ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2011, Vol. 40, No. 4

INITIAL TASK PLACEMENT FOR MACRO-PROGRAMMED WIRELESS SENSOR NETWORK

Vitalijus Martusevičius[1], Asta Martusevičienė[2], Egidijus Kazanavičius[1], Vaidas Jukavičius[1]

[1] *Department of Computer Engineering, Kaunas University of Technology*
Studentu St. 50-208, LT-51368 Kaunas, Lithuania
email:vitalijus.martusevicius@stud.ktu.lt

[2] *Department of Applied Mathematics, Kaunas University of Technology*
Studentu St. 50-324a, LT-51368 Kaunas, Lithuania

crossref http://dx.doi.org/10.5755/j01.itc.40.4.984

Abstract. One of the most important issues in wireless sensor network programming is to assign a set of tasks to a set of nodes with limited energy resources in order to minimize energy consumption. In this paper, we develop a task assignment model where cost function is formulated as a sum of computation and communication energy spent in the network. The model employs task placement constraints that ensure expected lifetime of individual node. We propose an efficient genetic algorithm with repair operator to obtain a minimal cost solution. The repair operator guarantees valid task assignments that meet model constraints as candidate solutions are generated during the process of evolution. Experiments reveal that provided heuristic takes a reasonable amount of time to produce near optimal results.

Keywords: wireless sensor network; task mapping; genetic algorithm; macro-programming.

1. Introduction

Energy is a critical performance metric of a wireless sensor network (WSN). In order to extend the lifetime of the WSN, energy efficient task placement on a given deployment of sensor nodes is required. Initial task placement utilizes global knowledge available at compile time with the aim to minimize energy consumption for application execution in the network. As WSN is a dynamic environment, changing energy levels and positions of nodes are further handled by task migration subsystem [21]. These techniques enable effective WSN application management as depicted in Figure 1.

The assignment of m tasks onto n nodes is a classical NP-complete problem, usually solved in parallel computing [1], [10]. Yet some aspects are specific for WSN: strict task placement constraints, communication of tasks between each other and the fact that nodes involved in routing also spend energy in the process.

Several approaches to task assignment problem have been identified in literature. Graph theoretical algorithms [16], [19] or mathematical programming [12], [26] require high time complexity and do not allow constrains to be easily incorporated into the model. These algorithms obtain an optimal solution. Otherwise, heuristic-based techniques like genetic algorithm [7], [15] or simulated annealing [14] are applicable to larger dimensional problems due to their low time complexity. However, these algorithms provide suboptimal solution. As well as aforementioned exact methods, most of genetic algorithms do not take into account constraints to meet various application requirements.

A number of researchers have already considered task placement in WSNs. Tian *et al.* [24] deal with task mapping problem from a protocol-centric point of view. Their aim is to map and schedule application tasks with minimum schedule length subject to energy consumption constrains. Here network assumptions are a bit different as sensors are grouped into single-hop clusters and attention is paid more on scheduling but not mapping of tasks. As opposite, we consider a multi-hop homogenous WSN. In [23], Hamouda and Phillips propose biological task mapping and scheduling algorithm claiming it outperforms an algorithm in [25]. Task allocation in these papers takes into account network dynamics during the lifetime of WSN whereas we deal only with initial task placement. Park *et al*. [14] use simulated annealing method to solve task transformation and assignment problem. In their case, tasks to be deployed are decomposable and transformable while we consider atomic tasks. In [5], energy consumption is addressed through energy-balanced task allocation that is most similar to our case.

However, Pathak and Prasanna only consider overall energy expenditure in the network whereas we also take into account energy dissipation of individual nodes. Besides, network communication and energy consumption mechanisms are more accurate in our simulations. We also propose an advanced adaptive genetic algorithm to solve task assignment problem.

Our goal is to near optimally assign tasks to the WSN nodes in terms of energy spent in the network and apply the given constraints. We minimize a classical performance metric of total energy spent in the system in order to find the best task placement. This optimization goal consists of computation and communication energy costs. We use genetic algorithm to explore the solution space efficiently. In order to comply with the given constraints, our constructed genetic algorithm employs a special repair operator to guarantee valid assignments during the process of evolution. Actual WSN application of greenhouse management is chosen to demonstrate the effectiveness of our solution to the task assignment problem.

The remainder of this paper is structured as follows. Initial task assignment onto the target network is introduced in detail in Section 2. Next, the case study is provided. In Section 4, we conclude and present directions of future work.

