Computer Networks 217 (2022) 109302

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Application placement with shared monitoring points in multi-purpose IoT wireless sensor networks

Mustafa Can Çavdar [*], Ibrahim Korpeoglu, Özgür Ulusoy

Department of Computer Engineering, Bilkent University, Ankara, 06800, Turkey

ARTICLE INFO

Keywords:
Wireless sensor networks
Virtualization
Resource allocation
Optimization

ABSTRACT

The main function of a wireless sensor network (WSN) is to gather data from a certain region and transfer the data to a center or remote locations for further processing. The collected data can be of interest for many applications. Therefore, a physical WSN owned by a single provider can be utilized by many customer applications. Additionally, the data of a particular point or sub-region can satisfy the need of multiple applications. Hence, sensing the data only once in such cases is beneficial to reduce the energy consumption, network traffic and acceptance ratio of the applications. We call this as monitoring point based shared data approach. In this paper, we focus on the placement of applications each of which requires several points to be monitored in an area a WSN covers. We first propose such a monitoring point based shared data approach for WSNs that will serve multiple dynamic applications. We also propose two methods for application placement over a shared physical WSN: one greedy method and one genetic algorithm based method called GABAP. We did extensive simulation experiments to evaluate our algorithms. The results show the effectiveness of our methods in fast and close-to-optimum placement of applications over a single network.

1. Introduction

Improvements in sensing technologies and devices, and cost-efficient evolution in wireless communications and mobile-computing enabled WSNs to become one of the key components for Internet of Things (IoT) and smart environments. Wireless Sensor Networks (WSNs) are heterogeneous systems which consist of sensor nodes that can collect various types of data from a set of monitoring points in an area of interest. Collected data are processed in sensor nodes or centralized units.

The range of applications has grown rapidly since the inception of WSNs. Various types of applications can be provided by using WSNs. For instance, a disaster prevention system can be built to monitor dangerous workplaces, like mines, refineries, etc., and alert officials accordingly in case of an emergency. Some other application examples include environmental monitoring, air quality monitoring with NO_2 sensors in cities [1], in-pipe monitoring of the quality of drinkable water [2], and collecting seismic and infrasonic signals from volcanoes [3]. A major domain of usage is the management of smart cities. A smart city can make use of IoT to provide the services of the city in a more efficient and interactive manner [4]. Noise monitoring in the urban areas [5], and intelligent parking systems [6] are among the smart city applications of WSNs. Applications of WSNs are so extensive that there are studies that group the use cases. Kandris et al. [7] categorize the WSN application domain into six groups: military, health, environmental, flora and fauna, industrial and urban. The broad range of applications makes the optimization and efficient use of WSNs very vital.

Traditionally, the realizations of WSNs were task-specific. A WSN was bundled with a single application and was optimized and used only for that application. Deploying another application to the already running WSN was impossible. This leads to redundant WSN deployments and inefficient utilization of WSN resources. As with many systems, the trend for WSNs is changing from being designed for a single application to designing WSNs that can support various applications with heterogeneous types of needs within a single network infrastructure [8]. Consider, for example, a smart building system with environment monitoring capabilities that are used to adjust general cooling system. Into this network, another application that uses moist and carbon dioxide sensors to estimate room occupancy can be deployed without disturbing the already existing cooling application. Another example may be a city-wide WSN infrastructure of sensor nodes with various sensing capabilities. Various applications, such as traffic monitoring, waste management, air quality monitoring, crime detection, etc., can be deployed over this single WSN infrastructure to realize a smart city.

[*] Corresponding author.
E-mail addresses: mustafa.cavdar@bilkent.edu.tr (M.C. Çavdar), korpe@cs.bilkent.edu.tr (I. Korpeoglu), oulusoy@cs.bilkent.edu.tr (Ö. Ulusoy).

https://doi.org/10.1016/j.comnet.2022.109302
Received 7 August 2021; Received in revised form 13 June 2022; Accepted 12 August 2022
Available online 19 August 2022

In this paper, we focus on application placement with sharing possibilities over a sensor network operated by a single provider. We assume the network is accessible by application providers or users to request their applications to be placed. Each application requires some set of points in the area to be monitored with certain data types, and the data gathered from those points need to be processed in centralized units. The goal is to place as many applications as possible, considering also data sharing possibilities. Such an application placement is a resource allocation problem and is NP-hard. Therefore, we propose two methods to place applications over a shared physical WSN. One is a simple greedy algorithm that is very fast and that also serves as a simple benchmark. The other is a genetic algorithm based solution that can perform close to optimum even for very large networks. We call our genetic algorithm solution GABAP. GABAP tries to increase the number of placed applications by choosing sensor nodes and base stations for monitoring points as good as possible. We also formulate the problem as a mathematical program to find out the optimal solution for reasonably-sized networks.

We conducted extensive simulation experiments and evaluated our algorithms. The experimental results show that our genetic algorithm, GABAP, is very effective in placing high number of applications and performing close to optimum. Our greedy algorithm, on the other hand, is very fast and can be preferred for the cases where speed is very important.

The rest of the paper is organized as follows: Section 2 describes the example motivational use of our network model. Section 3 discusses the related work in the literature and provides a brief comparison with our work. Section 4 provides the problem formulation. Section 5 introduces our greedy and genetic algorithms. Section 6 details our experimental results. Finally, Section 7 gives our conclusions.

2. Motivation

For multi-application operations, it is essential that WSNs be designed and utilized in an optimal way so that we can support as many applications as possible simultaneously over a single physical WSN infrastructure. This usually requires a centralized controller. Thanks to *Software-Defined Networking (SDN)* technology, we now have this opportunity. With SDN, decision-making process is moved from a distributed network of units to a logically implemented centralized controller software [9]. SDN allows sharing physical resources among multiple applications, and improves flexibility of the network [10]. Hence, Software-Defined Networking has a key role in the development of next generation networks and Internet of Things [11].

In WSNs that support multi-application deployment, some applications may require the same data type (i.e., image, video, temperature) to be sensed from the same monitoring point. For instance, consider an application that measures the average speed of vehicles between two points by capturing their image. There may be another application that monitors traffic density at those two points at certain times of the day. Both applications require image data from those two points but their data requirement frequencies are different. Measuring the speed of vehicles requires more frequent image data than monitoring traffic density. For such cases, we propose *monitoring point based shared data* approach, which enables sensing and transmission of data only once for multiple applications registered to sense the same monitoring point. In this example, data will be collected once to be shared among two applications, instead of sensing and collecting separate data for each application. This reduces the capacity and bandwidth usage of sensor nodes and links. In this way, a sensor node will have more resources available for applications yet to be deployed.

The differences between shareable and unshareable approaches are summarized in Table 1.

To illustrate the benefits of the shareable approach, we consider the following example scenario. Let us consider a sensor network consisting of sensor nodes and base stations. Assume that there are multiple applications requiring a specific monitoring point to be sensed and that monitoring point is in the sensing range of a single specific sensor node. Assume also that the sensor node can connect to just one particular base station which has enough processing capacity. With the shareable approach, we can place many applications whose sensing requirements are at most the sensing capacity of the sensor node. However, with the unshareable approach, each placed application consumes additional sensing resources of the sensing node, therefore we cannot place many applications. For instance, suppose that the sensor node has a sensing rate capacity of 400 kbps, and there are four applications with the sensing requirements of 300 kbps, 100 kbps, 150 kbps and 200 kbps. With the shareable approach, all four applications can be placed to the network, since the used sensing rate would be 300 kbps which is less than the capacity of the sensor node. However, with the unshareable approach, at most two applications can be placed to the network, considering the capacity of the senor node. Therefore, with the shareable approach, a WSN can support much more applications with the same amount of resources. Moreover, the average energy spent per application would be less compared to the unshareable approach.

3. Related work

Optimization of WSNs is among the topics that are extensively studied in computer networks field. Previous studies can be categorized based on optimization objective, such as decreasing energy cost, increasing efficiency of resource allocation and scheduling, increasing coverage area of the WSN, and so on.

In [12], Raee et al. assign tasks to sensor nodes in a way that energy consumption is minimized by using Integer Linear Programming. They compare their solution with a traditional WSN which is a non-virtualized network. Ojha et al. [13] describe a scheme for dynamic IoT applications to preserve energy efficiency in a cloud system. They model the interaction between cloud owners and sensor owners as a Stackelberg game. Lemos et al. [14] propose an Ant Colony Optimization method to handle virtual sensor provisioning in WSNs. Their algorithm selects an optimal set of sensor nodes to respond to user demands while withholding energy consumption in the whole network. In [15], Rahmati et al. present a load balancing algorithm for efficient routing in WSNs. They consider energy consumption and resource allocation and show that a uniform distribution of WSN load leads to the most efficient routing.

Delgado et al. [8] use mathematical programming to solve both application admission and network slicing problems in WSNs. They evaluate their method on realistic WSN infrastructures. The same authors expand their work to handle dynamic application admission in shared sensor networks in [16]. SenShare [17] is a platform that addresses the technical difficulties of transforming a physical sensor network into an infrastructure that supports multiple applications. In [18], Bhattacharya et al. present UMADE which is an application deployment system that allocates applications to sensor nodes by considering their Quality-of-Monitoring (QoM). Therefore, they aim to increase overall QoM in the whole network within resource constraints. The two QoM attributes they use are variance reduction and detection probability. Ajmal et al. [19] propose an admission control algorithm for WSNs to which applications are deployed for a certain time interval. Their method tries to minimize the total execution time of applications. Moreover, they also assess the feasibility of their scheduling based on co-arrived applications. Cionca et al. [20] propose Judishare, a framework that enables the reuse of sensing and communication resources in shared sensor networks. In [21], Bousnina et al. focus on the resource allocation problem in a virtual sensor network that has sensor nodes with various amounts of resources. They propose a greedy approach to solve the problem. This greedy method is faster than the methods which optimally solve the same problem, however, under-performs compared to them. Tynan et al. [22] propose a Multi-Agent System architecture that deals with embedding virtual sensor networks on a WSN. They

Table 1
Comparison of shareable and unshareable approaches.

	Shareable approach	Unshareable approach
Sensing	Required resources for each monitoring point are less compared to Unshareable since sensing requirement of each monitoring point is the maximum of individual requests of applications.	Required resources for each monitoring point are more since the sensing requirement of each monitoring point is the sum of individual requests of applications.
Transmission	Since less data are sensed by sensor nodes, amount of transmitted data is also less.	Since more data are sensed, amount of transmitted data is more compared to Shareable approach.
Benefit	Less resources are used for the same applications. More available resources for applications yet to be deployed.	More resources are used for the same applications. Less available resources for applications yet to be deployed.
	Less energy spent for the deployed applications.	More energy spent for the deployed applications.

use hibernation of idle resources to reduce power consumption. Xu et al. [23] present a local search algorithm for application allocation in shared sensor networks. They aim to maximize QoM by considering resource constraints such as memory and bandwidth. They compare their method with simulated annealing by using both real-world datasets and simulated networks that are randomly generated.

Uchiteleva et al. [24] propose a resource scheduling algorithm for WSNs. The scheduler provides a resource management solution for isolated profiles in a WSN and enables the virtualization of the network. They compare their approach, which is a branch and bound technique, with traditional Round Robin and Proportionally Fair algorithms. Wei et al. [25] present ISVM-Q, a Q-learning algorithm for task scheduling, to optimize application performance and energy consumption of WSNs. Li et al. [26] propose a framework that considers the load condition of every node in a wireless network and initiates a re-embedding operation if necessary. In [27], Abreu et al. describe a QoS-based application admission for WSNs for the biomedical domain. Instead of applications, their proposed work decides when to admit a new sensor node on the network. Edalat and Motani [28] consider a network which contains sensor nodes with solar panels that harvest energy. They tackle the problem of task scheduling and mapping of those tasks to sensor nodes. They aim to increase fairness in mapping by considering task priority and energy harvesting constraints. The method called EN-MASSE [29] deals with dynamic mission assignment in a WSN including sensor nodes with harvesting capabilities. It proposes a mixed-integer programming method to assign missions to sensor nodes within a time horizon. It aims to reduce the total execution time of the missions as its biggest constraint is the lifetime of the network. De Frias et al. [30] describe an application scheduling algorithm for shared sensor and actuator networks. They target to improve the energy efficiency of the system and experiment by using both simulations and real sensor nodes.

