodes.

The stimulus of a node to belong to the cluster i is given by $s_{recr_{v,i}} = k \cdot (p(R_i) \cdot g(R_i))$.

If two clusters are trying at the same time to attract the node, this equation is used with the higher stimulus. The stimulus is the combination of positive and negative feedbacks.

Aggregation Through Positive Feedback

Positive feedback is used to control the stimulus of neighboring nodes to enter a determined cluster. It is performed by considering the attraction force (or stimulus in the response function) to be proportional to the amount of resources R_i of the cluster i plus some bias, i.e., $p(R_i) = k_1 + k_2 \cdot R_i$. This equation denotes the relationship between the amount of resources and the "force" (that is reflected in the stimulus) to attract new nodes to the cluster.

Creating Structure Through Negative Feedback

The negative feedback is responsible for "controlling" the explosive nature of the positive feedback and to shape the emergent structures in the self-organizing process. In our case, we use Equation $g(R_i) = 1 - \left(\frac{R_i}{k_1 \cdot q}\right)^{\beta}$ as negative feedback.

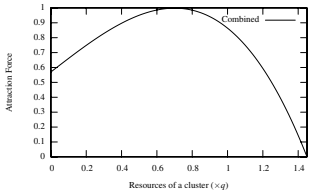

Fig. 1 Resulting attraction force after combination of the positive and the negative feedback

It is important to remark that the negative feedback in our case controls how much the positive feedback takes effect, i.e., the result stimulus is given by the multiplication of the feedbacks, a fact that is shown in Figure 1.

4.4 Cluster Construction Process

In this section we will present the steps performed by the heuristic to build the clusters based on the concepts presented in the previous sections.

At the beginning, there is no cluster in the network. Every node tests periodically whether it should become clusterhead (using the response function $T_{\theta_{ch_v}}$). An information flow based on beacons is used to provide the nodes with the necessary knowledge for the response function.

When the node v decides to become clusterhead, a new cluster (we call it cluster i, $i = clusterID$) comes into existence. Initially, this cluster has the resource $R_i = r(v)$.

Now, it starts to broadcast to the neighborhood periodically its current resource state (R_i). The message is called clusteringForward. The basic function of the clusteringForward message is to inform all members of the cluster and nearby neigh-

bors the actual amount of resources of the cluster. This is used by the nodes to calculate the current attraction force of the cluster. The clusteringForward message is forwarded by the members of the cluster until arriving at nodes outside the cluster. During this phase, a spanning tree having the clusterhead as root is generated. Nodes outside the cluster that receive a clusteringForward message will generate the clusteringBackward message that travels back to the clusterhead, gathering information about nodes with intention to enter or leave the cluster. Each node that is not a leaf of the spanning tree waits until receiving the clusteringBackward message from its children before sending a fused clusteringBackward message to its own parent.

We will call this process of sending the clusteringForward message and gathering information through the clusteringBackward message a *cluster construction round*.

As already said, the cluster construction round is started by the clusteringForward message issued by the clusterhead. When receiving this message, a node u stores it temporarily in order to select the message with the smallest link metric to the clusterhead. This is used to build a good spanning tree with the clusterhead as root.

The way of responding to the incoming message varies depending on the current status of the node u:

Node u is not a member of cluster i: The first action of the node is to determine whether it should enter the cluster i. This is done using the response function $T_{\theta_{recr_{v,i}}}$(recruitment function) to evaluate whether the node u wishes to enter the cluster (recruitment function). This response function uses the connectivity to the cluster as threshold (good connected nodes have less threshold to enter the cluster), and the stimulus is given by the combination of the positive/negative feedback presented in Section 4.3. If the test of the recruitment function returns positive, the clusteringBackward message will carry the membership intention of the node u. The next *clusteringForward* message will confirm (or not, if the cluster is overcrowded) the acceptance of the node u in the cluster.

