A Modified Discrete Differential Evolution based TDMA Scheduling Scheme for Many to One Communications in Wireless Sensor Networks

Sk. Minhazul Islam[1], Saurav Ghosh[1], Swagatam Das[1], Ajith Abraham[2, 3]and Subhrajit Roy[1],

[1]Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700 032, India
[2]Faculty of Computer Science and Electrical Engineering, VSB – Technical University of Ostrava, Czech Republic
[3]Machine Intelligence Research Labs (MIR Labs), Seattle, WA, USA
ajith.abraham@ieee.org

Abstract— **Time Division Multiple Access (TDMA) plays an important role in MAC (Medium Access Control) for wireless sensor networks providing real-time guarantees and potentially reducing the delay and also it saves power by eliminating collisions. In TDMA based MAC, the sensor are not allowed to radiate signals when they are not engaged. On the other hand, if there are too many switching between active and sleep modes it will also unnecessary waste energy. In this paper, we have presented a multi-objective TDMA scheduling problem that has been demonstrated to prevent the wasting of energy discussed above and also further improve the time performance. A Modified Discrete Differential Evolution (MDDE) algorithm has been proposed to enhance the converging process in the proposed effective optimization framework. Simulation results are given with different network sizes. The results are compared with the Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) and the original Discrete DE algorithm (DDE). The proposed MDDE algorithm has successfully outperformed these three algorithms on the objective specified, which is the total time or energy for data collection.**

Keywords- ***Differential Evolution, Particle swarm optimization, Genetic algorithm, TDMA scheduling, Wireless sensor networks***

I. INTRODUCTION

Wireless sensor networks have brought up a new field of research [1] due to their growing application in defense, pervasive commercial and scientific applications. In such WSN, sensors are destined to perform the job of sensing, processing, and wireless networking capability. They automatically collect the information and transmits the measurements to a particular access point. Till date, many WSN's have been designed and deployed for kinds of applications [2]. Although, there are many system challenges to resolve for WSN having the promising applications which are very attractive. The first thing is the energy which posses a great problem for WSN since sensors are usually battery-driven. The second thing is that the sensors need a very short time to perform data collection. In these cases, TDMA can be a good choice to satisfy the above requirements. At first, TDMA can save energy by eliminating collisions, avoiding idle listening, or entering inactive states until their allocated time slots and secondly, as a collision-free access method, TDMA can bind the delays of packets, guarantees reliable communication. Now, for the TDMA based MAC the main problem is to allocate time slots for each pair of neighboring nodes, where the time and the energy consumption are considered. There are some researches in the field of the energy saving problem [3, 4] for TDMA scheduling. Pei et al. [3] combined passive clustering and TDMA .Jolly et al. [4] saved energy by minimizing the number of state transitions between active and sleep modes, where tabu search based technique was used. For TDMA scheduling, there are some references for minimizing packet delay [5], improving fairness [6], maximizing parallel operation [7] [8], and shortening the total slots to finish a set of transmission tasks [9] etc. Cui et al. [5] adopted relaxation methods to solve the problem by convex optimization model and got parity optimal energy-delay curves. Sridharan et al. [6] developed a linear programming formulation and presented a distributed solution, which outperformed random MAC in terms of fairness and delay etc. Kalyan et al. [7] used PSO algorithm to minimize the overall transaction time, which was modeled as a graph partitioning problem. To the similar problem, Gandham et al. [8] proposed a distributed edge-coloring algorithm. In order to save time for data collection, Ergen *et al.* [9] proposed three algorithms based on coloring method in graph theory. However, these three references did not consider the energy saving problem in sensor networks. In this paper, we present a new multi-objective optimization framework for slot scheduling in many-to-one sensor networks, inspired by the scheduling model in [9].

Differential Evolution (DE) [10, 11] is simple yet very powerful real parameter optimization algorithm. In this article, a modified discrete Differential Evolution (MDDE) algorithm has been proposed for searching the optimal solution and the results are compared with the classical PSO (particle Swarm optimization) and GA (Genetic Algorithm) and the original discrete DE (DDE) [13] which itself is a efficient evolutionary algorithm. The desired objectives are to minimize the total time for collecting a set of data and to save the energy consumed on switching between the active and sleep states. The remainder of this paper is organized as follows: the problem statement is introduced in section 2. Section 3 describes the optimization framework, coding method and fitness function. The original DE algorithm and the MDDE algorithm is described in Sections 4 and 5 in details. In Section 6, the computational results are given. Some concluding remarks are made in Section 7.