Our work described in this paper differs from the above mentioned studies with the following features:

- A monitoring point based shared data approach: Data sensed from a monitoring point can be shared by multiple applications and our methods consider this whenever possible. Each application indicates its required sensing rate for a monitoring point, and our scheme multiplexes and satisfies these requests using a single stream of data sensed from the monitoring point. In this way, we reduce the sensing and communication resources used per application, without violating sensing requirements.
- Network structure: We focus on sensor networks that have limited bandwidth and computational resources at the edge. Data are sensed from monitoring points by sensor nodes and sent to base stations (or cluster-heads) to be processed and/or conveyed further towards one or more data centers. We assume the network among base stations and sinks is a high speed network. Hence, we are concerned with the efficient use of the limited bandwidth available between sensor nodes and base stations.
- A novel algorithm: We propose a novel genetic algorithm, called GABAP, that increases the number of placed applications by selecting applications to be admitted and assigning monitoring points to sensors and base stations in a close-to-optimal way. GABAP can also *migrate* monitoring points from their assigned sensor nodes and base stations to others.
- An LP solution for the problem: We provide a linear programming (LP) solution to the problem for small networks as well. In this way, we are able to compare our GABAP algorithm against the optimal solution. We also provide a greedy algorithm to compare our GABAP solution to a very fast algorithm.

4. Problem statement

We consider a wireless sensor network (WSN) that is owned by a single sensor network infrastructure provider and shared by multiple applications. Our WSN model consists of sensor nodes with equal sensing rates and sensing ranges, base stations with equal processing capacities, and connections between those sensors and base stations with equal bandwidth capacity. A sensor can gather data from the monitoring points within its sensing range and it is connected to the base stations within its communication range.

An example network model is provided in Fig. 1. In the figure, large circles represent sensor nodes in the network, small circles represent monitoring points and radio towers represent base stations. A dotted line between a sensor node and a base station means that the sensor node connects to the base station. A dashed line between a sensor node and a monitoring point means that the sensor node is actively sensing the monitoring point. Base stations are connected to sinks with a high speed connection. We try to deploy as many applications as possible to this network. The applications require different types of data to be sensed from some monitoring points. If all monitoring point requirements of an application can be satisfied by our network, then the application is placed. Otherwise, it is not placed. A monitoring point requirement of an application is satisfiable if with the extra load caused by the application's needs, we can find a sensor node and a base station with sufficient resources.

The parameters used for a formal description of our problem are shown in Table 2.

We aim to maximize the number of applications that are successfully deployed. We can formally express this problem as follows:

$$\sum_{j \in A} z_j \tag{1}$$

is maximized subject to

$$z_j = \prod_{k \in M_j} x_{jk} \qquad \forall j \in A \tag{2}$$

$$x_{jk} \leq x_k \qquad \forall k \in M \tag{3}$$

$$x_k = \sum_{i \in S_k} x_{ik} \leq 1 \qquad \forall k \in M \tag{4}$$

$$r_k = max(r_{jk} * x_{jk}) \qquad \forall k \in M \tag{5}$$

$$u_k = \sum_{j \in A} (r_{jk} * x_{jk}) \qquad \forall k \in M \tag{6}$$

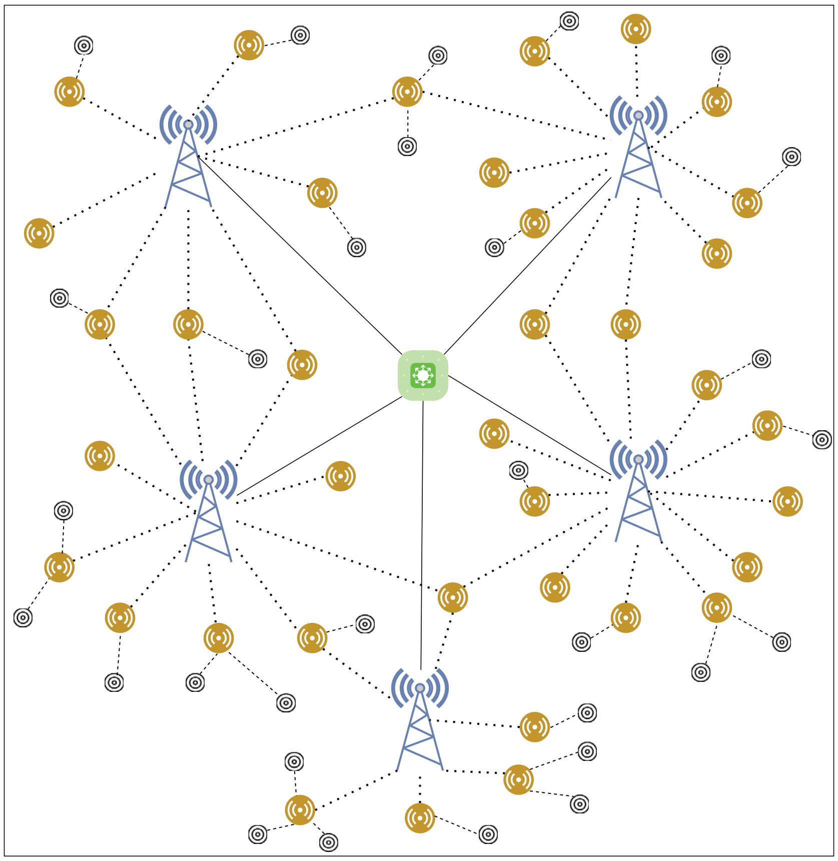


Fig. 1. Network model.

Table 2
Parameters used in problem statement.

S	Set of sensor nodes
B	Set of base stations
C	Set of connections
A	Set of applications
M	Set of monitoring points
S_k	Set of sensor nodes covering monitoring point k
M_j	Set of monitoring points required by application j
M_{il}	Set of monitoring points whose data is transferred from sensor node i to base station l
z_j	Binary variable indicating whether application j is deployed
x_k	Binary variable indicating whether monitoring point k is sensed
x_{ik}	Binary variable indicating whether sensor node i is actively sensing monitoring point k
x_{ilm}	Constant indicating whether sensor node i is connected to base station l through connection m
x_{jk}	Binary variable indicating monitoring point requirement of application j for monitoring point k is satisfied
r_{jk}	Sensing rate requirement by application j at monitoring point k
r_k	Sensing rate requirement of monitoring point k
u_k	Sum of sensing rate requirements of applications for monitoring point k
R_i	Sensing rate capacity of sensor node i
E_s	Energy budget for each sensor node
P_l	Processing capacity of base station l
C_m	Bandwidth capacity of connection m
R'_i	Used sensing resource of sensor node i
P'_l	Used processing resource of base station l
C'_m	Used bandwidth of connection m
TC	Transmission coefficient
PC	Processing coefficient

$$\sum_{k\in M} x_{ik} r_k \le R_i \qquad \forall i \in S \tag{7}$$

$$\sum_{k\in M} x_{ik} u_k \le R_i \qquad \forall i \in S \tag{8}$$

$$\sum_{k\in M_{il}} r_k TC \le \sum_{m\in C} x_{ilm} C_m \qquad \forall i \in S, \forall l \in B \tag{9}$$

$$\sum_{k\in M_{il}} u_k TC \le \sum_{m\in C} x_{ilm} C_m \qquad \forall i \in S, \forall l \in B \tag{10}$$

$$\sum_{i\in S} \sum_{k\in M_{il}} u_k PC \le P_l \qquad \forall l \in B \tag{11}$$

Eq. (2) shows that to deploy an application, all the monitoring point requirements of the application must be satisfied. Requirements of a monitoring point can be satisfied if the monitoring point is sensed by a sensor node as shown in Eq. (3). A monitoring point is sensed by at most one sensor node as shown in Eq. (4). With Eqs. (5) and (6), we indicate the calculation of the required sensing rate by a monitoring point for shared and unshared cases, respectively. Eqs. (7) and (8) are the sensing constraints for shareable and unshareable approaches, respectively. For shareable approach, our connection constraint is shown in Eq. (9). Eq. (10) is the connection constraint of unshareable approach. We have a *Transmission Coefficient* whose value is between zero and one because we assume that the data collected at sensor nodes can be compressed before they are sent to a base station to be processed. Eq. (11) is the processing constraint. We use a *Processing Coefficient* is since some data sent to base stations may be noise and they do not need to be processed. The sensing constraint indicates that the sensing capacity of a sensor node must be at least large enough to meet the sensing requirements of monitoring points it actively senses. The connection constraint specifies that a connection should have enough bandwidth to transfer the data from the sensor node to the base station it connects. With the processing constraint, we enforce that the total processing requirement of monitoring point data sent to a base station must not exceed the base station's processing capacity. A monitoring point's requirement is calculated by considering only admitted applications. If our algorithm does not admit an application to the network, the requirements of that application are ignored in calculating the monitoring point requirement.

We assume that the applications arrive at the network dynamically, in batches or one by one. Therefore, there exist both initially known applications that are already placed and new applications that are yet to be deployed.

4.1. Energy constraint

The following equations are to calculate the energy spent by the sensor nodes and base stations in the network. The total energy spent in the edge network is the sum of energy spent by each sensor node and base station.

$$E^t_{il} = x_{ilm} C'_m * (\beta_1 + \beta_2 * d^4_{il}) \qquad \forall i \in S, \forall l \in B \tag{12}$$

$$E^s_i = \rho * R'_i \qquad \forall i \in S \tag{13}$$

$$E_i = \sum_{l \in B} (E^t_{il}) + E^s_i + E_{act} + E_{mig} * N_i \qquad \forall i \in S \tag{14}$$

$$E^r_{il} = \beta_1 * x_{ilm} C'_m \qquad \forall i \in S, \forall l \in B \tag{15}$$

$$E^p_l = \gamma * P'_l \qquad \forall l \in B \tag{16}$$

$$E_l = \sum_{i \in S} (E^r_{il}) + E^p_l + E_{act} \qquad \forall l \in B \tag{17}$$

Eq. (12) is the calculation of energy that is spent by data transmission by a sensor node i to a base station l. Eq. (13) describes the calculation of energy cost for sensing data. For each sensor node, the sum of transmission energy cost to all connected base stations, sensing energy cost, activation cost and the cost of migration to the sensor node is equal to the total energy spent by the sensor node which is shown in Eq. (14). $\beta_1 = 50$ nJ/bit, $\beta_2 = 0.0013$ pJ/bit m^4, $\rho = 0.5$ nJ/bit. These values and the formulas are derived from [31]. E_{act} is the activation energy for sensor nodes and base stations. E_{mig} is the migration cost. Both E_{act} and E_{mig} are equal to 10 J [8].

Eq. (15) is to calculate the energy dissipation for the reception of the transmitted data to the base stations. Eq. (16) is to calculate the energy cost of processing received data at the base stations. Total energy spent by a single base station is the sum of total energy spent for the reception of the data, processing of the data, and the activation cost. It is shown in Eq. (17). $\gamma = 5$ nJ/bit. The value of γ and the formulas are derived from [31].

$$E_i \leq E_s \qquad \forall i \in S \tag{18}$$

Eq. (18) shows the only energy constraint in our model. It indicates that energy spent by each sensor node cannot exceed its energy budget. In our model, we do not have any energy constraints for base stations, since we assume that base stations can be plugged into the grid, as in [8].

4.2. Hardness of application placement problem

The application placement problem which is described above is a resource allocation problem. Here, to prove that the aforementioned problem is NP-hard, we reduce the 3-SAT problem which is a well-known NP-hard problem [32], to the application placement problem.

