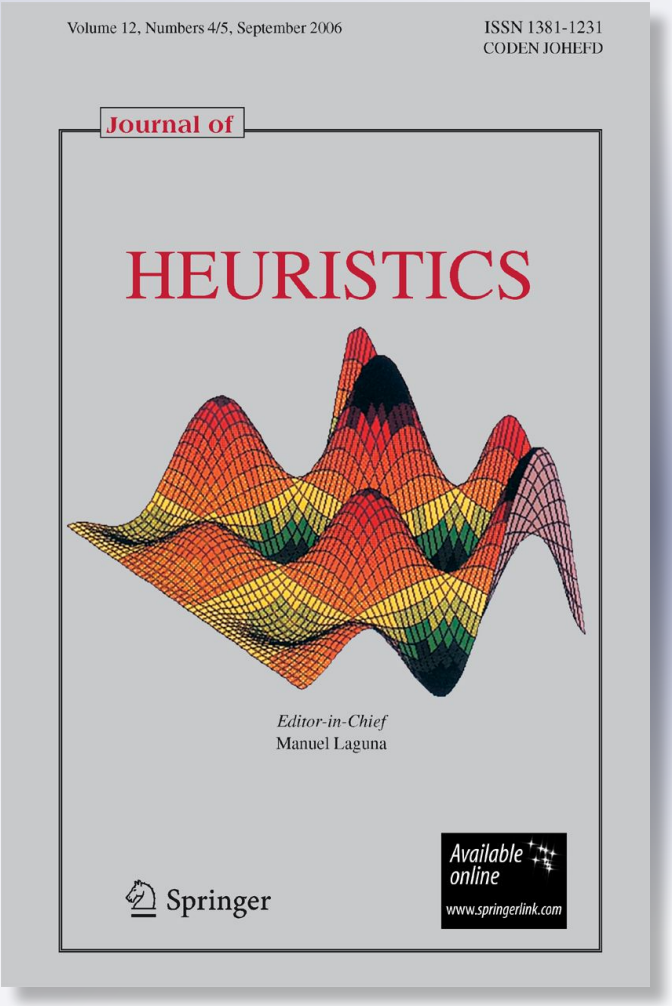
Strategies for designing energy-efficient clusters-based WSN topologies

Andréa Cynthia Santos, Christophe Duhamel, Lorena Silva Belisário & Lucas Moreira Guedes

Journal of Heuristics

ISSN 1381-1231
Volume 18
Number 4

J Heuristics (2012) 18:657-675
DOI 10.1007/s10732-012-9202-x



Springer

J Heuristics (2012) 18:657–675
DOI 10.1007/s10732-012-9202-x

Strategies for designing energy-efficient clusters-based WSN topologies

Andréa Cynthia Santos · Christophe Duhamel · Lorena Silva Belisário · Lucas Moreira Guedes

Received: 21 October 2011 / Accepted: 10 May 2012 / Published online: 24 May 2012

Abstract Wireless Sensor Networks are used in several practical applications such as environmental monitoring and risk detection. In this work, we deal with the problem of organizing the network topology into clusters in order to minimize the total energy consumption. The problem is modeled as an Independent Dominating Problem with Connecting requirements. We first present a state-of-the-art on the problems to optimize energy consumption in WSN. Then, we propose a mixed integer linear programming formulation, constructive heuristics, a local search procedure, and a GRASP-based metaheuristic. Results are provided for large scale WSN instances.

Keywords Wireless Sensor Network · Clusters-based topologies · Metaheuristic · GRASP

A.C. Santos (✉) · C. Duhamel
ICD-LOSI, UMR CNRS STMR, Université de Technologie de Troyes 12, rue Marie Curie, B.P. 2060, 10010, Troyes Cedex, France
e-mail: andrea.duhamel@utt.fr

C. Duhamel
e-mail: christophe.duhamel@utt.fr

L.S. Belisário
Departamento de Engenharia de Produção, Federal University of Minas Gerais, Av. Antônio Carlos, 6627, CEP 30161-010, Belo Horizonte, MG, Brazil
e-mail: lorenabelisario@gmail.com

L.M. Guedes
Graduate Program in Electrical Engineering, Federal University of Minas Gerais, Av. Antônio Carlos, 6627, CEP 31270-010, Belo Horizonte, MG, Brazil
e-mail: lucasguedes@cpdee.ufmg.br

1 Introduction

Wireless Sensor Network (WSN) has been widely investigated in the last years due to its technical challenges and the large number of applications. Potential WSN applications have emerged such as environmental monitoring, risk detection for natural and human disasters, fine-tuning agricultural inputs, and intelligent transportation.

A WSN contains a set of sensors which communicate on a logical network defined by radio transmission. WSNs differ from classic networks in many aspects, for instance: (i) sensors are limited in energy, in storage capacity and in processing, (ii) WSN requirements can change with the application, (iii) the traditional Internet Protocol (IP) does not apply due to the identity address cost maintenance, (iv) collecting data is sometimes more important than identifying who sent it, (v) the network topology changes along its lifetime (ad hoc structure), (vi) the flow of messages is non-conservative for cluster-based topologies, etc.

Some problems as the design of Media Access Control WSN protocols (Abbasi and Younis 2007) and of routing techniques (Al-Karaki and Kamal 2004) have been widely studied in the literature. Simulation tools are usually applied to analyze those strategies. However, one drawback of simulation is that it only evaluates the interest of a solution, it does not aim at providing the best possible solution. In fact, some inherent WSN issues rely on optimization problems. For instance, the energy consumption is a major concern in WSN due to the sensors limited resources. Energy consumption in a WSN can be optimized at different levels: the physical level (the circuits design), the logical level (the network architecture, the communication protocols, and the strategic WSN organization) and by an integration of logical and physical levels (to activate sensors or not). We focus on the strategic WSN organization by designing cluster-based topologies which minimize the energy consumption. This problem is modeled as an Independent Dominating Set Problem with Connecting requirements (IDSC). We propose a Mixed Integer Linear Programming (MILP) formulation, heuristics, and a GRASP-based metaheuristic. Three sets of instances are used to evaluate the proposed strategies.

The work is organized as follows: we first present a state-of-the-art on WSNs optimization issues in Sect. 2. Then, the IDSC problem is defined in Sect. 3. A MILP formulation is proposed in Sect. 4. Heuristics and a metaheuristic are respectively detailed in Sects. 5 and 6 followed by the computational results in Sect. 7. Finally, we present concluding remarks and perspectives in Sect. 8.

2 State-of-the-art

Different models can be used to optimize energy in WSN. They mainly depend on the WSN characteristics such as the application needs, the number of sensors, the sensors device features, the type of data detection, the number of sinks and the mobility. We briefly present below three main classes of WSN optimization problems, when it is more interesting to apply each of them, and the main approaches available in the literature.