II. Problem Statement

A. Scheduling Model

In this paper, we have assumed a static WSN and the access point (AP) which is responsible for tracking the information from all the transmitting nodes and the interference relationship, thus it becomes able to perform the slot assignment and then send it back to nodes in network. In TDMA scheduling problem, time gets divided into equal intervals in the form of time slots. A particular time slot is designed to accommodate a single packet to be transmitted and received between pairs of nodes in the network. A basic principle of TDMA scheduling is to assign time slots to nodes so that collisions would not happen when they are transmitting packets. In many-to-one sensor networks, as the sensor data are flowing toward AP, the TDMA scheduling problem may be formulated as followed. There is a set of sensor data packets which plan to transmit to AP over the routing tree [9]. Each data collection process where a packet flows to AP from its source node is called a task. According to the corresponding route, each task consists of a sequence of transmission actions called subtasks, where one subtask needs one slot occupation. The aim of the problem is thus to determine a sequence of the subtasks and the corresponding slots so that collisions would not happen, and some optimization criteria would be satisfied.

B. Optimization objective description

In this paper, we have incorporated two optimization objectives. At first, our goal is to minimize the energy consumed by the nodes and the other goal is to be to reduce the total time to finish a set of tasks. A sensor can be switched off to save energy when it is not transmitting. However, frequent switching off and on of the sensors would waste large amounts of power [10]. Hence, we take this part of energy into account, which has been ignored in many TDMA researches. According to Ref. [10], the consumption formula of energy of a network is followed:

$$EC = \sum_{i=1}^{N} \left[P_i^{tx} \cdot \left(t_i^{tx} + t_i^{s-tx} \right) + P_i^{rx} \cdot \left(t_i^{rx} + t_i^{s-rx} \right) \right] \tag{1}$$

Here, N denotes the number of nodes in the network, $P_i^{tx/rx}$ is the consumed power of transmitter/receiver at node *i*. $t_i^{tx/rx}$ is the total work time of the transmitter/receiver at node *i*. $t_i^{s-tx/rx}$ is the consumed time for the total transition between per sleep and active states. The total time is measured in the form of number of time slots which is regarded as a performance metric. This metric should be minimized, because lesser total time, the faster the data collection rate. We we take the weighted sum of the two objectives in our problem as the objective function, shown as follow:

$$\min\left(F(s)\right) = \alpha \cdot EC + (1-\alpha) \cdot totalslots \tag{2}$$

Here, *s* means a TDMA schedule. *EC* is the total energy consumption of the network, "totalslots" is the total number of the time slot under the schedule, α is a tradeoff factor between these two objectives.

III. Optimization Framework for TDMA Scheduling

A. Optimization Framework

In many-to-one WSN, assigning time slots for sensors for data collection tasks is an NP problem [9] and the energy constraints also make this problem more difficult. Differential Evolution (DE) algorithm is a stochastic evolutionary optimization technique which has been successfully applied to many NP-hard problems. Thus we propose to apply a modified version of the discrete DE (MDDE) to the slot scheduling problem and present an optimization framework, shown in Fig. 1. In Fig. 1, one problem solution is encoded to one sequence code, namely vector. By this way an initial population including a number of vectors can be generated. Then this population evolves according to the MDDE, where the fitness function provides the evolution direction. Finally a best vector can be found and decoded to an optimal TDMA scheduling solution.

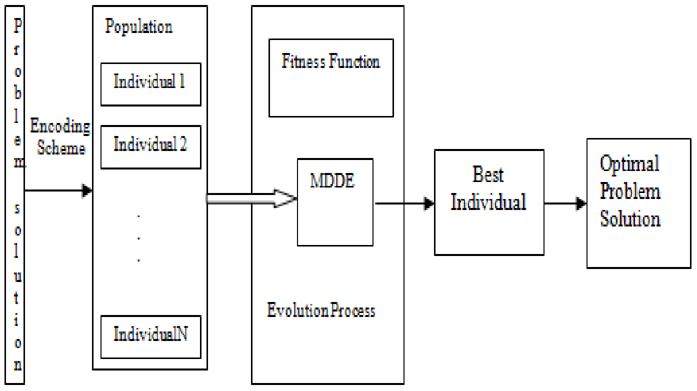

Figure 1: Optimization Framework

B. Encoding and Decoding

In the domain of Evolutionary Algorithm encoding scheme is a very important scheme. Here, our goal is to represent a solution by a sequence code, in the form of a vector to incorporate this in the population based algorithms. Now, following the description given in the Section 2.1 for scheduling problem in sequence a task for collection of data is divided into many hops which are called subtasks. So, a data collection task-set is nothing but a combination of all the subtasks. Therefore, we designate a subtask in the form of (TaskID, Hop-No.). The TaskID informs the task number for a subtask and the HopNo. denotes the sequence number of this subtask in set of all the subtasks of task TaskID. To demonstrate this, we are considering that the first task has two Hops, so the required subtasks will be designated as (1, 0) and (1, 1). Following this representation procedure, the TaskIDs from a particular subtask sequence is made to form a vector. We have given an example shown in Figure 2 to show the

above approach of encoding. In this example, two tasks are there, 1) A packet is getting transmitted from node 0 to AP and 2) A packet is getting transmitted from node AP to 0. Now, each task contains 4 subtasks, i.e., (0, 0)(0, 1)(0, 2)(0, 3) and (1, 0)(1, 1)(1, 2)(1, 3), respectively. A random combination of the above subtask is,
(0, 0) (1, 0) (0, 1) (0, 2) (1, 2)(0, 3) (1, 1) (1, 3)
And then taking the TaskIDs out, the above sequence can be encoded as follow: 01001011.

