Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 31 (2008) 750–759
computer communications
www.elsevier.com/locate/comcom

Efficient Load-Balanced Clustering Algorithms for wireless sensor networks [☆]

Chor Ping Low [*], Can Fang, Jim Mee Ng, Yew Hock Ang

School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore

Available online 22 October 2007

Abstract

Wireless sensor networks have been receiving increasing attention in recent years due to their potential applications in the establishment of dynamic communications for emergency/rescue operations, disaster relief efforts, and military networks. In this paper, we investigate the problem of grouping the sensor nodes into clusters to enhance the overall scalability of the network. A selected set of nodes, known as gateway nodes, will act as cluster-heads for each cluster and the objective is to balance the load among these gateways. Load-Balanced Clustering increases system stability and improves the communication between the various nodes in the network. We call the problem addressed in this paper as the *Load-Balanced Clustering Problem* (*LBCP*). We first show that a special case of LBCP (whereby the traffic load contributed by all sensor nodes are the same) is optimally solvable in polynomial time. We next prove that the general case of LBCP is NP-hard. We then proposed an efficient $\frac{3}{2}$-approximation algorithm for the problem.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Wireless sensor networks; Clustering; Load balancing; NP-hard approximation algorithm

1. Introduction

The availability of cheap, low power, and miniature embedded processors, radios, sensors, and actuators, often integrated on a single chip, is leading to the use of wireless communications and computing for interacting with the physical world in applications such as security and surveillance applications, smart classroom, monitoring of natural habitats and eco-systems, and medical monitoring. The resulting systems, often called wireless sensor networks, differ considerably from current networked and embedded systems. They combine the large scale and distributed nature of networked systems such as the Internet with the extreme energy constraints and physically coupled nature of embedded control systems.

A sensor network is composed of a large number of sensor nodes (or sensors), which are densely deployed either inside the *phenomenon* (i.e. something known by sense perception) or very close to it. Sensors are generally equipped with data processing and communication capabilities. These sensing circuit measures parameters from the environment surrounding the sensor and transforms them into an electric signal. Processing such signals reveals some properties about phenomenon and/or objects located in the vicinity of the sensors. Data collected from each sensor are routed back to a base station or command node, either periodically or based on events. To avoid long-haul communication with the command node, some high-energy nodes called *Gateways* are typically deployed in the network. Sensor nodes are group into distinct clusters by using each of these gateways as the cluster-head of a cluster. Each sensor node only belongs to one cluster and communicates with the command node through the gateway (or cluster-head) in the cluster. It is easy to see that by adopting a cluster-based network architecture, the overall traffic load can be better distributed among the nodes in the various clusters and the end-to-end transmission delay

[☆] A preliminary version of this paper has been accepted for publication in the Proc. IEEE International Conference on Communications (IEEE ICC 2007).
[*] Corresponding author.
E-mail address: icplow@ntu.edu.sg (C.P. Low).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.10.020

between a sensor node and the command node can be reduced. This in turn enhances the overall scalability of the network.

A multi-gateway architecture is required to cover a large area of interest without degrading the service of the system. But, if the sensor nodes and the gateways are not "well distributed", some gateways may be overloaded with increase in sensor density and detected phenomenons/targets. Such overload may increase latency in communication and cause degradation of overall network performance.

Hence, in this paper we address the problem of assigning sensor nodes to gateways to form clusters with the objective of minimizing the maximum load of each gateway in the given network. We note that the effort to minimize that maximum load of each gateway will result in a more balanced distribution of loads among the set of gateways. We refer to this problem as the *Load-Balanced Clustering Problem* (*LBCP*). The problem here is to determine for each sensor node, the gateway to which it should be assigned, under the constraint that each sensor node must be connected to one and only one gateway, in order to minimize the overall maximum load of the gateways.

We first consider a special case of the problem whereby the offered traffic loads from all sensor nodes are the same. We refer to this special case as the *Load-Balanced Clustering Problem with Uniform Traffic Load* (*LBCP–UTL*). We show that this special case is optimally solvable in polynomial time. We next consider the general case of the problem in which the traffic load from each sensor node may differ from one another. We show that the problem in this case is NP-hard and we propose $\frac{3}{2}$-approximation algorithm for the problem.

The rest of this paper is organized as follows: In the next two sections we describe the architectural model of sensor network and summarizes related work. In Section 4, we give a formal definition of the Load-Balanced Clustering Problem. A polynomial time algorithm that optimally solves a special case of LBCP is described in Section 5. The general case of LBCP is considered in Section 6. This problem is shown to be NP-hard and an approximation algorithm for the problem is given in this section. Simulation results to evaluate the performance of our proposed approximation algorithm will be described in Section 7. The paper concludes with Section 8.

2. System architecture

The system architecture for the sensor network is shown in Fig. 1. Gateway nodes are less-energy constrained compared to sensor nodes. All communication is over wireless links. A wireless link is established between two nodes only if they are in range of each other. Gateways are capable of long-haul communication compared to sensor nodes and all gateway nodes are assumed to be in communication range of one another. In this paper, we assume that the sensor nodes and gateways are stationary.

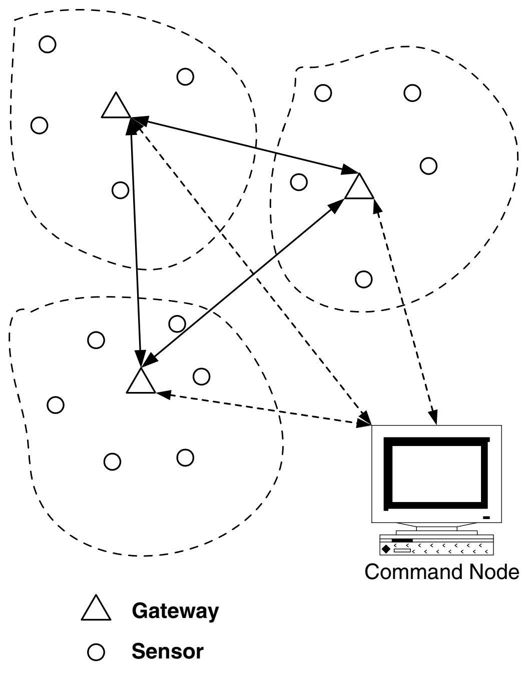

Fig. 1. Multi-gateway Clustered Sensor Network.