Minimizing the energy consumption can be addressed as providing optimal network coverage over the time (Meguerdichian and Potkonjak 2003; Rossi et al. 2011a,

2011b). The general idea is to put a set of sensors in a sleeping mode while ensuring a minimal coverage, and to schedule a subset of active sensors over the time. By doing so, energy consumption is minimized. Such a model is very useful when one aims at eliminating redundancy to cover a target. A non-linear mathematical model, a column generation, and a metaheuristic have been proposed to determine covers and scheduling in Rossi et al. (2010, 2011a). Integer linear programming formulations using a time period discretization are presented in Aioffi et al. (2007), Meguerdichian and Potkonjak (2003). The formulation found in Aguiar et al. (2008) is a generalization including heterogeneous sensors and connecting requirements. Strategies which couple the coverage requirements with a mobile sink are introduced in Aioffi et al. (2007). The mobile sink defines a topology in clusters or trees. Coverage using a distributed strategy is investigated in Li et al. (2002). The authors propose geometric algorithms based on Delaunay triangulation.

Another way to improve the network lifetime is to allow varying the sensing ranges. The idea is to adjust the sensors range. In this approach, the sensing radii for active sensors are minimized, while coverage is guaranteed for all targets (or target areas). This model is useful when dealing with powerful radio range devices, a random deployment and known target locations. Thus, controlling the transmission range for active sensors can reduce the amount of energy required. A formulation to maximize the number of set covers, heuristics methods and simulations to validate the proposed strategies are presented in Cardei et al. (2005). Connecting requirements are not taken into account though. An extended formulation for this problem which maximizes the network lifetime and heuristics to schedule covers are proposed in Dhawan et al. (2006). The authors consider sensors with non-uniform batteries (heterogeneous sensors). A multi-objective approach to minimize both the coverage and the energy consumption is studied in Jia et al. (2009). The authors use a Pareto function embedded within a genetic algorithm. The proposed strategy works for heterogeneous sensors. Moreover, the energy consumption for sensing is considered, but the strategy does not consider energy consumption for other activities (transmitting data, dissipation, etc.).

The design of topologies for WSN can also be defined to maximize the network lifetime. The problem we consider in this paper belongs to this class. The general goal of such problems is to define a logical communication structure (topology) for ad hoc WSNs that consume as little energy as possible and that ensure topological constraints. In general, the methods developed for this class of problems are able to quickly compute a new topology when sensors run out of energy. Thus, they are able to handle the possible changes in ad hoc WSNs along its lifetime. Works (Hajiaghayi et al. 2007; Moraes et al. 2009) provide solutions for designing WSN topologies such that the amount of energy assigned to each node is minimized, while disjoint paths between each pair of nodes are guaranteed. This approach ensures alternative paths to transmit messages in the network and thus improves its robustness. The objective in Li et al. (2003) is to determine topologies on unit disk graphs. The authors model the problem as a minimum connected dominating set and a polynomial-time approximation algorithm is proposed. Computational results are not reported.

The design of WSN in clusters-based topologies to minimize energy consumption has been independently developed in Hurinka and Niebergb (2008), Santos et

al. (2009). A cluster is a hierarchical way to organize the sensors. It contains a master node, slaves and bridges nodes. A master coordinates a cluster (it collects, aggregates and sends messages), slaves perform sensing activities and bridges establish inter-cluster communication. This structure is particularly suited for applications with highly-correlated data. The data can be aggregated in the master, thus reducing the number of messages to be sent. An analysis of the applications for cluster-based structures is found in Vlajic and Xia (2006). An approximation algorithm is proposed in Hurinka and Niebergb (2008) and the work focuses on the theoretical issues of the proposed strategy. The approximated algorithm is composed of two steps: an infeasible initial solution is first computed and then it is repaired. Computational results are not reported. We consider the same core problem, i.e. finding a minimum independent dominating set, but, we have integrated the connecting requirements in the model.

Our first ever work to the IDSC problem is found (Hou et al. 2008). We have initially analyzed the amount of energy needed to perform sensing activities, to receive a message and to forward a message. Thus, the energy level of each sensor node is transformed into a maximal amount of messages the node can handle. Topologies are computed while the optimization relies on a maximum flow. The method has been tested on small size instances with up to 300 nodes. Unfortunately, this strategy seems to be difficult to be applied in realistic scenarios. Properties, theorems and graphs with known optimal solutions for the IDSC have been investigated in Santos et al. (2009). Constructive heuristics are suggested and results are reported for instances with up to 500 nodes. The proposed strategies are proved to find optimal solution for some particularly graphs like a regular grid, which are very useful for agriculture applications. Improved heuristics are presented in Belisario et al. (2010) for WSN with up to 1,000 nodes.

Here, we propose new strategies with an emphasis on providing simple, efficient and scalable methods for solving the IDSC problem : constructive heuristics, a local search procedure and a metaheuristic. Solutions are provided for large scale WSN with up to 20,000 sensors. Another contribution is a MILP formulation based on spanning trees which is used to evaluate the performance of the heuristic strategies.

3 The IDSC problem

Let $G = (V, E)$ be a communication graph with a set V of vertices and a set E of edges (if $[i, j] \in E$, sensors i and j can communicate). An independent set $I \subseteq V$ of G contains only vertices not adjacent to each other. $D \subseteq V$ is a dominating set of G if every vertex in $V \setminus D$ is adjacent to at least one vertex of D. Thus, the set $M \subseteq V$ of masters must be an independent dominating set of G. Since $S = V \setminus M$ is the set of slaves, M and S define a partition of G, i.e. $(M \cap S = \phi)$ and $(V = M \cup S)$. The bridges correspond to slaves connected to at least two masters. They can be implicitly deduced from S.

The IDSC problem consists in defining an independent dominating set M such that there is at least one path from each master to the sink, hence the connecting requirements. The IDSC problem is NP-hard as shown in Clark et al. (1991). Minimizing the

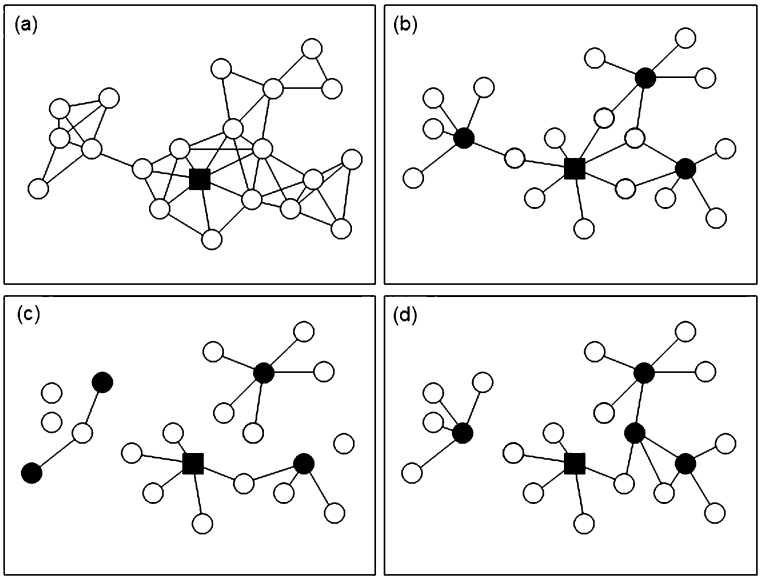


Fig. 1 Examples of feasible (**b**) and infeasibles (**c**) and (**d**) WSN topologies