4.2.1. 3-SAT

Boolean satisfiability problem (SAT) is the problem of determining whether there is an interpretation that satisfies a given Boolean formula. In, 3-satisfiability (3-SAT), the formula consists of clauses each having exactly three literals. For instance, $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ and $c_i = x_a \vee x_b \vee x_c$ where $a, b, c \in \mathbb{N}$ is in 3-SAT form.

4.2.2. Reduction to application placement

For each monitoring point request, we need to find a sensor to gather data from the monitoring point and a base station to process gathered data. The Boolean formula ϕ we use for reduction consists of three literals. Let $\phi = C_1 \wedge C_2 \wedge C_3$. If C_1 and C_2 have common attributes with non-contradictory values, we say that monitoring point is in the sensing range of the sensor with enough available sensing resources. Similarly, if C_2 and C_3 have common attributes with non-contradictory values, the sensor and the base station with enough processing power have connection with enough bandwidth. Therefore, if we can find a feasible solution for ϕ, then we can sense that monitoring point. For instance, let $\phi = (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_5)$. We can reduce it as in Fig. 2. Common literals of C_1 and C_2 are x_2 and x_4 and the only common literal of C_2 and C_3 is x_2. With $x_1 = false$, $x_2 = false$, $x_3 = true$, $x_4 = true$ and $x_5 = true$, ϕ is feasible and we can find a suitable sensor and a suitable base station for the monitoring point.

5. Proposed method

To solve the aforementioned resource allocation problem in WSNs, we propose two different methods: a greedy approach, and GABAP, a genetic algorithm based solution.

5.1. Greedy algorithm

The greedy approach we propose is a modified version of the *Worst Fit* algorithm. Therefore, we use the least utilized sensor node or base station to distribute the load more evenly to have more options available for subsequent requests. At each time instant t, our algorithm processes arriving applications one by one in an unordered way. For each application, we check whether all of its monitoring point requirements are satisfiable. If they are, then we place that application into our network and update resources accordingly. The satisfiability of a requirement is checked as follows:

- If the required monitoring point is already sensed, then, we check whether the resources that monitoring point uses can meet the extra demands by the new application. Resources mentioned here are the resources of the sensor that is already sensing that monitoring point, the base station at which that point's data is processed, and the connection between them. If new demand can be met, then this requirement is satisfiable. Here, we do not provide any migration operation as it contradicts the approach being greedy.

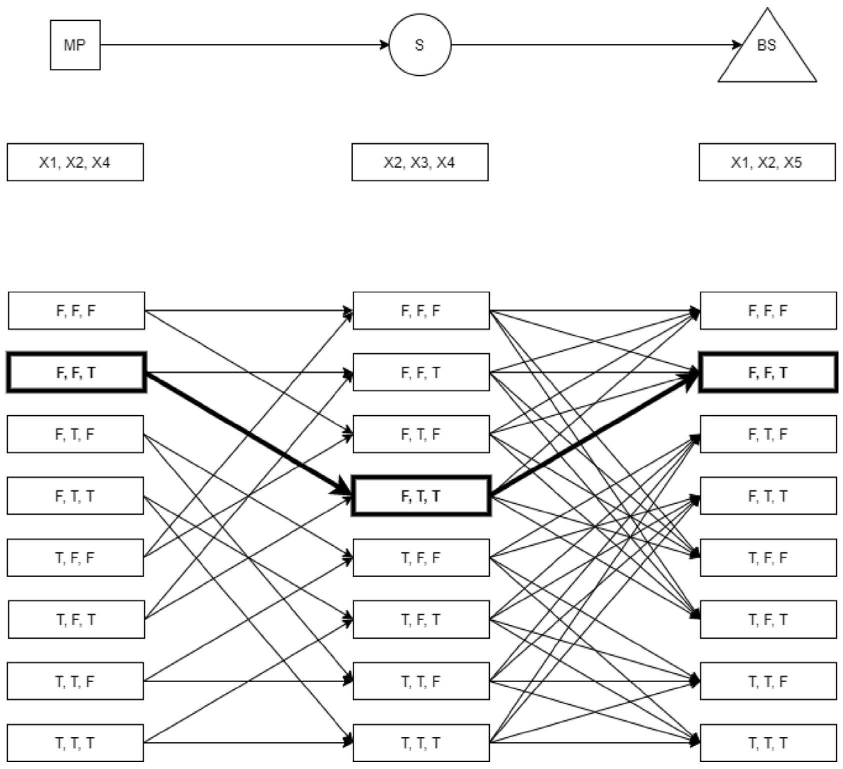


Fig. 2. 3-SAT reduction to application placement.

- If the required monitoring point is not sensed, then we apply the *Worst Fit* logic to find a sensor node and a base station to sense the monitoring point and process the related data. We search through sensor nodes which have that point in their sensing range and select the sensor node with the largest available resource. Then, among the base stations that the sensor node has a connection, we select the one with the largest available processing resource. If the connection between the selected sensor node and the base station has enough bandwidth, then we assign that monitoring point to the selected sensor node and base station. If the connection cannot meet the demand, we skip that base station and consider the *next* base station with the largest available resource. If none of the base stations connected to the selected sensor node satisfies the demand, then we skip this node and consider the *next* sensor node with the largest available resource. If our algorithm cannot find such a sensor node either, then we consider the requirement as unsatisfiable.

If all requirements of an application are satisfied, then the application is placed. Our algorithm outputs the number of placed applications.

5.2. Genetic Algorithm Based Application Placement (GABAP)

Genetic algorithms are proven to be useful for the optimization of networks [33–35]. Therefore, we decided to use a genetic algorithm based solution for the aforementioned problem. A genetic algorithm runs for *generations* each having a number of candidate solutions called *individuals*. Each individual has chromosomes representing the actual solution it offers. The chromosome structure in the algorithm we propose consists of *Application Genes*, *Sensor Genes* and *Base Station Genes*, which are all integer lists. The size of *Application Genes* is the number of applications arriving at that time instant that needs to be placed and the sizes of the other two are $|M|$, i.e., the number of monitoring points. The values in *Application Genes* are binary (0 or 1) and they represent which applications need to be admitted. *Sensor Genes* represent the proposed sensor nodes sensing each monitoring point requested by at least one application, and *Base Station Genes* represent the proposed base stations in which each monitoring point is processed. Example lists are shown in Fig. 3.

5.2.1. Initial population creation

In genetic algorithms, each individual of a generation is created by crossover operations whose parameters are individuals from the previous generation. Since there is not any generation before the initial population (the first generation), we need to create its individuals randomly.

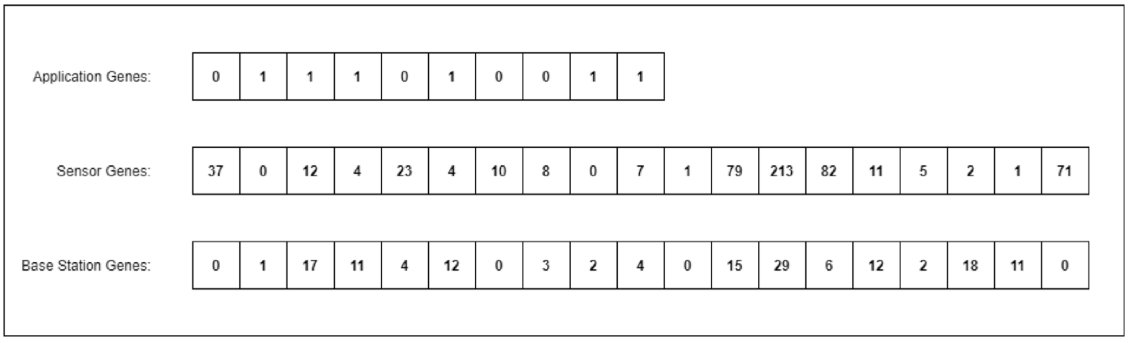
Since *Application Genes* of an individual determines whether the application should be admitted, each application gene of an individual from the initial population is determined with a 50% probability. It is either 0 or 1.

Sensor Genes and *Base Station Genes* of the initial population are determined together. For each monitoring point that is requested by at least one application from the current batch, we first determine a sensor gene. It is randomly determined among the sensor nodes whose sensing range covers the monitoring point. Each possible sensor node has an equal probability. After the sensor gene is determined, we randomly select a base station among the ones the chosen sensor node has a connection to. Again, each possible base station has an equal chance to be selected. For monitoring points that are not requested by any application, we do not determine a sensor gene or base station gene to avoid unnecessary migrations in the current or future batch of applications. Sensor and base station genes for monitoring points that are not requested are set to -1.

5.2.2. Fitness calculation

Our fitness calculation is designed to measure how close an *Individual* (a candidate solution to the problem) is to the optimum solution. The value obtained from the calculation is called the individual's *fitness score*. Eq. (19) shows the fitness calculation of our genetic algorithm.

$$fitness = PAC - \alpha * MC - \beta * (WAPS + WMS) \tag{19}$$



Application Genes:	0	1	1	1	0	1	0	0	1	1									
Sensor Genes:	37	0	12	4	23	4	10	8	0	7	1	79	213	82	11	5	2	1	71
Base Station Genes:	0	1	17	11	4	12	0	3	2	4	0	15	29	6	12	2	18	11	0

Fig. 3. Example genes.

Placed Application Count (PAC) is the number of the applications that can be placed into the network. For each application *j* with *ApplicationGene(j)=1*, we check whether all of its requirements can be met without violating network constraints. If we cannot place application *j*, then we increase *Wrong Application Placement Suggestion (WAPS)* by one.

Our algorithm can also *migrate* a monitoring point from one sensor node and base station to another pair which is explained in Section 5.2.6. If a monitoring point has an already assigned sensor node and an already assigned base station, and the genes of the individual suggest a different sensor node or a different base station, we say that the individual suggests a migration. If a monitoring point whose migration is suggested is required by a successfully placed application, then we increase *Migration Count (MC)* by one. However, if the migration violates the network constraints, then we increase *Wrong Migration Suggestion (WMS)* by one.

Values of α and β have a direct effect on the fitness score. A migration operation may be helpful to place a larger number of applications, however, it is not a free operation. If an application can be placed both with and without migration, we favor placing it without migration because of the cost. Nevertheless, placing an application with migration is preferable to not placing the application at all. Therefore, The value of α is in the interval of (0,1). The sum $WAPS + WMS$ indicates the number of times that the genes of the individual violate the constraints of our network. Since violating the network constraints is undesired, β has a value that is big enough to ensure that not attempting to place any applications at all is a better outcome than having network constraint violations. In our experiments, we observe that $\alpha = 0.1$ has the best value for the application placement and migration trade-off. For β, 1000 is a value that is large enough to prevent violations. A feasible solution for the network means a non-negative fitness score for the individual.

5.2.3. Selection operation

We use the tournament selection method to find an individual for each individual in the population to pair up for the crossover operation. Tournament selection has its own population (i.e. the tournament population) which includes a subset of individuals that are randomly chosen from the general population. It outputs the individual with the highest fitness score. There is a small possibility of there may be multiple individuals with the highest fitness score in the tournament population. In such cases, we randomly select one of them. We repeat this process for every individual in the general population. We do not use the same tournament population for each individual which means that for each individual in the general population, we create a new tournament population. Therefore, the individuals included in the tournament population are not the same every time. The size of the tournament population is 5% of the general population since this amount is small enough to increase the variety and large enough to have better individuals after the crossover operation.