3. Related work

Clustering of nodes in wireless networks has been addressed by a number of researchers [1–7]. Most of the existing work for clustering base the selection of cluster-heads on various factors which include cluster ID [5], degree of connectivity [6,7] or randomization [4].

However, most of the published clustering protocols do not consider any load balancing among clusters due to variable density of nodes in the system. In [8], a clustering scheme which aims to produce clusters of bounded size was proposed. However such a scheme will require all nodes to have Uniform Traffic Load to ensure that the overall load is sufficiently balanced among all clusters in the network. Since a system that is not load-balanced will give rise of clusters with high-density and very low-density, the high-density cluster-head will be overwhelmed with the processing and communication load and while be depleted of energy at a much faster rate than low-density cluster-heads.

The main objective of our work in this paper is to cluster sensor nodes in a network efficiently around several high-energy gateway nodes. Clustering enable network scalability to large number of sensor nodes and extends the life of the network by allowing the sensor nodes to conserve energy through communication with closer nodes and by balancing the load among the gateway nodes. Clusters are formed based on the load of the gateways. The network topology is assumed to be static, as in sensor networks or slowing changing.

The procedure for cluster formation consists of two phases: gateway (cluster-head) election and assignment of sensor nodes to gateways. In this paper, we consider a network scenario where gateway nodes are chosen a priori and are fixed throughout the network lifetime. We then address the problem of cluster formation by assigning sensor nodes to gateways to with the aim of balancing the load among the gateways.

4. Problem formulation

In this section, a formal definition of the Load-Balanced Clustering Problem will be given. We adopt the following assumptions and notations in the problem formulation:

- we consider a network scenario where gateways are chosen apriori and the locations of sensor nodes are known;
- the load (which is a function of processing load and communication load) contributed by each sensor node can be estimated;
- the set of sensor nodes is denoted by T and $|T| = n$;
- the set of gateways is denoted by C and $|C| = m$;
- $n > m$, i.e. the number of sensor nodes is greater than the number of gateways;
- d_i denotes the traffic load contributed by sensor t_i where $t_i \in T$ and $d_i \in Z^+$;
- C_i denotes the set of gateways onto which sensor t_i may be assigned, where $t_i \in T$. We note that some constraints may be imposed such that a given sensor t_i can only be assigned to a member of a selected set of gateways C_i, where $C_i \subseteq C$. Examples of constraints that may restrict the assignment of sensors to gateways include that of ensuring that a given pair of sensor and gateway are within the communication range of each other. We refer to such constraints as the *assignment constraints*.

4.1. Formulation as a mathematical program

Prior to the problem formulation, the following variables are defined:

- x_{ij}: a binary variable, 1 if sensor t_i is assigned to gateway c_j, otherwise 0;
- α: the maximum load that may be assigned to a gateway.

The Integer Linear Programming (ILP) formulation of the Load-Balanced Clustering Problem is defined as follows:

Objective function:

$$\text{Minimize } \alpha \tag{1}$$

Subject to

$$\sum_{c_j \in C_i} x_{ij} = 1, \quad \forall t_i \in T \tag{2}$$

$$\sum_{t_i \in T} d_i \cdot x_{ij} \leqslant \alpha, \quad \forall c_j \in C_i \tag{3}$$

The objective (1) is to minimize the overall maximum load of the gateways. Constraint (2) states that each sensor should be assigned to one (and only one) gateway. Constraint (3) imposes the condition that the total traffic load of all sensors assigned to a particular gateway should not exceed the maximum load permitted.

5. The Load-Balanced Clustering Problem with Uniform Traffic Load

In this section we consider a special case of LBCP whereby the traffic load from each sensor is the same, i.e. $d_i = \beta$ for some constant $\beta\ \forall i \in T$. Without loss of generality, lets assume that $\beta = 1$. We refer to this problem as the *Load-Balanced Clustering Problem with Uniform Traffic Load* (*LBCP–UTL*). We note that the problem of minimizing the maximum load of each gateway in this case is equivalent to that of minimizing the maximum number of sensors that are assigned to each gateway. Let $l(j)$ denote the number of sensors that are assigned to gateway j, where $j \in C$. The Load-Balanced Clustering Problem with Uniform Traffic Load is that of finding as assignment $A{:}T \to C$ such that $A(i) \in C_i$ and $l_{\max}$ is minimized, where $l_{\max} = \max_{j \in C} l(j)$ and $l(j) = |\{i \in T{:}A(i) = j \text{ and } j \in C_i\}|$.

We say that sensor i is *assigned* to gateway j if $A(i) = j$. An *optimal solution* for LBCP–UTL is an assignment A for which the resulting $l_{\max}$ is the least possible. In this section, we propose an algorithm, called the *Load-Balanced Clustering Algorithm* (*LBCA*), that optimally solves LBCP–UTL in $O(mn^2)$.