energy consumption for IDSC is done by first minimizing the number of clusters and second by minimizing the hop average. Few clusters reduce the amount of messages sent to the sink. Moreover, shorter paths from the masters to the sink require fewer inter-cluster communications. Routing a message in a cluster-based topology is done by paths which alternate masters and bridges. Thus, more energy is left to sensing activities and the topology lifetime is increased. The lifetime is here considered as the first time an area is disconnected, mostly because a master or a bridge has run out of energy. In such a case, the topology can be quickly repaired by setting a slave to replace the failing node, and propagating the modifications.

An example of a communication graph G is shown in Fig. 1(a). The black square corresponds to the sink. Figure 1(b) shows a feasible WSN topology for the IDSC problem, where the black, the white and the gray nodes are respectively the set of masters, slaves and bridges. Figure 1(c) illustrates an independent set (the black nodes) of G. It is not a feasible solution because some nodes are not connected to a master and the topology is disconnected. Figure 1(d) displays a dominating set of G (the black nodes) that is not a feasible solution since some masters are directly connected, and the topology is disconnected.

4 A MILP formulation for the IDSC problem

Two optimization criteria are considered for the IDSC problem: minimizing the number of masters and minimizing the average number of hops. The first one is a priority objective. Thus, we propose a two-step optimization approach. In the first step, the first criterion is addressed to produce an optimal number of master. Then, the second

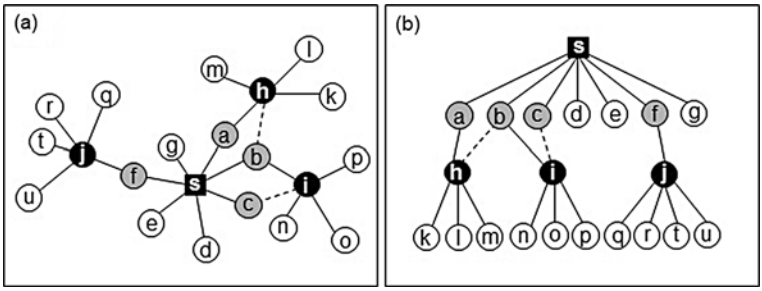


Fig. 2 A cluster-based topology solution in a tree structure

step considers the second optimization criterion, while imposing the previously computed minimal number of clusters. Since a cluster can be identified by its master, the number of clusters equals the number of masters (sink included).

Given a connected communication graph $G = (V, E)$, there is at least one embedded spanning tree $T = (V, E')$ in G, $E' \subseteq E$ and $|E'| = |V| - 1$. The sink node s is considered as a master. Thus, the general idea of the proposed model is to find a spanning tree of G such that the total number of nodes at the even levels (master nodes) of T and the average number of hops are minimized. Figure 2 illustrates an example of a solution for the IDSC problem represented by a tree. The topology in Fig. 2(a) is also represented in hierarchical levels from the sink toward the network border in Fig. 2(b). As before, the black square correspond to the sink, and the black, the gray and the white nodes are respectively the masters, the bridges and the slaves. Bridges can be deduced in a post-optimization phase since they correspond to slaves in the radio range of at least two masters.

A network flow structure is used to guarantee the connecting requirements and it is defined over a directed graph. Thus, the digraph $G' = (V, A)$ is obtained from G by transforming every edge $[i, j] \in E$ in two arcs (i, j) and (j, i) belonging to A. The sink s sends one flow unit to each node (the sensors) and each node consumes one unit. Let $x_i \in \{0, 1\}$ be the decision variable to set i as a master or not. Let $f_{ij} \geq 0$ be the flow variable on arc $(i, j) \in A$. The first optimization step for the IDSC is given as follows:

$$\min Z = \sum_{i \in V} x_i \tag{1}$$

$$\text{s.t.} \quad \sum_{(i,j) \in A} f_{sj} = |V| - 1 \tag{2}$$

$$\sum_{j:(j,i) \in A} f_{ji} - \sum_{j:(i,j) \in A} f_{ij} = 1 \quad \forall i \in V \backslash \{s\} \tag{3}$$

$$f_{ij} \leq \big(|V| - 1\big)(x_i + x_j) \quad \forall (i, j) \in A \tag{4}$$

$$x_i + x_j \leq 1 \quad \forall (i, j) \in A \tag{5}$$

$$x_s = 1 \tag{6}$$

$$x_i \in \{0, 1\} \quad \forall i \in V \tag{7}$$

$$f_{ij} \geq 0 \quad \forall (i, j) \in A \tag{8}$$

The objective function 1 aims at minimizing the total number of masters. Constraints 2 ensure the sink node sends $|V| - 1$ units of flow (one for each node of G). Restriction 3 guarantee the flow conservation and state that each node consumes one unit of flow. Constraints 4 allow flow on an arc only if one of its extremities is a master. Inequalities 5 forbid more than one extremity of an arc to be a master. Variables x_i and f_{ij} are respectively defined in Eqs. 7 and 8.

The connecting requirements are ensured by the flow constraints. Moreover, constraints 5 ensure the set of masters is an independent set. As a consequence of constraints 4 and 5, the set of masters is also a dominating set.

Let Z^* be the optimal number of masters found in the first optimization step. Let $f'_{ij} \geq 0$ be the amount of flow to send from the sink to the masters, one unit for each master. The general idea of the second optimization phase is to use this auxiliary flow from the sink to the masters. Then, the average number H of hops to the masters can be expressed as:

$$\min H = \sum_{(i,j)\in A} f'_{ij}/Z^* \tag{9}$$

$$\text{s.t.} \quad \sum_{i \in V} x_i = Z^* \tag{10}$$

$$\sum_{j:(j,i)\in A} f'_{ji} - \sum_{j:(i,j)\in A} f'_{ij} = x_i \quad \forall i \in V \backslash \{s\} \tag{11}$$

Constraints 2–8

$$f'_{ij} \geq 0 \quad \forall (i, j) \in A \tag{12}$$

Equation 10 imposes the optimal number of masters found in the first optimization phase. Constraints 11 guarantee the topology is connected. Constraints 2–8 correspond to the requirements of the IDSC core. Finally, specific variables to the second optimization phase are defined in Eq. 12. Note that the constant in the objective function could be dropped as well.

5 Heuristic strategies

The constructive heuristics we propose differ on the way to extend the partial current solution and on the way to select the master at each iteration. In the first constructive approach, denoted here as "DOWN", solutions are build from the sink towards the network border. The second one, denoted as "UP", builds solutions from the network border towards the sink. Both are detailed in this section.

