Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 05, No. 01, April 2019.

MODIFIED RANDOM BIT CLIMBING (λ -MRBC) FOR TASK MAPPING AND SCHEDULING IN WIRELESS SENSOR NETWORKS

Yousef E. M. Hamouda

(Received: 8-Nov.-2018, Revised: 17-Dec.-2018, Accepted: 23-Dec.-2018)

ABSTRACT

This paper examines the problem of Task Mapping and Scheduling (TMS) in Wireless Sensor Networks (WSNs). The application, which is supposed to be executed in WSNs, can be divided into interdependent tasks. The key objectives of TMS in WSNs are the improvement of execution time, energy consumption and network lifetime. A modified version of Random Bit Climbing (RBC) optimization method, also called λ-Modified Random Bit Climbing (λ-mRBC), is developed to get better and faster optimal or near-optimal solution. In the proposed λ-mRBC method, a new operator, called transposition operator, is added to improve the exploration of search space and hence to escape from the local optima. The deepth of exploration is controlled by using a single parameter (λ). Firstly, a number of sensor nodes is selected to cooperatively execute the application with the purpose of improving the network lifetime. After that, the proposed λ-mRBC method is performed to get the optimal or near-optimal task/sensor pair solution, so that the execution time and energy consumption are minimized.

The simulation results show that λ-mRBC method enhances the TMS performance. Compared with the traditional RBC method, the proposed λ-mRBC method converges to better fitness value, make-span and total energy consumption by 19.1%, 19.6% and 22.3%, respectively. Furthermore, the network lifetime is prolonged through using the proposed selection algorithm. The distribution of remaining energy among sensor nodes is improved about three times, compared with the random selection scheme. Furthermore, compared with the random selection, the number of neighbours for sensor nodes is improved by 20.1% using the proposed selection algorithm.

KEYWORDS

Application DAG, Optimization methods, Random bit climbing, Task mapping and scheduling, Wireless sensor networks.

1. INTRODUCTION

Sophisticated technologies and applications, such as smart homes, Internet of Things (IoT), smart grid, precision agriculture and automated control have provoked the need of developing self-organized, multi-hop and *ad-hoc* Wireless Sensor Networks (WSNs). WSNs are made up of hundreds or thousands of tiny and cheap sensor nodes with limited resources. Sensor nodes cooperate with each other to execute the applications. In addition, sensor nodes are scattered randomly or in a planned manner to monitor and control the field of interest [1]. Sensor node consists of energy unit, processing unit, sensing unit, wireless communication unit and storage unit [2]. In several applications, WSNs are positioned in sites that are difficult to be physically accessed; i.e., forest. Therefore, network lifetime is an essential requirement for WSNs to prolong the lifetime of the sensor nodes and the network connectively [3, 4]. Lots of civil and military applications employ WSNs. Civil applications, for example, include healthcare [5], precision irrigation [6], smart grid [7], home automation [8] and surveillance [9], while military-based applications usually include intrusion detection and detection of illegal crossings [10].

Given the fact that the sensor nodes have limited resources, improving the energy-efficiency and application execution time of WSNs seems to be plausible to increase the network lifetime [11]. In fact, energy is consumed from the battery during sensing, communicating and processing activities. In

Y. E. M. Hamouda is with Department of Computer, Al-Aqsa University, Gaza, Palestine. Email: ye.hamouda@alaqsa.edu.ps

addition, numerous applications in WSNs require massive in-network processing capability. For instance, smart visual sensor networks usually go through several subsequent executional jobs, including image processing, computer vision and image sensing [12]-[13]. Furthermore, application should complete its execution at the right time, after which the execution of the application will not be useful anymore. In most cases, sensor node is not fast enough to execute the complex application in a reasonable amount of time.

Parallel computing refers to the methods that solve a problem within a reasonable amount of time (i.e., called make-span) by dividing it into smaller parts and solving the parts using multiple physical processors [14]. As a result, a complicated application of WSNs is decomposed into smaller tasks. Afterward, Task Mapping and Scheduling (TMS) techniques are developed in order to share the execution of divided tasks among sensor nodes [15]. As shown in Figure 1, four phases are required to perform a parallel system [35]: The first phase is called task decomposition. In this phase, the application is divided into small tasks. These small tasks depend on each other by using dependencies. The dependence analysis is performed in the second phase to order the tasks in line with the dependencies. Task graphs, such as Direct Acyclic Graphs (DAGs), are employed to model the tasks with dependencies. The third phase is called task mapping and scheduling. The main purpose of task mapping is to allocate the tasks to the processors or computing nodes. Consequently, the task scheduling aims to order the task execution according to the dependencies through determining the start times of task execution. The fourth phase is called parallel programming and aims to develop the application, based on the result of task mapping and scheduling. This paper considers the mechanism of task mapping and scheduling, explained in the third phase.

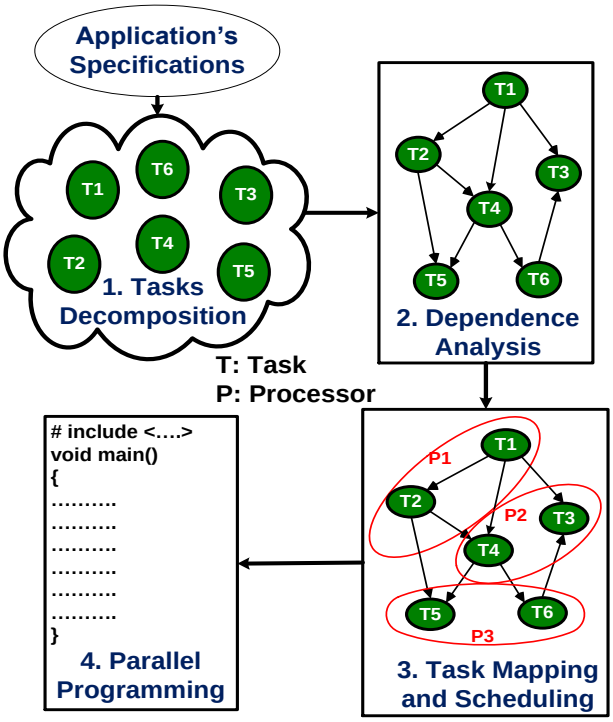


Figure 1. Task mapping and scheduling concept.

However, unlike the traditional parallel computing systems, TMS in WSNs focuses not only on the execution time of the application, but also on energy-efficient schemes that prolong the network lifetime. Therefore, TMS is generally modelled as a multi-objective optimization problem, which has been proved to be non-deterministic polynomial-time (NP)-hard [16].

Metaheuristics are used to get a satisfactory solution of optimization problem that fulfils the required objective function. Metaheuristics are strategies that depend on a guide to examine the search space to get the optimal (or near-optimal) solution, without the need of testing every solution in the search space [17]. A metaheuristic has two main properties: diversification and intensification [18]. Diversification is the exploration of search space in order to escape from the local optimal. Intensification refers to the process of exploitation of accumulated search space. Indeed,

diversification needs a time to be performed, whereas intensification looks deeply and locally for high-quality solutions. Therefore, dynamic balancing between exploration and exploitation is necessary for a good metaheuristic [17]. In literature, metaheuristics are classified into single-solution metaheuristics and population-based metaheuristics. According to the operation form of metaheuristics, single-solution metaheuristics are of more intensification, while population-based metaheuristics are of more diversification [18].

This paper introduces λ-Modified Random Bit Climbing (λ-mRBC) optimization method to attempt to solve the problem of TMS in WSNs. The claim of this research is to improve the traditional RBC optimization method. The main contributions of the proposed λ-mRBC optimization method are: (1) a novel modification in RBC method is developed to improve the convergence speed and fitness value of the final solution. (2) The researcher thinks that this research might be the first one to apply the RBC method and its proposed modified version (λ-mRBC) in TMS problem. (3) Lifetime Awareness Sensor Node Selection Algorithm (LA-SNSA) is incorporated to select the sensor nodes so that the lifetime of the network is improved. (4) Heterogeneous sensor nodes with different processing, energy level and energy consumption are used in the proposed algorithms.

The paper contains seven sections as follows: Section (2) explores work related to task allocation in WSNs. In Section (3), the network framework for TMS is introduced. Then, Section (4) defines and formalizes the research problems. After that, the proposed λ-mRBC and LA-SNSA schemes are introduced in Section (5). Simulation results are shown and discussed in Section (6). Finally, the paper is concluded in Section (7).