2. Energy Efficient Task Placement

2.1. Application Model

The task mapping problem emerges while compiling data-driven macro-programs for WSNs. As a case study for the task assignment problem, we consider a monitoring system to prevent dew condensation in a greenhouse environment [6], [11]. Dew condensation on the plants surface can promote various crop diseases. Our system is composed of three types of sensor nodes that collect ambient temperature and air humidity as well as plants surface temperature. Also, actuators that control ventilating fans are deployed for adjusting the environment inside the greenhouse.

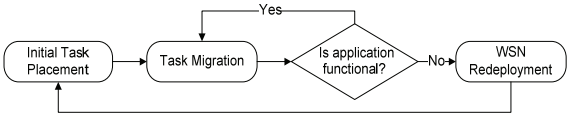

Figure 1. Application management in WSN

We design the application as a directed acyclic graph (DAG) that describes tasks dependencies (Fig. 2). The tasks of the DAG are to be placed onto the underlying target network. Each sensor node has an associated ambient temperature (T_1, T_2) or plant surface temperature (T_5, T_6) or humidity (T_3, T_4) sampling task. Additional five tasks T_7–T_{11} that control application functionality are to be placed onto any of the nodes of the WSN. These are the tasks for optimization. Here T_7, T_8, and T_9 collect and average ambient temperature, air humidity and plants surface temperature respectively, while T_{10} calculates the dew point and T_{11} compares it with the averaged plants surface temperature.

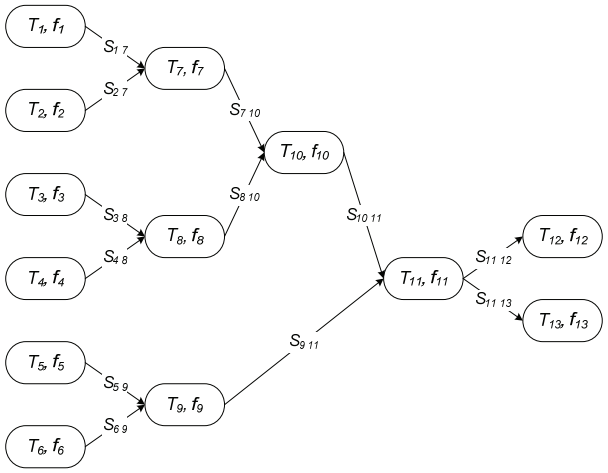

Figure 2. Task graph of the greenhouse management application

The temperature of the dew point is calculated by Barenbrug formula [11] using averaged values of ambient temperature and air humidity. If the dew point temperature of the greenhouse is higher than the averaged plants surface temperature, the environment is adjusted by actuating tasks (T_{12}, T_{13}) that control ventilating fans.

It is important to highlight that the assignment of tasks onto the target nodes should achieve the following goals:

- minimize total energy spent in the entire system,
- evaluate task placement constraints.

According to these goals, we create a mathematical model and formulate a cost function as described in the next section.

2.2. Problem Formalization

The concept of the model is based on the results presented in [5], where the framework of mathematical description consists of four matrices that define task assignment problem. We complete these matrices with respect to actual WSN operation, whereas Pathak and Prasanna [5] provide inaccurate assumptions to perform simulations. Thus, in our case, elements of the model are supplemented with techniques that evaluate platform-dependent computation and communication energy consumption.

A target network $\mathbf{TN} = \{1,\ldots,k,\ldots n\}$ consists of n nodes with initial properties that depend on the platform in use. We explore a homogenous sensor network, where all nodes have the same initial energy level e_k.

A task graph $\mathbf{D} = (\mathbf{DT}, \mathbf{DE})$ consists of a set of vertices $\mathbf{DT} = \{1,\ldots,i,\ldots m\}$ representing the tasks to be executed and a set of directed edges

$\mathbf{DE} \subseteq \mathbf{DT} \times \mathbf{DT}$ depicting communication dependencies among tasks. Each task i has a firing rate f_i depending on the nature of the task. It indicates the number of times the task is invoked in a period of system behavior. Each edge has an associated size of data s_{ij} that task i provides to task j per invocation.

We construct task execution energy matrix $\mathbf{T}_{m\times n}$ to present energy consumption of each task execution on every node and comply with task placement constraints. Here entry T_{ik} of the matrix $\mathbf{T}$ defines the energy spent by node k per invocation of task i.

Further, we create a routing energy cost matrix $\mathbf{R}_{n\times n\times n}$ for target network, where each entry $R_{\alpha\beta k}$ depicts the energy required at node k to transmit a unit of data from node α to node β. We used Dijkstra shortest path algorithm to fill the routing matrix. The construction of routing matrix $\mathbf{R}$ is elaborated in Section 3.