Several strategies to select the master candidate at each iteration are proposed. In the "MEMORY" approach, a memory mechanism is used by setting a weight to each node. Each iteration a candidate is not chosen as master, its weight increases

of one unit. Thus, the idea is to select as a master the oldest candidate not selected along the heuristic iterations. The advantage of such criterion is to diversify solutions and a drawback is that the memory mechanism works once several iterations have been performed. In the "NEIGHBOR" strategy, the selected master candidate is the one with the largest number of neighbors not yet belonging to the current topology. This type of criterion is inspired by the classic greedy heuristics. In the context of the IDSC problem, it means that the largest is the number of neighbors the less is the number of clusters.

Both master selection mechanism have been applied with DOWN and UP approaches. In the DOWN constructive heuristic, a new candidate (master) is inserted in M at each iteration, starting from the sink, and its neighbors enter S. Thus, the candidates to become masters are the nodes that do not belong to $M \cup S$ and which are adjacent to at least one node $i \in S$, i.e. $\{j \in V \backslash (M \cup S) \mid \exists [i, j] \in E, i \in S\}$. Let N_i, the neighborhood of i, be the set of nodes connected to node i. A pseudo-code for this procedure is presented in Algorithm 1.

Input: $G = (V, E)$, s
Output: $T = (V', E')$, M

1 $M \leftarrow \{s\}$;
2 $S \leftarrow$ all neighbors of N_s;
3 $V' \leftarrow M \cup S$;
4 $E' \leftarrow \{[s, i] \mid \text{for all } i \in N_s\}$;
5 update candidate list;
6 **while** $(|V'| \neq |V|)$ **do**
7 select m from the candidate list;
8 $M \leftarrow M \cup \{m\}$;
9 $S \leftarrow S \cup N_m$;
10 $V' \leftarrow M \cup S$;
11 $E' \leftarrow E' \cup \{[m, i] \mid \text{for all } i \in N_m\}$;
12 update candidate list;
13 **end**
14 **return** T, M;

Algorithm 1: The DOWN constructive procedure

Given a communication graph G and the sink node s, the procedure returns a topology $T = (V', E')$ and the set M of masters. The initialization is done in lines 1 to 4 by respectively selecting the sink node s, updating the set of slaves S and the topology T. Then, candidates are selected in line 5 as mentioned above. The loop from 6 to 13 iteratively adds a new master to the topology. It stops when each node is set as a master or as a slave. The selection of a master from the candidate list is done in line 7 by using one of the criteria mentioned above (MEMORY or NEIGHBOR).

The UP constructive heuristic works differently. It builds initial solutions from the network border towards the sink. We use the property that only nodes in even levels

can become masters. Thus, the procedure to update the candidate list selects candidates with the highest even level. A pseudo-code is given in Algorithm 2. Initially, the sink is set in level zero in line 1 and a Breadth First Search (BFS) is applied on the communication graph to compute the smallest level of each node from the sink in line 2. Thus, the maximum even level of G is computed in lines 3 to 6. The candidate list is updated in line 7. It selects only nodes in the maximum even level. The loop from 8 to 15 is performed while there are still candidates to enter the solution. A candidate is selected as master in line 9 by using one of the two selection criteria mentioned above (MEMORY or NEIGHBOR). Thus, the set M of masters and the set S of slaves are respectively updated in lines 10 and 11. The topology is updated in lines 12 to 13 as well as the candidate list in line 14. Since some nodes might be left disconnected, a repair procedure is performed. It consists in connecting those nodes as masters and in updating the topology accordingly (lines 16 to 18).

Input: $G = (V, E)$, s
Output: $T = (V', E')$, M

1 $level(s) \leftarrow 0$;
2 apply a BFS(G) and set the level of each node $i \in V$;
3 max_level $\leftarrow$ the highest level of $i \in V$;
4 **if** (max_level is odd) **then**
5 | max_level $\leftarrow$ max_level - 1;
6 **end**
7 update candidate list;
8 **repeat**
9 | select m from the candidate list;
10 | $M \leftarrow M \cup \{m\}$;
11 | $S \leftarrow S \cup N_m$;
12 | $V' \leftarrow M \cup S$;
13 | $E' \leftarrow E' \cup \{[m, i] \mid \text{for all } i \in N_m\}$;
14 | update candidate list;
15 **until** ($candidate\ list \neq \emptyset$);
16 **if** ($|V| \neq |V'|$) **then**
17 | apply a repair procedure;
18 **end**
19 **return** T, M;

Algorithm 2: The UP constructive procedure

6 A GRASP for the IDSC problem

The Greedy Randomized Adaptive Search Procedure (GRASP) has been proposed by Feo and Resende (1989, 1995). It basically consists in building at each iteration

an initial solution using a randomized constructive heuristic, then in improving it by a local search. The best solution found is kept. GRASP is especially interesting as it only requires two components (a randomized heuristic and a local search) and very few parameters.

The first component in a GRASP is the constructive randomized heuristic. The NEIGHBOR heuristic using DOWN and UP constructive strategies has been randomized by using a Restricted Candidate List (RCL) instead of the complete candidate list. Let C_i be the number of neighbors at node i which do not already belong to the current solution. C_{max} and C_{min} denote respectively the maximal and the minimal values of C_i among all the candidates. Let $\alpha \in [0, 1]$ be the RCL parameter which controls the greediness of the selection. The selection is based on the RCL. It contains the candidates i such that the condition 13 holds. Then, a candidate is randomly chosen in the RCL. When $\alpha = 1$, the selection is greedy and one looks for candidates with the largest number of neighbors. When $\alpha = 0$, the selection is randomly done over all the candidates.

$$C_i \geq (1 - \alpha)(C_{min} - C_{max}) + C_{max} \tag{13}$$

We did not apply the RCL with the MEMORY mechanism because both features enhance the solutions diversity at solutions. They are different since the MEMORY is deterministic and works on the accumulated history, while the RCL is a memoryless stochastic process.

The second important component of a GRASP is the local search. As the IDSC encloses several features (independent set, dominating set and connecting requirements), defining moves for the local search is not a trivial task. In fact, transforming a slave into a master is not a good idea because it will systematically lead to an infeasible solution (a set which is neither independent nor dominating). Bridges nodes are more valuable targets to define a move since they connect two masters or more and since the number of masters can be reduced. Thus, a move in the local search consists in transforming a bridge into a master and its neighborhood is updated accordingly. Such a move is accepted since it reduces the number of masters and whenever the resulting WSN topology is connected. An example of such move is given in Fig. 3.

Figure 3(a) depicts a feasible topology where black, gray and white nodes stand for masters, bridges and slaves. The nodes i and k in Fig. 3(b) are bridges candidates to become master. A move is illustrated in Fig. 3(c) where node k is transformed into a master. In Fig. 3(d), a move using node i is rejected because the resulting topology is disconnected.