To summarize the above process, the encoding rule is as follows. Given N tasks and each task i includes M_i hops, an vector is a sequence composed of all the task ID numbers from 0 to $N-1$, where each task number i appears for M_i times. Obviously, the length of the vector is $\sum_{i=0}^{N-1} M_i$. According to this rule, a vector can be generated at random. The decoding method of a vector is as follows. First, a vector is transformed to a sequence of subtasks. Then, from the first subtask to the last one we assign slots by sequence, at the same time, trying to assign those subtasks to one slot without rousing collisions. Take an vector from the above example network, i.e., 0001111, the corresponding sequence of subtasks is (0, 0)(0, 1)(0, 2)(0, 3)(1, 0)(1, 1)(1, 2)(1, 3). Then the slot allocation scheme is shown in Table 1. From the decoding scheme, it can be seen that an vector can finally be decoded to a slot scheduling, which allows parallel operations. The mapping between an vector and a TDMA scheduling solution is now completely set up.

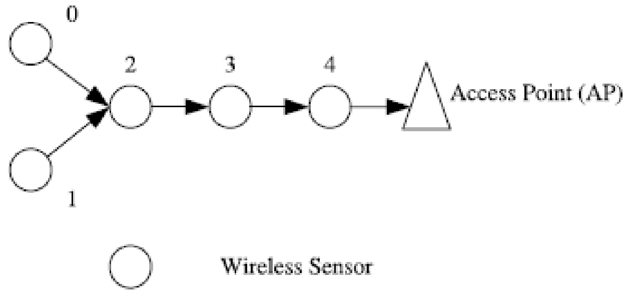

Figure 2: Example Network

Table 1: Slot allocation of the example

Slot No.	1	2	3	4	5	6	7
Sub task	(0,0)	(0,1)	(0,2)	(0,3) (1,0)	(1,1)	(1,2)	(1,3)
Execution Node	0	2	3	4,1	2	3	4

C. *Fitness Function*

During the evolution process, fitness is used as the performance evaluation of vector of population. Usually the selection of fitness function depends on the optimization goals. In this paper, we directly take the objective function in (2) as the fitness function, i.e.

$$Fit(i) = F(s) = \alpha \cdot EC + (1-\alpha) \cdot totalslots \qquad (3)$$

Here, i represents an vector of the evolution algorithms, the schedule s is decoded from the vector i.

IV. Classical Differential Evolution

DE is a simple real-coded evolutionary algorithm. In this section we describe the basic operations of DE and introduce necessary notations and terminologies, which facilitate the explanation of our adaptive DE algorithm later.

A. *Initialization of the Parameter Vectors*

DE searches for a global optimum point in a D-dimensional continuous hyperspace. It begins with a randomly initiated population of Np (population number) D dimensional real-valued parameter vectors. Each vector, also known as *genome/chromosome*, forms a candidate solution to the multi-dimensional optimization problem. We shall denote subsequent generations in DE by $G = 0,1,....,G_{\max}$ .Here we adopt the following notation for representing the i-th vector of the population at the current generation:

$$x_i^{(G)} = [x_{1,i}^{(G)}, x_{2,i}^{(G)}, x_{3,i}^{(G)}, ..., x_{D,i}^{(G)}]. \qquad (4)$$

The initial population (at $G=0$) should cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds: $x_{\min} = \{x_{1,\min}, x_{2,\min}, ..., x_{D,\min}\}$ and $x_{\max} = \{x_{1,\max}, x_{2,\max}, ..., x_{D,\max}\}$ .Hence we may initialize the j-th component of the i-th vector as:

$$x_{j,i}^{(0)} = x_{j,\min} + rand_{i,j}(0,1) \cdot (x_{j,\max} - x_{j,\min}) \qquad (5)$$

where rand is a uniformly distributed random number lying between 0 and 1 (actually $0 \le rand_{i,j}(0,1) < 1$) and is instantiated independently for each component of the i-th vector.