The task mapping is a surjective function *map*: $\mathbf{DT} \rightarrow \mathbf{TN}$, assigning task i to node *map(i)*.

After model elements are defined, computation and communication energy costs at a node in a period of system behavior are evaluated as follows:

$$C_{\text{comp}}^{k} = \sum_{i:map(i)=k} f_i \cdot T_{ik}, \quad k \in 1,\ldots,n, \tag{1}$$

$$C_{\text{comm}}^{k} = \sum_{(i,j)\in\mathbf{DE}} f_i \cdot s_{ij} \cdot R_{map(i)map(j)k}, \quad k \in 1,\ldots,n. \tag{2}$$

Finally, the total energy spent in the WSN is expressed as the sum of energy consumptions of individual nodes:

$$C_{\text{total}} = \sum_{k\in\mathbf{TN}} \left(C_{\text{comp}}^{k} + C_{\text{comm}}^{k} \right), \tag{3}$$

$$C_{\text{comp}}^{k} \le \lambda \frac{e_k}{\text{t}}, \quad k \in 1,\ldots,n\,. \tag{4}$$

The system level metric (3) is to be minimized in order to find an optimal mapping of tasks in terms of energy consumption. This is a classical goal function estimating the performance of distributed systems. The constraint (4) puts a limit on computation energy for the tasks on node k in a period of system behavior. It ensures that tasks are spread over the network avoiding high workload on a few nodes. In (4), λ is the computation energy component and t denotes the expected lifetime of a node expressed in periods of system behavior. For performance metric (3) a feasible solution is possible only when all nodes have enough energy left at the end of period of system behavior (4), otherwise the task mapping algorithm should report failure.

The time required to minimize the cost function (3) increases exponentially as the size of the problem $n^{\text{m}_{\text{opt}}}$ grows. Here m_{opt} is the number of tasks to be optimized. Thus, various heuristic and approximation methods are used to speed up the process of finding a good enough solution, where an exhaustive search is impractical. In [5], the problem is formulated as a mixed integer programming (MIP) problem and also special greedy algorithms are provided to solve it with computational complexity $O(n^3|\mathbf{DE}|^2)$. On the other hand, we propose a well established genetic-based approach to find a solution that minimizes a given cost function. We adapt general genetic algorithm for task assignment in WSN using repair operator. Also, the parameters of genetic algorithm are tuned. Introduced heuristic is adjustable to complex applications and effective for large networks.

2.3. Genetic Algorithm for Task Mapping

A genetic algorithm (GA) is a heuristic [3] that mimics natural biological evolution. GAs operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. This heuristic is usually used to generate useful solutions to optimization and search problems. Closest to our case, it has been applied to assign tasks for multiprocessors with limited memory in parallel computing [1]. Other applications were also considered [2], [13], [17], [18]. There are several basic points that all genetic algorithms are based on.

Firstly, a genetic representation of the solution domain should be created. This means that a solution to a problem should be represented as a genome (chromosome). We assume that setting the i-th cell, called gene, of the chromosome array to j means assigning of the i-th task to node j (Figure 3). A chromosome is called valid if tasks are allocated to nodes in such a way that each node has enough energy to execute assigned tasks. The total cost for a given assignment is the total energy spent while executing all tasks assigned to the nodes of the target network.

Task i	1	2	3	4	5
Node j	3	18	15	3	9

Figure 3. Chromosome representation of task assignment

Further, we have to define a fitness function to evaluate the potential solution. Fitness function is the key element in the whole genetic algorithm and depends on the problem. It measures the quality of the chromosome and correlates closely with the objective function. Also, computation time of fitness value of a chromosome should not be high. Hence, in our case, the fitness function *fit* is defined as follows:

$$fit = \frac{1}{\sum_{k\in\mathbf{TN}} \left(C_{\text{comp}}^{k} + C_{\text{comm}}^{k} \right)}. \tag{5}$$

Eventually, genetic operators should be defined and implemented. Crossover operator generates child chromosomes from two parent chromosomes by combining the information extracted from the parents. In our genetic algorithm we use one-point crossover. Mutation operator is used to explore the whole search

space and to maintain diversity. Our mutation operator involves a probability that a randomly selected gene in a genetic sequence will be changed from its original state.