The local search uses a first improvement strategy. The procedure starts by initially computing the list of all the bridges not connected to the sink. Then, they are considered and selected in decreasing order of number of neighbors (masters) in the current solution. The selected bridge is transformed into a master and each connected master is transformed into a slave. Those new slaves become bridges whenever they are in the communication radius of others masters. The topology is then updated accordingly. The disconnected slaves are connected to a master in their neighborhood, whenever it exists. Finally, a BFS is applied to check the global connectivity. If the new solution is feasible, the move is done and the new solution is necessarily better

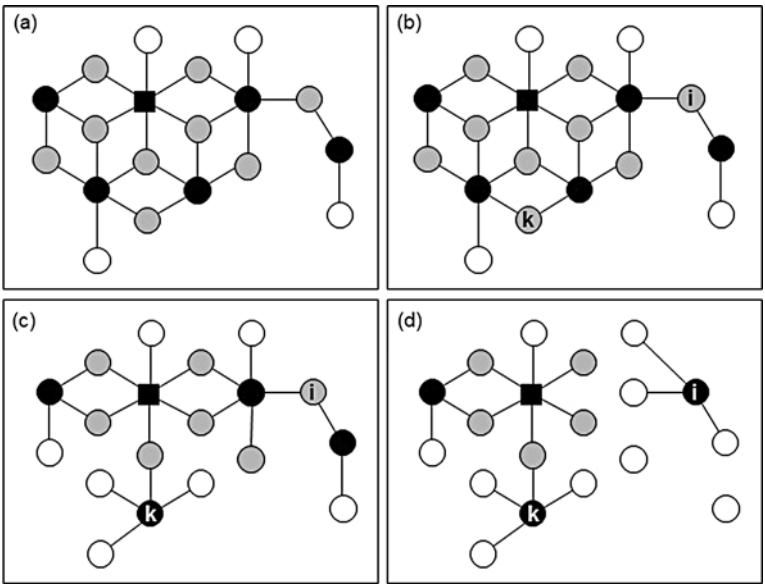


Fig. 3 Example of feasible and infeasible moves

Input: stopping criterion, *seed*
Output: T^*

```
1 while (stopping criterion not met) do
2 |   T ← greedy randomized constructive heuristic solution;
3 |   apply a local search procedure to T;
4 |   if (T is the best solution found so far) then
5 |   |   update best solution T*;
6 |   end
7 end
8 return T*;
```

Algorithm 3: GRASP pseudo-code

than the current one since the number of masters is reduced. The process stops when the current solution cannot be improved after investigating all moves.

A general view to the GRASP procedure is given in the Algorithm 3. The loop in lines 1 to 7 is repeated while the stopping criterion is not met. The constructive random heuristic and the local search procedures are called in lines 2 and 3, respectively. The procedure checks in line 4 if the best solution has been improved. If so, the best solution is updated in line 5. The best solution found so far is returned by the procedure (line 8).

7 Computational experiments

Experiments were carried out on an Intel Core Duo with 3.00 GHz clock and 8 Gb of RAM. The algorithms were developed in C. Three test sets have been generated to evaluate both the quality and the performance of the approaches. The first one contains 19 instances from 100 to 1,000 nodes randomly located in a 100×100 m^2 area and with a 20 m radio range. The second set is made of 5 instances of 200 nodes for which the radio range varies from 20 m to 60 m. The third set contains 8 instances from 5,000 to 20,000 nodes, randomly located in a 500×500 m^2 area and with a 20 m radio range. Results are presented in the following subsections: the first one 7.1 contains the calibration and comparison experiments for the constructive heuristics. In Sect 7.2, experiments for the GRASP are reported.

7.1 Constructive heuristics

A calibration for the constructive heuristics NEIGHBOR DOWN and NEIGHBOR UP has been done using different values for the parameter $\alpha = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. For each run using a specific α value, 1,000 iterations have been performed. "Score" and "average" ranks are used to evaluate the impact of this parameter over both heuristics, considering the three test sets. The score denotes how many times a heuristic found the best overall result using the corresponding α parameter. The average rank is computed in two steps. First, a classification is done for each heuristic and each instance. Whenever a heuristic has found the best result for an instance, it is set in the first rank and it is assigned a value $= 1$. The second best values are assigned a value $= 2$, and this procedure is repeated until every result has been classified. The average rank for a heuristic is equal to the arithmetic average classification values over all instances. If the average rank is equal to one, the heuristic found the best results for all instances for a test set, considering a specific α value.

Table 1 summarizes results for the NEIGHBOR DOWN heuristic. Results for tests sets 1 and 2 are quite similar: $\alpha = 0.8$ performs better than the others. However, results indicate the greedy version ($\alpha = 1.0$) becomes very interesting for the test set 3. Concerning the average rank for this test set, $\alpha = 1.0$ and $\alpha = 0.8$ have given respectively the first and the second best results. Table 2 depicts results for the NEIGHBOR UP heuristic. This heuristic behaves differently according to the α values. For the

Table 1 Results for the NEIGHBOR DOWN heuristic

Instances	Ranks	α values					
		0.0	0.2	0.4	0.6	0.8	1.0
Test set 1	score	4/19	8/19	10/19	12/19	**18/19**	4/19
	average	1.95	1.68	1.52	1.36	**1.05**	2.37
Test set 2	score	3/5	3/5	4/5	3/5	**5/5**	0/5
	average	1.60	1.40	1.20	1.40	**1.00**	2.60
Test set 3	score	0	0	0	0	0	**8/8**
	average	6	5	4	3	**2**	**1**

Table 2 Results for the NEIGHBOR UP heuristic

Instances	Ranks	α values					
		0.0	0.2	0.4	0.6	0.8	1.0
Test set 1	score	8/19	11/19	14/19	**15/19**	12/19	3/19
	average	1.58	1.47	1.26	**1.21**	1.37	2.37
Test set 2	score	5/5	5/5	5/5	5/5	5/5	3/5
	average	1.00	1.00	1.00	1.00	1.00	1.40
Test set 3	score	0	0	0	0	0	**8/8**
	average	6	5	4	3	**2**	**1**

Table 3 Comparison between the constructive heuristics

Instances	Ranks	NEIGHBOR DOWN		NEIGHBOR UP		MEMORY DOWN	MEMORY UP
		($\alpha = 0.8$)	($\alpha = 1.0$)	($\alpha = 0.6$)	($\alpha = 1.0$)		
Test set 1	score	**17/19**	4/19	8/19	1/19	4/19	2/19
	average	**1.16**	2.74	1.79	3.21	2.00	2.26
Test set 2	score	**5/5**	0/5	0/5	0/5	3/5	3/5
	average	**1.00**	3.00	2.00	2.80	1.40	1.80
Test set 3	score	0	**8/8**	0	0	0	0
	average	2	**1**	4	3	5	5.63

test set 1, $\alpha = 0.6$ produces the best results while for test set 2, the greedy ($\alpha = 1.0$) strategy performs worse than the others. Results for the test set 3 are similar to those obtained for NEIGHBOR DOWN.