B. *Mutation with Difference Vectors*

After initialization DE, creates a donor vector $v_i^{(G)}$ corresponding to each population member or target vector $x_i^{(G)}$ in the current generation through mutation. It is the method of creating this donor vector, which differentiates between the various DE schemes. Five most frequently referred mutation strategies implemented in the public-domain DE codes available online *at http://www.icsi.berkeley.edu/~storn/code.html* are listed below:

"DE/rand/1": $v_i^{(G)} = x_i^{(G)} + F \cdot (x_{r_1^i}^{(G)} - x_{r_2^i}^{(G)}).$ (6)

"DE/best/1": $v_i^{(G)} = x_{best}^{(G)} + F \cdot (x_{r_1^i}^{(G)} - x_{r_2^i}^{(G)}).$ (7)

"DE/target-to-best/1":

$$v_i^{(G)} = x_i^{(G)} + F \cdot (x_{best}^{(G)} - x_i^{(G)}) + F \cdot (x_{r_1^i}^{(G)} - x_{r_2^i}^{(G)}). \tag{8}$$

"DE/best/2":

$$v_i^{(G)} = x_{best}^{(G)} + F \cdot (x_{r_1^i}^{(G)} - x_{r_2^i}^{(G)}) + F \cdot (x_{r_3^i}^{(G)} - x_{r_4^i}^{G}). \tag{9}$$

"DE/rand/2":

$$v_i^{(G)} = x_{r_1^i}^{(G)} + F \cdot (x_{r_2^i}^{(G)} - x_{r_3^i}^{(G)}) + F \cdot (x_{r_4^i}^{(G)} - x_{r_5^i}^{(G)}). \tag{10}$$

The indices r_1^i, r_2^i, r_3^i, r_4^i, and r_5^i are distinct integers uniformly chosen from the set {1, 2,. . . *Np*} with *i* being the vector concerned in the population. These indices are randomly generated once for each donor vector. The scaling factor *F* is a positive control parameter for scaling the difference vectors. $x_{best}^{(G)}$ is the best individual vector with the best fitness (i.e. lowest objective function value for minimization problem) in the population at generation g.

C. *Crossover*

To enhance the potential diversity of the population, a crossover operation comes into play after generating the donor vector through mutation. The donor vector exchanges its components with the target vector $x_i^{(g)}$ under this operation to form the *trial* vector $u_i^{(G)} = [u_{1,i}^{(G)}, u_{2,i}^{(G)}, u_{3,i}^{(G)}, ..., u_{D,i}^{(G)}]$ .The scheme may be outlined as:

$$u_{j,i}^{(G)} = \begin{cases} v_{j,i}^{(G)}, & \text{if } rand_{i,j}(0,1) \le Cr \text{ or } j = j_{rand} \\ x_{j,i}^{(G)}, & \text{otherwise,} \end{cases} \tag{11}$$

where, $rand_{i,j}(0,1)$ is a uniformly distributed random number, for each *j*-th component of the *i*-th parameter vector. $j_{rand} \in \{1, 2, ..., D\}$ is a randomly chosen index, which ensures that $u_i^{(G)}$ gets at least one component from $v_i^{(G)}$. These *F*, *Cr* and population number are the control parameters of the basic DE.

D. *Selection*

The next step of the algorithm calls for *selection* to determine whether the target or the trial vector survives to the next generation i.e. at $G = G + 1$. The selection operation is described as:

$$x_i^{(G+1)} = \begin{cases} = u_i^{(G)}, & if \quad f(u_i^{(G)}) \le f(x_i^{(G)}) \\ = x_i^{(G)}, & if \quad f(u_i^{(G)}) > f(x_i^{(G)}) \end{cases} \tag{12}$$

where $f(x)$ is the objective function to be minimized.

V. MODIFIED DISCRETE DIFFERENTIAL EVOLUTION

The TDMA scheduling mechanism is a discrete optimization problem. But the continuous nature of DE does not permit the algorithm to apply to discrete optimization problems. In this article a novel discrete version of a modified DE is designed to solve the combinatorial optimization problem. According to our TDMA problem, some parameters in DE are defined as follows. Given *N* tasks, each task *i* includes M_i hops. The dimension of vectors is set to the number of the total subtasks, i.e. $\sum_{i=0}^{N-1} M_i$. The border of the searching space is defined by $(N-1)$.

A. *Initial Population*

The appropriate selection of initial population is necessary for any evolutionary algorithm to approach optimal fitness. Here, we present the excellent concept of simulated orthogonal array (SOA) [14], which are used in experimental simulation methods. For an experiment having m factors and each having n levels, an orthogonal array is an array with m rows and n columns which is a representative sample of some testing experiments that satisfies the condition that for the factors in any column, every level occurs the same number of times. For each column of $[SOA]_{m\times n}$, a random permutation of 1, …, m is generated and denoted as sequence Z. Then the elements in Z are picked sequentially one by one and filled randomly in the column. When all elements in Z were picked, the process starts once again from the beginning of sequence. So in every column of $[SOA]_{m\times n}$, each of m elements will appear the same number of times. In the beginning, the proposed algorithm generates initial solutions by using the simulated orthogonal array, which ensures uniform initialization within the search range [0, N-1].