After crossover and mutation operators are applied, it is possible some child chromosomes to be invalid due to node capacity restrictions. Thus we deliver an additional repair operator that converts the invalid chromosomes to valid ones (Fig. 4). The input to the repair procedure consists of a chromosome **chrom**, which defines the mapping of tasks onto the nodes, as well as **Enode** and **Etask** arrays, which represent the energy levels of nodes and tasks, respecttively. It is assumed that a proportion of node energy is reserved for computation as WSNs deplete most of energy for communication. Dividing this energy by a number of periods of system behavior we get an amount of energy that can be consumed in one period to achieve expected lifetime of a node. Thus, each cell of the **Enode** array represents the available energy of particular node and its value is obtained by subtracting energy required for predetermined tasks from the energy that can be consumed in one period. Further, **Etask** array represents the tasks under optimization, where the value in each cell defines computation costs for particular task in a period of system behavior. Here, energy consumption for each task is calculated using formula (6) which is multiplied by firing rate f_i in order to obtain computation costs in a period. Additional parameters, i.e. stringlength, that represents the number of optimization tasks, and optg_p, that is required to compute the value of the objective function for a given task assignment, are also needed as inputs of the repair operator. The repair operator verifies the validity of a chromosome and, if required, corrects it according to constraints. Finally, a valid chromosome **rchrom** is provided as an output.

```
Input: chrom, Enode, Etask,
stringlength, optg_p
Output: rchrom // repaired chromosome
for i=1 to stringlength do
  y = chrom(i)
  if Enode[y]-Etask[i] < 0
   valid_nodes = []
   index = 0
   for k=1 to n do
     if Enode[k]-Etask[i] >= 0
      index=index+1
      valid_nodes(index)=k
     end if
   end for
// target node for a task is selected
// randomly
   tnode = valid_nodes(randi([1,index]))
   Enode(tnode)= Enode(tnode)-Etask(i)
   rchrom(i) = tnode
  else
   Enode(y) = Enode(y) - Etask(i)
   rchrom(i) = chrom(i)
  end if
end for
// fitness of a new individual is
computed
// in the last element of an array
rchrom(stringlength+1)=1/C_total(optg_p,
rchrom(1:stringlength))
return rchrom
```

Figure 4. Method of the repair operator. Function *C_total* calculates the value of the objective function for a chromosome

Once we have the genetic representation of the solution, the fitness function and genetic operators defined, GA proceeds to initialize a population of valid solutions randomly and then improves it through repetitive application of crossover, mutation, repair and selection operators. There are many ways to select chromosomes to the next generation like roulette-wheel selection, stochastic universal sampling or tournament selection. We executed various schemes of selection to improve the performance of GA and found the most effective one as follows:

- Firstly, 10% of elite chromosomes with large values of the fitness function are moved to the next generation without any change.
- The crossover operator with the probability pc=0.7 is applied to the next 70% of chromosomes.
- The mutation operator is applied to the rest of chromosomes with small values of the fitness function. It is a mandatory mutation, so mutation probability *pm* is set to 1.
- Finally, the repair operator corrects invalid child chromosomes.

Evolutional process of genetic algorithm is repeated until a termination condition is reached. We stop the evolution when the fitness of generation does not significantly increase over time (Figure 8). The general structure of genetic algorithm for task mapping is depicted in Figure 5.

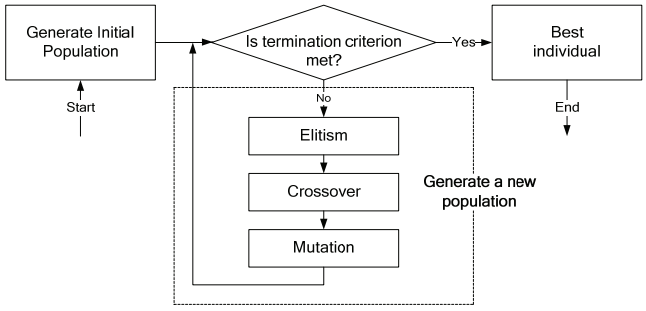

Figure 5. Procedure of genetic algorithm

Setting of parameters and type of selection operator are very important in GAs as these determine the convergence rate of the algorithm. Experiments were performed to find accurate settings that are presented in the next section.

3. Results

As a case study of task assignment, we adopted a greenhouse management application discussed in Section 2.1. Firstly, a generalized task graph, which describes the application tasks and their dependencies, is produced (Figure 2). It consists of ambient temperature, ambient humidity and plants temperature sampler tasks as well as actuator tasks. The number of sampler and actuator tasks depends on the number of nodes with appropriate capabilities. Besides, five application management tasks, which process the data from samplers and control actuators, are part of the task graph. These tasks can be assigned to any node in the target network, thus allowing optimizations. Also, each task in the task graph represents an indivisible unit of processing as opposed to task partitioning approach [14].