The calibration results indicate the NEIGHBOR DOWN heuristic performs better with parameters $\alpha = 0.8$ and $\alpha = 1.0$, while for the NEIGHBOR UP heuristics $\alpha = 0.6$ and $\alpha = 1.0$ are the best. A comparison among the NEIGHBOR DOWN and NEIGHBOR UP using the results of the calibration, and the MEMORY UP and MEMORY DOWN heuristics is given in Table 3. Results produced by the NEIGHBOR DOWN heuristic dominate the others. The impacts of using the setting ($\alpha = 0.8$) and ($\alpha = 1.0$) are similar to the previous results.

7.2 Results for the GRASP

The computational experiments for the proposed GRASP are described below. Experiments with the proposed MILP formulation are reported for small and medium size instances (first and second test sets). The first test set is used to check the limit for the MILP formulation. The optimal solutions computed by CPLEX 12 solver is used to evaluate the absolute performance of GRASP. Following the calibration experiments, the NEIGHBOR DOWN heuristic is used. Moreover, $\alpha = 0.8$ is set for small and medium size instances (test sets 1 and 2), while $\alpha = 1$ is applied for large size instances (test set 3). Five runs were done under different seeds $= \{8, 12, 67, 100, 259\}$, for all test sets and the Mersenne Twister random number generator (Matsumoto and

Table 4 Results for the test set 1 using the GRASP

$\|V\|$	R	O^*	Time(O^*)	Heuristics	GRASP						
				Best	Clusters			Hops			Time
					Best	Worst	Avg.	Best	Worst	Avg.	
100	20	12	2.59	12	12	12	12.00	3.82	3.82	3.82	0.05
150	20	12	39.20	13	12	12	12.00	4.18	4.18	4.18	0.11
200	20	11	80.73	14	11	12	11.80	4.55	5.09	4.87	0.19
250	20	11	174.11	14	11	11	11.00	4.40	4.40	4.40	0.29
300	20	11	577.08	15	11	12	11.60	4.73	6.20	5.31	0.44
350	20	11	1,333.86	15	11	12	11.80	4.55	6.60	5.03	0.63
400	20	12	2,372.48	15	12	12	12.00	2.91	3.09	2.95	0.71
450	20	11	236,377.83	14	11	12	12.00	4.55	4.91	4.80	0.97
500	20	*12*	–	14	12	12	12.00	4.55	4.73	4.66	1.22
550	20	*12*	–	15	12	13	12.20	3.50	4.55	3.90	1.34
600	20	*12*	–	15	12	12	12.00	4.55	5.09	4.77	1.85
650	20	*12*	–	14	12	12	12.00	4.00	4.18	4.11	2.03
700	20	*12*	–	16	12	12	12.00	3.09	3.45	3.27	2.22
750	20	*12*	–	15	12	12	12.00	3.82	4.36	4.04	2.74
800	20	*12*	–	15	12	13	12.80	3.83	4.18	3.97	3.19
850	20	*12*	–	16	12	13	12.40	3.67	4.00	3.86	3.42
900	20	*13*	-	16	13	13	13.00	3.33	3.83	3.57	3.86
950	20	*12*	–	15	12	13	12.60	4.17	4.91	4.50	4.48
1,000	20	*12*	–	16	12	12	12.00	2.91	3.27	3.02	4.52

Nishimura 1998) are used. Each run consists of 200 iterations. This stopping criteria has been set after analyzing the experiments for the instances in the first test set where the optimal solutions are known. In fact, 200 iterations seems to be a good trade off between the running time and the solution quality.

Numerical results are presented in Tables 4, 5 and 6, respectively for the first, the second and the third set of instances. Each line corresponds to an instance. The number of nodes $|V|$ and the radio range R are given for each instance. The optimal value (O^*) and the corresponding running time in seconds (Time(O^*)) are reported whenever the optimal solution has been obtained by CPLEX in Tables 4 and 5. Otherwise, the best known solution found by the GRASP is given in italic in column (O^*). The best number of clusters obtained on the previous works (Belisario et al. 2010; Hou et al. 2008; Santos et al. 2009) is given in column Heuristics. It is important to mention that these results correspond to quite simple heuristics (greedy, constructive and using multiflow strategy). The next columns correspond to the results produced by GRASP. Considering the five runs and the clusters and hop average criteria, "Best" stands for the best solution values found over all runs, "Worst" and "Avg." are respectively the worst value and the average of the best solution values over all runs. "Time" is the running time average in seconds for the five runs. Since GRASP stopping criteria is an absolute number of iterations, the CPU time does not vary much over the runs (2 % average variation).

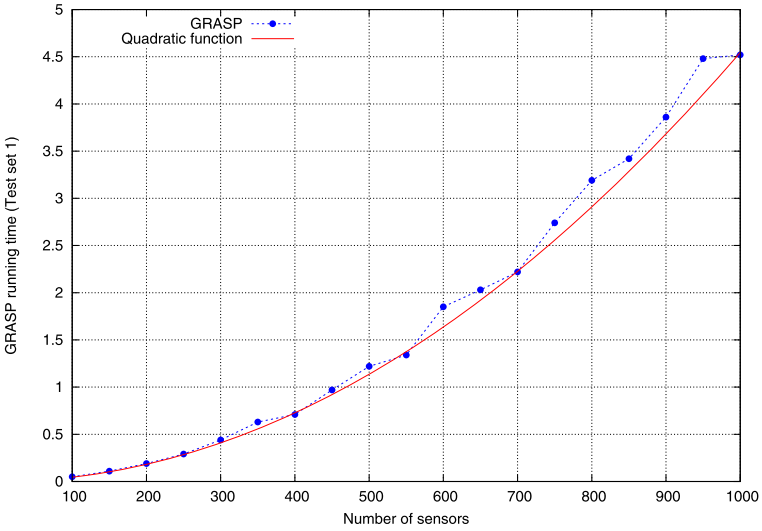


Fig. 4 GRASP running time evolution for the first test set

GRASP is able to find the optimal values for the instances, when it is known. For this set of instances, the deviation from the best values to the optimal solution are of at most one cluster. Thus, the GRASP seems to be quite stable on the value of the best solutions. CPLEX found the optimal solution for instances with up to 450 nodes. In terms of running time, it becomes very time consuming, more than 65 hours, for the instance with 450, while the GRASP stays very time efficient. CPLEX cannot solve the instances with 500 to 1,000 sensors and it runs out of memory in about 45 minutes. Figure 4 displays the CPU time evolution when the instance size grows for this test set. It appears to be nearly quadratic and the GRASP CPU time remains affordable for WSNs with up to 1,000 nodes.

