International Journal of Scientific Research & Engineering Trends
Volume 6, Issue 4, July-Aug-2020, ISSN (Online): 2395-566X

Reinforcement Learning for Wireless Sensor Networks based on ISVM

Palaghat Hariharan Lakshmi
Dept.of Computer science and engineering
Jawaharlal Nehru Technological University,
Kakinada, Andhra Pradesh,India

***Abstract*-Application performance and energy consumption deep exposed to task scheduling of nodes in wireless sensor networks (WSNs). Unreasonable task scheduling of nodes leads to excessive network energy consumption. Thus, a Q-learning algorithm for task scheduling based on Improved Support Vector Machine (ISVM) in WSNs, called ISVM-Q, is proposed to optimize the application performance and energy consumption of networks. Energy consumption of task scheduling is associated with a reward of nodes in the learning process. To solve the "dimensionality disaster" problem of Q-learning, SVM is introduced as a value function approximation. Parameterizations of SVM function can strengthen the interpretation characteristics by using experience knowledge. Experiments show that ISVM-Q has the ability to make nodes perform tasks reasonably in a dynamic environment. Compared with classic task scheduling algorithms, ISVM-Q achieves better application performance with less energy consumption and keeps the learning system stable.**

***Keywords*- SVM function, Task scheduling , Q-learning, ISVM-Q**

I.INTRODUCTION

1. Wireless Sensor Networks Using Improved Support Vector Machine

In Wireless Sensor Networks (WSN), when a usual event is noticed in the networks, an event is detected through the sensor devices placed at distributed locations. This event detection information is passed to the base station and intelligent decision is taken. In the existing system, an ensemble distributed machine learning approach for detecting events. This approach works in 3 steps: collection of data, defining levels of fires and division of dataset where more packet drops are there. Regression and SVM are the approaches used in proposed architecture for detection of events and prediction of forest fires. This method uses regression for calculating the detection accuracy and errors and levels of fires are defined by SVM.

The predictors considered in the dataset are significant and thus help in better prediction of forest fires. A comparison between proposed approached and other machine learning techniques has also been done which helps to prove that the proposed approach has better detection accuracy and less errors. The R- squared calculated of the R-Squared is very high as compared to other machine learning techniques. Also, the analysis time taken by the proposed approach is less as compared to other techniques. Thus, the proposed algorithm and the approach used are better in terms of detection of forest fires. The proposed algorithm helps in fast and accurate detection of forest fires.

2. Q-Learning:

Q-learning is an approach of Reinforcement Learning to learn the usefulness of tasks over time in a particular environment. As the system developed in the proposed techniques, the agent observes the state "s" of the current environment and performs action "a" according to its learning policy. Then, the environment changes from state "s" to a new state s and also produces a reward r to the agent. If the action a makes the environment produce a positive reward, the learning policy will strengthen the trend to choose the action "a", otherwise, the learning policy will weaken this trend. The task scheduling problem for sensor nodes in a WSN can be mapped to the Q-learning problem as follows:

- Each sensor node in the WSN and environment of the WSN can be mapped to an agent and the environment in Reinforcement Learning model, respectively.
- The task set and the working state set can be mapped to the set of state space and action space for the agent. The task set is associated with the applications of WSNs, and the working state set is determined by the environment of the WSN and the internal working state of the sensor node.
- The incentive reward of task executing is mapped to the re- ward of the agent, receiving from the environment after executing a task.

The task scheduling policy is mapped to the learning policy of agent. Generally, when executing a task successfully, the task scheduling algorithm based on

Independent Q-learning (IQ). Some basic learning elements are defined, including the set of states, set of actions and delayed rewards. Each sensor node is allowed to self-schedule its tasks according to ε-greedy method

3. Basic Functionality of Q-Learning:

Let's say we know the expected reward of each action at every step. This would essentially be like a cheat sheet for the agent! Our agent will know exactly which action to perform.