B. *Type of Mutation*

In DE, greedy strategies like DE/current-to-best/*k* and DE/best/*k* benefit from their fast convergence by guiding the evolutionary search with the best solution so far discovered, thereby converging faster to that point. But, as a result of such exploitative tendency, in many cases, the population may lose its diversity and global exploration abilities within a relatively small number of generations, thereafter getting trapped to some locally optimal point in the search space. Taking into consideration these facts and to overcome the limitations of fast but less reliable convergence performance of DE/current-to-best/1 scheme, in this article, we propose a less greedy and more explorative variant of the DE/current-to-best/1 mutation strategy by utilizing the best vector of a dynamic group of *q*% of the randomly selected population members for each target vector. The new scheme, which we call DE/current-to-gr_best/1, can be expressed as:

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F \cdot (\vec{X}_{gr\_best,G} - \vec{X}_{i,G} + \vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) \tag{13}$$

where $\vec{X}_{gr\_best,G}$ is the best of q % vectors randomly chosen from the current population whereas $\vec{X}_{r_1^i,G}$ and $\vec{X}_{r_2^i,G}$ are two distinct vectors picked up randomly from the current population. Under this scheme, the target solutions are not always attracted towards the same best position found so far by the entire population and this feature is helpful in avoiding premature convergence at local optima. The parameter q is known as the group size which controls the greediness of the mutation scheme DE/target-to-gr_best/1. As the mutation process is continuous the components of the mutated vectors generated takes on continuous values. These values are encoded in the discrete domain.

C. Scale Factor Adaptation

At every generation, the scale factor F_i *of* each individual target vector is independently generated as:

$$F_i = Cauchy(F_m, 0.1), \tag{14}$$

where $Cauchy(F_m, 0.1)$ is a random number sampled from a Cauchy distribution with location parameter F_m and scale parameter 0.1. The value of F_i is regenerated if $F_i \le 0$ or $F_i > 1$. Denote $F_{success}$ as the set of the successful scale factors, so far, of the current generation generating better trial vectors that are likely to advance to the next generation. Also let $mean_A(F_{G-1})$ is the simple arithmetic mean of all scale factors associated with population members in generation $G-1$. Location parameter F_m of the Cauchy distribution is initialized to be 0.5 and then updated at the end of each generation in the following manner:

$$F_m = w_F \cdot F_m + (1 - w_F) \cdot mean_{Pow}(F_{success}) \tag{15}$$

The weight factor w_F is set in the following way:

$$w_F = (0.8 + 0.01 \cdot (rand(0,1))) \tag{16}$$

where $rand(0,1)$ stands for a uniformly distributed random number in (0, 1) and $mean_{Pow}$ stands for power mean given by:

$$mean_{Pow}(F_{success}) = \sum_{x \in F_{Success}} \left(x^n / \left| F_{success} \right| \right)^{1/n}, \tag{17}$$

with $\left| F_{Success} \right|$ denoting the cardinality of the set $F_{Success}$. We took n = 1.5. Small random perturbations to the weight terms of F_m and $mean_{pow}$ puts slightly varying emphasis on the two terms each time an F is generated, and improves the performance of MDDE in this TDMA problem which is found experimentally.

D. Discretization of Variables

In this TDMA scheduling problem if we have N number of tasks and each task having M_i hops the dimension of the vectors are $\sum_{i=0}^{N-1} M_i$ .and the maximum number of value that a particular dimension of a vector can take is N. In this paper, we opted for a quantization rule for the discretization. Suppose a dimension is having a value of 7.2 after mutation, then after the discretization the value would be 7 and if it were to be 7.6 then it would have been 8 and if it were to be 7.5 then after discrtization it would have been either 7 or 8 just like a toss of a coin so that each has a equal probability. So, after the discrete rule the value that a dimension of a vector takes up is the nearest integer from it.

E. The p-Best Crossover

The crossover operation named as p-best crossover where for each donor vector, a vector is randomly selected from the p top-ranking vectors (according to their objective function values) in the current population and then normal binomial crossover is performed as per eqn. (11) between the donor vector and the randomly selected p-best vector to generate the trial vector at the same index. In this way, the information contained in the top-ranking individuals of the population is incorporated into the trial vector through component exchange so that the population can evolve in a much better manner. The parameter p is linearly reduced with generations in the following way:

$$p = ceil\left\lfloor \frac{NP}{2}\left(1 - \frac{G-1}{G_{max}}\right) \right\rfloor, \tag{18}$$

where NP is the population size, G is the current generation number, $G_{\max}$ is the maximum number of generations, $G = [1, 2, ..., G_{max}]$, and $ceil(y)$ is the 'ceiling' function returning the lowest integer greater than its argument y. The reduction routine of p favors exploration at the beginning of the search and exploitation during the later stages by gradually downsizing the elitist portion of the population.