Experiments with the second test set aim at analyzing the evolution of the number of clusters when the radio range varies. In fact, increasing the radio range implies a reduction in the number of clusters. In the worst case, the number of clusters stays the same. Results in Table 5 illustrate this behavior. These results are particularly interesting in the context of WSNs for which radio ranges vary along the network lifetime. Varying radio ranges has a direct impact on a cluster-based topology since it may drastically reduce the number of clusters. On the other hand, the energy consumption increases as it requires more power to transmit messages.

Experiments with the third test are reported in Table 6. They illustrate the limits of the proposed method. The results from previous heuristics (Belisario et al. 2010; Hou et al. 2008; Santos et al. 2009) are not reported since they were only applied on the first and second test sets. The deviation over the best solutions found over the 5 runs is slightly larger than for other test sets. Figure 5 shows the running time evolution is above quadratic. In the context of WSNs, it is not applicable since it becomes too time consuming. Changing the number of GRASP iterations will impact on the solution quality. Thus, improved strategies to efficiently compute topologies for WSNs of such

Table 5 Results for the test set 2 using GRASP

$\|V\|$	R	O^*	Time(O^*)	Heuristics	GRASP						
				Best	Clusters			Hops			Time
					Best	Worst	Avg.	Best	Worst	Avg.	
200	20	11	2.59	14	11	12	11.80	4.55	5.20	4.93	0.19
200	30	6	185.61	7	6	6	6.00	3.20	3.20	3.20	0.20
200	40	4	485.93	4	4	4	4.00	2.67	2.67	2.67	0.16
200	50	4	253.25	4	4	4	4.00	2.00	2.00	2.00	0.19
200	60	3	41.78	3	3	3	3.00	2.00	2.00	2.00	0.18

Table 6 Results for the test set 3 using GRASP

$\|V\|$	R	GRASP						
		Clusters			Hops			Time
		Best	Worst	Avg.	Best	Worst	Avg.	
5,000	20	251	252	251.80	21.18	22.96	21.89	515.75
6,000	20	254	257	256.00	17.60	19.35	18.39	770.22
7,000	20	254	259	257.40	14.56	16.78	15.59	1,103.31
8,000	20	262	263	262.60	23.69	27.36	24.78	1,426.59
9,000	20	264	267	265.40	19.02	19.88	19.49	2,034.54
10,000	20	266	270	267.80	14.27	14.98	14.70	2,609.11
15,000	20	273	276	274.40	18.71	19.80	19.27	7,092.50
20,000	20	275	278	276.40	15.88	17.20	16.68	11,477.15

size can be investigated. This can be done, for instance, by partitioning a WSN or by developing distributed algorithms.

Finally, it is important to mention the GRASP improves the best results found by the constructive heuristics in an average of 14 %, 2 % and 15 %, respectively for the first, the second and the third test sets. The biggest are the instances the more work is required to the local search. For the test set 2, the constructive heuristics basically are able to find some optimal results. This explains the low improvement percentage for this test set. The best known solutions for the instance with 1,000 sensors and 5,000 sensors are illustrated in Figs. 6 and 7. The location of the masters seems to be quite homogeneous since the sensors position was randomly generated using the uniform law.

8 Concluding remarks

In this work, we have developed several strategies to solve the IDSC problem including a MILP formulation, several heuristics and a GRASP metaheuristic. To the best of our knowledge, this it is the first ever mathematical formulation for the IDSC problem using spanning trees with special properties to design cluster-based topologies.

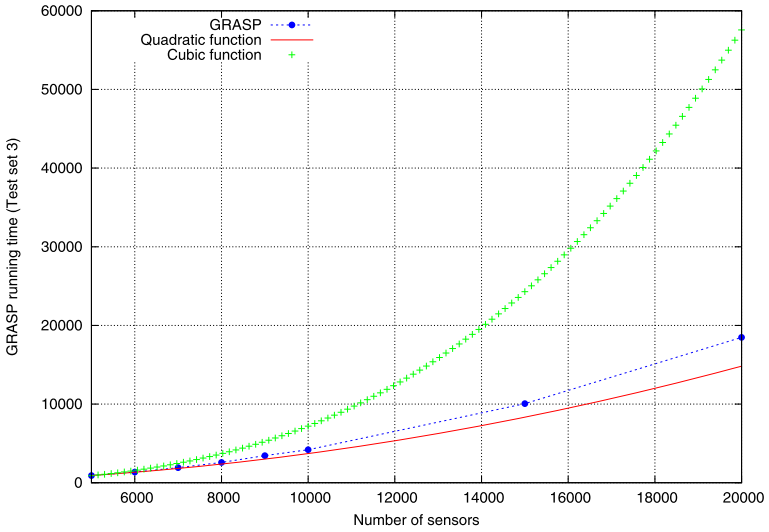


Fig. 5 GRASP running time evolution for the third test set

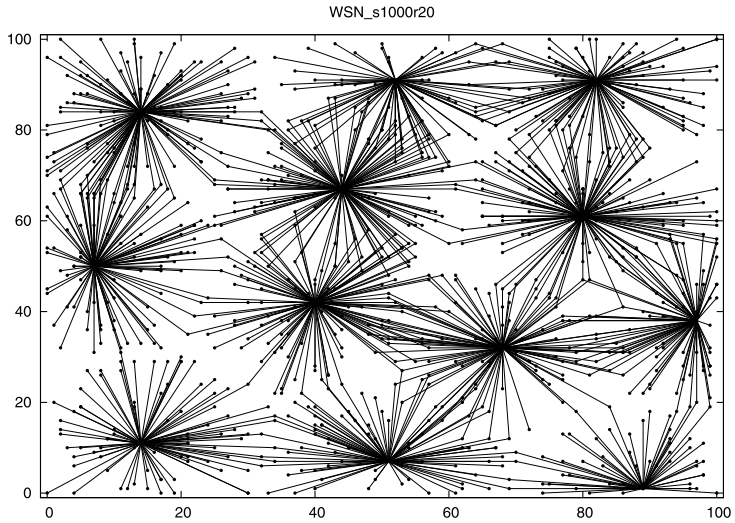


Fig. 6 Solution for a WSN with 1,000 sensors

This has inspired the design of efficient tree-based heuristics for the IDSC. In spite of the running time, optimal solutions have been computed for instances with up to 450 sensors. These results have been used to measure the performance of the GRASP.