It will perform the sequence of actions that will eventually generate the maximum total reward. This total reward is also called the Q-value and we will formalise our strategy as:

$$Q(s,a) = r(s,a) + \gamma \max_a Q(s',a)$$

The above equation states that the Q-value yielded from being at state s and performing action a is the immediate reward r(s,a) plus the highest Q-value possible from the next state s'. Gamma here is the discount factor which controls the contribution of rewards further in the future.

Q(s',a) again depends on Q(s",a) which will then have a coefficient of gamma squared. So, the Q-value depends on Q-values of future states as shown here:

$$Q(s,a) \rightarrow \gamma Q(s',a) + \gamma^2 Q(s'',a) \dots\dots\dots \gamma^n Q(s''^{\dots n},a)$$

Adjusting the value of gamma will diminish or increase the contribution of future rewards.

Since this is a recursive equation, we can start with making arbitrary assumptions for all q-values. With experience, it will converge to the optimal policy. In practical situations, this is implemented as an update:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)\right]$$

where alpha is the learning rate or step size. This simply determines to what extent newly acquired information overrides old information.

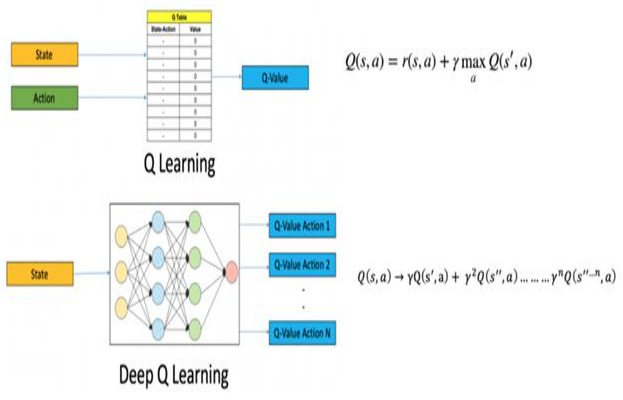


Fig 1 Q-learning and Deep Q-Learning.

4. Why Deep Q-Learning:

Q-learning is a simple yet quite powerful algorithm to create a cheat sheet for our agent. This helps the agent figure out exactly which action to perform.

But what if this cheat sheet is too long? Imagine an environment with 10,000 states and 1,000 actions per state. This would create a table of 10 million cells. Things will quickly get out of control!

It is pretty clear that we can't infer the Q-value of new states from already explored states. This ststes two problems:

- Firstly, the amount of memory required to save and update that table would increase as the number of states increases
- Secondly, the amount of time required to explore each state to create the required Q-table would be unrealistic

Now, what if we approximate these Q-values with machine learning models such as a neural network? Well, this was the idea behind DeepMind's algorithm that led to its acquisition by Google for 500 million dollars!

4.1. Support Vector Machine:

SVM is introduced as an approximation to solve the problem of generalization ability of Q-learning. As a typical representation of the non-parametric model, SVM has the disadvantage of does not use experience or historical information. Therefore, interpretation ability of the model is weaker than the parametric model such as a neural network. ISVM is presented in this paper to form a semi-parametric model.

SVM stores data in a manner of the kernel matrix. The state of the learning system is constantly changing while new data generates constantly. As a result, the size of the kernel matrix keeps increasing, and the calculation speed of SVM greatly reduces

In addition, when a small amount of data is changed in the training sample collection of the SVM model, the SVM model can be retrained to change slightly. Therefore, the sliding window mechanism is introduced. The sample is divided into the new datasets (ND) and the work datasets (WD). The Sample for learning system will first get into ND. When the ND is full, the data in ND enter the WD by the first-in-first-out way. The scale of WD and ND are L and 0.2 L , respectively. L is the number of samples. The specific process is as follows: Sample of the system was filtered by sliding window to train the SVM model for Q value estimation; Pairing the current state s t with each pending discrete action, Q k is obtained from the SVM model according to the continuous state-discrete action pair (s t , a k)(k is the number of discrete actions), which is the estimated value of Q . Then, based on the estimated Q value Q k , the action selector selects action a

t to act on the environment, and the state changes to s t+1 while it gets reward r t . Finally, according to formula (1), the Q value is iteratively calculated, and the target value Q t is obtained, TD error δt = r t + γQ t - Q t−1 is then calculated