2. RELATED WORK

Task mapping and scheduling problems have been deeply discussed in WSNs. In [19]-[20], Genetic Algorithm (GA) is used to provide well-performing task allocation. A Modified Binary Particle Swarm Optimization (MBPSO) algorithm is presented in [21] to find the optimal task allocation solution. In [22], logic gate-based evolutionary algorithm is used to solve the problem of task allocation in WSNs. However, the population-based metaheuristics used in the above research require high processing power, energy consumption and execution time. Furthermore, high complexity optimization algorithms are not appropriate for limited resource WSNs.

Integer linear programming is used in [23] to optimally assign complex tasks to sensor nodes to minimize total energy consumption. In [24], task allocation is introduced so that energy consumption and network lifetime are improved. However, the execution time (i.e., make-span) has not been taken into account [23]-[24], which leads the application to take long time to be executed.

In [25], a distributed task allocation is introduced. The task is made to move from a sensor node to another. The suitable sensor node with enough capacity to execute the task is found. In [26], Topology-Aware Task Allocation and Scheduling (TATAS) is introduced to map and schedule the tasks to the sensor nodes. However, the task allocation presented in [25]-[26] assumes independent tasks which are not practical for complex application, such as visual surveillance [27].

In [28], a real-time task mapping and scheduling (RT-MapS) algorithm is developed for collaborative in-network processing in single-hop cluster WSN using Dynamic Voltage Scaling (DVS) feature. In [27], Multi-hop Task Mapping and Scheduling (MTMS) solution is developed for TMS in multi-hop cluster WSN. Nevertheless, MTMS and RT-MapS prevent task mapping to sensor nodes that execute the immediate predecessors of the task. As a result, this leads to using more sensor nodes for TMS and including all sensor nodes in the task mapping decision-making.

In [29], Biological Task Mapping and Scheduling (BTMS) approach is introduced, where the application is executed by a group of sensor nodes so that the execution time and energy consumption are improved. However, the network lifetime related to sensor neighbour count is not considered. In [30], Light Allocation of Tasks (LAT) algorithm is presented to enhance energy efficiency, network lifetime and application execution time. However, LAT algorithm includes all sensor nodes in decision-making for TMS.

Task Level Parallelism (TLP) in WSN is introduced in [31] to parallelize the execution of smart health care applications so that the processing time is reduced. Nevertheless, scheduling of the task execution

is not considered. An energy-efficient Complicated Task Solution scheme for real-time task processing based on node Cooperation (CTSC) is tackled in [32] to allocate more tasks to sensor nodes with a higher energy-level. However, CTSC maps all dependent tasks to the same sensor nodes which could cause exhaustion for the energy level of sensor node.

In [33], Machine-to-Machine (M2M) architecture with sensor devices and limited resources is considered. Tasks are allocated to the nodes of M2N so that the lifetime is maximized. However, the task allocation algorithm proposed in [33] finds all possible task allocation possibilities which need high processing and time. In addition, the execution time is not considered in [33]. In [34], complex application is allocated for different clustered wireless sensors. Firstly, tasks are distributed to clusters so that the energy consumption is minimized. Then, tasks allocated to each cluster are assigned to the nodes within the cluster so that energy cost and load balancing are improved. In [35], the problem of task allocation in IoT applications is considered, where the embedded devices of IoT are assumed to have limited resources. The tasks are allocated so that the energy consumption is minimized. In [36], tasks are allocated locally to the slave sensor nodes or globally to the master sensor node, so that the network life time is maximized. However, the execution time is not considered in [34]-[36].

In this paper, λ-Modified Random Bit Climbing (λ-mRBC) optimization method is developed to solve the problem of TMS in WSNs. The proposed method supports heterogeneous sensor nodes. Actually, the proposed λ-mRBC method is different from the previous one through the use of a modified version of RBC method with faster conversion speed and better final solution. Moreover, the proposed λ-mRBC is a single-solution metaheuristic with single algorithm parameter. Therefore, it needs less processing capabilities to be executed and in turn it is suitable for the sensor nodes with limited resources. Finally, Lifetime Awareness Sensor Node Selection Algorithm (LA-SNSA) is developed to select the sensor nodes to enhance network lifetime.

3. NETWORK FRAMEWORK

As shown in Figure 2, the sensor nodes are randomly distributed in the monitoring area. The sensor nodes are connected with each other wirelessly. The sink node aims to pass on the data from the monitoring area to the main controller *via* Internet, satellite or cellular networks. Sensor node knows its location using Global Positioning System (GPS) [37]. Nonetheless, only few sensor nodes use GPS to know their locations and other sensor nodes can calculate their locations using triangulation [38]. At time step (k), the neighbours of a target sensor node (s_{TSN}) are a set $N_{s_{TSN}}(k)$. The neighbours with remaining energy level above a predefined threshold value (E_{th}) can participate to execute an application DAG (A_d). These particular neighbours are saved in an $S_s(k, A_d)$ set of $n_s(k, A_d)$ sensor nodes. After that, a set $(S_g(k, A_d))$ of $n_g(k, A_d)$ nodes is selected from $S_s(k, A_d)$ to execute the application DAG (A_d). The selection of $n_g(k, A_d)$ sensor nodes is performed to improve the network lifetime. The application DAG (A_d) is assumed to be decomposed into interdependent tasks. Then, TMS is incorporated to cooperatively execute the application tasks using the selected sensor nodes, so that the time and energy required to execute the application DAG are reduced.

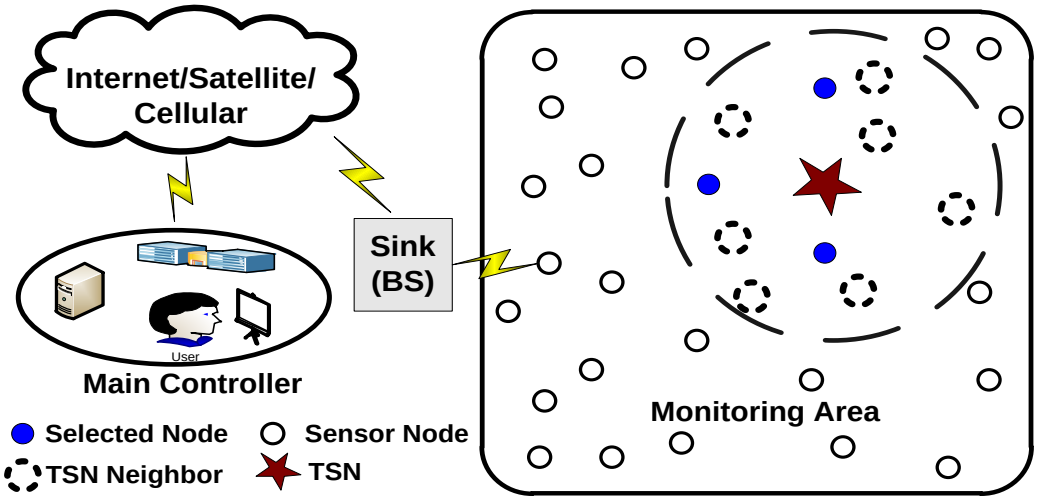


Figure 2. Network framework.

4. PROBLEM DEFINITION AND FORMALIZATION

4.1 Application Model

In this paper, the application is modelled using Direct Acyclic Graphs (DAG) [21]. So, the application is divided into smaller tasks. DAG can also model the interdependencies among tasks [39]. Figure 3 shows as example of application DAG.

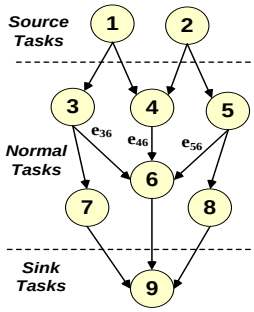


Figure 3. Application DAG.

The application DAG is modelled as $A_d = (V, E)$. The set V represents "n" application tasks, where $V = \{v_i : i = 1,2,..n\}$. Similarly, the set E represents "q" communication interdependencies, where $E = \{e_k : k = 1,2,..q\}$. The edge $e_k \in E$ between the tasks v_i and v_j is denoted as e_{ij}, where v_j is called the immediate successor of v_i and v_j is called the immediate predecessor of v_i. Accordingly, the task is executed when it receives all of its immediate predecessor's output. The entry-tasks or source-task do not have immediate predecessors. In addition, a task without immediate successors is called an exit-task or a sink-task. In WSNs, the entry-tasks are used for sensing or gathering the raw data to detect physical phenomena. Therefore, task placement constraints can be defined as an only one source task that can be assigned to the sensor node. In Figure 3, v_1 and v_2 are source-tasks, v_9 is the sink-task, v_1 and v_2 are the immediate predecessors of v_4 and v_9 is the immediate successor of v_8. The task v_6 cannot be executed until it receives the communication interdependencies (e36, e46 and e56) from its immediate predecessors (v_3 , v_4 and v_5).