Let the set of sensors to be assigned be denoted as $T = \{t_1, t_2, \ldots, t_n\}$ and the set of gateways available be denoted by $C = \{c_1, c_2, \ldots, c_m\}$. Starting with sensor t_1, LBCA will construct a breadth-first search (BFS) tree rooted at t_1. The set of gateways onto which sensor t_1 may be assigned, i.e. C_1, is next included into the tree. Since t_1 is the first sensor to be assigned, it is easy to see that each of the gateways in C_1 will have zero load at the moment. The algorithm will arbitrarily assign t_1 to one of these gateways. Next the algorithm proceeds to construct a breadth-first search tree rooted at t_2 and so on. In general a BFS tree rooted at t_i is constructed level-by-level with t_i in level 1 and then gateways onto which it may be assigned in level 2. The tree is next extended to the 3rd level by including the set of sensors that was previously assigned to the set of gateways in level 2 into the 3rd level of the tree. Observe that the tree alternates between sensors and gateways from one level of the tree to the next, with the sensors and gateways occupying the odd levels and the even levels, respectively. A queue is used to maintain the list of gateways and sensors that appears in the BFS tree. In particular, the set of gateways onto which sensor t_i may be assigned is first inserted in the queue Q. Next, an element v of Q is removed from the front of Q and the sensors that have been assigned to gateway v are inserted at the back of Q; if the element v that is removed from the front of Q is a sensor, then the corresponding gateways onto which sensor v may be assigned, are inserted at the back of Q. This process continues until either one of the following two conditions is true: (i) a gateway with zero load is found or (ii) the set Q is empty. Once the tree rooted at sensor t_i is constructed, the next step is to make adjustments to the previous assignment of sensors $\{t_1, t_2, \ldots, t_{i-1}\}$ to the set of gateways in C so that sensor t_i can be assigned to one of the gateway in C_i while at the same time ensuring that the maximum load of the set of gateways in C is minimized. This process is carried out as follows. We first identify a gateway, say $r_k \in C$, with the least load in the BFS tree rooted at t_i and let us assume that r_k is at level k of the BFS tree. The predecessor of r_k will be a sensor (in level $k-1$) that may be assigned to gateway r_k. Let this sensor be denoted by s_{k-1}. The predecessor of sensor s_{k-1} will be a gateway at level $k-2$ onto which sensor s_{k-1} was assigned in previous assignment. Let this gateway be denoted by r_{k-2}. This process continues until the root of the tree, namely t_i, is reached. At this stage we would have found a path P from t_i to the gateway r_k which is comprised of a set of edges which connects a sequence of vertices which alternate between sensors and gateways. In particular, let P be denoted by: $t_i \to r_2 \to s_3 \to \cdots \to s_{k-1} \to r_k$, where sensor s_3 was previously assigned to gateway r_2, sensor s_5 was assigned to gateway r_4 and in general sensor s_{2i+1} was assigned to gateway r_{2i}, where $i = 1, 2, \ldots, k/2 - 1$. Having identified the path P, the sensors are next reassigned to the gateways as follows:

(i) assign sensor t_i to gateway r_2,
(ii) next reassign sensor s_3 to gateway r_4, and
(iii) in general reassign sensor s_{2i-1} to gateway r_{2i}, where $i = 2, 3, \ldots, k/2$.

The pseudocode of the algorithm is shown in Table 1.

Table 1
Load-Balanced Clustering Algorithm

```
INPUT: A set of sensors $T = \{t_1, t_2, \ldots, t_n\}$, a set of gateways $C = \{c_1, c_2, \ldots, c_m\}$ and traffic load $\beta$ for each sensor $t_i$ .
OUTPUT: An assignment $A : T \to C$ such that $A(i) \in C_i$ and $l_{max}$ is minimized, where $l_{max} = \max_{j \in C} l(j)$ and $l(j) = |\{i \in T : A(i) = j\}|$.
Begin
Step1:
  for $j = 1$ to $m$ do
    set $l(j) = 0$;
  endfor
Step2: /* construction of BFS tree */
  for $j = 1$ to $n$ do
    set $l_{min} = \infty$;
    $Q = \{t_j\}$;
Step3:
    while ($Q \neq \emptyset$) and ($l_{min} > 0$) do
      let $v$ be the front element of $Q$;
      remove $v$ from $Q$;
Step4:
      if $v$ is a sensor then
        for each unmarked gateway $w$ onto which $v$ may be assigned do
          mark $w$;
          insert $w$ to the end of $Q$;
          set $pred(w) = v$;
        endfor
Step5:
      else /* $v$ is a gateway */
        if $l(v) < l_{min}$ then $l_{min} = l(v)$;
        for each sensor $w$ that was assigned to gateway $v$ do
          insert $w$ to the end of $Q$;
          set $pred(w) = v$;
        endfor
    endwhile
Step6: /* reassignment of sensors to gateways */
    let $v$ be a gateway with the least load;
    let $w = pred(v)$;
    assign sensor $w$ to gateway $v$;
    increase load of gateway $v$ by $\beta$;
    while $w \neq t_j$ do
      $v = pred(w)$;
      remove the previous assignment of sensor $w$ to gateway $v$;
      let $w = pred(v)$;
      assign sensor $w$ to gateway $v$;
    endwhile
  endfor
End
```

Lemma 1. *The load of each gateway in the path $P = t_i \to r_2 \to s_3 \to \cdots \to s_{k-1} \to r_k$ will remain unchanged after the reassignment of the sensors in P to the set of gateways in P except for gateway r_k whose load will be increased by one.*

Proof. After the reassignment procedure, sensor t_i will be a new sensor to be assigned to gateway r_2. However, sensor s_3 which was previously assigned to gateway r_2 will now be reassigned to gateway r_4. Hence the load of gateway r_2 remains unchanged. Similarly, sensor s_5 that was previously assigned to r_4 will now be reassigned to r_6. Hence the load of gateway r_4 will also remain unchanged. By similar arguments, it is easy to see that the load of gateway r_{2i}, where $i < k/2$, will remain unchanged. Next, we see that since sensor s_{k-1} will be reassigned to gateway r_k, the load of gateway r_k will be increased by one following the reassignment. □

Lemma 2. *The Load-Balanced Clustering Algorithm (LBCA) produces an optimal solution for LBCP–UTL.*

Proof. Let A be an assignment that is obtained using LBCA. Let A_j denote the assignment of (any) j sensors from T to the gateways in C using LBCA. Let the set of sensors in A_j be denoted by $T_j = \{t_1^j, t_2^j, \ldots, t_j^j\}$. Let c_i^j denote the gateway onto which sensor t_i^j is assigned in assignment A_j, where $1 \leqslant i \leqslant j$. Let $R_{t_i^j}$ denote the BFS tree rooted at t_i^j that is constructed using LBCA. Let P_j denote the path in $R_{t_i^j}$ that connects t_i^j to a least loaded gateway, say c, in $R_{t_i^j}$. We will prove by induction that the following two conditions, referred to as *optimality conditions* are satisfied for all values of j: (i) A_j is an optimal assignment for T_j and (ii) $l(c) \leqslant l(w)$, where w is a gateway with the least load in a given assignment.