4.2. Reward function

Reward function reflects the value of rewards and punishments after task execution. Each sensor node will obtain positive feedback when executing a task successfully. Otherwise, it will receive negative feedback. Applicability predicates are quite simple and self-explanatory. For example, to ensure the nodes available for routing messages or other requirements, the node cannot allow performing sampling if its energy is below a certain threshold. Considering the impact of task scheduling energy consumption on the reward function [2,3,5,6] , and combined with applicability predicates. Number of sensing packets, sending packets, receiving packets and aggregating packets are denoted as N sense, N send, N receive, and N aggre , respectively.

II. SYSTEM ANALYSIS

1. Methodology

The proposed work can be categorized in to three phases: source, receiver, base station.

1.1. Source

Selects the packets and initialize the key address based on content of the packets and send to the base station. Source is forming energy efficient nodes to forward the data packets to base station.

1.2. Routing Path

Initialize the key address for all the nodes. In router to send the packets to the destination path will be find by considering the shortest distance between the nodes. Since packets content will be assigned to all the nodes it checks the energy level of sensors because always (fsize<sensor energy). It check the key address of the sensor and sends attacker details to the IDS manager Base Station In this module, Receiver request for packets name and energy details and receives the packet content from the routing path of the source. Time delay will be calculated by sending the packets from source to destination and time taken to reach the destination.

2. System Architecture

2.1. Improved support vector machine (ISVM)

SVM has advantages of strong adaptability, global optimization, good generalization performance, and complexity of the algorithm is irrelevant to feature space dimension.

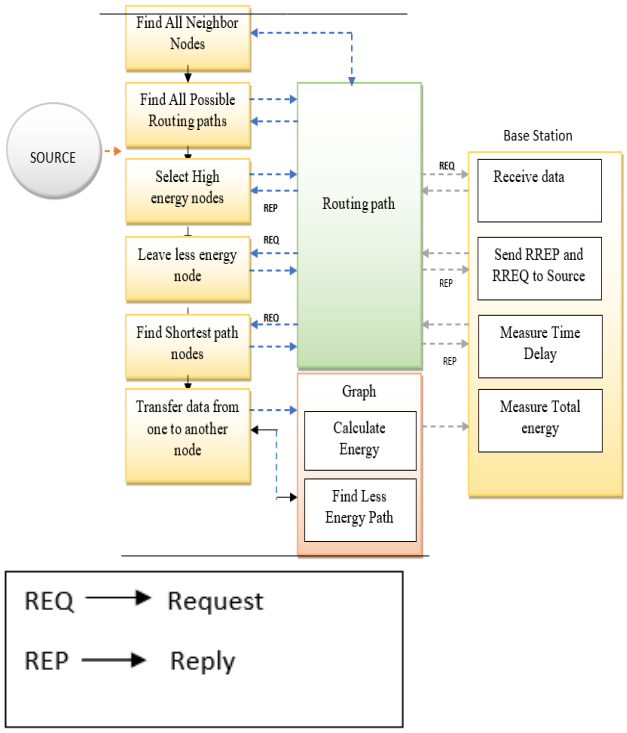


Fig: 2System Architecture.

To improve the interpretation ability of the SVM model by using empirical knowledge, the structure of SVM has been improved parametrical. SVM function approximation and parameterizations are as follows. The environmental estimation model produces value function Q k in the SVM planning process. To maintain the sparsity of the SVM algorithm and remove most of the smaller error sample points in the original variable space, the most common ε- insensitive loss function is introduced [10] . | Q k −f(x) | ε = max { 0 , | Q −f(x) | −ε} , SVM is similar to a neural network, and the output is linear combination of intermediate nodes. Therefore, the form of function can be expressed as: Q k = f(x) = ωφ(x) + b.