Each task, $v_i \in V$ is modelled as a tuple of the form: $\{N_{v_i}, t_{v_i}, E_{v_i}\}$, where N_{v_i} is the number of the computational cycles of the task, E_{v_i} is computational energy consumption of the task and t_{v_i} is computational time of the task. Each edge (e_{ij}) between the tasks v_i and v_j is modelled as a tuple of the form: $\{l_{e_{ij}}, t_{e_{ij}}, E_{e_{ij}}\}$. $l_{e_{ij}}$ is the data size generated from v_i and is required to execute v_j. $E_{e_{ij}}$ and $t_{e_{ij}}$ are the communication energy consumption and communication time required to send e_{ij} from the sensor node that executes the task v_i to the sensor node that executes the task v_j.

4.2 The Wireless Sensor Network Model

The WSN is composed of a number of heterogeneous sensor nodes distributed randomly in the area of interest. The sensor nodes have different specifications, such as processing speed, power consumption and transmission distances. WSN is modelled as a graph $W = (S, D)$, where $S = \{s_x : x = 1,2,..m\}$ is the set of heterogeneous sensor nodes and $D = \{d_k : k = 1,2,..p\}$ is a set of communication links among sensor nodes. The edge $d_k \in D$ between the sensor nodes s_x and s_y is denoted as d_{xy} and is the physical distance between sensor nodes s_x and s_y.

Sensor node, s_x is modelled as a tuple of several properties and states as follows: $s_x = \{ID_{s_x}, x_{s_x}, y_{s_x}, E_r(k, s_x), f_{s_x}, e_{s_x}, a_{s_x}\}$, where ID_{s_x} is the sensor node identification, x_{s_x}, y_{s_x} are the xy coordination of sensor node, $E_r(k, s_x)$ is the battery remaining energy of sensor node at time k, e_{s_x} is the average power consumption for the processor of node (s_x), a_{s_x} is the time at which the sensor node is available to execute a task and f_{s_x} is the processing speed of sensor node. Sensor nodes s_x and s_y can directly communicate if the distance between them, d_{ij} is less than or equal to the radio range, R_r. The distance between sensor nodes (s_x and s_y) is calculated using Euclidean distance according to the following equation:

$$d_{xy} = \sqrt{(x_x - x_y)^2 + (y_x - y_y)^2}. \quad (1)$$

Therefore, at time k, the sensor node, s_x has a set of $m_{s_x}(k)$ neighbours, $N_{s_x}(k)$, where: $N_{s_x}(k) = \{s_l: \forall k\ satisfies\ d_{xl} \leq R_r\}$.

4.3 Cost Functions

4.3.1 Execution Time

The CPU clock frequency is defined as the number of computational cycles that can be executed per second. Therefore, the computational time (t_{v_i}) required to execute the task (v_i) is computed using the following formula:

$$t_{v_i} = \frac{N_{v_i}}{f_{s_x}} \quad (2)$$

where f_{s_x} is the CPU clock frequency of sensor node (s_x) which executes the task (v_i). The total computational time (called serial execution time) required to computationally execute the application tasks is the sum of computational times for all tasks and is calculated as follows:

$$t_T^p = \sum_{k=1}^{n} t_{v_k} \quad (3)$$

The communication time ($t_{e_{ij}}$) required to send the e_{ij} from the sensor node that executes the task v_i to the sensor node that executes the task v_j is made up of transmission time, queue time and propagation time. The queue time is the latency caused by media access to avoid interference and collision. Therefore, the communication time ($t_{e_{ij}}$) is computed as follows:

$$t_{e_{ij}} = t_{e_{ij}}^t + t_{e_{ij}}^p + t_{e_{ij}}^q = \frac{l_{e_{ij}}}{R_b} + \frac{d}{c} + t_{e_{ij}}^q \quad (4)$$

where $t_{e_{ij}}^t$ is the transmission time which is the data size ($l_{e_{ij}}$) divided by the data rate or communication bandwidth (R_b), $t_{e_{ij}}^p$ is the propagation time which is the distance between the sensor nodes that exchange the edge (d) divided by the speed of light ($c = 3 \times 10^8 m/s$) and $t_{e_{ij}}^q$ is the queue time. The total communication time required to exchange all the interdependences of the application tasks is the sum of all communication times required to send all dependencies and is determined as follows:

$$t_T^c = \sum_{k=1}^{q} t_{e_{ij}} \quad (5)$$

The node/task pairs are modelled as $P(v, s)$, where $P(v, s)$ shows the "n" mapped tasks of application DAG (the set) with its corresponding "$n_g(k, A_d)$" assigning sensor nodes which are the set $S_g(k, A_d)$. Hence, the overall time required to execute the application tasks using node/task pair ($P(v, s)$) is the sum of the serial execution time and total communication time and is calculated as:

$$t_T[P(v, s)] = t_T^p + t_T^c \quad (6)$$

Each task (v_i) mapped to sensor node (s_x) is starting to be executed at a time called starting executing time of the task ($t_s(v_i, s_x)$). The task is executed when the sensor node is available after it receives all the task dependencies. It is assumed that $t_{max}[pred(v_k)]$ is the time at which the last dependency (i.e., predecessor) of task (v_k) is received by the node (s_x). Therefore, after receiving the last dependency, the task (v_k) can be executed if the CPU of sensor node (s_x) is available. The time to which the sensor node (s_x) is available is referred as (a_{s_x}). Thus, $t_s(v_i, s_x)$ is the maximum of one of the two: ($t_{max}[pred(v_k)]$) or (a_{s_x}). $t_s(v_i, s_x)$ and is calculated as:

$$t_s(v_i, s_x) = max\{a_{s_x},\ t_{max}[pred(v_k)]\} \quad (7)$$

When the sensor node (s_x) starts to execute the tasks, it finishes after a time equal to the task execution time. The time at which the task is completely executed is called the finishing execution time of the task ($t_f(v_i, s_x)$), which is the time at which the task is started to be executed ($t_s(v_i, s_x)$)

plus the task execution time (t_{v_i}) and is given by:

$$t_f(v_i, s_x) = t_s(v_i, s_x) + t_{v_i} \tag{8}$$

The make-span of the application DAG is the time at which the application execution completely finishes. Due to parallelism, the make-span will be less than ($t_T[P(v,s)]$). The execution of application is completed after finishing of execution of last task. Thus, the finishing execution time of the last task will be the biggest finishing execution time. Hence, the biggest finishing execution time is the make-span and is calculated as follows:

$$ms[A_d, P(v,s)] = max_{\forall s_x \in S_g(k,A_d)}\{\, t_f(v_i, s_x)\,\} \tag{9}$$

4.3.2 Energy Consumption

The computational energy consumption (E_{v_i}) required to execute the task (v_i) is computed using the following formula:

$$E_{v_i} = e_{s_x}.t_{v_i} \tag{10}$$

where e_{s_x} is the average power consumption for the processor of node (s_x). The energy consumption ($E_{e_{ij}}$) required to send the e_{ij} from the sensor node that executes the task v_i to the sensor node that executes the task v_j is calculated as:

$$E_{e_{ij}} = E_{e_{ij}}^{TX} + E_{e_{ij}}^{RX} \tag{11}$$

where $E_{e_{ij}}^{TX}$ is the transmitted energy consumption dissipated from the source node and $E_{e_{ij}}^{RX}$ is the received energy consumption dissipated from the destination node. $E_{e_{ij}}$ is equal to zero if the tasks v_i and v_j are mapped to the same sensor node. $E_{e_{ij}}^{TX}$ and $E_{e_{ij}}^{RX}$ are calculated as follows [40]-[41]:

$$E_{e_{ij}}^{TX} = \left(e_{elec} + \varepsilon_{amp}.d^2\right).l_{e_{ij}} \tag{12}$$

$$E_{e_{ij}}^{RX} = e_{elec}.l_{e_{ij}} \tag{13}$$

where e_{elec} is the electronic energy required to transmit a bit that depends on coding, modulation and filtering and ε_{amp} is related to the radio energy. The total processing energy consumption (called serial energy consumption) required to computationally execute the application tasks is determined as follows:

$$E_T^p = \sum_{k=1}^{n} E_{v_k} \tag{14}$$

The total communication energy consumption required to exchange the interdependences of the application tasks is calculated as follows:

$$E_T^c = \sum_{k=1}^{q} E_{e_{ij}} \tag{15}$$

The overall energy consumption required to execute the application tasks using node/task pair, $P(v,s)$ is calculated as:

$$E_T[P(v,s)] = E_T^p + E_T^c \tag{16}$$

4.4 Problem Definition

At time step k, a target sensor node (s_{TSN}) triggers a request to collaboratively execute an application DAG (A_d). The number of neighbours of s_{TSN} at time step k is $n_s(k, A_d)$ and is contained in a set $S_s(k, A_d)$. $S_s(k, A_d)$ participates to execute the application. However, only $n_g(k, A_d)$ sensor nodes are selected from $S_s(k, A_d)$ to execute the application DAG (A_d). The set $S_g(k, A_d)$ includes the selected $n_g(k, A_d)$ sensor nodes. The objective function is defined as the weighted sum of the total energy consumption and the make-span. It is calculated as follows:

$$F_{obj}[A_d, P(v,s)] = \alpha * \frac{ms[A_d, P(v,s)]}{t_T^p[A_d, P(v,s)]} + (1-\alpha) * \frac{E_T[A_d, P(v,s)]}{E_{T(max)}[A_d, P(v,s)]} \tag{17}$$

where $0 \leq \alpha \leq 1$ is a weighted controlled parameter, $ms[A_d, P(v,s)]$ is the make-span to execute the

application DAG (A_d) using the mapped task/sensor ($P(v,s)$), $t_T^p[A_d, P(v,s)]$ is the serial execution time of application DAG (A_d) using the mapped task/sensor ($P(v,s)$), $E_T[A_d, P(v,s)]$ is the total energy consumption to execute the application DAG (A_d) using the mapped task/sensor ($P(v,s)$) and $E_{T(max)}[A_d, P(v,s)]$ is the maximum energy consumption to execute the application DAG A_d using the mapped task/sensor ($P(v,s)$). The make-span in Equation (17) is normalized by dividing it by the serial execution time ($t_T^p[A_d, P(v,s)]$) which is the maximum time required to execute the application. Similarly, the total energy consumption in Equation (17) is normalized by dividing it by the maximum total energy consumption ($E_{T(max)}[A_d, P(v,s)]$). The main goal is to get the task/node pair ($P^*(v,s)$) which is used to execute the application. $P^*(v,s)$ is obtained so that the objective function defined in Equation (17) is minimized according to the following objective function:

$$P^*(v,s) = arg\, min_{P(v,s)} \{F_{obj}[A_d, P(v,s)]\} \tag{18}$$

5. THE MODIFIED RANDOM BIT CLIMBING

5.1 Lifetime Awareness Sensor Node Selection Algorithm (LA-SNSA)

Awareness Sensor Node Selection Algorithm (LA-SNSA) aims to select a number of $n_g(k, A_d)$ sensor nodes from the $S_s(k, A_d)$ set. The selected nodes are then kept in the $S_g(k, A_d)$ set. In addition, the selected sensor nodes ($S_g(k, A_d)$) are used to execute the application DAG (A_d). The LA-SNSA takes into account the network lifetime. Since reducing the gaps which appear because of node death in the network increases the network lifetime, the sensor nodes with higher number of neighbours are preferred to be selected. Furthermore, LA-SNSA also takes into account the current remaining energy of the sensor nodes. Thus, sensor nodes with higher remaining energy are favoured to be selected to increase the network lifetime. Therefore, the objective function of the LA-SNSA is the weighted sum of the ratio of energy of sensor node with respect to the sum of remaining energy for all nodes in $S_s(k, A_d)$ and the ratio of the number of neighbours of the sensor node with respect to the sum of the number of neighbours for all nodes in $S_s(k, A_d)$. It is computed as follows:

$$F_{obj}(k, s_x, A_d) = \beta * \frac{E_r(k,s_x)}{\sum_{\forall s_l \in S_s(k,A_d)} E_r(k,s_l)} + (1-\beta) * \frac{N_c(k,s_x)}{\sum_{\forall s_l \in S_s(k,A_d)} N_c(k,s_l)} \tag{19}$$

As seen in the above formula, β is a weighting parameter and varies in the interval [0, 1]; and $N_c(k, s_x)$ is the number of neighbours of sensor node s_x at time k . Algorithm 1 shows the LA-SNSA. The weighting parameter (β) is firstly selected. Then, the objective function for sensor nodes in $S_g(k, A_d)$ set is calculated based on Equation (19). After that, a number of $n_g(k, A_d)$ sensor nodes, with the highest objective function, are selected and added to $S_g(k, A_d)$ set.

Algorithm 1: Lifetime Awareness Sensor Node Selection Algorithm (LA-SNSA)

select β;
set $\ell = 0$;
while $\ell \leq n_g(k, A_d)$ **do**:
  for each sensor node $s_x \in \mathbf{S}_s(k, A_d)$ **do**:
    calculate $F_{obj}(k, s_x, A_d)$ based on Equation (19);
  end for;
  find the sensor s_x^* with maximum $F_{obj}(k, s_x, A_d)$;
  add s_x^* to $\mathbf{S}_g(k, A_d)$;
  remove s_x^* from next search;
  increment ℓ: $\ell = \ell + 1$;
end while;

5.2 Random Bit Climbing (RBC)

Random Bit Climbing (RBC) optimization [42]-[43] is a metaheuristic local search-based algorithm that employs a trajectory-based approach to guide the search and obtain a (near) optimal solution.

RBC is a single-solution metaheuristic, which adopts the exploitation in its operation through memorizing the best current solution. In RBC, single stochastic solution is used for each round. Firstly, an initial single-parent (p) is randomly generated and set as the current solution. After that, the objective function of the initial parent is evaluated. Then, a random arrangement of the index positions for the current solution is created and kept in the π vector. Next, a child is produced by mutating a single dimension of the current solution at a time. The child replaces the current solution if it fulfils the objective function. The evaluation of children either continues for all possible children or is terminated when the first better child is found. Then, a new random permutation is generated for the current solution. The process continues until a predefined number of iterations have been exhausted. However, the main limitation of RBC is the trap of local optimal solution because of its deficiency for exploration ability.

5.3 λ-Modified Random Bit Climbing (λ-mRBC)

Algorithm 2 shows the proposed λ-Modified Random Bit Climbing (λ-mRBC). Because the exploration is tied up to randomness [17], the λ-mRBC adopts a random parameter (λ) to use exploration in RBC operation. The solution is represented as a vector of n elements. The vector index represents the task number (from 1 to n). On the other hand, the vector value represents one of the selected sensor node numbers. In Step (1), an initial parent solution ($P(v,s)$) is generated randomly. This initial parent solution is then set as the current best solution and is stored in $C_s(v,s)$. The evaluation is performed in Step (2) to calculate the fitness value of the current best solution. In Step (3), the random permutation for the current best solution is achieved to produce the permutation vector (π). In Step (4), a new operator named random transposition operator (trans) is added into the RBC method to escape from local optima and to increase the exploration of the search space. The random transposition operator is performed on the current solution according to the following rule:

$$C_s(v,s) = \begin{cases} trans\left(C_s(v,s)\right) & r < \lambda \\ C_s(v,s) & otherwise \end{cases} \quad (20)$$

where r is a random number which uniformly distributes between [0, 1], λ an algorithm parameter number which ranges between 0 and 1 and trans is the transposition operation that randomly exchanges the places of the current best solution. The children are generated in Step (5) by cloning $C_s(v,s)$ and flipping the position π_l. After that, the child is evaluated in Step (6). In Step (7), the child replaces the current best solution if it has a better fitness value. In Step (8), the children are generated and evaluated. The algorithm flow continues to the next iteration in Step (9). The operations are repeated until the maximum iterations are exhausted in Step (10). After termination, the current best solution is returned as the suboptimal solution of the problem.