Algorithm 1 Selection Operation

```
Require: The tournament population, $Pop$
Ensure: an individual chromosome
 1: procedure SELECTION
 2:     $bestScore \leftarrow 0$
 3:     $bestInd \leftarrow Null$
 4:     for each Individual $x$ in $Pop$ do
 5:         calculate its fitness score, $newScore$
 6:         if $bestScore \leq newScore$ then
 7:             $bestScore \leftarrow newScore$
 8:             $bestInd \leftarrow x$
 9:         end if
10:     end for
         return $bestInd$
11: end procedure
```

5.2.4. Crossover operation

Crossover operation is performed to create individuals of the next generation. Paired-up candidate solutions in the selection operation are used at this stage. This operation decides which genes are inherited from which parent. The value of *uniformRate* determines the probability of selecting genes from either parent. In our experiments, we choose *uniformRate = 0.5* not to favor either parent to improve diversity. Crossover of *Application Genes* is realized separately from the other two. For each application, a randomly selected parent's gene is inherited. For each monitoring point that is requested by the applications from the current batch, according to the generated random value, a parent is chosen and both *Sensor Gene* and *Base Station Gene* are taken from that parent. This operation guarantees that each sensor gene of the offspring covers the corresponding monitoring point and the base station gene of the corresponding monitoring point is connected to the sensor node since the offspring inherits these from either parent and initially we ensure that these constraints are satisfied as explained in Section 5.2.1. However, the offspring may violate the constraints related to sensing, communication, or processing capabilities and if that is the case, the offspring is punished in terms of fitness score as explained in Section 5.2.2.

The operation is presented in Algorithm 2.

5.2.5. Mutation operation

Mutation operation is realized after individuals of the new generation are created. For each individual, we apply the mutation operation with the probability determined by the *mutationRate*. It generally has a small value. In our experiments, the value of *mutationRate* is 0.05 which means that there is a 5% chance for each individual to be mutated.

For a randomly selected application, our mutation operation flips the value of the corresponding gene of that application. For each

Algorithm 2 Crossover Operation

```
Require: Six chromosomes from two parents: $A_1$, $S_1$, $BS_1$, $A_2$, $S_2$, and $BS_2$
Ensure: The chromosomes of Offspring: $A_{new}$, $S_{new}$ and $BS_{new}$
 1: procedure CROSSOVER
 2:     for $x = 1$ to $|A|$ do
 3:         randomly create a value between 0 and 1, $r$;
 4:         if $r \leq uniformRate$ then
 5:             set gene $x$ of $A_{new}$ as gene $x$ of $A_1$
 6:         else
 7:             set gene $x$ of $A_{new}$ as gene $x$ of $A_2$
 8:         end if
 9:     end for
10:     for $x = 1$ to $|M|$ do
11:         randomly create a value between 0 and 1, $r$;
12:         if $r \leq uniformRate$ then
13:             set gene $x$ of $S_{new}$ as gene $x$ of $S_1$
14:             set gene $x$ of $BS_{new}$ as gene $x$ of $BS_1$
15:         else
16:             set gene $x$ of $S_{new}$ as gene $x$ of $S_2$
17:             set gene $x$ of $BS_{new}$ as gene $x$ of $BS_2$
18:         end if
19:     end for
        return Offspring
20: end procedure
```

monitoring point requested by the applications from the current batch, our mutation operation selects a random sensor node and a random base station and change the related genes accordingly. This operation is aware of the network structure: The selected sensor node can cover the monitoring point and there is an active connection between the selected sensor node and the selected base station. Mutation on application genes and the mutation on sensor and base station genes are independent of each other.

The pseudocode of our mutation operation is presented in Algorithm 3.

Algorithm 3 Mutation Operation

```
Require: Three chromosomes of Individual Ind: $A_{old}$, $S_{old}$ and $BS_{old}$
Ensure: Three mutated chromosomes of Individual Ind: $A_{new}$, $S_{new}$ and $BS_{new}$
 1: procedure MUTATION
 2:     randomly create a value between 0 and 1, $r1$;
 3:     if $r1 \leq mutationRate$ then
 4:         randomly create a value between 0 and $|M|$, $a$;
 5:         set gene $a$ of $A_{new}$ as 1 - gene $a$
 6:     end if
 7:     for $x = 1$ to $|M|$ do
 8:         randomly create a value between 0 and 1, $r2$;
 9:         if $r2 \leq mutationRate$ then
10:             randomly select a sensor from possible sensors, $i$; where $1 \leq i \leq |S|$
11:             randomly select a base station from possible base stations, $l$; where $1 \leq l \leq |B|$
12:             set gene $x$ of $S_{new}$ as gene $i$
13:             set gene $x$ of $BS_{new}$ as gene $l$
14:         else
15:             set gene $x$ of $S_{new}$ as gene $x$ of $S_{old}$
16:             set gene $x$ of $BS_{new}$ as gene $x$ of $BS_{old}$
17:         end if
18:     end for
19: end procedure
```

5.2.6. The genetic algorithm

Our genetic algorithm is described in Algorithm 4. The termination condition for the algorithm is that either a candidate solution places all available applications, or no improvement is observed in the best individual's fitness score for 3 generations. The population size is 500.

For each discrete time instant t (which may correspond to a time interval), we run our genetic algorithm. We feed the algorithm with applications arriving at time *t*. After our algorithm finds a solution, we apply this close-to-optimal solution to the network, and we continue with time instant *t+1*. For the sensor and the base station of each monitoring point we have three cases:

- At time instant *t*, if a monitoring point is not sensed yet, we set the assigned sensor and base station of that monitoring point according to the solution we have by running our algorithm at time t and mark that monitoring point as *sensed*. We update the remaining resources of the corresponding sensor, base station, and the connection between them.
- At time instant *t*, if a monitoring point is already sensed and the solution we have by running our algorithm at time t suggests the same sensor and base station, we update the required sense rates of the monitoring point, and remaining resources of the corresponding sensor, the base station and the connection between them.
- At time instant *t*, if a monitoring point is already sensed and the solution we have by running our algorithm at time t suggests a sensor and/or base station different from the monitoring point that is already assigned to, we *migrate* the monitoring point from the old sensor and base station to the new sensor and base station. The resources of two sensors, two base stations, and two connections are updated accordingly.

Algorithm 4 The Genetic Algorithm

```
 1: procedure THE GABAP
        generate a population of random individuals, $POP$;
 2:     while THE TERMINATION CONDITION is not true do
 3:         for each Individual $x$ in $POP$ do
 4:             calculate its fitness value $f(x)$
 5:         end for
 6:         for each Individual $x$ in $POP$ do
 7:             invoke the SELECTION Operation, that is using tournament selection technique (Alg. 1) to select another individual to pair
 8:         end for
 9:         for each pair of parents do
10:             use CROSSOVER Operation to produce an offspring which is described in Alg. 2
11:         end for
12:         for each offspring do
13:             apply MUTATION Operation which is described in Alg. 3
14:         end for
15:         find the best individual among offsprings, $newBest$
16:         if $newBest$ is better than current best individual then
17:             replace current best individual with $newBest$
18:         end if
19:     end while
        return best individual
20: end procedure
```

6. Experimental results

We did extensive simulation experiments to evaluate our algorithms. In our simulations, we consider a model of a sensor network

Table 3
Sensing rate requirements.

Data type	Sensing rate
Data type 0	5 kb/s–20 kb/s
Data type 1	15 kb/s–40 kb/s
Data type 2	25 kb/s–60 kb/s

Table 4
Network parameters.

Parameter	Value
Monitoring points per application	1–3
Communication range of sensors	200 m
Sensing range of sensor nodes	50 m
Sensing rate capacity of sensor nodes	400 kb/s
Bandwidth capacity of connections	100 kb/s
Processing capacity of base stations	1000 kb/s
Energy budget of sensor nodes	20 000 J
Transmission coefficient	0.7
Processing coefficient	0.9

spanning a 2D plane which has 1000 m width and 1000 m height. Each sensor node, base station, and monitoring point has its own coordinates (x and y) on this plane. Coordinates of these elements are randomly determined and they do not overlap but can be very close to each other. The method used to create a network model for simulations guarantees that each monitoring point is within the sensing range of at least one sensor node and each sensor node can be connected to at least one base station.

In our network, we assume that applications can require sensing rates from monitoring points in 3 different major scales (rate types). These 3 different sensing rate types, i.e., rate ranges, are listed in Table 3. The data rate requirements of applications (the rate type and exact rate in that range) are randomly selected. However, we enforce that from one monitoring point only one data rate type can be demanded.

The quality of the connections in a WSN has a vital role in reliable transmission of sensed data from sensor nodes to base stations. To simulate the effect of the link quality, we assign a Packet Delivery Ratio (PDR) to each sensor node base station connection in the network. PDR is one of the main metrics for modeling link quality [36]. For a particular link, the PDR is calculated as the number of packets delivered to the receiving node divided by the number of packets sent by the transmitting node. In this way, we model the delivery probability of a packet to be equal to the PDR of the connection over which the packet is sent. An undelivered packet is retransmitted. That means, of course, more energy cost per packet. We limit the number of retransmissions for a packet to 10 [37]. We additionally assume that a packet is definitely delivered after the 10th retransmission, if not delivered earlier. In our simulations, the PDR of each connection is randomly determined between 0.7 and 1. We assume that the packet size is fixed and 512 bits.

Network parameters are shown in Table 4. Each sensor node can be connected to base stations within its communication range. The communication range of a sensor node is 200 m. We consider symmetric communication range between sensor nodes and base stations. Each application requires 1 to 3 points to be monitored. The sensing range of each sensor node is 50 m. If the distance between a monitoring point and a sensor node is less than the sensing range, the monitoring point can be sensed by that sensor.

Applications arrive in ten different batches. We assign a batch number to each application randomly. At each time instant t, one batch of applications arrives. When a batch of applications is requested to be placed, a selected subset of those applications is admitted, and the resources used by these admitted applications become unavailable for the next batches. Applications run in our network for a limited amount of time. When an application completes, it releases all resources it has used. The released resources become available for the applications in later batches. The run time for each application is 12 h.

We compared our two proposed algorithms, GABAP and the Greedy algorithm, with the algorithms proposed in [8,26] which we call DH and RSVN, respectively. The four are algorithms run with both shared and unshared approaches. In the plots, for each algorithm, results of shared and unshared approaches are presented with suffix "S" and "U", respectively. We experiment with various numbers of applications, monitoring points, sensors, and base stations. Evaluation of the methods is reported in terms of the number of placed applications and the average energy spent per placed application. Our experiments are realized in 7 cases:

- *Case 1*: The number of monitoring points is fixed at 300. The application count is started at 500 and is incremented by 200 until 1500.
- *Case 2*: The number of applications is fixed at 1000. The monitoring point count is started at 50 and is incremented by 50 until 250.
- *Case 3*: The number of applications is fixed at 1000, and the number of monitoring points is fixed at 300. For the number of monitoring point requests per application, instead of randomly determining between the values shown in Table 4, we start with 1 monitoring point per application and increment it by 1 until 6.
- *Case 4*: The number of applications is fixed at 1000, and the number of monitoring points is fixed at 300. For the communication range of sensors, instead of randomly determining between the values shown in Table 4, we start with 50 m as the communication range for each sensor node and increment it by 50 m until 250 m.
- *Case 5*: The number of applications is fixed at 1000, and the number of monitoring points is fixed at 300. For the sensing range of sensor nodes, instead of randomly determining between the values shown in Table 4, we start with a 30 m sensing range for each sensor node and increment it by 5 until 50 m.
- *Case 6*: The number of monitoring points is fixed at 300, and the batch count is fixed at 10. We experiment with various batch sizes: 50, 100, 200, 250, and 500. Application count is calculated as the number of batches times the batch size.
- *Case 7*: Same constraints as in Case 6 except that application count is fixed at 1000. The batch count is calculated as the number of applications divided by the batch size.

In all the settings of all cases, the number of sensor nodes is 250 and the number of base stations is 30. We chose these values because they are sufficient enough to cover the whole network area while enabling us to analyze the impacts of other parameters on the performances of the algorithms. For each setting, we run the algorithms 1000 times with both shareable and unshareable approaches. The results of the experiments are plotted in vertical bars that show the average result of 1000 runs. Error bars show maximum and minimum results.

In Case 1, we investigate how the change in the number of arriving applications affects the performance of the algorithms. Figs. 4 and 5 show the average count of placed applications and average energy cost per placed application, respectively averaged over 1000 runs. Each algorithm performs better with the shareable approach. Obviously, as the number of applications arriving to the network gets larger, all algorithms achieve to place larger number of applications. GABAP-S is clearly superior to all other algorithms. RSVN-S and GABAP-u have the next-best performance. DH and Greedy come the last, where DH has a slightly better performance than Greedy.