F. Crossover Probability Adaptation

At every generation the crossover probability Cr_i of each individual vector is independently generated as:

$$Cr_i = Gaussian(Cr_m, 0.1), \tag{19}$$

where $Gaussian(Cr_m, 0.1)$ is a random number sampled from a Gaussian distribution according with mean Cr_m and standard deviation 0.1. Cr_i is regenerated if it falls outside the interval [0, 1]. Denote Cr_{success} as the set of all successful crossover probabilities Cr_i 's at the current generation. The mean of the normal distribution Cr_m is initialized to be 0.6 and then updated at the end of each generation as:

$$Cr_m = w_{Cr} \cdot Cr_m + (1 - w_{Cr}) \cdot mean_{Pow}(Cr_{success}), \tag{20}$$

with the weight being set as:

$$w_{Cr} = 0.9 + 0.001 * rand(0,1), \tag{21}$$

The power mean is calculated as:

$$mean_{Pow}(Cr_{success}) = \sum_{x \in Cr_{Success}} \left(x^n / \left| Cr_{success} \right| \right)^{1/n}, \qquad (22)$$

where $\left| Cr_{Success} \right|$ denotes the cardinality of the set $Cr_{Success}$. We took n = 1.5 here also.

G. Putting it all together

In what follows we shall refer to this adaptive discrete DE algorithm as MDDE (Modified Discrete Differential evolution with *p*-best crossover). The proposed algorithm incorporates modifications in all the algorithmic components of classical DE-mutation, crossover and parameter adaptation.

1) Modification 1: Firstly a less greedy and more explorative variant of the DE/current-to-best/1 mutation strategy is proposed. We call it DE/current-to-gr_best/1 (gr stands for group). Unlike DE/current-to-best/1 that always uses the fittest (hence best) member of the entire current population to perturb a target (parent) vector, DE/current-to-gr_best/1 forms a group, corresponding to each target vector, by randomly selecting population members to form a group whose size is say *q*% of the total population size. The best member of this dynamic group is used to perturb the target vector.

2) Modification 2: Secondly we propose an exploitative crossover scheme called "*p*-best crossover". Under this scheme a mutant vector is allowed to exchange its components not with the parent at the same index, but with a randomly selected member from the *p* top-ranked individuals from the current generation through uniform (binomial) crossover.

3) Modification 3: Thirdly novel parameter adaptation schemes for F and Cr are proposed to update the values of these control parameters in each generation, guided by the knowledge of the successful values of F and Cr that were able to generate better offspring (trial vectors) in the last generation.

VI. Simulation Results

A. Settings

We have taken the results on a 2.13GHz quad to duo machine with 2 GB RAM memory in MATLAB 7.5 environment. The channel capacity is set to 500 kbps and the packet lengths are 1 kbit. Some parameters in energy aspect are as follows: similar to Ref. [4], the transition time between the sleep and active states is assumed to 470lsec. The power consumed in transmission and reception of a packet set to 81and 180 mW, respectively. The power consumed in idle state is set as same as the reception state. In addition, we assume that the clock drift can be ignored by lengthening the slot time slightly. In this paper, our MDDE is compared with three algorithms: the classical PSO Algorithm (PSO) and Genetic Algorithm (GA) and the original discrete DE algorithm. The running parameters of the MDDE are listed in Table 2. For fair comparisons, all these algorithms are running in energy efficient modes, i.e., a node with no packet to send keeps its radio off during its allocated time slots. The setting of task amount in our simulations is a sampling round. A sampling round is defined as a data collection process, which starts when each sensor has generated one packet to send, and ends when all the packets have arrived at AP. The following metrics are used to measure the performance of the proposed algorithm: (1) number of time slots to finish a sampling round and (2) total energy consumption of the network to finish a sampling round, this metric should be minimized to extend network lifetime; (3) slot utility, which is defined as number of empty slot divided by the number of total time slots. This metric reflects the channel efficiency of an algorithm.

Table 2: Parameters of the MDDE

Initial population	50
Number of Generation	600
Value of *q*	10
Initial *mean (F)*[Scale factor] F_m	0.5
Initial *mean (Cr)*[crossover Probability] Cr_m	0.6

B. Simulation in a Simple Network

The goal of this simulation is to show the slot allocation results in detail and to provide experimental basis for scheduling performance analysis. Fig. 3 depicts the simulation network topology with 7 sensor nodes and 1 AP, where each node's ID is marked in the circles. Both the arrows and the dot lines mean links between two nodes, besides, the arrows show the routing direction. The slot allocation results are shown in Table 3, where the working nodes in each slot are given. Table 3 shows the energy performance results under the above slot allocation schemes.

C. Simulation on four random networks

The goal of this experiment is to evaluate the performance of the algorithms under the different network sizes. The experiments are done with four random networks, and its specifications are shown in Table 5.The routers have been computed based upon the GPSR algorithm [12], such that each sensor will transmit the information to its neighbor which is closest to the destination. AP node is set as the node in center of the zone where the sensor are scattered. The time and energy results of finishing a sampling round are given in Table 6 and Table 7, respectively, where $\alpha = 0$ means the objective is the optimal time performance, $\alpha = 1$ means the objective is the optimal energy performance. Here, we have chosen only 2 values of α to only show the optimum time and energy performance. Table 6 shows that MDDE ($\alpha = 0$) is optimal and DDE ($\alpha = 0$) is suboptimal and rest of the algorithm follows when $\alpha = 1$ while all the algorithms keep 100% slot utilization. Table 7 shows that MDDE ($\alpha = 1$) is optimal and DDE ($\alpha = 1$) is suboptimal and rest of the algorithm follows when $\alpha = 0$. Again, both the Table 6and 7 shows that time and energy performance cannot be optimized at the same time for any algorithm. So, there should exist a tradeoff between these two metrics.