Several constructive heuristics have been developed. The construction of solutions is done from the sink towards the network border (DOWN) and from the network border towards the sink (UP). Moreover, different criteria have been developed to select the candidates to become masters. The MEMORY approach uses a memory

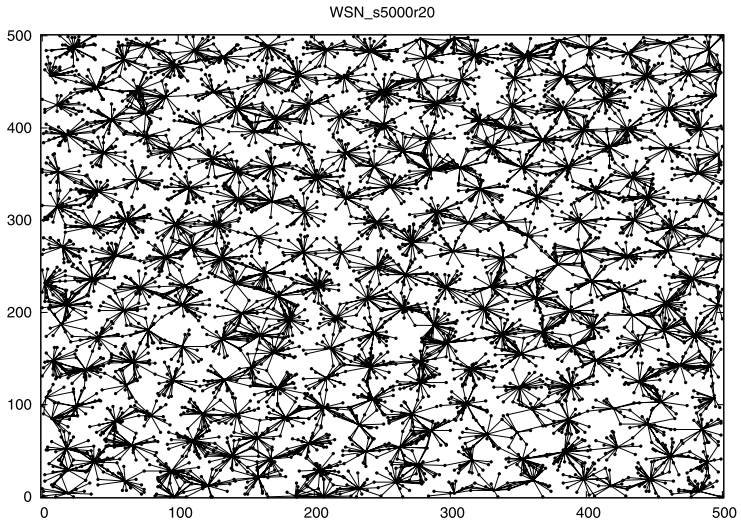


Fig. 7 Solution for a WSN with 5,000 sensors

mechanism, while NEIGHBOR strategy is more classic and uses a quantitative criteria strongly related with the problem (the number of neighbors). The NEIGHBOR DOWN and NEIGHBOR UP heuristics have been randomized by using a RCL. Results show the NEIGHBOR DOWN strategy dominates the others. In spite of the results obtained using the MEMORY mechanism, it remains an interesting idea which could work well in another context.

From a theoretical point of view, we can mention as the main advantages of the proposed metaheuristic: the quality of the solutions and the efficiency for WSNs with thousands of sensors. The GRASP metaheuristic is specially efficient for instances with up to 1,000 nodes. Moreover, a new topology can be quickly computed whenever a region becomes unreachable. This result suggests the centralized approach proposed here can be applied in practice for designing cluster-based topologies. As a consequence, the network lifetime is improved as it benefits from the clusters organization: the number of messages to be sent is reduced and the amount of hops to route messages to the sink is reduced as well.

As future works, the model can be extended to multi-sink and multi-range contexts. Furthermore, the connection between cluster-based topologies and area coverage remains an open issue. In terms of algorithms, different strategies can be investigated like distributed and multi-agent algorithms. For the centralized approaches, other metaheuristics can be designed for the problem such as genetic algorithms and tabu search.

References

Abbasi, A.A., Younis, M.: A survey on clustering algorithms for wireless sensor networks. Comput. Commun. **30**, 2826–2841 (2007)

Aguiar, A., Pinheiro, P.R., Coelho, A.L.V., Nepomuceno, N., Neto, A., Cunha, R.P.P.: Scalability analysis of a novel integer programming model to deal with energy consumption in heterogeneous wireless sensor networks. Lect. Notes Comput. Sci. **14**, 11–20 (2008)

Aioffi, W., Mateus, G.R., Quintão, F.P.: Optimization issues and algorithms for wireless sensor networks with mobile sink. In: Proceedings of International Network Optimization Conference (INOC), Spa, Belgium (2007)

Al-Karaki, J.N., Kamal, A.E.: Routing techniques in wireless sensor networks: a survey. IEEE Wirel. Commun. **11**(6), 6–28 (2004)

Belisario, L.S., Guedes, L.S.M., Santos, A.C., Duhamel, C., Hou, K.-M.: Heuristiques pour la conception de rseaux wsn. In: Proceedings of 5th International Conference on Operations Research (CIRO), Marrakech, Maroc, pp. 91–92 (2010)

Cardei, M., Wu, J., Lu, M., Pervaiz, M.O.: Maximum network lifetime in wireless sensor networks with adjustable sensing ranges. In: Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Montreal, Canada, pp. 438–445 (2005)

Clark, B.N., Colbourn, C.J., Johnson, D.J.: Unit disk graphs. Discrete Math. **86**, 165–177 (1991)

Dhawan, A., Vu, C.T., Zelikovsky, A., Li, Y., Prasad, S.K.: Maximum lifetime of sensor networks with adjustable sensing range. In: Proceedings of the International Workshop on Self-Assembling Wireless Networks (SAWN'06), Las Vegas, EUA (2006)

Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. **8**, 67–71 (1989)

Feo, T.A., Resende, M.G.C.: Greedy randomized adaptative search procedures. J. Glob. Optim. **6**, 109–133 (1995)

Hajiaghayi, M.T., Immorlica, N., Mirrokni, V.S.: Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks. IEEE/ACM Trans. Netw. **15**, 1345–1358 (2007)

Hou, K.-M., Mailfert, J., Bendali, F., Duhamel, C., Santos, A.C.: An optimization approach for designing wireless sensor networks. In: The NTMS Workshop on Wireless Sensor Networks: Theory and Practice, Tanger, Maroc (2008). Page Eletronic diffusion

Hurinka, J.L., Niebergb, T.: Approximating minimum independent dominating sets in wireless networks. Inf. Process. Lett. **109**, 155–160 (2008)

Jia, J., Chena, J., Changa, G., Wena, Y., Songa, J.: Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. Comput. Math. Appl. **57**(11–12), 1767–1775 (2009)

Li, X., Wang, P., Frieder, O.: Coverage in wireless ad hoc sensor networks. IEEE Trans. Comput. **52**(6), 753–763 (2002)

Li, D., Wu, W., Du, D.-Z., Cheng, X., Huang, X.: A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. Networks **42**, 202–208 (2003)

Matsumoto, M., Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. Trans. Model. Comput. Simul. **8**(1), 3–30 (1998)

Meguerdichian, S., Potkonjak, M.: Low power 0/1 coverage and scheduling techniques in sensor networks. Technical Report 030001, University of California, Los Angeles (2003)

Moraes, R.E.N., Ribeiro, C.C., Duhamel, C.: Optimal solutions for fault-tolerant topology control in wireless ad hoc networks. In: IEEE Transactions on Wireless Communications, vol. 8, pp. 5970–5981 (2009)

Rossi, A., Singh, A., Sevaux, M.: A column generation scheme for a collection of sensor network scheduling problems. In: Proceedings of 11ème Congrés de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Toulouse, France (2010)

Rossi, A., Sevaux, M., Singh, A., Geiger, M.J.: On the cover scheduling problem in wireless sensor networks. In: Proceedings of the 5th International Conference on Network Optimization (INOC), Hamburg, Germany, pp. 657–668 (2011a)

Rossi, A., Singh, A., Sevaux, M.: Column generation algorithm for sensor coverage scheduling under bandwidth constraints. Networks (2011b, to appear). doi:10.1002/net.20466

Santos, A.C., Bendali, F., Mailfert, J., Duhamel, C., Hou, K.-M.: Heuristics for designing energy-efficient wireless sensor network topologies. J. Netw. **4**(6), 436–444 (2009)

Vlajic, N., Xia, D.: Wireless sensor networks: to cluster or not to cluster. In: Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, Buffalo, New York, pp. 258–268. IEEE Comput. Soc., Los Alamitos (2006)