In terms of energy spent, all algorithms perform better with the shareable approach. In fact, the average energy spent with the shareable approach is around half of the energy spent with the unshareable approach. As expected, we see that with a larger number of available applications, the shareable approach costs less energy per placed application since the sensing and transmission overheads are far less (even

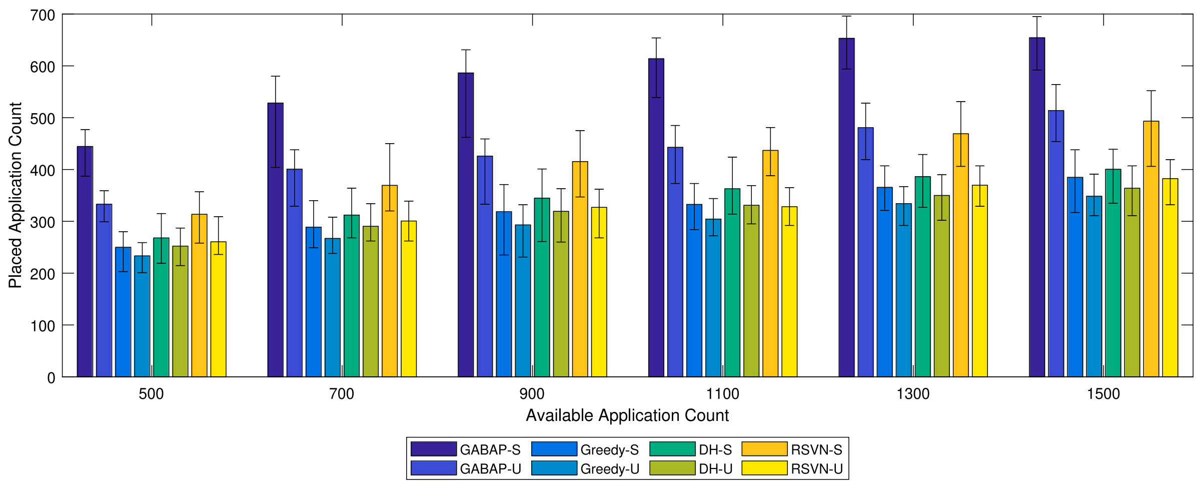


Fig. 4. Comparison of algorithms in terms of placed application count in Case 1.

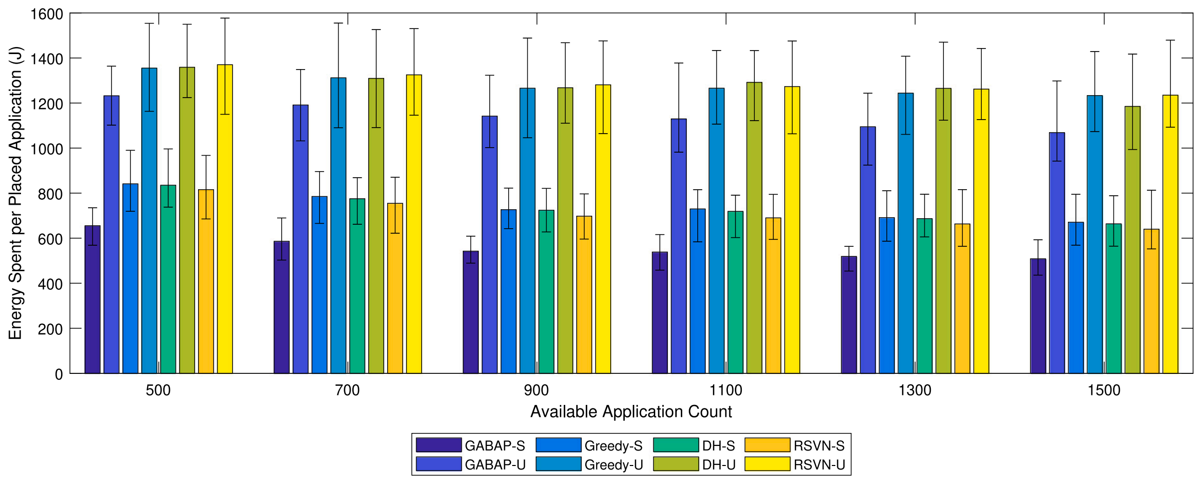


Fig. 5. Comparison of algorithms in terms of energy cost in Case 1.

zero in some cases) than they are in the unshareable approach. However, we also observe a small reduction in the results of the unshareable approach with all algorithms. The reason for this result is that the possibility of the arriving applications with less data sense requirements increases with the larger number of available applications. Therefore, the average energy spent decreases slightly. With the unshareable approach, all four algorithms perform similarly when GABAP-U has slightly better performance. With the shareable approach, GABAP-S has the best performance followed by RSVN-S.

In Case 2, we evaluate the impact of the number of monitoring points on the performance of the algorithms. Figs. 6 and 7 show the results for this case. We expect that with less number of monitoring points, placing applications becomes harder because there will be more application requests for each monitoring point as the number of applications is fixed. Moreover, fewer monitoring points affect the performances of the algorithms with the unshareable approach more since the sensing and transmission overheads increase drastically. Fig. 6 shows that the performance of the algorithms improves clearly with the increasing number of monitoring points. With the shareable approach, the improvement is not that visible since the sensing and transmission overheads are much smaller. GABAP-S still has the best performance, the gap between its performance and the others increases as the number of monitoring points decreases.

We again observe that with the shareable approach, the energy cost is much less than the unshareable one. With increasing number of monitoring points, the energy cost is also increasing since there are more data to sense, transmit, and process. With the unshareable approach, all four algorithms show similar performance in terms of the energy cost. With the shareable one, GABAP-S produces the lowest energy cost.

In Case 3, we have a fixed number of applications and a fixed number of monitoring points. We experimented with different numbers of monitoring points each application requires to be sensed, instead of randomly determined values according to Table 4. Figs. 8 and 9 show the results of Case 3. With the increasing number of monitoring point requests per application, it is getting harder to place applications since the demand of a single application increases while network resources remain the same. We observe that the shareable approach makes algorithms place more applications compared to the unshareable one. Again, GABAP has the best performance, RSVN comes the second, while Greedy and DH have the least number of applications placed. DH performs slightly better than Greedy.

In terms of the energy cost, it is expected that the cost per placed application increases when the applications require more monitoring points to be sensed. Since the sensing and transmission overheads are more with the unshareable approach, the energy cost is more compared to the shareable approach. The performance gap between the shareable and unshareable approaches is proportional to the number of requests per application. GABAP-S has the best performance, and again, RSVN-S is the second-best. Energy cost results of the algorithms do not differ much with the unshareable approach, however with the shareable one,

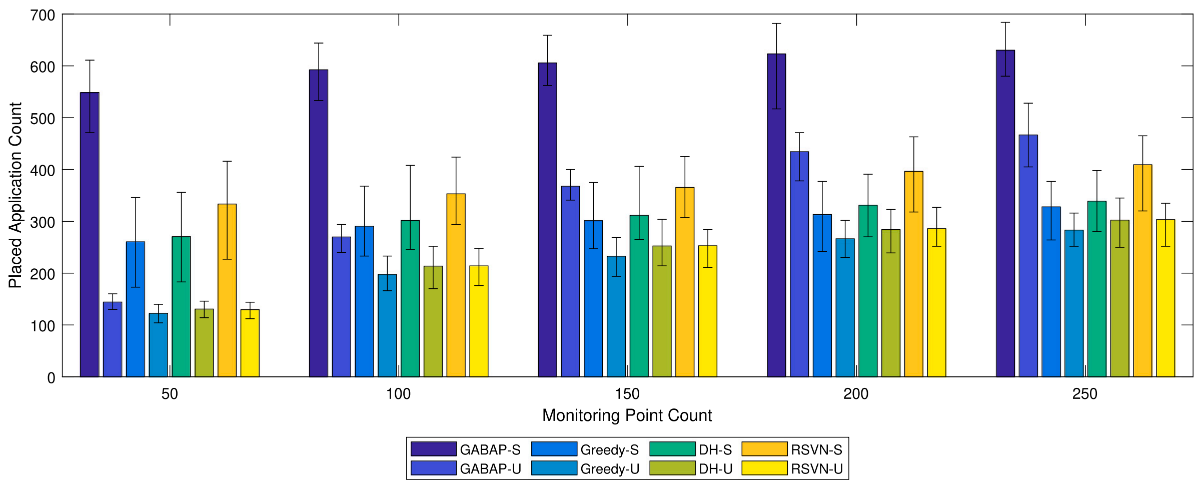


Fig. 6. Comparison of algorithms in terms of placed application count in Case 2.

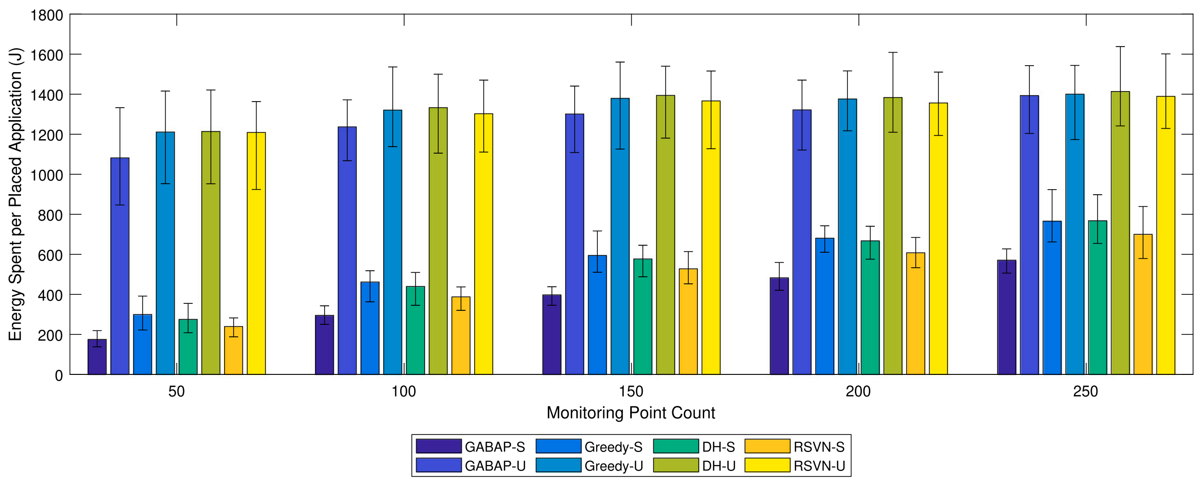


Fig. 7. Comparison of algorithms in terms of energy cost in Case 2.

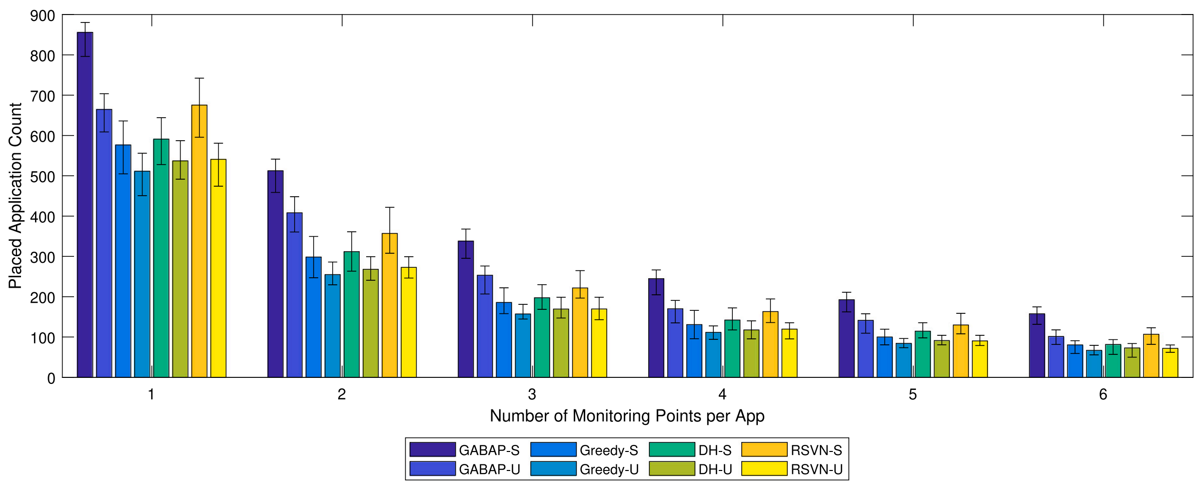


Fig. 8. Comparison of algorithms in terms of placed application count in Case 3.

especially with higher number of requests, the performance difference between GABAP-S and others become more visible.