It is clear that A_1 is an optimal assignment (as there were no other assignment prior to this and sensor t_1^1 is being assigned to a gateway, say c_1^1, with zero load). The resultant assignment has a maximum load of one. In addition, we note that a gateway c with the least load in $R_{t_1^1}$ has load equal to (i) zero if $|C(t_1^1)| > 1$ or (ii) one if $|C(t_1^1)| = 1$. In either case, it is easy to see that for any given assignment with a least loaded gateway w, $l(w) \geqslant l(c)$.

Next we assume that both the above-mentioned optimality conditions are satisfied when $j = k - 1$. Let the maximum load of A_{k-1} be denoted by $l^{k-1}_{A_{\max}}$, where $k \geqslant 2$. We will next argue that A_k is also an optimal assignment, i.e. the resulting maximum load is the least possible. Let P_k denote the path in $R_{t^k_k}$ that connects t^k_k to a least loaded gateway, say c, in $R_{t^k_k}$. Let $l_{A_{k-1}}(c)$ denote the load of c which results from assignment A_{k-1}. After the reassignment of the sensors in P_k to the set of gateways in P_k, the load of gateway c will be increased by 1 (Lemma 1), i.e. $l_{A_k(c)} = l_{A_{k-1}(c)} + 1$. The load of all other gateways remains the same (Lemma 1). Hence the resultant maximum load is $\max[l_{A_k}(c), l^{k-1}_{A_{\max}}]$.

We claim that there cannot exist another assignment for the set of sensors in T_k that will result in a lower maximum load. Suppose otherwise and let B_k be such an assignment for the sensors in T_k. Let $l^{k-1}_{B_{\max}}$ denote the maximum load that results from the assignment of sensors from the set $T_k - \{t^k_k\}$ using assignment B_{k-1}. Suppose that sensor t^k_k is assigned to gateway w using assignment B_k. Then the maximum load of B_k is $l^k_{B_{\max}} = \max[l_{B_k}(w), l^{k-1}_{B_{\max}}]$. By induction assumption, $l^{k-1}_{A_{\max}} \leqslant l^{k-1}_{B_{\max}}$. Hence, if $l^k_{B_{\max}} < l^k_{A_{\max}}$, then $l_{B_k}(w) < l_{A_k}(c)$. Since $l_{B_k}(w) = l_{B_{k-1}}(w) + 1$ and $l_{A_k}(c) = l_{A_{k-1}}(c) + 1$, $l_{B_{k-1}}(w) < l_{A_{k-1}}(c)$. But this contradicts the second optimality condition for A_{k-1}. Hence, A_k is an optimal assignment. By induction, A is therefore an optimal assignment. □

Lemma 3. *The time complexity of LBCA is $O(mn^2)$.*

Proof. The initialization of the gateway load in step 1 can be done in $O(m)$. Each iteration of the for loop in step 2 deals with the construction of a BFS tree which is done within the while loop (step 3). Each sensor and gateway are inserted at most once into the queue Q. Hence there are at most $2(m + n)$ additions and removals of elements from Q. For each sensor i, $|C_i| \leqslant m$. Hence at most m gateways that will be inspected for each sensor that is removed from Q (in step 4). As there are at most n sensors that are inserted into Q, the total number of gateways that are inspected within the while loop in step 3 is $O(mn)$. Next we note that each removal of a gateway from Q results in the inspection of sensors that are assigned to it (step 5). Since each sensor is assigned to only one gateway, the total number of sensors that are inspected (due to the removal of the set of gateways from Q) within the while loop is $O(n)$. Hence each BFS tree can be constructed in $O(m + n + mn)$ and thus step 2 can be done in $O(n[m + n + mn])$. The reassignment of sensors to gateways in step 6 can be done in $O(m + n)$ (as there are at most $n + m$ elements in the path from t_j to v in the BFS tree rooted at t_j). Thus, the time complexity of the proposed algorithm is $O(mn^2)$. □

6. The Load-Balanced Clustering Problem (LBCP)

In this section, we consider the Load-Balanced Clustering Problem in which the traffic load from each sensor may differ from one another. We first analyse the computational complexity of the problem and prove that it is NP-hard. Having understood its computational complexity, we next propose an efficient approximation algorithm for the problem.

6.1. The intractability of LBCP

LBCP is related to the following machine scheduling problem:

Problem 1. *Minimum Makespan Scheduling Problem on Identical Machines (MMSPIM)*

We are given m machines and n jobs with respective processing times $p_1, p_2, \ldots, p_n \in Z^+$. The processing times are the same no matter on which machine a job is run and pre-emption is not allowed. Find an assignment of jobs to m identical machines such that the *makespan* (which is the latest completion time among all machines) is minimized.

Lemma 4. *LBCP is NP-hard.*

Proof. Consider a special case of LBCP whereby each sensor can be assigned to any gateways (i.e. no assignment constraints). It is easy to see that this special case of LBCP is identical to MMSPIM and LBCP is thus a generalization of MMSPIM. Since the MMSPIM is known to be NP-hard [9], LBCP is also NP-hard. □

6.2. Some observations about LBCP

In this section, we highlight some observations about LBCP.

Observation 1. We first observe that LBCP is also related to another machine scheduling problem, namely the Minimum Makespan Scheduling Problem on Unrelated Machines (MMSPUM), which is defined as follows:

Problem 2. *Minimum Makespan Scheduling Problem on Unrelated Machines (MMSPUM)*

We are given a set J of n jobs and a set M of m machines. The processing time for a job $j \in J$ on machine $i \in M$ is $p_{ij} \in Z^+$ and pre-emption is not allowed. Find an assignment of jobs in J to the machines in M such that the *makespan* is minimized.

In particular, we note that each instance of LBCP can be transformed into an instance of the Minimum Makespan Scheduling Problem on Unrelated Machines (MMSPUM) whereby the sensors and the gateways of LBCP correspond to the jobs and machines of MMSPUM, respectively. For each sensor $t_i \in T$, let $p_{ij} = d_i \, \forall c_j \in C_i$ and let $p_{ij} = \infty \, \forall c_j \notin C_i$. Then it is easy to see that an optimal solution for LBCP corresponds to a schedule for MMSPUM with minimum makespan and vice versa. MMSPUM is also known to be NP-hard [9] and Lenstra et al. [10] gave a 2-approximation algorithm for the problem. This performance bound was further improved to $2-\frac{1}{m}$ by Shchepin et al. [11] and this is currently the best-known approximation ratio that can be achieved in polynomial time.