The fact of SVM is a typical non-parametric kernel function. To minimize the structural risk of SVM, an improved SVM hybrid model is constructed with the help of parametric linear basis function and used in the regression estimation of Q value: Q k = f(x) = _ ω, φ(x) _ + b,where x = (s, a) is the state-action pair of the system. φ(x) rep- resents a non-linear map that maps inputs to high-dimensional space. ω is the weight of non-parametric SVM model

2.2. Task scheduling algorithm

This section defines the basic elements of Q-learning map- ping in the application including state space, reward function (re- ward and punishment), and exploration-exploitation policy (task scheduling strategy). Finally, the optimization objective is defined.

2.2.1. Task sets

The WSN is composed of n sensor nodes, location coordinates p(x i , y i) (0 <i _ n, n ∈ Z) , sleep time t sleep , collection radius r i and additional attributes are included. Assuming that each task cannot be preempted and is an atomic task, taking the data collection application as an example, the task model is described as follows:

- **Sleeping Task:** This task puts sensor modules, communication modules to sleep mode so as to save energy consumption.
- **Sensing Task:** This task makes to collect data in the monitoring area when the trigger event arrives. Define the trigger event as

{ t start , tduration , (x, y)}, where t start represents the start time of event, t duration represents the duration of the event, and (x, y) is the location coordinates of the event.

- **Sending task**: This task opens the sending module, picks packets out of the sending queue and transmits them to the next hop.
- **Receiving task:** This task opens the receive module and receives a packet from the last hop to the receiving queue.
- **Aggregation task:** This task is to process the received packet and transfer it to the send queue. Although the processing mechanisms of each application are not identical, they will be added to the sending queue and sent to the next hop. To simplify the analysis, the aggregation task in this paper is tantamount to transfer the packets from the receiving queue to the sending queue.

2.2.2. State space

State space represents all the state set of the sensor node in the task scheduling process. For data collection applications, the state space in this paper is described by the external environment of the sensor network and the internal sending and receiving states of the sensor nodes. We define the state of sensing area as S probe , the state of transmitting queue as S send and the state of receiving queue as S receive . S probe reflects the uncertain external environment. S send and S receive reflect the internal working state of the sensor node. The state space is defined as S = (S probe , S send , S receive) . S probe ∈ [0, 1] represents whether the agent needs sensing task and collecting packets or not. S send ∈ [0, 1] represents whether the transmitting queue is empty or not. S receive ∈ [0, 1] represents whether the receiving queue is empty or not. At time step t , the state of the ith sensor node meets the condition of s i t ∈ S.

III. SOFTWARE AND HARDWARE REQUIREMN

1.Hardware Requirements:

- Processor :Pentium IV
- Hard Disk :200GB or more
- RAM :4GB or more

2.Software Requirements:

- Operating System :Windows (Any version)
- User Interface :ns2
- Programming Language : Tclscripting,python3.0

IV. SYSTEM DESIGN

As we discussed the 3 phases are shown with source , destination and base station where we can send the packets by energy efficient nodes by queue system

Level 0:

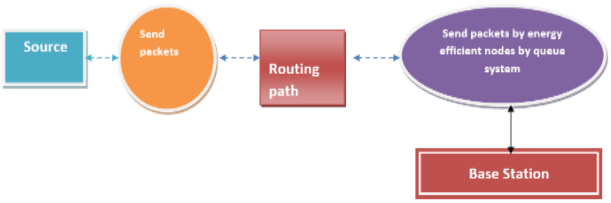
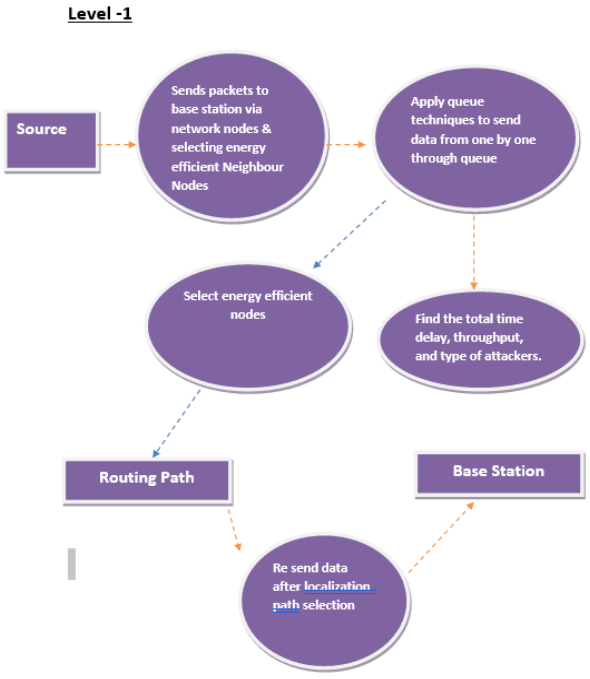
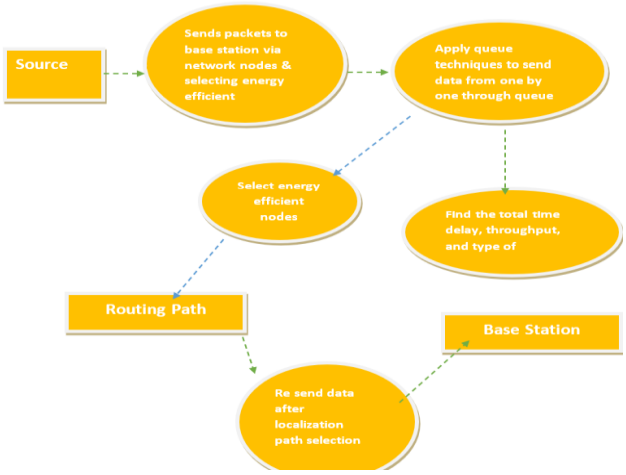


Fig:3 Data Flow Diagram.

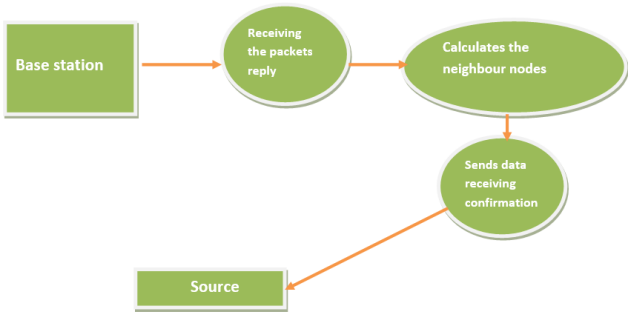
Level-1



Level-2



Level 3:



V. SYSTEM STUDY

1. Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

1.1.Economical Feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

1.2.Technical Feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

1.3.Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

VI.SYSTEM IMPLEMENTATION

1. Required simulator: NS2(NETWORK SIMULATOR)

1.1 Introduction to NS2

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviours. NS2 is an object-oriented simulator written in OTcl and C++ languages. While OTcl acts as the frontend (i.e., user interface), C++ acts as the backend running the actual simulation.

Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator, the foundation which NS is based on. Since 1995 the Defense Advanced Research Projects Agency) supported development of NS through the Virtual Internetwork Testbed project currently the National

Science Foundation has joined the ride in development. Last but not the least, the group of researchers and developers in the community are constantly working to keep NS2 strong and versatile.