Case 4 is investigating how the communication range of sensor nodes affect the performance of the algorithms. Instead of the 200 m communication range as in Table 4, we experimented with the communication ranges between 50 to 250 m. Figs. 10 and 11 present the results. We observe that with a longer communication range for sensor nodes, the number of placed applications increases slightly. Similar to

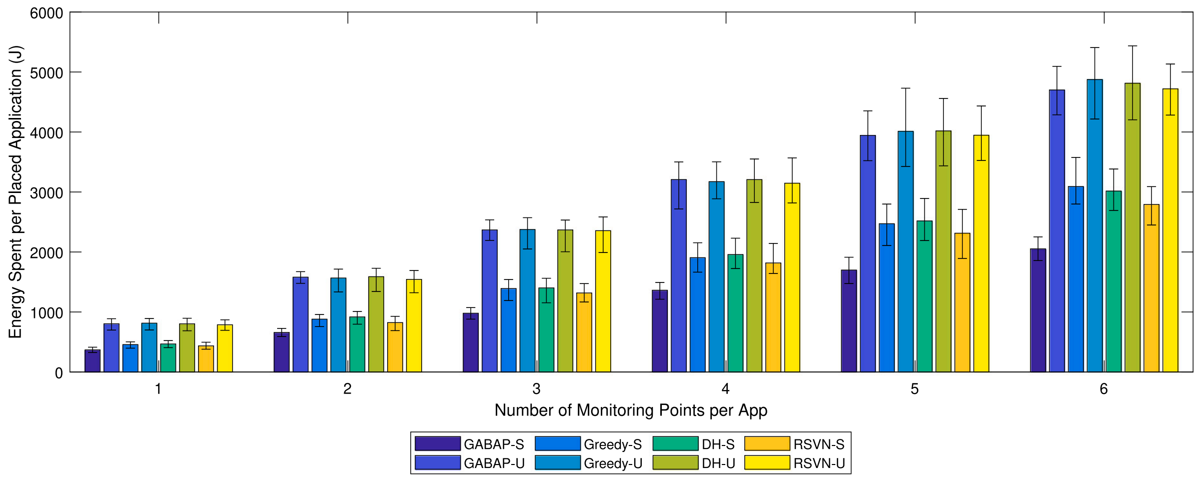


Fig. 9. Comparison of algorithms in terms of energy cost in Case 3.

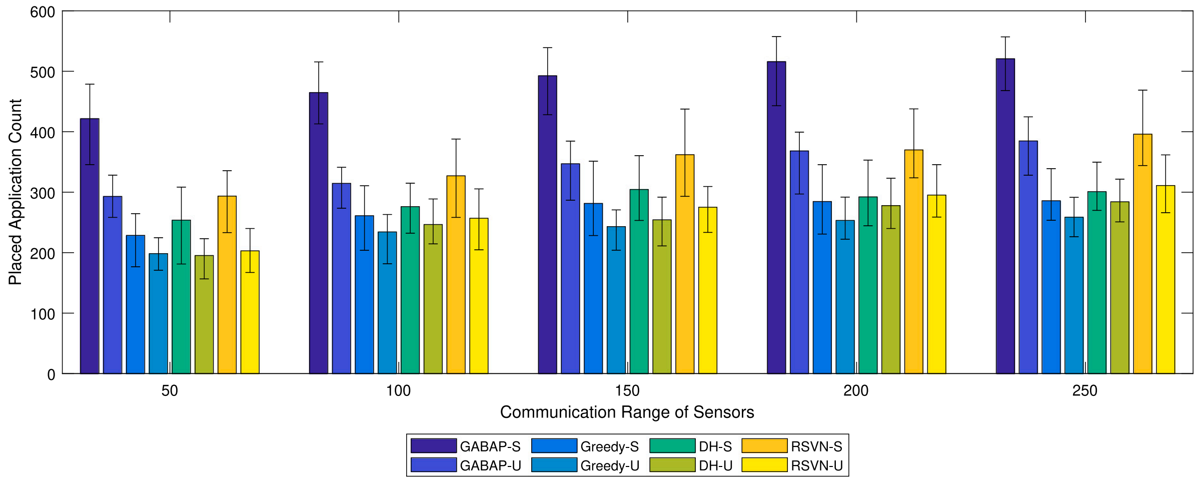


Fig. 10. Comparison of algorithms in terms of placed application count in Case 4.

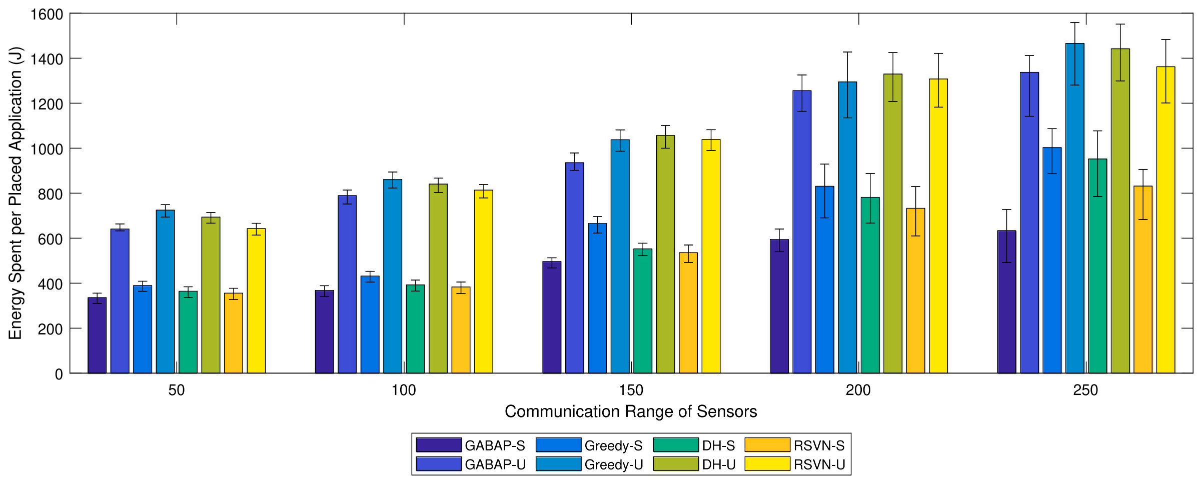


Fig. 11. Comparison of algorithms in terms of energy cost in Case 4.

the previous cases, GABAP-S is superior among all compared. RSVN-S and GABAP-U have the next-best performances.

The energy spent increases with the broader communication range of sensor nodes. The reason for this result is that with a longer range, sensor nodes can connect to base stations that are more distant and sending data to more distant base stations costs more compared to sending to closer base stations as shown in Eq. (12). The gap between the shareable and unshareable approaches in terms of energy cost increases with the longer communication ranges. This also applies to the gap among the results of the algorithms. GABAP-S has the least energy

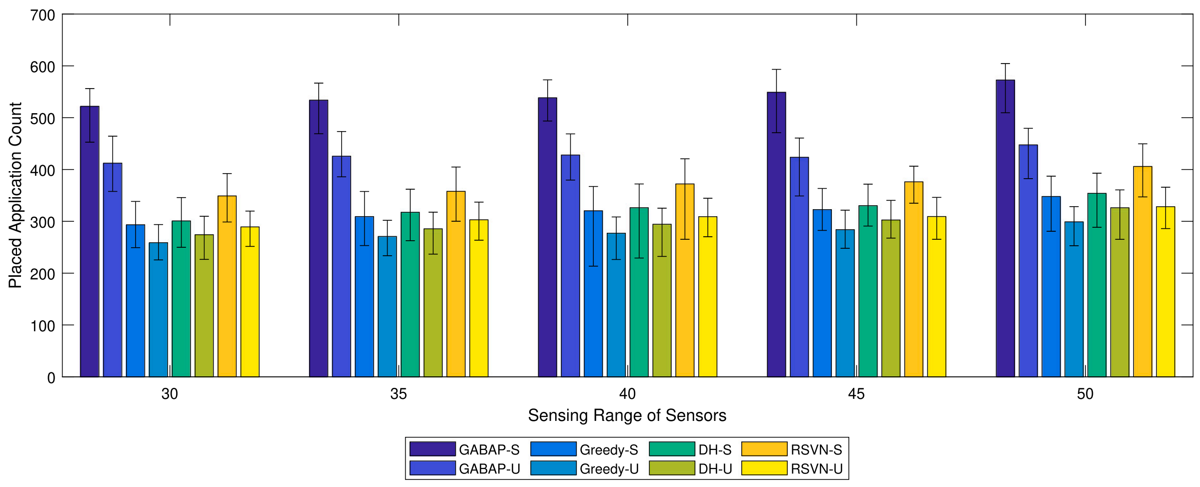


Fig. 12. Comparison of algorithms in terms of placed application count in Case 5.

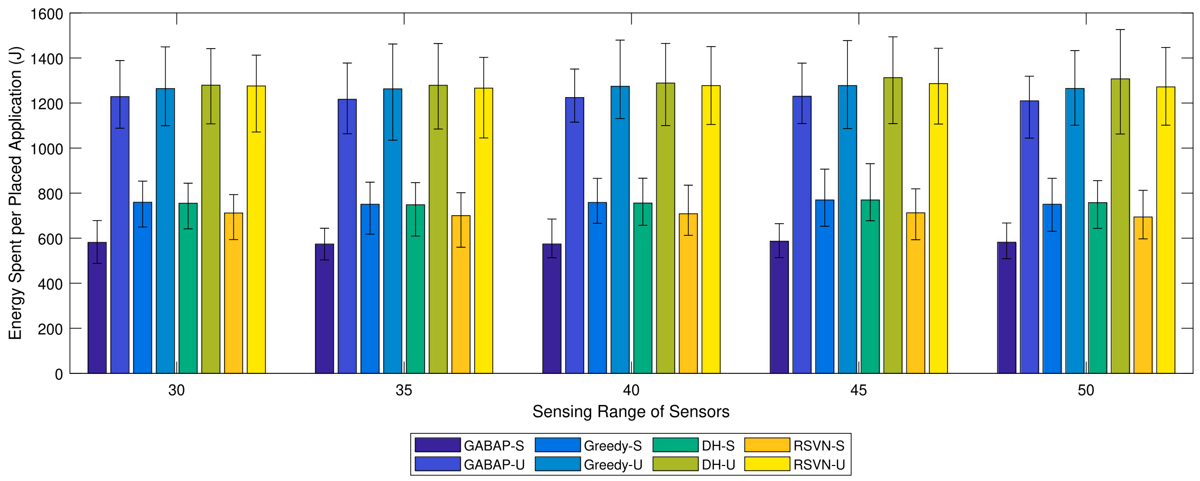


Fig. 13. Comparison of algorithms in terms of energy cost in Case 5.

cost per application which is followed by RSVN-S. DH-S is slightly better than Greedy-S. With the unshareable approach, the energy cost is similar among all compared algorithms.

The experiments corresponding to Case 5 are conducted to investigate the performance with various sensing ranges of sensor nodes. Figs. 12 and 13 show the placed application count and the average energy cost, respectively, for Case 5. The sensing range of the sensor nodes does not affect the results too much. Placed application count increases slightly when sensing range increases. Energy cost does not change much since the sensing range is not involved in energy calculations in our model directly as the communication range. The performance in terms of both the placed application counts and the energy cost is similar to the previous four cases. GABAP-S is the best for both metrics, GABAP-U and RSVN-S has the next-best performance. In terms of the energy cost, the shareable approach consumes far less energy than the unshareable one.