Observation 2. We next observe that each instance of LBCP can be represented using a bipartite graph as follows. Let $G=(T\cup C,E)$ denote a bipartite graph where E corresponds to a set of edges connecting the vertices in T to the vertices in C. An edge is said to exists between a pair of vertices (t_i,c_j) where $t_i\in T$ and $c_j\in C$ if $c_j\in C_i$. Let $q=|E|$ and let M be a maximum matching for G.

Lemma 5. *The maximum number of gateways that may be used in any assignment of sensors in T to gateways in C is equal to* $|M|$.

Proof. Let M be a maximum matching for the bipartite graph G. Let T_M and C_M denote the set of matched vertices corresponding to sensors and gateways, respectively. It is easy to see that each sensor in T_M can be assigned to the corresponding gateway in C_M to which it is matched, thus utilizing $|M|$ gateways in this partial assignment. Next we argue that the sensors in $T-T_M$ can only be assigned to the gateways in C_M. This in turn implies that the maximum number of gateways that may be used in any assignment is equal to $|M|$.

The argument is as follows. Since M is a maximum matching, there does not exist any augmenting path in G with respect to M. Hence each path that begins with an unmatched vertex $t\in T-T_M$ must terminates at some matched vertex $t^*\in T_M$ and each of the vertices on this path are matched vertices. Hence each vertex $t\in T-T_M$ is only be adjacent to the vertices in C_M. This in turn implies that each vertex $t\in T-T_M$ can only be assigned to one of the vertices in C_M. Hence the maximum number of gateways that may be used in any assignment is equal to $|C_M|=|M|$. □

Lemma 6. *There exists an optimal assignment that uses exactly* $|M|$ *gateways.*

Proof. Let X be an optimal assignment and $V_X\subseteq C$ denote the set of gateways utilized by X. Let $|V_X|=\delta$ and suppose that $\delta<|M|$. For each $v\in V_X$, let $U_v^X\subseteq T$ denote the set of sensors that are assigned to v (refer to Fig. 2a and b for an illustration). We next construct a bipartite matching as follows. For each gateway $v\in V_X$, match v to a vertex, say u, where $u\in U_v^X$. Let the resultant matching be denoted by M' (refer to Fig. 2c for an illustration). Clearly, $|M'|=\delta$. Since M' is not a maximum matching[1] in G, there must exists $|M|-\delta$ augmenting paths[2] G with respect to M'. Each augmenting path in G will begin at some unmatched vertex $v\in C$ (a gateway with zero load) which is adjacent to a vertex u (corresponding to a sensor) which has been assigned to a gateway, say w, using X, i.e. $w\in V_x$. We consider the following two possibilities.

Case (*i*): u is an unmatched vertex.
In this case, we will reassign u to v and removes it assignment to w. We note that there will be no increase in the overall maximum load following the reassignment. In addition, by matching u to v, the size of the bipartite matching will be increased by one.
Case (*ii*): u is matched to w.
Let the augmenting path that begins with v be denoted by $P_u=v-u-w-x$, where u, x and v, w denote sensors and gateways, respectively. In addition, we note that $u,x\in U_w^X$. Since we have matched u to w, x must be an unmatched vertex. Hence the gateway v may be assigned with a sensor as follows. Assign u to v and removes its assignment to w. Match u to v in G and removes its matching to w. We note that by reassigning u to v, there will not be any increase in the maximum load of the overall assignment. In addition, by matching x to w, the size of the matching (and the utilization of gateways) will be increased by one (refer to Fig. 2d for

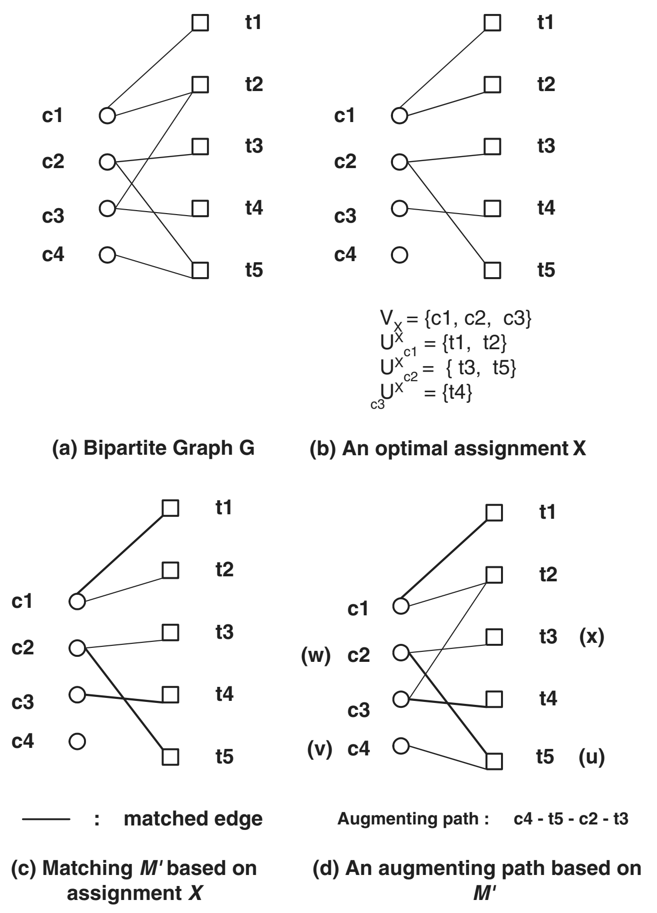

Fig. 2. An optimal assignment using $|M|$ gateways.

[1] A matching M on a graph G is a maximum matching if and only if there is no augmenting path in G with respect to M [12].