Secondly, we simulated various WSN topologies where tasks of DAG are to be deployed. The number of nodes varied from 10 to 50 and they were randomly distributed in the area of $100m^2$. We assume line of sight (LOS) communication between the nodes within the same coverage area. Two nodes are in the same coverage area if the distance between them is equal to or less than the radio range, which is set to 30m. An example of topology is depicted in Figure 6.

Since it is important to use realistic values during the simulation, the proposed application was implemented to run on Sun SPOT nodes [20]. These nodes are equipped with 400MHz 32bit ARM926EJ-S processor, 1M SRAM, 8M Flash memory and 2.4GHz IEEE 802.15.4 radio with integrated antenna. The Sun SPOT has lithium-ion rechargeable battery with 770mAh capacity and a normal voltage output of 3.7V. Thus, each node has its initial energy e_k equal to 10256J.

In order to estimate computation costs in the network, we firstly defined task execution energy matrix **T** and selected firing rates f_i. The task execution energy matrix **T** is an $m \times n$ matrix that represents computational energy required for tasks and ensures that tasks which perform particular sensing and actuating would be tied up to nodes with the relevant capabilities. Thus, each entry T_{ik} of the matrix is either computed using formula (6) and depicts the energy spent by node k per invocation of task i or is set to ∞ if it is impossible for node k to execute task i. We explore homogenous network, where energy consumption for task execution is the same on each node it is invoked on. In order to accurately estimate task execution cost, each task is expressed in CPU cycles [9]. Then the energy consumption of executing N_i clock cycles with CPU clock frequency f_{cpu} and core supply voltage V_{dd} is evaluated as follows:

$$T_{ik}\left(V_{dd}, f_{cpu}\right) = N_i C V_{dd}^2 + V_{dd} I_{leak} \frac{N_i}{f_{cpu}}, \qquad (6)$$

where C and I_{leak} are processor dependent parameters [8]. It should be noted that this energy consumption model only considers the energy expenditure directly related with application executions. Thus energy dissipation during idle time is not taken into account. The parameters of Sun SPOT processor ARM926EJ-S in formula (6) are set as follows: V_{dd}=1V, C=0.52nF, I_{leak}=12mA, and f_{cpu}=400MHz. Further, we selected higher firing rates f_i for sensing tasks compared to actuating and controlling tasks in order to comply with data-driven approach.

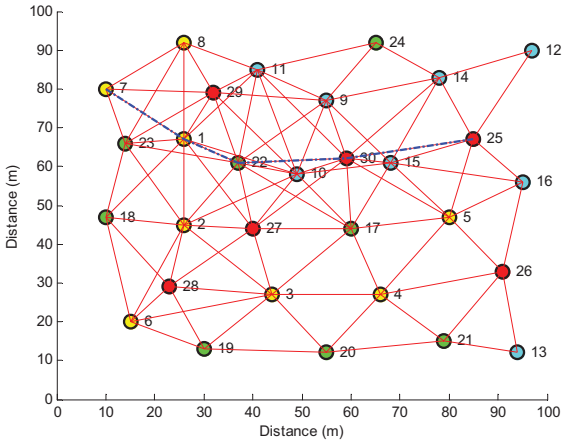

Figure 6. Network topology with three types of sensors and one type of actuators. The shortest route is depicted between Node7 and Node25

Network communication costs are evaluated by formula (2), where routing matrix **R** and data size in packets s_{ij} are to be provided. Routing matrix entry $R_{\alpha\beta k}$ is obtained as follows: the shortest path between nodes α and β in the target network is found using Dijkstra algorithm and the energy consumption per data packet at intermediate node k, that belongs to this route, is computed via radio model [8], [22]:

$$E_{tx}(l,d) = E_{tx\_elec} \cdot l + \varepsilon_{amp} \cdot l \cdot d^2, \quad d \le d_0, \qquad (7)$$

$$E_{rx}(l) = E_{rx\_elec} \cdot l, \qquad (8)$$

where E_{tx_elec}, E_{rx_elec} and ε_{amp} are node hardware property parameters. Formula (7) defines that power consumption on a sensor node to transmit l-bit data packet depends on transmission distance d and the energy dissipated by transmit electronics and power amplifier. Likewise, formula (8) states that energy required to receive l-bit data packet depends only on the energy dissipated to run the receive electronics. The appropriate parameter values for Chipcon CC2420 transceiver used in Sun SPOT nodes were obtained from the datasheet [20]: E_{tx_elec}=130nJ/bit, E_{rx_elec}=144nJ/bit, ε_{amp}=100pJ/bit/m^2, d_0=30m. It should be mentioned that if a particular node k in WSN does not belong to the shortest route from node α to node β, then its energy consumption value $R_{\alpha\beta k}$ equals to 0 in the routing matrix **R**. Also, s_{ij} denotes the number of packets per data item that task i produces to task j per invocation. In our simulation, we assume s_{ij}=1 for all data items.