Algorithm 2: λ-Modified Random Bit Climbing (λ-mRBC)

Step (1) Compute the initial parent task/node pairs $P(v,s)$; set the current best solution $C_s(v,s) = P(v,s)$; and set $iter = 1$.
Step (2) Calculate the fitness value $F_{obj}[A_d, C_s(v,s)]$ of $C_s(v,s)$.
Step (3) Generate the random permutation $\boldsymbol{\pi} = (\pi_1, \pi_2 \ldots \pi_m)$ of the position of $C_s(v,s)$; and set $l = 1$.
Step (4) if (r < λ): execute transposition operation of $C_s(v,s)$ positions.
Step (5) Generate the child (offspring) $O_l(v,s)$ by cloning $C_s(v,s)$ and flipping the position π_l;
Step (6) Calculate the objective function $F_{obj}[A_j, O_l(v,s)]$ of $O_l(v,s)$.
Step (7) If ($\boldsymbol{F_{obj}[A_d, O_l(v,s)] < F_{obj}[A_d, C_s(v,s)]}$): replace $C_s(v,s) = O_l(v,s)$; and go to Step (9).
Step (8) If ($\boldsymbol{l > m}$): Go to Step (9)
else: Increment l: $l = l + 1$; and go to step (5);
Step (9) Increment $iter = iter + 1$.
Step (10) If ($\boldsymbol{iter > max\,iter}$): go to step (11)
else: go to Step (3)
Step (11) Return C_s as the suboptimal solution and finish.

5.4 The Complete TMS Approach

Figure 4 explains the proposed TMS approach. First of all, heterogeneous sensor nodes are created and WSN is randomly distributed. Then, the algorithm parameters for λ-mRBC and LA-SNSA are set and defined. When a target sensor node requests execution of an application, a DAG of the requested application is created. LA-SNSA is performed based on Algorithm 1 to select the sensor nodes that will cooperatively execute the application so that the network lifetime is improved. λ-mRBC is achieved based on Algorithm 2 to optimally get the best task/node pairs with minimum execution time and energy consumption. After that, λ-mRBC method is repeated until termination condition is met. Finally, the final solution of task/node pair is obtained and simulation statistics are recorded.

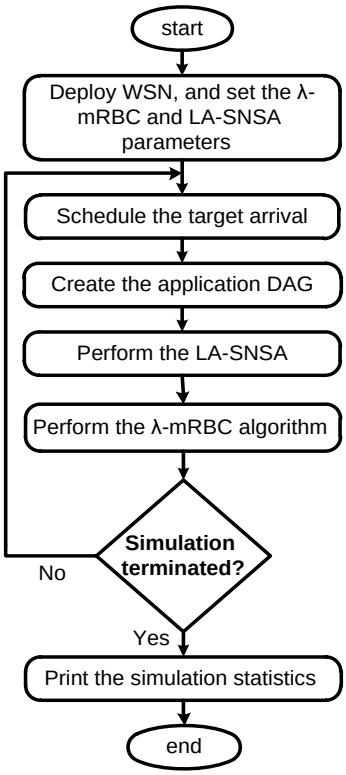


Figure 4. The proposed TMS approach.

6. SIMULATION RESULTS

This section evaluates the proposed λ-mRBC method. C++ is used to build the simulation environment using core i5 of 2.5 GHz processor and 4 GB memory.

6.1 Simulation Setting

6.1.1 The Parameters for Application DAG

Unless it is clearly stated, the application DAG consists of fifteen tasks ($n = 15$) as follows: four tasks are used as entry tasks, ten tasks are used as normal tasks and one task is used as an exit task. The immediate successors for each entry and normal tasks are selected to be uniformly distributed in the range of [1, 3]. The computation load of each task (N_{v_i}) is initialized to be uniformly distributed in the range of [300, 600] Kilo Clock Cycles (KCC). The communication load for edges among tasks (e_{ij}) is initialized to be uniformly distributed in the range of [500, 800] bytes of data.

6.1.2 The Parameters for WSN

WSN is implemented using 224 heterogeneous sensor nodes which are deployed randomly in a monitoring region of 200 m × 200 m. The transmission radio is set to $R_r = 100$ m. The radio channel with bandwidth (i.e., bit rate) of 250 Kbps is used in the simulation environment. The processing speed for sensor nodes (f_{s_x}) stands for the total number of clock cycles which can be executed within

one second. It is set to be uniformly distributed in the range of [30, 100] Million Cycles per Second (MCPS).The power consumption of the processors for sensor nodes (e_{s_x}) is set to be uniformly distributed in the range of [4, 10] mJ. The initialized energy level ($E_r(0, s_i)$) of each sensor is set to be uniformly distributed in the range of [0, 1] J.

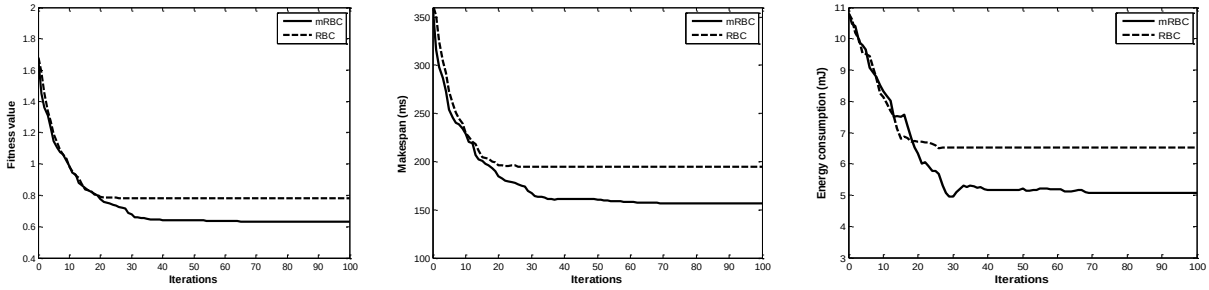
6.1.3 The Parameters for the λ-mRBC

The weighting parameters are set as follows: β=0.5 and α=0.5. The number of iterations for λ-mRBC is assumed to be 100. Unless it is clearly stated, the number of selected nodes to execute the application is $n_g(k, A_d) = 3$ sensor nodes. The λ-mRBC algorithm parameter λ is set to 0.5.

6.2 Results and Analysis

6.2.1 Impact of Number of Iterations

The fitness value of the best solution is plotted in Figure 5 (a) for the RBC and λ-mRBC methods. It becomes clear that the RBC method has a lower convergence speed, compared with the proposed λ-mRBC method. Additionally, the RBC is trapped in local minima. On the other hand, the proposed λ-mRBC method converges to better fitness value by 19.1%, compared with RBC method. This is because of using the transposition operator (trans), where the positions of current best solution elements are randomly swapped. The transposition operator (trans) which is controlled by adjusting the λ parameter occurs in some selected iterations. When the elements of the current best solution are randomly swapped, more exploration in the search space occurs. Hence, the λ-mRBC method tries to escape from the trap of local minima. Consequently, better solution can be found. Figure 5 (b) and Figure 5 (c) show the make-span and total energy consumption *versus* iteration for both RBC and λ-mRBC methods. Compared with RBC method, the proposed λ-mRBC method converges to better make-span and total energy consumption by 19.6% and 22.3%, respectively. Since the fitness value of λ-mRBC method has better convergence speed and lower values, the performance of the proposed λ-mRBC method, in terms of make-span and total energy consumption, is improved, compared with the RBC method.



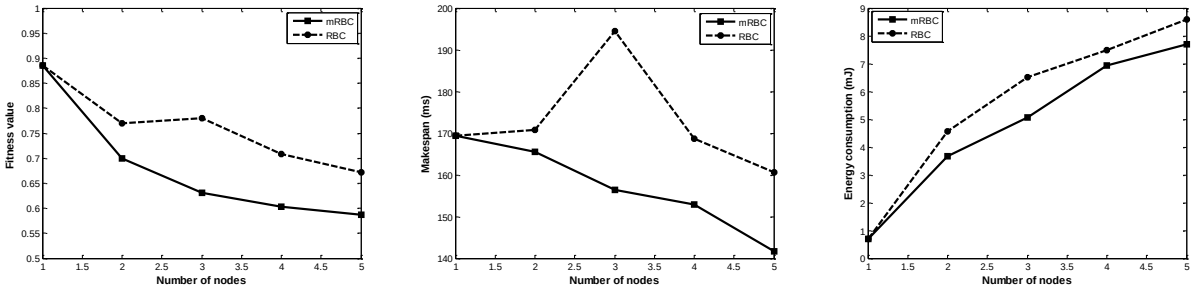
(a) Fitness value (b) Make-span (c) Energy consumption

Figure 5. The effect of iterations for the RBC and proposed λ-mRBC methods.