Node u is a member of cluster i: The node will test whether it should leave the cluster using the response function $T_{\theta_{leave_v}}$. If the test returns negative, the node just retransmits (forwards) the message clusteringForward in order to continue the construction of the spanning tree. If the node is willing to leave the cluster (because its connection is getting loose), it also forwards the clusteringForward message, but indicating this intention of leaving the cluster. This will force previous children to select another parent because this node is going to be disconnected from the cluster. If they could not find another parent, they must also disconnect themselves from the cluster.

The clusteringBackward message is used to inform the clusterhead about nodes with intention to enter the cluster and nodes willing to leave. Moreover, the *id* of all members of the cluster is collected in this message. Therefore, the clusterhead can re-check the complete membership of the cluster to see if some node has for example disappeared due to failure or a drastic topology change.

When the clusterhead receives the clusteringBackward message from all its direct children, it can decide which nodes that are willing to enter the cluster will be accepted. This decision is based on their thresholds to enter the cluster: nodes with less threshold have higher priority.

It is important to state here that after the cluster is complete, the clusterhead ceases to start new rounds. When some member of the cluster detects a large topology change, the clusterhead is informed and a new round is started to re-check the complete cluster (reactive response to topology changes).

An example of the cluster construction round is shown in Figure 2.

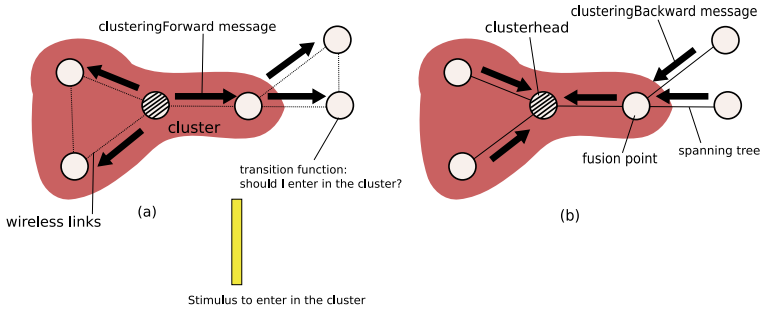

Fig. 2 Example of cluster construction round. (a) Clusterhead starts the round sending the message clusteringBackward with the current amount of resources of the cluster. (b) When arriving at nodes outside the cluster, they decide whether they are willing to join the cluster. This information is sent back using the clusterBackward message

The first purpose of the positive/negative feedback is to reduce the amount of information aggregated in the *clusteringBackward* message. Nodes badly connected to the cluster will decide not to enter the cluster, thus reducing the amount of information that the *clusteringBackward* message must carry.

The second purpose of the positive/negative feedback mechanism is to control the competition among neighboring clusters and belongs to the dynamic part of our heuristic (which is not the main focus of this paper). The feedback curves are designed in such a way that an already formed cluster may just loose some members till the q limit is achieved, because, when this limit is achieved, the desire to attract new members is at maximum. In the same way, if there are two clusters under construction, this method avoids that one cluster steals members from the other one, reaching the state where no cluster has fulfilled its requirement on resources.

5 Results

We implemented our emergent clustering heuristic using an event-based wireless ad hoc simulator called ShoX [9]. Some parts of the heuristic were also implemented in the specification and modeling language AsmL. As input, we generated 35

instances of the problem with 16 nodes in a field of 50m by 50m. These instances were generated by random selection of the node positions.

We used the received signal strength (RSSI) for the free space model with isotropic point sources as link metric. We decided to test our heuristic for dense networks, therefore the radio range was 70m, covering the complete field.

In this paper, we evaluate our heuristic for static networks and for networks of homogeneous devices (i.e., each node has a unit of resource). We run our algorithm, let it converge to a stable configuration, and compare the heuristic result with the optimal one and also with an existing heuristic called expanding ring [10]. In order to calculate the optimal result, we model our *minimum intracommunication-cost clustering* as an integer linear programming model and, for each simulated instance, we find the optimal solution for that configuration with the lp_solve program.

Figure 3 shows results of performed experiments. In average, the cost of our heuristic was 1.98 times higher than the optimal solution and the expanding ring was about 4.29 times higher. To run the simulation of the complete network, it took 10 seconds while the optimal solution needed more than 10 hours in Intel Core Duo 2.7 GHz computers.