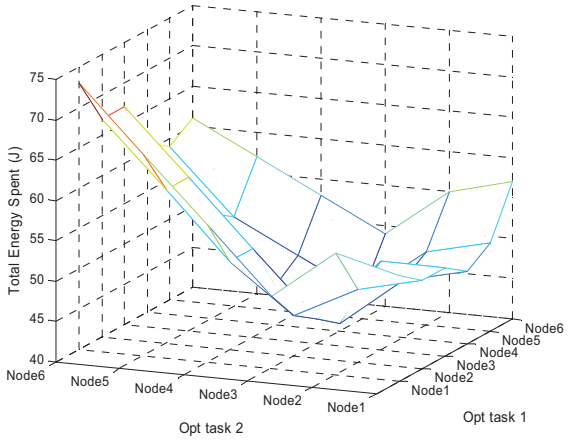

Figure 7. Task mapping in terms of energy consumption

After all elements required for computation and communication costs are defined, the total energy spent in the entire network C_{total} is computed using formula (3). It takes 0.05s to find an optimal mapping for the network with 6 nodes and 2 tasks for optimization, as depicted in Figure 7, while the total energy spent in the network is 41J.

As the number of optimization tasks and WSN nodes increases, computations grow exponentially and it is impossible to solve the problem within a reasonable amount of time. Therefore, we introduced a low time complexity genetic algorithm with repair operator which leads to near optimal results. A number of experiments were performed to identify suitable GA parameters presented in Table 1.

Table 1. Values of GA parameters

Parameter	Value
Maximum number of generations	200
Population size	100
Number of genes in a chromosome	5
Crossover probability	0.7
Mutation probability	1.0

We select maximum number of generations as stopping criterion according to the analysis of the convergence rate of fitness over populations as presented in Figure 8.

In order to evaluate the efficiency of the proposed heuristic, we performed optimization of five tasks (Figure 2) on various network topologies with increasing number of nodes. Time taken for task assignment using exhaustive search and genetic algorithm is exhibited in Figure 9. It demonstrates the effectiveness of genetic algorithm, particularly for complex applications to be mapped onto large WSNs.

In our experiments the solution C_{total} produced by the genetic algorithm was the same as the one provided by exhaustive search for deployments with 10 and 20 nodes and introduced only 2.5% error for the network with 50 nodes (Figure 10). Figures 9 and 10 show the time taken and the energy spent averaged over 100 runs of genetic algorithm.