Table 3.a: Slot allocation of the algorithms (For $\alpha = 0$)

Slot No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GA	1,3	0,4	0,5	2,6	2,6	2	2	3	3	3	4	4	4	4	2
PSO	1,4	0,5	2,6	2,6	2	2	3	3	3	4	4	4	4		
DDE	3,5	1,4	2,6	2,6	0,4	2	3	3	3	4	4	4			
MDDE	2,4	1,3	3,6	2,5	3,5	2	2	3	3	4	4				

Table 3.b: Slot allocation of the algorithms (For $\alpha = 1$)

Slot No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
GA	0,3	1,4	0,5	2,6	2,6	2	2	3	3	3	4	4	4	4	2	2
PSO	2,4	0,5	1,6	2,6	2	2	3	3	3	4	4	4	4	2		
DDE	1,2	3,4	1,5	2,4	3,6	2	2	2	3	3	3	4	4	4		
MDDE	2,5	1,4	2,6	2,6	1,5	2	3	3	3	4	4	4				

Table 4.a: Scheduling performance of the algorithms ($\alpha = 0$)

Algorithms	No. of total slots	Total Energy Consumption (mJ)	Slot utilization (%)
GA	15	15.846	83%
PSO	13	12.371	94%
DDE	12	12,012	96%
MDDE	11	10.391	100%

Table 4.b: Scheduling performance of the algorithms ($\alpha = 1$)

Algorithms	No. of total slots	Total Energy Consumption (mJ)	Slot utilization (%)
GA	16	14.214	78%
PSO	14	11.527	91%
DDE	14	11,296	94%
MDDE	12	9.992	100%

Table 5: Problem Specification

Problem No.	Number of nodes	Number of link	Average degree
1	25	61	4.88
2	49	134	5.47
3	121	352	5.82
4	169	489	5.79

Table 6: Time performance of the algorithms

Problem No.			No. of total time slots						Slot Utilization
		$\alpha = 0$				$\alpha = 1$			
	GA	PSO	DDE	MDDE	GA	PSO	DDE	MDDE	
1	31	29	29	**28**	32	31	31	30	100% for algorithms
2	57.0	56.1	55.9	**55.0**	63	60	59	58	
3	151	147	144	**143**	159	156	154	152	
4	209	202	200	**198**	213	210	207	203	

Table 7: Energy consumption of the algorithms

Problem No.		$\alpha = 0$				$\alpha = 1$		
	GA	PSO	DDE	MDDE	GA	PSO	DDE	MDDE
1	39.23	38.14	37.92	37.67	37.2	36.8	36.1	**35.6**
2	108.3	106.1	105.6	105.1	100.1	99.8	99.6	**99.1**
3	418.71	415.2	417.8	419.2	403.8	402.2	402.0	**401.7**
4	667.87	660.1	655.3	650.9	648.3	647.5	646.3	**644.1**