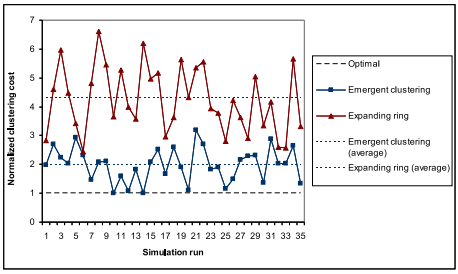

Fig. 3 Normalized results of performed experiments

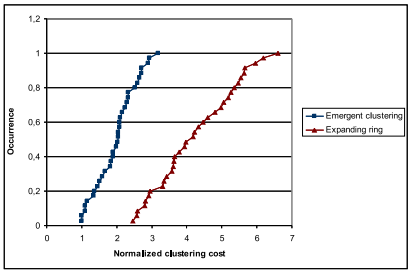

Fig. 4 Cumulative distribution of normalized simulation results

In Figure 4, the cumulative distribution of the normalized results can be seen. It is possible to notice that for more than 80% of all simulations, the emergent clustering heuristic could find results that were below 2.3 times the optimal one. In the case of the expanding ring, results were below 5.2 times the optimal one.

6 Conclusion

In this paper, we introduce a useful clustering problem and develop an efficient heuristic inspired by biological systems to solve it. The heuristic has two parts: the clusterhead election, which is responsible for selecting a subset of nodes and vesting them with extra responsibility of representation of the cluster, and membership selection, which is responsible for selecting members in order to fulfill the resource requirements of each cluster.

The selection of the task for a node is based on its suitability for that task. In the same way that ants with different morphology have tendency to perform different tasks, different nodes have different probabilities of assuming the clusterhead or cluster member roles. This concept is combined with a positive/negative feedback stimulus, which is responsible to shape the size and form of the cluster.

The results of the simulations show that the heuristic performs well, with cost in average just 1.98 times the optimal one. This was achieved in a distributed manner and using only locally available information to make decisions. This makes this heuristic suitable for ad hoc networks with resource-constrained devices or sensor networks.

The results obtained here re-enforces our confidence that methods found in nature can be successfully transferred to computer systems.

In the future, we plan to simulate the heuristic in networks with moderate topology changes, evaluating our approach with dynamic scenarios.

References

1. A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 32–41vol.1, 26-30 March 2000.
2. S. Bannerjee and S. Khuller. A clustering scheme for hierarchical control in wireless networks. In *Proceedings of the IEEE INFOCOM*, Anchorage, AK, April 2001.
3. Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, Santa Fe Institute Studies in the Sciences of Complexity, New York, NY, 1999.
4. Scott Camazine, Jean-Louis Deneubourg, Nigel R. Franks, James Sneyd, Guy Theraulaz, and Eric Bonabeau. *Self-Organization in Biological Systems*. University Presses of CA, 2003.
5. Y. Chen, A. Liestman, and J. Liu. Clustering algorithms for ad hoc wireless networks. In Ad Hoc and Sensor Networks, 2004.
6. Tales Heimfarth. *Biologically Inspired Methods for Organizing Distributed Services on Sensor Networks*. PhD thesis, University of Paderborn, 2008.
7. R. Krishnan and D. Starobinski. Message-efficient self-organization of wireless sensor networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, USA, March 2003.
8. Rajesh Krishnan and David Starobinski. Efficient clustering algorithms for self-organizing wireless sensor networks. In *Ad Hoc Networks*, volume 4, pages 36–59, January 2006.
9. 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 Modelling and Simulation*, Cambridge, England, April 2008.
10. C. V. Ramamoorthy, A. Bhide, and J. Srivastava. Reliable clustering techniques for large, mobile packet radio networks. In *Proceedings of the 6th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 87)*, San Francisco, USA, April 1987.
11. Ivan Stojmenovic, editor. *Handbook of Sensor Networks*. John Wiley and Sons Inc, 2005.