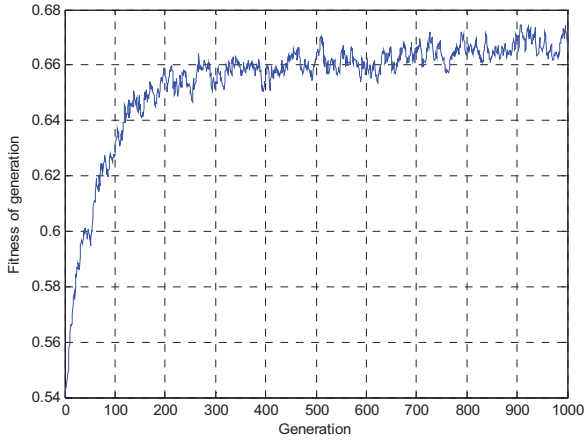

Figure 8. Performance of GA

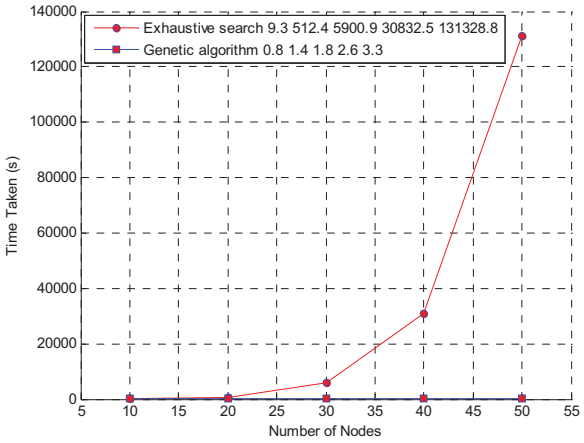

Figure 9. Time of task mapping

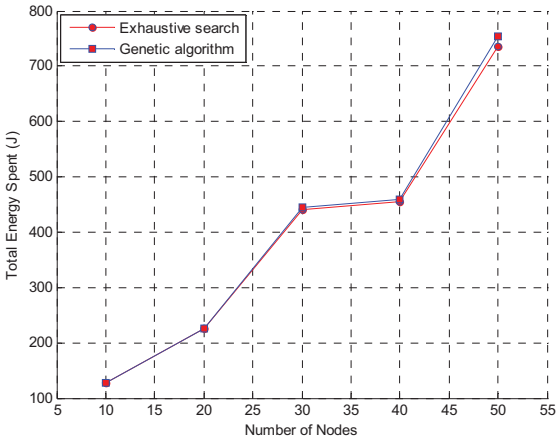

Figure 10. Total energy spent in the WSN

Implemented optimization technique enables to save up to 52% of energy for the network depicted in Figure 6. Therefore, network lifetime is extended to 46.13 hours. Our experiments indicate that the proposed genetic algorithm with repair operator is fast and accurate for actual WSN applications and therefore is an important contribution to WSN macro-programming.

4. Conclusions

In this paper, an approach for optimal initial task mapping that improves system lifetime was presented. Firstly, mathematical model for tasks assignment onto target network nodes was developed. Model constraints prevent the assignment of optimization tasks to the same node in order to extend the longevity of individual node with the aim to evenly distribute the energy depletion in the WSN. Network performance was evaluated using energy balance as optimization goal. To minimize this goal a genetic algorithm with repair operator was implemented. The repair operator is required to comply with model constraints.

Experiments were performed to tune parameters of the genetic algorithm in order to achieve best convergence rate of its fitness function. Simulation results have shown that our genetic algorithm efficiently determines the mapping of tasks that leads to minimal energy consumption in the WSN. Even in large sensor networks genetic algorithm finds suboptimal solution with minimal error.

To sum up, the proposed technique is dedicated to the compiler task allocation module [4]. We introduced accurate energy and communication models that lead to better cost estimates.

In the future, we intend to extend our work to design space exploration for wireless sensor network in order to evaluate various performance metrics in differrent platforms.

References

[1] **A. Mehrabi, S. Mehrabi, A. D. Mehrabi.** An Adaptive Genetic Algorithm for Multiprocessor Task Assignment Problem with Limited Memory. *Proc. of World Congress on Engineering and Computer Science*, October 2009, *Vol*.2, 1018-1023.

[2] **A. Misevičius, D. Rubliauskas, V. Barkauskas.** Some Further Experiments with the Genetic Algorithm for the Quadratic Assignment Problem. *Information Technology and Control*, *Vol*.38, *No*.4, 2009, 325-332.

[3] **A. Misevičius, T. Blažauskas, J. Blonskis, J. Smolinskas.** An Overview of Some Heuristic Algorithms for Combinatorial Optimization Problems. *Information Technology and Control*, *Vol*.30, *No*.1, 2004, 21-31.

[4] **A. Pathak, L. Mottola, A. Bakshi, V.K. Prasanna, G.P. Picco.** A Compilation Framework for Macroprogramming Networked Sensors. *Proc. of the* 3*rd Intl. Conf. on Distributed Computing in Sensor Systems*, *Vol*.4549, *June* 2007, 189-204.

[5] **A. Pathak, V. K. Prasanna.** Energy-Efficient Task Mapping for Data-Driven Sensor Network Macroprogramming. *Proc. of the* 4*th IEEE Intl. Conf. on Distributed Computing in Sensor Systems*, *Vol*.5067, 2008, 516-524.

[6] **A. Pawlowski, J.L. Guzman, F. Rodriguez, M. Berenguel, J. Sanchez, S. Dormido.** Simulation of Greenhouse Climate Monitoring and Control with Wireless Sensor Network and Event-Based Control. *Sensors*, *Vol*.9, *No*.1, *January* 2009, 232-252.

[7] **A.S. Wu, H. Yu, S. Jin, K. Lin, G. Schiavone.** An Incremental Genetic Algorithm Approach to Multiprocessor Scheduling. *IEEE Trans. on Parallel and Distributed Systems*, *Vol.*15, *No.*9, *September* 2004, 824-834.