In Case 6, we investigate the impact of batch size on the performance of the algorithms. Total application count is not limited, and since we have 10 batches, in this case, it is equal to $10 \times BatchSize$. Figs. 14 and 15 present the results of Case 6. Since we do not limit the total application count, the results are similar to Case 1, where the effect of the number of applications on the performance is measured.

Since we do not limit the number of applications in Case 6, the application count may affect the results we gathered from the experiments. Therefore, to observe the effect of the batch size on the algorithms in a clearer way, in Case 7, the number of applications is set to 1000. Thus, the total batch count changes for each batch size. Figs. 16 and 17 present the results for Case 7. Because we investigated the impact of the batch size in Case 7, only the performance of GABAP is affected since the others admit applications one by one when GABAP selects a subset of arriving applications at each batch. With larger batch sizes, GABAP performs better.

Energy cost results are similar to the previous cases. The shareable approach is better compared to the unshareable one. GABAP performs the best and RSVN follows it with both shareable and unshareable approaches.

We can infer the following conclusions from our experimental results explained above:

- In all cases, the shareable approach has much better performance than the unshareable one both in terms of placed application and the energy cost per application.
- GABAP is clearly superior to all compared algorithms. RSVN has the second-best performance. DH is slightly better than the Greedy algorithm.
- The performance difference between the shareable and unshareable approaches is greater when network resources are more limited with a smaller number of applications. With an application count that is large enough, the performance gap also increases with plenty of network resources.

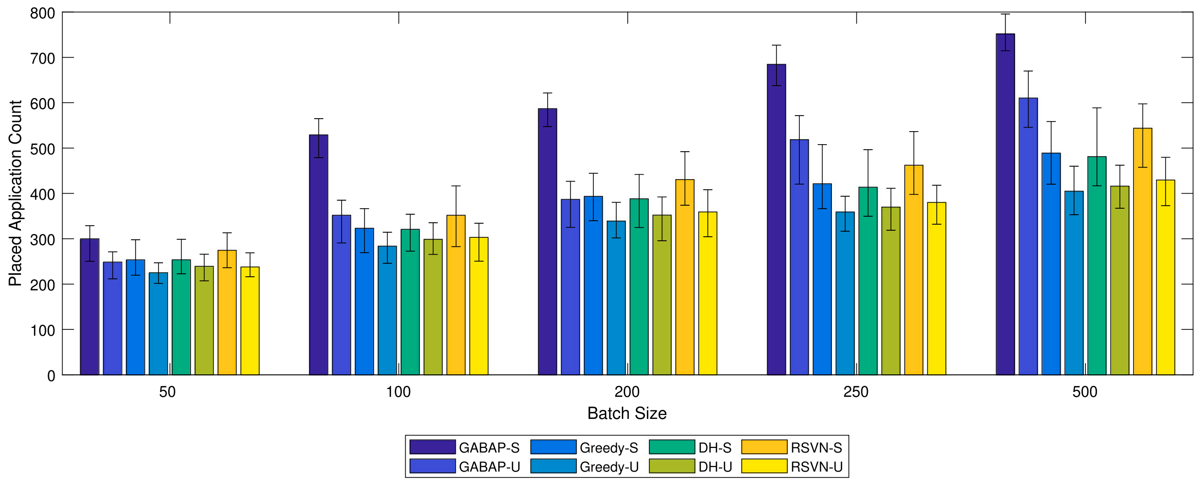


Fig. 14. Comparison of algorithms in terms of placed application count in Case 6.

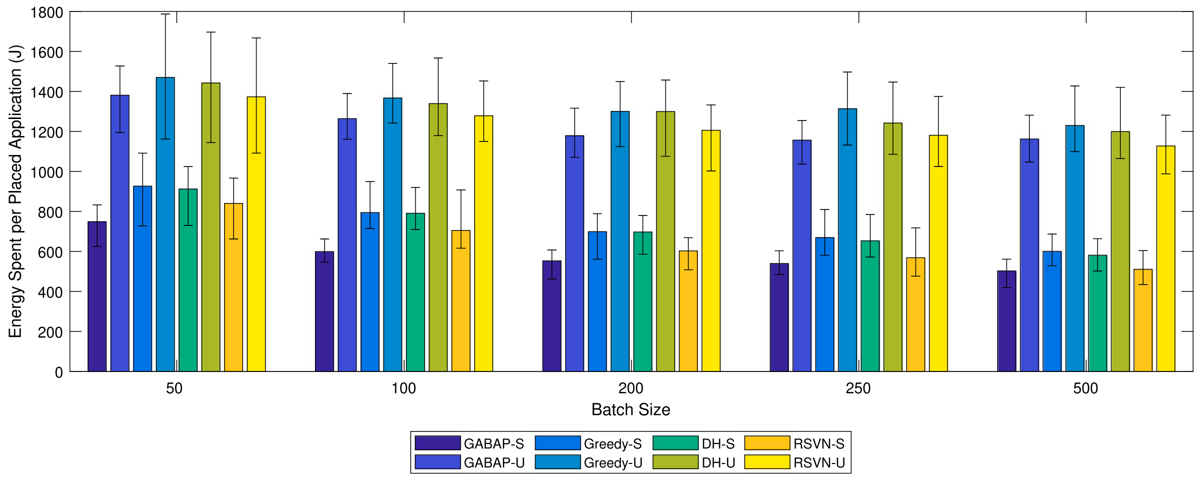


Fig. 15. Comparison of algorithms in terms of energy cost in Case 6.

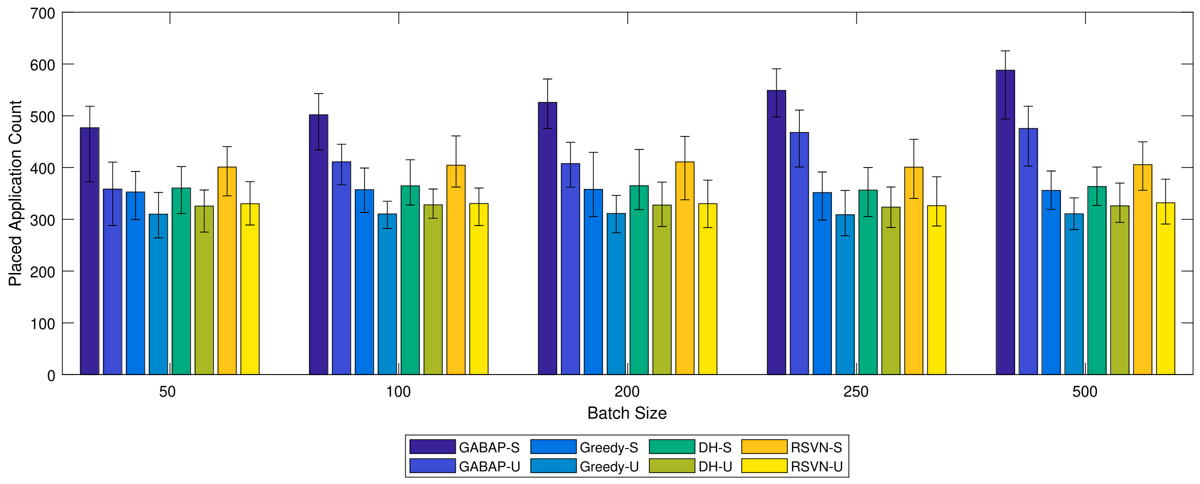


Fig. 16. Comparison of algorithms in terms of placed application count in Case 7.

- The performance difference between the shareable and unshareable approaches is larger in GABAP's results compared to other algorithms.
- Since GABAP selects a subset of applications among applications that arrive at the network at each batch while the other three try to admit applications one by one, GABAP can place applications much more optimally.
- DH and Greedy have very similar performance in all cases since their logic to place applications is similar. In some cases DH

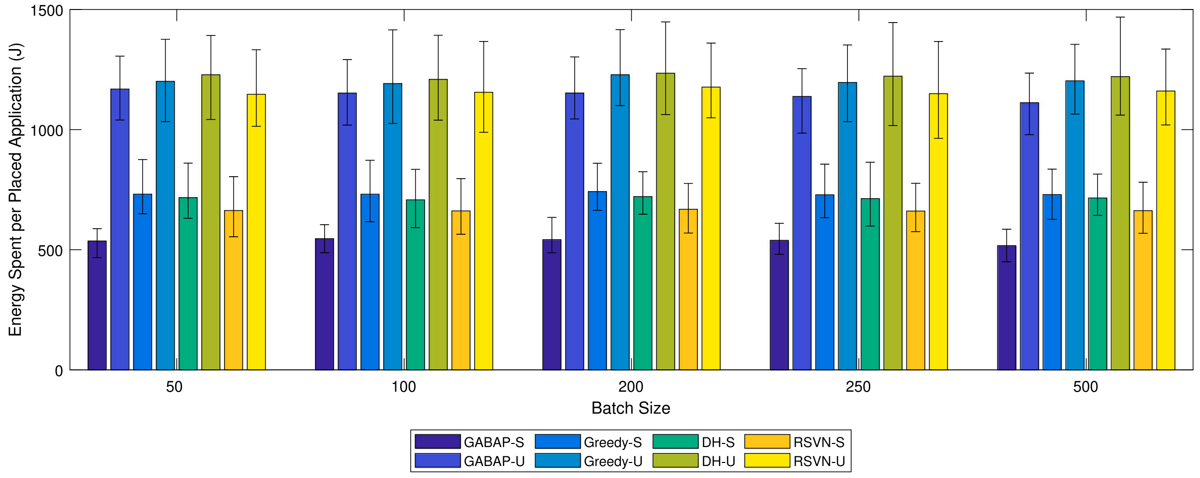


Fig. 17. Comparison of algorithms in terms of energy cost in Case 7.

Table 5
Comparison with optimal solution (Linear programming).

Application count	Mon. point count	Sensor count	Base station count	Online GABAP		Offline GABAP		Linear Prog.	
				Result	Time (ms)	Result	Time (ms)	Result	Time (ms)
30	20	30	3	1.23	27	1.54	21	1.54	3320
30	20	30	4	1.57	36	1.72	19	1.72	3397
30	20	30	5	2.01	42	2.17	27	2.17	4245
50	20	30	4	2.60	51	2.92	38	2.92	3973
50	40	50	7	3.14	121	3.97	93	3.97	23 806
50	100	50	7	3.71	198	4.60	99	4.60	53 749
100	100	25	25	2.30	240	3.17	199	3.21	65 651
20	150	100	30	4.02	1007	4.02	291	4.14	914 380
30	150	100	30	4.74	1092	4.98	426	5.20	955 542
50	150	100	30	8.12	944	8.12	487	8.26	964 397

performs slightly better than Greedy, since it allows migrations while Greedy does not.

6.1. Comparison with optimal results obtained from linear programming

To investigate how close our algorithm performs to the optimal solution, we compare the results of our algorithm with optimal results of a linear programming model. We use the Java API of IBM ILOG Cplex optimizer to code our linear program.

We use the constraints described in Section 4 to build our linear programming model. Some constraints there, however, are not linear. Therefore, we linearized those constraints.

Linearization of maximum function is realized as follows. Let,

$$C = max(a_1, \dots, a_n) \tag{20}$$

Then, we transform this maximum function into a linear form with the following:

$$C \geq a_i \qquad \forall i \in N \tag{21}$$

$$C \leq a_i + (1 - b_i) * M \quad \forall i \in N \tag{22}$$

$$\sum_{i \in N} b_i = 1 \tag{23}$$

where b_i is a binary variable that indicates the maximum value, x_i. Therefore, if $b_i = 1$, then x_i is the maximum of all. M is a very large number.