6.2.2 Impact of Varying Number of Sensor Nodes

This section evaluates the effects of selected sensor node size ($n_g(k, A_d)$). The proposed λ-mRBC method supports different sizes of the sensor nodes. The size of the sensor nodes is changed from 1 to 5 with one sensor node for each step. Figure 6 shows the performance of the RBC and proposed λ-mRBC methods with the sensor node size. As shown in Figure 6 (a), the fitness value is getting better whenever the size of the selected sensor nodes increases. This is because the computational load of tasks is parallelized in more powerful fashion whenever the size of the selected sensor nodes rises. However, the proposed λ-mRBC method gives lower fitness values, compared with traditional RBC method. In addition, the fitness value of RBC method at a sensor node size of 3 does not improve, compared with its value at a sensor node size of 3. This is due to the trapping in the local minima. The make-span shown in Figure 6 (b) is reduced whenever the sensor node size goes up, because the computational load is distributed to more sensor nodes. As shown in Figure 6 (c), the communication activities used to exchange the communication edges increase whenever the sensor node size rises up, because tasks can be distributed to more sensor nodes. Therefore, according to Equation (15), the total

energy consumption rises with increasing the sensor node size. Ultimately, the proposed λ-mRBC method improves the energy consumption. Additionally, compared with RBC method, the proposed λ-mRBC method has better fitness value, make-span and total energy consumption by 11.8% , 10.3% and 12.6%, respectively.

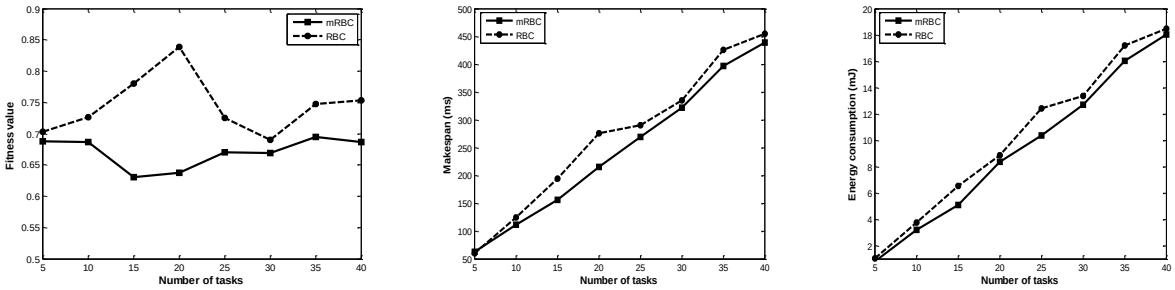


(a) Fitness value (b) Make-span (c) Energy consumption

Figure 6. The effect of node size for the RBC and proposed λ-mRBC methods.

6.2.3 Impact of Number of Tasks

The proposed λ-mRBC method supports different numbers of tasks. In Figure 7, the number of tasks varies from 5 to 40 with five tasks for each step. The fitness value, make-span and energy consumption for each step are plotted. In fact, according to Equation (14) and Equation (15), increasing the number of tasks leads to the increasing of the computational and communicational loads. Therefore, make-span and energy consumption increase with increasing the number of tasks. This is shown in Figure 7 (b) and Figure 7 (c). The aim of the objective function of Equation (17) is to reduce the energy consumption and the make-span as well. Thus, some solutions give better improvement in terms of energy consumption and other solutions give improvement in terms of make-span. Therefore, the fitness values shown in Figure 7 (a) fluctuate with increasing the number of tasks. It is worth mentioning that the proposed λ-mRBC method can cope with different numbers of tasks due to the small fluctuation of fitness values, compared with the RBC method. Besides, λ-mRBC method gives better performance in terms of make-span and energy consumption. Furthermore, compared with RBC method, the proposed λ-mRBC method has better fitness value, make-span and total energy consumption by 3.6%, 2.4% and 8.8%, respectively.



(a) Fitness value (b) Make-span (c) Energy consumption

Figure 7. The effect of varying number of tasks for the RBC and proposed λ-mRBC methods.

6.2.4 LA-SNSA Evaluation

This section evaluates the performance of LA-SNSA. The performance metrics used to evaluate the proposed LA-SNSA are the Remaining Energy Performance (*REP*) and the Neighbour Count Performance (*NCP*). *REP* is defined as the normalized sum of normalized remaining energy of all sensor nodes and *NCP* is defined as the normalized sum of the number of neighbours of all sensor nodes. Therefore, *REP* and *NCP* are calculated as follows:

$$REP(k) = \frac{\sum_{l=1}^{m}[E_r(k,s_l)/E_{r(max)}(k,s_l)]}{REP_{max}} \quad (21)$$

$$NCP(k) = \frac{\sum_{l=1}^{m}[N_c(k,s_l)]}{NCP_{max}} \quad (22)$$

Where REP_{max} and NCP_{max} are the maximum values of *REP* and *NCP* which are calculated at the beginning of the simulation. Therefore, $REP(k)$ and $NCP(k)$ values are in the range of [0, 1]. In Figure 8 and Figure 9, the proposed LA-SNSA and random selection schemes are compared by calculating these performance metrics. In random selection scheme, however, the sensor nodes ($n_g(k, A_d)$) are chosen randomly from the neighbouring target sensor node (s_{TSN}).

As shown in Figure 8, the $REP(k)$ is decreasing with time. This is due to the increasing energy consumption of communication and processing activities, which are caused by the application executions. It is observed from Figure 8 that the rate of $REP(k)$ reduction with time using the proposed LA-SNSA is smaller than those in the random selection scheme. The reason behind this is that LA-SNSA aims to select the nodes with higher remaining energy, while the random selection scheme selects the sensor nodes randomly without any knowledge of the node remaining energy. The proposed LA-SNSA enhances the $REP(k)$ about three times, compared with the random selection scheme.

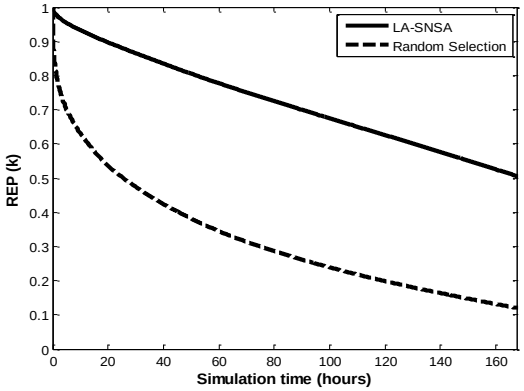


Figure 8. REP for the RBC and proposed λ-mRBC methods.

The value of $NCP(k)$ remains 1 until the first node death takes place. Thus, $REP(k)$ and number of dead nodes are calculated and plotted in Figure 9 after the death of the first node. Application executions lead to energy consumption, caused by processing and communicating. Therefore, sensor node energy level decreases. When the energy level of sensor node is exhausted, the sensor node dies and all activities stop. After the first death, $NCP(k)$ is decreased due to the increasing of death nodes. As shown in Figure 9, $NCP(k)$ is decreased sharply in case of random selection scheme, because there are no directional guides to select sensor nodes. Additionally, the rate of increased dead nodes is higher in case of random selection scheme. Another advantage of the proposed LA-SNSA is that it takes a long time for first node to die. Furthermore, compared with random selection, the $NCP(k)$ is improved by 20.1% using the proposed LA-SNSA.

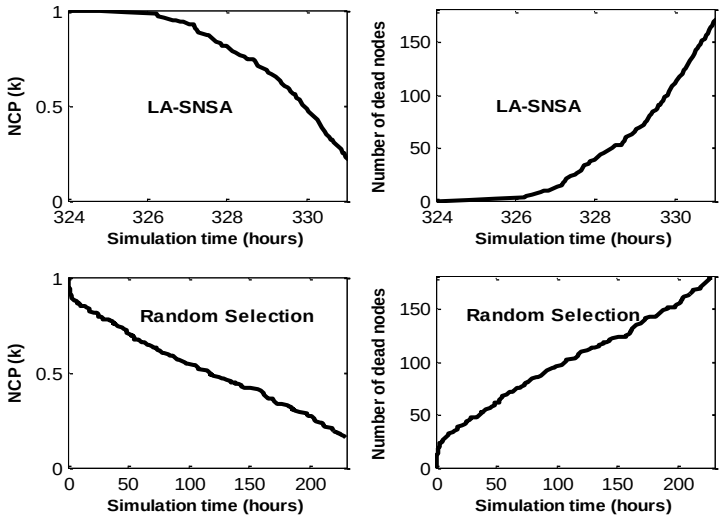


Figure 9. NCP and number of dead nodes for the RBC and proposed λ-mRBC methods.