[8] **A. Wang, A. Chandrakasan.** Energy-Efficient DSPs for Wireless Sensor Networks. *IEEE Signal Processing Magazine*, *Vol*.19, *No*.4, *July* 2002, 68-78.

[9] **B.L. Titzer, D.K. Lee, J. Palsberg.** Avrora: Scalable Sensor Network Simulation with Precise Timing. *Proc. of the* 4*th Intl. Conf. on Information Processing in Sensor Networks*, *April* 2005, 477-482.

[10] **B. Ucar, C. Aykanat, K. Kaya, M. Ikinci.** Task assignment in heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, *Vol*.66, *No*.1, *January* 2006, 32-46.

[11] **D.H. Park, J.W. Park.** Wireless Sensor Network-Based Greenhouse Environment Monitoring and Automatic Control System for Dew Condensation Prevention. *Sensors*, *Vol*.11, *No*.4, *March* 2011, 3640-3651.

[12] **F.H. Bijarbooneh, P. Flener, E. Ngai, J. Pearson.** Energy-Efficient Task-Mapping for Data-Driven Sensor Network Macroprogramming Using Constraint Programming. *Proc. of the* 12*th INFORMS Computing Society Conference,* January 2011, 1-13.

[13] **G. Narvydas, R. Simutis, V. Raudonis.** Autonomous Mobile Robot Control Using IF-THEN Rules and Genetic Algorithm. *Information Technology and Control*, *Vol*.37, *No*.3, 2008, 193-197.

[14] **H. Park, M.B. Srivastava.** Energy-Efficient Task Assignment Framework for Wireless Sensor Networks. *Center for Embedded Network Sensing*, *Technical Report* 26, *September* 2003, 1-10.

[15] **J. Zhu, J. Li, H. Gao.** Energy Efficient and QoS-Aware Tasks Allocation for Sensor Networks. *Journal of Digital Information Management*, *Vol*.5, *No*.2, *April* 2007, 88-94.

[16] **K. Kaya, B. Ucar.** Exact Algorithms for a Task Assignment Problem. *Parallel Processing Letters*, *Vol*.19, *No*.3, *May* 2009, 451-465.

[17] **M. Paulinas, A. Ušinskas.** A Survey of Genetic Algorithms Applications for Image Enhancement and Segmentation. *Information Technology and Control*, *Vol*.36, *No*.3, 2007, 278-284.

[18] **N. Nedic, S. Vukmirovic, A. Erdeljan, L. Imre, D. Capko.** A Genetic Algorithm Approach for Utility Management System Workflow Scheduling. *Information Technology and Control*, *Vol*.39, *No*.4, 2010, 310-316.

[19] **S.M.S. T. Yazdi, S.A. Savari.** A Max-Flow/Min-Cut Algorithm for a Class of Wireless Networks. *Proc. of the* 21*st ACM-SIAM Symp. on Discrete Algorithms*, *January* 2010, 1209-1226.

[20] Sun SPOT World. *Sun™ SPOT Main Board Technical Datasheet, Rev. 8.0*, *www.sunspotworld.com*.

[21] **V. Martusevičius, E. Kazanavičius, V. Jukavičius.** Task Migration Subsystem for Macro-Programmed Wireless Sensor Network. *Proc. of the* 16*th Intl. Conf. on Information and Software Technologies*, April 2010, 175-182.

[21] **W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan.** Energy-Efficient Communication Protocol for Wireless Microsensor Networks. *Proc. of the* 33*rd*

Hawaii Intl. Conf. on System Sciences, *Vol*.8, *January* 2000, 3005-3014.

[22] Y.E. M. Hamouda, C. Phillips. Biological Task Mapping and Scheduling in Wireless Sensor Networks. *IEEE International Conference on Communication Technology and Applications*, *October* 2009, 914-919.

[23] Y. Tian, E. Ekici, F. Ozguner. Energy-Constrained Task Mapping and Scheduling in Wireless Sensor Networks. *Proc. of the IEEE Intl. Workshop on Resource Provisioning and Management in Sensor Networks*, *November* 2005, 211-218.

[24] Y. Tian, E. Ekici. Cross-Layer Collaborative In-Network Processing in Multihop Wireless Sensor Networks. *IEEE Trans. on Mobile Computing*, *Vol*.6, *No*.3, March 2007, 297-310.

[25] Y. Yu, V. K. Prasanna. Energy-Balanced Task Allocation for Collaborative Processing in Networked Embedded Systems. *Proc. ACM SIGPLAN Conference on Languages, Compilers, and Tools for Embedded Systems*, June 2003, 265-274.

Received March 2011.