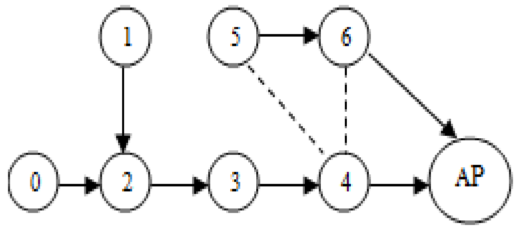

Figure 3: A simple network topology

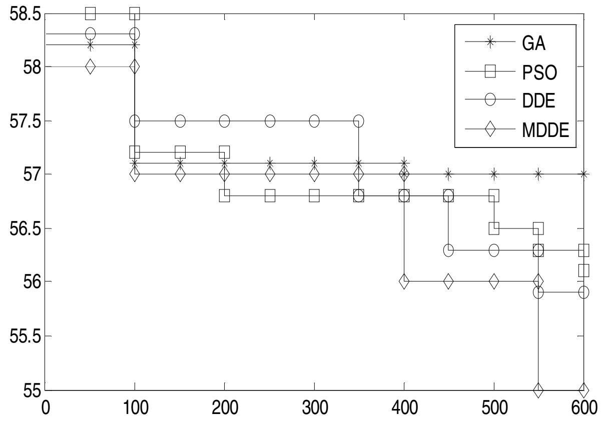

Figure 4: The time performance over generations in Problem 2

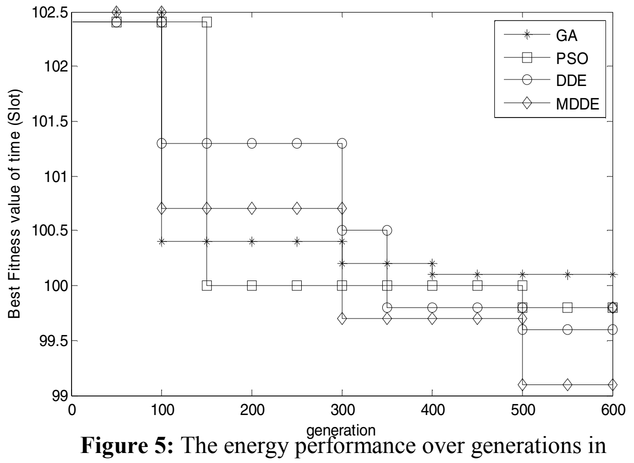

Figure 5: The energy performance over generations in Problem 2

VII. CONCLUSIONS

In this paper, we focus on two topics. At first, we present a general optimization framework for TDMA scheduling in many-to-one sensor networks, which can optimize a number of targets. Secondly, we solve the TDMA scheduling problem by using a problem-specific modified discrete Differential Evolution algorithm which possesses a good searching ability to find the optimal slot allocation scheme. We have demonstrated a multi-objective optimization framework for TDMA scheduling. In this optimization framework, we highlighted two premier things: An innovative encoding method being problem specific and a modified version of Differential Evolution algorithm (DE). The first thing is solely responsible for absence of empty slot during scheduling, and flexible adjustment of sequence of subtasks. The second one ensures a powerful evolutionary algorithm based optimization technique or searching in the hyperspace to find the optimal slot allocation scheme. Simulation results demonstrated that our algorithm (MDDE) outperforms the PSO and GA algorithm and also the discrete DE algorithm. Moreover, the proposed framework is flexible enough to change the desired objective from the time to energy or vice versa or even make trade-off between the above two by a proper weight factor.

ACKNOWLEDGEMENTS

This work was supported by the Czech Science Foundation, under the grant no. GA102/09/1494.

REFERENCES

[1] J.M. Kahn, R.H. Katz, K.S.J. Pister, “Next century challenges: mobile networking for Smart Dust”, *Mobicom, Seattle Washington*, USA,1999, 271–278.

[2] N. Xu, “A survey of sensor network applications”, *IEEE Communications Magazine* 40 (8) (2002) 102–114.

[3] G. Pei and C. Chien, “Low power TDMA in large wireless sensor networks”, *Military Communications Conference*, 2001, 1, 347–351.

[4] G. Jolly, M. Younis, “An energy-efficient scalable and collision-free MAC layer protocol for wireless sensor networks”, *Wireless Communications and Mobile Computing* 5 (3) (2005) 285–304.

[5] S. Cui, *et al*, “Energy-delay tradeoffs for data collection in TDMA-based sensor networks”, *The 40th annual IEEE International Conference on Communications*, Seoul, Korea, May 16–20, 2005.

[6] A. Sridharan, B. Krishnamachari, “Max-min fair collision-free scheduling for wireless sensor networks”, *Workshop on Multihop Wireless Networks* (MWN’04), April 2004.

[7] K. Veeramachaneni, L.A. Osadciw, “Optimal scheduling in sensor networks using swarm intelligence”, *Proceedings of 38th Annual Conference on Information Systems and Sciences,* Princeton University, Princeton, New Jersey, 2004, pp. 17–19.

[8] S. Gandham, M. Dawande, R. Prakash, “Link scheduling in sensor networks: Distributed edge coloring revisited”, *INFOCOM*, 2005, 2492–2501.

[9] S.C. Ergen, P. Varaiya, “TDMA scheduling algorithms for sensor networks”, Technical Report, Department of Electrical Engineering and Computer Sciences University of California, Berkeley, July, 2005.

[10] R. Storn and K. Price, “Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces”, *Journal of Global Optimization*, 11(4) 341–359, 1997

[11] S. Das and P. N. Suganthan, “Differential evolution – a survey of the state-of-the-art”, *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 1, pp. 4 – 31, Feb. 2011

[12] B. Karp, H.T. Kung, “GPSR: Greedy perimeter stateless routing for wireless networks”, *Proc. ACM Mobicom*, Boston, MA, 2000.

[13] M. F. Tasgetiren, Q. K. Pan, Y. C. Liang, and P. N. Suganthan, “A Discrete Differential Evolution Algorithm for the No-Wait Flowshop Scheduling Problem with Total Flow time Criterion”, *IEEE Symposium on Computational Intelligence in Scheduling*, pp. 251-258, Hawaii, April 2007.

[14] L.-Y. Tseng and C. Chen, “Multiple Trajectory Search for Large Scale Global Optimization”, *IEEE World Congress on Computational Intelligence*, Hong Kong, June 1-6, 2008.