Table 1 shows the Remaining Energy Performance (*REP*), Neighbour Count Performance (*NCP*) and first node death time for the proposed LA-SNSA and random selection schemes. The proposed LA-SNSA ends with better *REP* which implies that the remaining energy of the sensor nodes is better distributed in the network. In addition, there is an improvement in *NCP* in case of using the proposed LA-SNSA. This improvement leads to less gaps without sensor nodes in the network. The network lifetime can be defined as the time when the first node dies [44]-[45]. First node death time is bigger in case of the proposed LA-SNSA. Since *REP*, *NCP and* first node death time are improved in the proposed LA-SNSA scheme, the network lifetime is also enhanced in case of using the proposed LA-SNSA.

Table 1. REP, NCP and first node death time for random and LA-SNSA schemes.

Method	Remaining Energy Performance (*REP*)	Neighbour Count Performance (*NCP*)	First Node Death time (Hours)
LA-SNSA	0.51	0.19	324.79
Random	0.12	0.16	0.03

6.2.5 The Effect of λ-mRBC Parameter

Figure 10 shows the fitness values *versus* the iteration using λ parameter values of 0.1, 0.3, 0.5, 0.7 and 0.9. According to Equation (20), the probability of running the transposition operation is increasing with increasing the λ parameter. Therefore, the convergence speed for λ parameter of 0.1 and 0.3 is the slowest, compared with other λ parameter values. Furthermore, when using λ parameters of 0.1 and 0.3, the λ-mRBC method converges to the highest fitness value. On the other hand, when using λ parameters of 0.5 and 0.7, the λ-mRBC method converges to the lowest fitness value. The fastest convergence speed occurs when using λ parameter of 0.9. However, λ-mRBC method converges to larger fitness value than the fitness value when using λ parameters of 0.5 and 0.7. It is worth mentioning that the number of optimization algorithm parameters increases the complexity of the algorithm [17]. λ-mRBC uses only one parameter (λ) which indicates its low complexity.

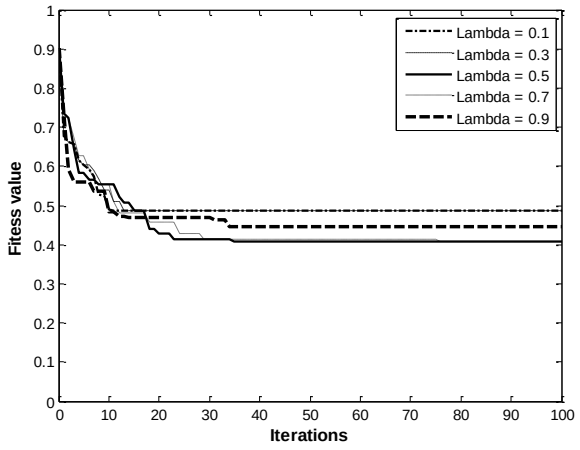


Figure 10. The effect of λ parameter.

7. CONCLUSION

In this paper, a Task Mapping and Scheduling (TMS) approach for WSN is introduced to look for the best tasks/nodes mapping solution. The proposed λ-mRBC, which is a modified version of RBC optimization method, is proposed to improve the performance of the search. To escape from local optima and to increase the exploration of the search space, the λ-mRBC method employs a new operator, which is named random transposition. The transposition operator changes the elements' positions of current best solution. The λ-mRBC method is controlled by using only one parameter (λ). Energy consumption and application execution time (make-span) are taken into consideration in the fitness objective to get the best performance of TMS. In addition, LA-SNSA is proposed to select a

number of sensor nodes needed to execute the applications, so that the network lifetime is improved. The simulation results show that the proposed λ-mRBC method improves the energy consumption, make-span and fitness value, compared with traditional RBC method. Furthermore, using LA-SNSA enhances the network lifetime, compared with random selection approaches. Although the proposed λ-mRBC uses a new operator called transposition operator to escape from local optima, it is still a single-solution metaheuristic. Unlike population-based metaheuristics, the proposed λ-mRBC is of less exploration of search space. The future work aims to add a new operator that employs more than one solution to increase the exploration of search space.

REFERENCES

[1] B. Sharma and T. C. Aseri, "A Comparative Analysis of Reliable and Congestion-aware Transport Layer Protocols for Wireless Sensor Networks," International Scholarly Research Network (ISRN), Sensor Networks, 2012.

[2] M. Katiyar, H. P. Sinha and D. Gupta, "On Reliability Modeling in Wireless Sensor Networks-A Review," IJCSI International Journal of Computer Science, vol. 9, no. 6, pp. 134–146, 2012.

[3] H. Yetgin, K. T. K. Cheung, M. El-Hajjar and L.H. Hanzo, "A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Networks," IEEE Communication Surveys & Tutorials, vol. 19, no. 2, pp. 828-854, 2017.

[4] P. R. Pereira, A. Grilo, F. Rocha, M. S. Nunes, A. Casaca, C. Chaudet, P. Almstrom and M. Johansson, "End to End Reliability in Wireless Sensor Networks: Survey and Research Challenges," Proceedings of the EuroFGI Workshop on IP QoS and Traffic Control, 2007.

[5] M. A. Kafi, J. B. Othman, M. Bagaa and N. Badache, "CCS_WHMS: A Congestion Control Scheme for Wearable Health Management System," Journal of Medical Systems, vol. 39, no. 12, 2015.

[6] Y. E. Hamouda and M. M. Msallam, "Smart Heterogeneous Precision Agriculture Using Wireless Sensor Network Based on Extended Kalman Filter," Neural Computing and Applications, pp.1-17, 2018.

[7] P. R. C. Araújo, R. H. Filho, J. J. Rodrigues, J. P. Oliveira and S. A. Braga, "Middleware for Integration of Legacy Electrical Equipment into Smart Grid Infrastructure Using Wireless Sensor Networks," Inter.

Journal of Communication Systems, vol. 31, no. 1, pp. e3380, 2018.

[8] B. L. R. Stojkoska and K. V. Trivodaliev, "A Review of Internet of Things for Smart Home: Challenges and Solutions," Journal of Cleaner Production, vol. 140, no. 3, pp.1454-1464, 2017.

[9] Y. E. Hamouda and C. Phillips, "Adaptive Sampling for Energy-efficient Collaborative Multi-Target Tracking in Wireless Sensor Networks," IET Wireless Sensor Systems, vol. 1, no. 1, pp.15-25, 2011.

[10] J. Luo and S. Zou, "Strong k-barrier Coverage for One-way Intruders Detection in Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 12, no. 6, 2016.

[11] J. Manikannu and V. Nagarajan, "A Survey of Energy Efficient Routing and Optimization Techniques in Wireless Sensor Networks," IEEE International Conference on Communication and Signal Processing (ICCSP), 2018.

[12] M. Wolf, Smart Camera Design, Springer, 2018.

[13] M. Karakaya and H. Qi, "Coverage Estimation in Heterogeneous Visual Sensor Networks," Proceedings of the 8th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 41-49, 2012.

[14] C. A. Navarro, N. Hitschfeld-Kahler and L. Mateu, "A Survey on Parallel Computing and Its Applications in Data-parallel Problems Using GPU Architectures," Communications in Computational Physics, vol. 15, no. 2, pp. 285-329, 2014.

[15] L. Dai, H. Xu, T. Chen, Q. Chao and L. Xie, "A Multi-Objective Optimization Algorithm of Task Scheduling in WSN," International Journal of Computers, Communications & Control, vol. 9, no. 2, pp. 160-171, 2014.

[16] Y. Yang, X. Qiu, L. Meng and K. Long, "Task Coalition Formation and Self-adjustment in the Wireless Sensor Networks,". Int J. Commun. Syst., vol. 27, no. 10, pp. 2241–2254, 2014.

[17] C. Blum and A. Roli,"Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," ACM Computing Surveys (CSUR), vol. 35, no. 3, pp. 268-308, 2003.

[18] I. Boussaïd, J. Lepagnot and P. Siarry, "A Survey on Optimization Metaheuristics," Information Sciences, vol. 237, pp. 82-117, 2013.