[2] An augmenting path with respect to M is one whose edges are alternately in M and not in M and the first and last vertices of the path are not incident to any edge of M.

an illustration). By repeating the above procedure for each augmenting path, we will establish a new optimal load-balanced assignment which utilizes exactly $|M|$ gateways. □

6.3. An approximation algorithm

The algorithm proposed in [10] and [11] relies on solving a linear programming relaxation of the problem and uses the information obtained from the solution to allocate jobs to machines. In this section we present a new algorithm, called the *Greedy Load-Balanced Clustering Algorithm (GLBCA)*, for LBCP that achieves a lower performance ratio than that of [11] without solving a linear program. In addition, our algorithm runs in $O(n[n+m+q])$ and is hence more efficient than the algorithm proposed in [11].

6.4. The Greedy Load-Balanced Clustering Algorithm (GLBCA)

Our proposed algorithm adopts the approach of utilizing the maximum number of gateways possible in the assignment of sensors to gateways in order to distribute the traffic load among as many gateways as possible. Based on Lemma 5, we know that the maximum number of gateways that may be used in any assignment is equal to $|M|$, where $|M|$ is the size of a maximum matching in the corresponding bipartite graph G. Hence our algorithm will attempt to find a maximum matching M in the graph G. We first sort the list of sensors in non-increasing order of traffic load. Let the resultant list be denoted by $T = \{t_1, t_2, \ldots, t_n\}$, where $d_1 \geqslant d_2 \geqslant \ldots \geqslant d_n$. Starting with the first sensor t_1 in the sorted list, we will attempt to match (or assign) t_1 to a gateway with zero load in the corresponding bipartite graph G. Next, the algorithm will proceed to match t_2 with another gateway with zero load in G. The algorithm will iterate in this manner where in each iteration, we will attempt to find an augmenting path P that connects a given sensor t_i to some unmatched gateway (with zero load) in G. If such a path is found, we will augment the edges in P which results in the assignment of t_i to some gateway in P and the reassignment of the other sensors to gateways in P. In addition the size of the resultant matching will be increased by one. If there does not exists any augmenting path that begins with t_i in G, then we will assign t_i to a least loaded gateway in C_i. The algorithm terminates when all sensors have been assigned to some gateway. The pseudocode of the algorithm is given in Table 2.

Table 2
Greedy Load-Balanced Clustering Algorithm

```
INPUT: A set of sensors $T = \{t_1, t_2, \ldots, t_n\}$, a set of gateways $C = \{c_1, c_2, \ldots, c_m\}$ and traffic load $\beta$ for each sensor $t_i$ .
OUTPUT: An assignment $A : T \rightarrow C$ such that $A(i) \in C_i$ and $l_{max}$ is minimized, where $l_{max} = \max_{j \in C} l(j)$ and $l(j) = |\{i \in T : A(i) = j\}|$.
Begin
Step1:
  for $j = 1$ to $m$ do
    set $l(j) = 0$;
  endfor
Step2: /* construction of BFS tree */
  for $j = 1$ to $n$ do
    set $l_{min} = \infty$;
    $Q = \{t_j\}$;
Step3:
    while ($Q \neq \emptyset$) and ($l_{min} > 0$) do
      let $v$ be the front element of $Q$;
      remove $v$ from $Q$;
Step4:
      if $v$ is a sensor then
        for each unmarked gateway $w$ onto which $v$ may be assigned do
          mark $w$;
          insert $w$ to the end of $Q$;
          set $pred(w) = v$;
        endfor
Step5:
      else /* $v$ is a gateway */
        if $l(v) < l_{min}$ then $l_{min} = l(v)$;
        for each sensor $w$ that was assigned to gateway $v$ do
          insert $w$ to the end of $Q$;
          set $pred(w) = v$;
        endfor
    endwhile
Step6: /* reassignment of sensors to gateways */
    let $v$ be a gateway with the least load;
    let $w = pred(v)$;
    assign sensor $w$ to gateway $v$;
    increase load of gateway $v$ by $\beta$;
    while $w \neq t_j$ do
      $v = pred(w)$;
      remove the previous assignment of sensor $w$ to gateway $v$;
      let $w = pred(v)$;
      assign sensor $w$ to gateway $v$;
    endwhile
  endfor
End
```

Lemma 7. *The time complexity of the proposed algorithm is $O(n[n+m+q])$.*

Proof. The sorted list in step 1 can be done in $O(n\log n)$. The initialization of the load of gateways in step 2 can be done in $O(m)$. The while loop in step 3 will iterates n times. In step 3.1, each augmenting path can be found in $O(n+m+q)$ using breadth-first search. The augmentation of the edges in step 3.2 can be done in $O(n+m+q)$; the reassignment of sensors to gateways and computation of the new load in step 3.3 can be done in $O(n+m)$. In step 3.4, the assignment of a sensor to a least loaded gateway can be done in $O(m)$ and the computation of the resultant load can be done in $O(1)$. Hence step 3 can be completed in $O(n[n+m+q])$. Thus, the overall complexity of the algorithm is $O(n[n+m+q])$. □

6.5. Performance ratio

Without loss of generality, we assume that the list of traffic loads $\{d_1, d_2, \ldots, d_n\}$ are all distinct. We will prove that our proposed algorithm is able to achieve a performance ratio of $\frac{3}{2}$ for LBCP.

Lemma 8. *The Greedy Load-Balanced Clustering Algorithm (GLBCA) is a $\frac{3}{2}$-approximation algorithm for LBCP and this bound is tight.*

Proof. Let W denote the sum of the traffic loads from all sensors, i.e. $W = \sum_{i=1}^{n} d_i$. Let OPT denote the maximum load of an optimal solution. Then it is clear that the sum of the traffic loads from all sensors is no more than $m \cdot \mathrm{OPT}$. Hence $\mathrm{OPT} \geqslant \frac{W}{m}$. In addition, it is easy to see that $\mathrm{OPT} \geqslant d_i \, \forall i$.

Let I be an instance with the smallest number of sensors such that an assignment of I which is obtained using proposed algorithm has a maximum load $>$ OPT. Let Φ denote this assignment. Let t_i be a sensor whose assignment to some gateway, say v^*, using Φ results in the overall maximum load of the assignment, i.e. $l(v^*) = \max l(v) \, \forall v \in C - \{v^*\}$, and $l(v^*) > \mathrm{OPT}$. Suppose that $i \neq n$. Consider an instance I' which is equal to I without sensor t_n. Then I' is a smaller instance of I for which the proposed algorithm computes an assignment with maximum load $>$ OPT. But this contradicts the choice of I. Hence we can assume that $i = n$. Note that this also implies that the load of each gateway prior to the assignment of sensor t_n to some gateway using Φ, is no more than OPT (otherwise we will again have a smaller problem instance with maximum load exceeding OPT).