Linearization of the product of a binary variable and a continuous variable is done as follows. Let,

$$z = A \times x \tag{24}$$

where A is a continuous variable and x is a binary variable. Moreover, let $\bar{A}$ be the upper bound of A. Then, transformation into linear form is done as follows:

$$z \leq \bar{A} \times x \tag{25}$$

$$z \leq A \tag{26}$$

$$z \geq A - (1 - x)\bar{A} \tag{27}$$

$$z \geq 0 \tag{28}$$

In the equations above, if x is equal to zero, then Eqs. (25) and (28) ensure that z is also equal to zero. If x is equal to one, Eqs. (26) and (27) state that z is equal to A. This transformation is a variation of the general product linearization where the lower bound of the continuous variable is zero. In our problem statement, the continuous variable is application demands which cannot be negative. Therefore, this transformation is applicable to our work.

Because the run time required by a linear programming solver is too long, we use small values for parameters of our network. We experiment with the shareable approach only. Sensing rate requirements of various data-rate types and network constraints are shown in Tables 3 and 4, respectively. The results are presented in Table 5. Both the placed application count and running-time results given in the table are the averages of 100 runs. At each run, we input the same network to all algorithms. The results show that our GABAP algorithm has a very good performance. For each parameter set, GABAP has a very close performance to the optimal results obtained from the linear programming. Besides, the running-time of our algorithm is much less than the running-time of the linear programming. Time values are also the average of 100 runs. The gap between running-times of GABAP and linear programming solution becomes larger as the number of network elements increases.

7. Conclusion and future work

In this paper we study the application placement problem in WSNs. We first propose a shareable approach with which we provide the option of sharing the data gathered from a single monitoring point among applications. Therefore, we reduce the sensing and communication overhead for a single point. Then, we propose two algorithms, a greedy algorithm and a genetic algorithm called GABAP, to solve the allocation problem. The algorithms decide which applications should be admitted, which monitoring point is sensed by which sensor and which base station will process the related data. We provide extensive simulation results, which show the effectiveness of the sharing data approach and our algorithms. We compare the performance of our GABAP and greedy algorithms with both shareable and unshareable approaches. We demonstrate that GABAP is clearly superior to the greedy algorithm for large networks. Lastly, we compare GABAP's performance with optimal results obtained from our linear programming formulation. The results show that GABAP performs very close to optimum while taking much less time compared to the linear programming solution.

Potential future work includes support for WSNs that have multi-hop architecture, considering possible interference effect on the connections, and considering other communication QoS metrics such as reliability and delay.

CRediT authorship contribution statement

Mustafa Can Çavdar: Conceptualization, Methodology, Software, Investigation, Validation, Writing – original draft, Writing – review & editing, Visualization. **Ibrahim Korpeoglu:** Conceptualization, Writing – review & editing, Supervision, Project administration. **Özgür Ulusoy:** Conceptualization, Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Q. Jiang, F. Kresin, A.K. Bregt, L. Kooistra, E. Pareschi, E. Van Putten, H. Volten, J. Wesseling, Citizen sensing for improved urban environmental monitoring, J. Sensors 2016 (2016).

[2] A. Nasir, B.-H. Soong, S. Ramachandran, Framework of WSN based human centric cyber physical in-pipe water monitoring system, in: 2010 11th International Conference on Control Automation Robotics & Vision, IEEE, 2010, pp. 1257–1261.

[3] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, M. Welsh, Deploying a wireless sensor network on an active volcano, IEEE Internet Comput. 10 (2) (2006) 18–25.

[4] R. Mondal, T. Zulfi, Internet of things and wireless sensor network for smart cities, Int. J. Comput. Sci. Issues (IJCSI) 14 (5) (2017) 50–55.

[5] N. Maisonneuve, M. Stevens, M.E. Niessen, P. Hanappe, L. Steels, Citizen noise pollution monitoring, in: Proceedings of the 10th International Digital Government Research Conference, Association for Computing Machinery, 2009, pp. 96–103.

[6] S. Lee, D. Yoon, A. Ghosh, Intelligent parking lot application using wireless sensor networks, in: 2008 International Symposium on Collaborative Technologies and Systems, IEEE, 2008, pp. 48–57.

[7] D. Kandris, C. Nakas, D. Vomvas, G. Koulouras, Applications of wireless sensor networks: an up-to-date survey, Appl. Syst. Innov. 3 (1) (2020) 14.

[8] C. Delgado, M. Canales, J. Ortín, J.R. Gállego, A. Redondi, S. Bousnina, M. Cesana, Joint application admission control and network slicing in virtual sensor networks, IEEE Internet Things J. 5 (1) (2017) 28–43.

[9] F.-Y. Wang, L. Yang, X. Cheng, S. Han, J. Yang, Network softwarization and parallel networks: beyond software-defined networks, IEEE Netw. 30 (4) (2016) 60–65.

[10] M. Casado, M.J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, S. Shenker, Rethinking enterprise network control, IEEE/ACM Trans. Netw. (ToN) 17 (4) (2009) 1270–1283.

[11] S. Sezer, S. Scott-Hayward, P.K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, N. Rao, Are we ready for SDN? Implementation challenges for software-defined networks, IEEE Commun. Mag. 51 (7) (2013) 36–43.

[12] V.M. Raee, D. Naboulsi, R. Glitho, Energy efficient task assignment in virtualized wireless sensor networks, in: 2018 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2018, pp. 976–979.

[13] T. Ojha, S. Misra, N.S. Raghuwanshi, H. Poddar, DVSP: Dynamic virtual sensor provisioning in sensor-cloud based internet of things, IEEE Internet Things J. (2019).

[14] M. Lemos, R. Rabêlo, C. de Carvalho, D. Mendes, V. Costa, et al., An energy-efficient approach to enhance virtual sensors provisioning in sensor clouds environments, Sensors 18 (3) (2018) 689.

[15] V. Rahmati, Near optimum random routing of uniformly load balanced nodes in wireless sensor networks using connectivity matrix, Wirel. Pers. Commun. 116 (4) (2021) 2963–2979.

[16] C. Delgado, S. Batista, M. Canales, J.R. Gállego, J. Ortín, M. Cesana, An implementation for dynamic application allocation in shared sensor networks, in: 2018 11th IFIP Wireless and Mobile Networking Conference (WMNC), IEEE, 2018, pp. 1–8.

[17] I. Leontiadis, C. Efstratiou, C. Mascolo, J. Crowcroft, SenShare: transforming sensor networks into multi-application sensing infrastructures, in: European Conference on Wireless Sensor Networks, Springer, 2012, pp. 65–81.

[18] S. Bhattacharya, A. Saifullah, C. Lu, G.-C. Roman, Multi-application deployment in shared sensor networks based on quality of monitoring, in: 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium, IEEE, 2010, pp. 259–268.

[19] S.M. Ajmal, S. Paris, Z. Zhang, F.N. Abdesselam, An efficient admission control algorithm for virtual sensor networks, in: 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), IEEE, 2014, pp. 735–742.

[20] V. Cionca, R. Marfievici, R. Katona, D. Pesch, JudiShare: Judicious resource allocation for qos-based services in shared wireless sensor networks, in: 2018 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2018, pp. 1–6.

[21] S. Bousnina, M. Cesana, J. Ortín, C. Delgado, J.R. Gállego, M. Canales, A greedy approach for resource allocation in virtual sensor networks, in: 2017 Wireless Days, IEEE, 2017, pp. 15–20.

[22] R. Tynan, G.M. O'Hare, M.J. O'Grady, C. Muldoon, Virtual sensor networks: An embedded agent approach, in: 2008 IEEE International Symposium on Parallel and Distributed Processing with Applications, IEEE, 2008, pp. 926–932.

[23] Y. Xu, A. Saifullah, Y. Chen, C. Lu, S. Bhattacharya, Near optimal multi-application allocation in shared sensor networks, in: Proceedings of the Eleventh ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, 2010, pp. 181–190.

[24] E. Uchiteleva, A. Shami, A. Refaey, Virtualization of wireless sensor networks through MAC layer resource scheduling, IEEE Sens. J. 17 (5) (2016) 1562–1576.

[25] Z. Wei, F. Liu, Y. Zhang, J. Xu, J. Ji, Z. Lyu, A Q-learning algorithm for task scheduling based on improved SVM in wireless sensor networks, Comput. Netw. 161 (2019) 138–149.

[26] Y. Li, Z. Zhang, S. Xia, H.-H. Chen, A load-balanced re-embedding scheme for wireless network virtualization, IEEE Trans. Veh. Technol. 70 (4) (2021) 3761–3772.

[27] C. Abreu, F. Miranda, P. Mendes, Smart context-aware QoS-based admission control for biomedical wireless sensor networks, J. Netw. Comput. Appl. 88 (2017) 134–145.

[28] N. Edalat, M. Motani, Energy-aware task allocation for energy harvesting sensor networks, EURASIP J. Wireless Commun. Networking 2016 (1) (2016) 28.

[29] T.L. Porta, C. Petrioli, C. Phillips, D. Spenza, Sensor mission assignment in rechargeable wireless sensor networks, ACM Trans. Sensor Netw. 10 (4) (2014) 60.

[30] C.M. de Farias, L. Pirmez, F.C. Delicato, W. Li, A.Y. Zomaya, J.N. de Souza, A scheduling algorithm for shared sensor and actuator networks, in: The International Conference on Information Networking 2013 (ICOIN), IEEE, 2013, pp. 648–653.

[31] L. Malathi, R. Gnanamurthy, K. Chandrasekaran, Energy efficient data collection through hybrid unequal clustering for wireless sensor networks, Comput. Electr. Eng. 48 (2015) 358–370.

[32] R.M. Karp, Reducibility among combinatorial problems, in: Complexity of Computer Computations, Springer, 1972, pp. 85–103.

[33] E. Yakıcı, M. Karatas, Solving a multi-objective heterogeneous sensor network location problem with genetic algorithm, Comput. Netw. 192 (2021) 108041.

[34] S. Rani, S.H. Ahmed, R. Rastogi, Dynamic clustering approach based on wireless sensor networks genetic algorithm for IoT applications, Wirel. Netw. 26 (4) (2020) 2307–2316.

[35] A.M. Maia, Y. Ghamri-Doudane, D. Vieira, M.F. de Castro, An improved multi-objective genetic algorithm with heuristic initialization for service placement and load distribution in edge computing, Comput. Netw. 194 (2021) 108146.
[36] A. Vlavianos, L.K. Law, I. Broustis, S.V. Krishnamurthy, M. Faloutsos, Assessing link quality in IEEE 802.11 wireless networks: Which is the right metric? in: 2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications, IEEE, 2008, pp. 1–6.
[37] B. Demirel, A. Aytekin, D.E. Quevedo, M. Johansson, To wait or to drop: On the optimal number of retransmissions in wireless control, in: 2015 European Control Conference (ECC), IEEE, 2015, pp. 962–968.

Mustafa Can Çavdar received his BS and MS degrees in computer engineering from Bilkent University, Ankara, Turkey, in 2014 and 2016, respectively. He is currently working towards the Ph.D. degree in Computer Engineering at Bilkent University. His research interests include cloud computing, wireless networks, and computer networks.

Ibrahim Korpeoglu received his Ph.D. and M.S. in Computer Science from University of Maryland at College Park in 2000 and 1996, respectively. He received his B.S. degree (summa cum laude) in Computer Engineering, from Bilkent University in 1994. Currently he is a full professor in Department of Computer Engineering at Bilkent University. Before joining Bilkent, he worked in Ericsson, IBM T.J. Watson Research Center, Bell Laboratories, and Bellcore, in USA. He received Bilkent University Distinguished Teaching Award in 2006. He speaks Turkish, English and German. His research interests include computer networks, wireless networks, cloud computing, and distributed systems. He is a member of ACM and a senior member of IEEE.

Özgür Ulusoy is a professor at the Department of Computer Engineering, Bilkent University, Ankara, Turkey. He has a Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign, USA. His current research interests include web databases and web information retrieval, multimedia database systems, social networks, and cloud computing. He has published over 130 articles in archived journals and conference proceedings. He is a member of IEEE and ACM.