[19] Y. Jin, J. Jin, A. Gluhak, K. Moessner and M. Palaniswami, "An Intelligent Task Allocation Scheme for Multihop Wireless Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 3, pp. 444-451, 2012.

[20] R. Shams and F. Khan, "Solving Wireless Network Scheduling Problem by Genetic Algorithm," IAMURE International Journal of Mathematics Engineering & Technology, vol. 2, no. 11, pp. 63-70, 2012.

[21] J. Yang, H. Zhang, Y. Ling, C. Pan and W. Sun, "Task Allocation for Wireless Sensor Network Using Modified Binary Particle Swarm Optimization," IEEE Sensors Journal, vol. 14, no. 13, pp. 882-892, 2014.

[22] A. A. Ferjani, N. Liouane and I. Kacem, "Task Allocation for Wireless Sensor Network Using Logic Gate-based Evolutionary Algorithm," International Conference on Control, Decision and Information Technologies (CoDIT), pp. 654-658, 2016.

[23] V. Papataxiarhis, "Optimal Task Assignment in Sensor Networks," Proc. of the 17th IEEE International Conference on Mobile Data Management (MDM), pp. 26-31, 2016.

[24] S. Abdelhak, C. S. Gurram, S. Ghosh and M. Bayoumi, "Energy-balancing Task allocation on Wireless Sensor Networks for Extending the Lifetime," Proceedings in the 53rd IEEE Int. MWSCAS, pp. 781-784, 2010.

[25] X. Yin, W. Dai, B. Li, L. Chang and C. Li, "Cooperative Task Allocation in Heterogeneous Wireless Sensor Networks", Inter. Journal of Distributed Sensor Networks, vol. 13, no. 10, pp. 1-12, 2017.

[26] D. R. Bolla, J. J. Jijesh and M. S. Pramod, "Real-Time Data Fusion Applications in Embedded Sensor Network Using TATAS," Indian Journal of Science and Technology, vol. 10, no. 13, pp. 1-7, 2017.

[27] Y. Tian and E. Ekici, "Cross-Layer Collaborative in Network Processing in Multihop Wireless Sensor Networks," IEEE Trans. Mobile Comput., vol. 6, no. 3, pp. 297-310, 2007.

[28] Y. Tian, B. Jarupan, E. Ekici and F. Ozguner, "Real-Time Task Mapping and Scheduling for Collaborative in Network Processing in DVS-Enabled Wireless Sensor Networks," Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006), pp. 1-10, 2006.

[29] Y. E. M. Hamouda and C. Phillips, "Biological Task Mapping and Scheduling in Wireless Sensor Network," Proc. of the IEEE International Conference on Communications Technology and Applications, pp. 914 – 919, 2009.

[30] Y. E. M. Hamouda, "Light Allocation of Tasks in Clustered-based Wireless Sensor Networks", Al-Aqsa University Journal (Natural Sciences Series), vol. 21, pp. 90-119, 2017.

[31] K. N. Devi and R. Muthuselvi, "Parallel Processing of IoT Health Care Applications," Proc. of the 10th IEEE International Conference on Intelligent Systems and Control (ISCO), pp. 1-6, 2016

[32] J. Jiang, G. Han and C. Zhu, "A Complicated Task Solution Scheme Based on Node Cooperation for Wireless Sensor Networks," Proc. of the 22nd IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 264-269, 2016.

[33] P. Skocir, M. Kusek and G. Jezic, "Energy-efficient Task Allocation for Service Provisioning in Machine-to-Machine Systems," Concurrency and Computation: Practice and Experience, vol. 29, no. 23, pp. e4269, 2017.

[34] X. Yin, K. Zhang, B. Li, A. K. Sangaiah and J. Wang, "A Task Allocation Strategy for Complex Applications in Heterogeneous Cluster–based Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 14, no. 8, 2018.

[35] E. A. Khalil, S. Ozdemir and S. Tosun, "Evolutionary Task Allocation in Internet of Things-based Application Domains," Future Generation Computer Systems, vol. 86, pp.121-133, 2018.

[36] W. Yu, Y. Huang, E. Ding and A. Garcia-Ortiz, "Joint Task Allocation Approaches for Hierarchical Wireless Sensor Networks," Proc. of the IEEE 7th International Conference on Modern Circuits and Systems Technologies (MOCAST), pp. 1-4, 2018.

[37] N. Bulusu, J. Heidemann and D. Estrin, "GPS-less Low-Cost Outdoor Localization for Very Small Devices," IEEE Personal Communications, vol. 7, no. 5, pp. 28-34, 2000.

[38] T. C. Karalar, S. Yamashita, M. Sheets and J. Rabaey, "A Low-Power Localization Architecture and

System for Wireless Sensor Networks," IEEE Workshop on Signal Processing Systems, USA: Signal Processing Society, pp. 89-94, 2004.

[39] O. Sinnen, Task Scheduling for Parallel Systems, New Jersey: John Wiley & Sons, Inc., Hoboken, 2007.

[40] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient Communication Protocol for Wireless Microsensor Networks," Proceedings of the IEEE 33rd Annual Hawaii International Conference on System Sciences (HICSS '00), pp. 1-10, 2000.

[41] A. Wang and A. Chandrakasan, "Energy-efficient DSPs for Wireless Sensor Networks," IEEE Trans. Signal Process. Mag., pp. 68-78, 2002.

[42] L. Davis, "Bit-Climbing, Representational Bias and Test Suite Design," Proc. of the 4th International Conference on Genetic Algorithms, pp. 18-23, 1991.

[43] H. Aguirre and K. Tanaka, "Random Bit Climbers on Multiobjective MNK-Landscapes: Effects of 49 and Population Climbing," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 88, pp. 334-345, 2005.

[44] M. A. Abd, S. F. M. Al-Rubeaai, B. K. Singh, K. E. Tepe and R. Benlamri, "Extending Wireless Sensor Network Lifetime with Global Energy Balance," IEEE Sensors Journal, vol. 15, no. 9, pp. 5053–5063, 2015.

[45] I. Dietrich and F. Dressler, "On the Lifetime of Wireless Sensor Networks," ACM Trans. Sen. Netw., vol. 5, no. 1, 2009.

ملخص البحث:

تعالج هذه الورقة مشكلة تخطيط المهام وجدولتها في شبكات المجسّات اللاسلكية. هذا التطبيق المفترض أن ينفَّذ في شبكات المجسّات اللاسلكية يمكن تقسيمه الى مهام يعتمد بعضها على بعض. وتتمثل الأهداف الأساسية لتخطيط المهام وجدولتها في تحسين زمن التنفيذ واستهلاك الطاقة وعُمر الشبكة. تم تطوير طريقة معدَّلة تقوم على التسلق العشوائي من أجل الحصول على حلول أفضل وأسرع تكون مثالية أو قريبة من المثالية.

في الطريقة المعدّلة المقترحة، يُضاف عامل جديد يسمى عامل الإزاحة من أجل تحسين استكشاف فضاء البحث ومن ثم الهروب من القيم المثلى المحلية. ويتم التحكم بعمق الاستكشاف باستخدام متغير واحد (λ). في البدء، يتم اختيار عدد من عُقد المجسّات من أجل التنفيذ التعاوني للتطبيق بهدف تحسين عُمر الشبكة. بعد ذلك، يتم تنفيذ الطريقة المعدّلة المقترحة للحصول على الحلّ الأمثل أو القريب من الأمثل فيما يتعلق بأزواج المهامّ / المجسّات، بحيث يتم التقليل الى الحد الأدنى الممكن من زمن التنفيذ واستهلاك الطاقة.

وقد بينت نتائج المحاكاة أن الطريقة المعدّلة المقترحة تُحسّن أداء تخطيط المهام وجدولتها. ومقارنة بالطريقة التقليدية، فإن الطريقة المقترحة تؤدي الى تحسُّن قيمة الملاءمة وفترة الفِعل والاستهلاك الكلي للطاقة بنسبٍ بلغت 19.1% و 19.6% و22.3% على الترتيب. من جانب آخر، تمت إطالة عُمر الشبكة عبر استخدام خوارزمية الاختيار المقترحة. وقد تحسّن توزيع الطاقة المتبقية بين عُقد المجسّات بمقدار 3 مرات تقريباً مقارنة بطريقة الاختيار العشوائي. علاوة على ذلك، ومقارنة بالاختيار العشوائي، فقد تحسّن عدد الجيران لعُقد المجسّات بنسبة 20.1% باستخدام خوارزمية الاختيار المقترحة.