Let A be an optimal assignment which uses the same number of gateways as Φ, i.e. $|M|$ (it follows from Lemma 6 that there exists such an optimal assignment). We first claim that $d_n \leqslant \frac{\mathrm{OPT}}{2} - \epsilon$, where ϵ is a small positive constant. Suppose otherwise and assume that $d_n > \frac{\mathrm{OPT}}{2} - \epsilon$. Since $d_i \geqslant d_n \, \forall i < n$, each gateway can be assigned with at most two sensors using A. Without loss of generality, we may assume that each gateway is assigned with two sensors (by adding dummy sensors with zero weight). We normalize the optimal assignment A as follows:

- for each gateway, place the sensor with the higher traffic load first;
- sort the gateways so that the first sensors assigned are in descending order of traffic loads. Let the resultant set of gateways be denoted by $\{v_1, v_2, \ldots, v_m\}$.

For each gateway v_q, let the first and second sensors assigned to v_q using A be denoted by t_q^1 and t_q^2, respectively. The corresponding traffic loads of t_q^1 and t_q^2 are denoted by d_q^1 and d_q^2, respectively. The assignment A may be further normalized as follows. Starting from $j = m$ *downto* 1, we compare the traffic load of t_j^1 with the traffic load of t_k^2 where $k < j$. Let t_h^2 be a sensor with highest traffic load among all sensors t_k^2, where $k < j$ which satisfies the following conditions, referred to as the *swapping conditions*:

- t_j^1 may be assigned to v_h;
- t_h^2 may be assigned to v_j;
- $d_h^2 > d_j^1$.

We note that by interchanging the assignment of sensors t_h^2 and t_j^1, we will have sensors t_h^2 and t_j^2 assigned to v_j and sensors t_h^1 and t_j^1 assigned to v_h. Since $d_j^2 \leqslant d_h^1$, $d_h^2 + d_j^2 \leqslant d_h^2 + d_h^1 \leqslant \mathrm{OPT}$. Similarly since $d_j^1 < d_h^2$, $d_h^1 + d_j^1 < d_h^2 + d_h^1 \leqslant \mathrm{OPT}$. Hence, we can interchange the assignment of sensors t_h^2 and t_j^1 and yet keep an optimal assignment (refer to Fig. 3 for an illustration). The resultant list of gateways (after considering all sensors t_j^1 for $j = m$ *downto* 1) is then sorted again so that the first sensors assigned are in descending order of traffic loads.

Following that, we will again check for the possibility of swapping the first sensor assigned to v_j with a second sensor assigned to another gateway which satisfy the above-mentioned swapping conditions for $j = m$ *downto* 1. If such a possibility exists, then the above-mentioned procedure is repeated. Otherwise, we next proceed the compare the traffic loads of the second sensors assigned to the gateways. Starting from $j = m$ *downto* 1, we compare the traffic load of t_j^2 with traffic load of t_k^2 where $k < j$. Let t_h^2 be a sensor with highest traffic load among all sensors t_k^2 where $k < j$, which satisfies the following conditions:

- t_h^2 may be assigned to v_j;
- t_j^2 may be assigned to v_h;
- $d_h^2 > d_j^2$.

We note that by interchanging the assignment of sensors t_h^2 and t_j^2, we will have sensors t_j^1 and t_h^2 assigned to gateway v_j and sensors t_h^1 and t_j^2 assigned to v_h. Since $d_j^2 < d_h^2$, $d_h^1 + d_j^2 < d_h^1 + d_h^2 \leqslant \mathrm{OPT}$. Similarly since $d_j^1 \leqslant d_h^1$, $d_j^1 + d_h^2 \leqslant d_h^1 + d_h^2 \leqslant \mathrm{OPT}$. Hence, we can interchange the assignment of sensors t_h^2 and t_j^2 and yet maintain an optimal assignment (refer to Fig. 4 for an illustration). By iterating exchanges of this kind for $j = m$ *downto* 1, it is easy to see that our proposed algorithm gives an assignment that is equivalent to this. But this contradicts that the assumption that the maximum load of an assignment obtained by proposed algorithm, i.e. Φ, is >OPT.

Hence $d_n \leqslant \frac{\mathrm{OPT}}{2} - \epsilon$. As noted earlier, the load of each gateway prior to the assignment of t_n to some gateway using Φ, is no more than OPT. Hence, following the assignment of t_n to some gateway by Φ, the overall maximum load $L \leqslant \mathrm{OPT} + d_n \leqslant \frac{3}{2}\mathrm{OPT} - \epsilon$.

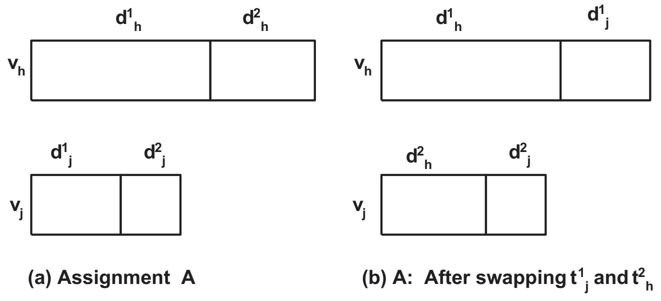

Fig. 3. Further normalization of assignment A.

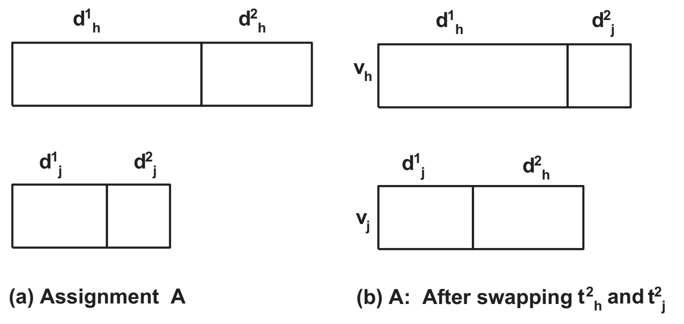

Fig. 4. Swapping the assignment of the second sensors in A.

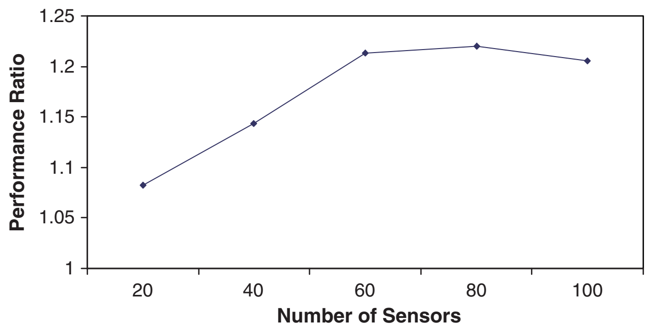

Fig. 6. Performance ratio of GLBCA.

We next show that the bound is tight using the following problem instance. Consider a problem instance whereby we have 2 gateways and 4 sensors with traffic loads of $5+\epsilon$, 5, $5-\epsilon$ and $5-\epsilon$, respectively. The assignment constraints of sensors to gateways are depicted in Fig. 5. Using the proposed algorithm, sensor t_1 will be assigned to gateway c_1 while sensors t_2, t_3 and t_4 will be assigned to gateway c_2 giving an overall maximum load of $15-2\epsilon$. However an optimal assignment will assign t_1 and t_2 to c_1 and assign t_3 and t_4 to c_2 and the overall maximum load is $10+\epsilon$. The solution obtained by our algorithm is a factor 1.5 worse than the optimum. □

7. Simulation results

We study the performance of GLBCA by comparing its solutions with optimal solutions which are obtained by solving the ILP formulated program using CPLEX. In our empirical studies, we consider the scenario of a sensor network whereby the sites for the gateways and sensors are generated randomly on a grid plane of 200×200, where each grid represents 10×10 square metre. The traffic load from each sensor is randomly selected from the range of 100–500 Kbps. We assume that a connection can be established between a gateway and a sensor if the distance between them is no larger than 550 m.

Fig. 6 shows the performance of GLBCA by varying the number of sensors (which ranges from 20 to 100) while the fixing the number of gateways at 20. For each data point in Fig. 6, 50 runs are taken and the average calculated. We observe that the solutions obtained using GLBCA differs from the optimal values by no more than 25%. In particular, we observe that the performance ratio of GLBCA ranges between 1.08 and 1.23 in all instances generated. This in turn implies that the average-case performance of our proposed algorithm is much better than the worst-case performance ratio derived.

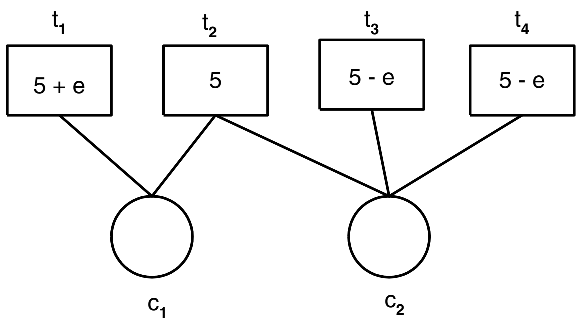

Fig. 5. A problem instance: performance bound is tight.

8. Conclusion

In this paper, we address a problem that arise in the design of cluster-based wireless sensor networks. In particular, we consider the problem of assigning sensors to gateways in a wireless sensor network with the objective of distributing the traffic load among the gateways so as to ensure that no gateway is overloaded. We show that this problem is optimally solvable in polynomial time if all sensors have Uniform Traffic Load. However, the problem turns out to be NP-hard if the sensors have differing traffic loads. We proposed an approximation algorithm for the NP-hard problem and prove that our proposed algorithm is able to guarantee a performance ratio of $\frac{3}{2}$. Empirical studies have shown that our proposed algorithm is able to perform much better on the average as compared to the worst-case performance ratio derived. Hence one direction for future research is to analyse the average-case performance of our proposed algorithm. Our proposed algorithm adopts a centralized approach which assume that each node is aware of the network topology. Another direction for future work is to develop a distributed algorithm which would be more scalable for the design of cluster-based sensor networks.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (2002) 393–422.
[2] M. Younis, M. Youssef, K. Arisha, Energy-aware routing in cluster-based sensor networks, in: Proc. IEEE/ACM Intl. Symp. on Modeling, Analysis and Simulation of Computer Systems (MASCOTS 2002), TX, USA, 2002.
[3] A. Cerpa, D. Estrin, ASCENT: Adaptive self-configuring sensor networks topologies, in: Proc. INFOCOM 2002, New York, June 2002.
[4] W. Rabiner Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocols for wireless microsensor networks, in: Proc. Hawaii Intl. Conf. on System Sciences (HICSS 00), January 2000.

[5] D.J. Baker, A. Emphemides, A distributed algorithm for organizing mobile radio telecommunication networks, in: Proc. Intl. Conf. in Distributed Computer Systems, April 1981.
[6] M. Gerla, J.T.C. Tsai, Multicluster, mobile, multimedia radio network, ACM/Baltzer, Journal of Wireless Networks 1 (1995) 255–265.
[7] A.K. Parekh, Selecting routers in Ad-hoc wireless networks, in: Proc. SBT/IEEE Intl. Telecommunications Symposium, August 1994.
[8] R. Krishnan, D. Starobinski, Efficient clustering algorithms for self-organizing wireless sensor networks, Ad Hoc Networks 4 (2006) 36–39.
[9] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[10] J.K. Lenstra, D.B. Shmoys, E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, Mathematical Programming 46 (1990) 259–271.
[11] E.V. Shchepin, N.V. Vakhania, Task distributions on multiprocessor systems, in: Lecture Notes in Computer Science, vol. 1872, 2000, pp. 112–125.
[12] C. Berge, Two theorems in graph theory, Proceedings of the National Academy Science of United States America 43 (1957) 842–844.