1.2. Basic Architecture of NS2

Figure shows the basic architecture of NS2. NS2 provides users with executable command ns.
which take on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (Otcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles.

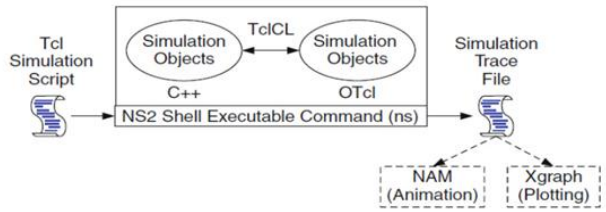


Fig 4 Basic Architecture Of Ns2.

Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g., _o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may define its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own C++ objects, and use an OTcl configuration interface to put together these objects. After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behavior of the network, users can extract a relevant subset of textbased data and transform it to a more conceivable presentation.

1.3. Two-Language Concept in NS2

Why two languages? Loosely speaking, NS2 uses OTcl to create and configure a network, and uses C++ to run simulation. All C++ codes need to be compiled and linked to create an executable file. Since the body of NS2 is fairly large, the compilation time is not negligible. A typical Pentium 4 computer requires few seconds (long enough to annoy most programmers) to compile and link the codes with a small change such as including “int i=0;” into the codes.
OTcl, on the other hand, is an interpreter, not a compiler. Any change in aOTcl file does not need compilation. Nevertheless, since OTcl does not convert all the codes into machine language, each line needs more execution time. In summary, C++ is fast to run but slow to change. It is suitable for running a large simulation. OTcl, on the other hand, is slow to run but fast to change. It is therefore suitable to run a small simulation over several repetitions (each may have different parameters). NS2 is constructed by combining the advantages of these two languages.

NS2 manual provides the following guidelines to choose a coding language:
• Use OTcl
• For configuration, setup, or one time simulation, or
• To run simulation with existing NS2 modules.
This option is preferable for most beginners, since it does not involve complicated internal
mechanism of NS2. Unfortunately, existing NS2 modules are fairly limited. This option is perhaps not sufficient for most researchers.

2. Algorithm for SVM-Q

- Algorithm 1 ISVM-Q.
- initialization: Q-learning controller and ISVM model parameters;
- repeat
- Detect the current system state s ;
- Construct the training sample of ISVM, obtain the optimal solution α∗from the optimization problem;
- Obtain the parameters ω and b of the regression model;
- Obtain the Q value for the selected action , and then select the action a t corresponding to the current value Q k according to the action selection strategy;
- Execute the action a t then get the state s t+1 in next time step and the immediate reward r t ;
- Obtain the target Q t value at time t;
- if (x t , Q t) not satisfied the KKT condition then
- Update the sliding window to retrain the model;
- end if 12: t = t + 1 ; 13: until Learning Condition is achieved

3.Karush-Kuhn-Tucker (KKT) Conditions:

The necessary conditions for a constrained local optimum are called the Karush Kuhn Tucker (KKT) Conditions, and these conditions play a very important role in constrained optimization theory and algorithm development

1. Feasible Constraints
gi(x∗)−biisfeasiblegi(x∗)−biisfeasible
2. No Feasible Descent
∇f(x∗)−m∑i=1λ∗i∇gi(x∗)=0∇f(x∗)−∑i=1mλi∗∇gi(x∗)=0
3. Complementary Slackness
λ∗i(gi(x∗)−bi)=0λi∗(gi(x∗)−bi)=0
4. Positive Lagrange Multipliers
λ∗i≥0λi∗≥0
The feasibility condition (1) applies to both equality and inequality constraints and is simply a statement that the constraints must not be violated at optimal conditions. The gradient condition (2) ensures that there is no feasible direction that could potentially improve the objective function. The last two conditions (3 and 4) are only required with inequality constraints and enforce a positive Lagrange multiplier when the constraint is active (=0) and a zero Lagrange multiplier when the constraint is inactive (>0).

Results:

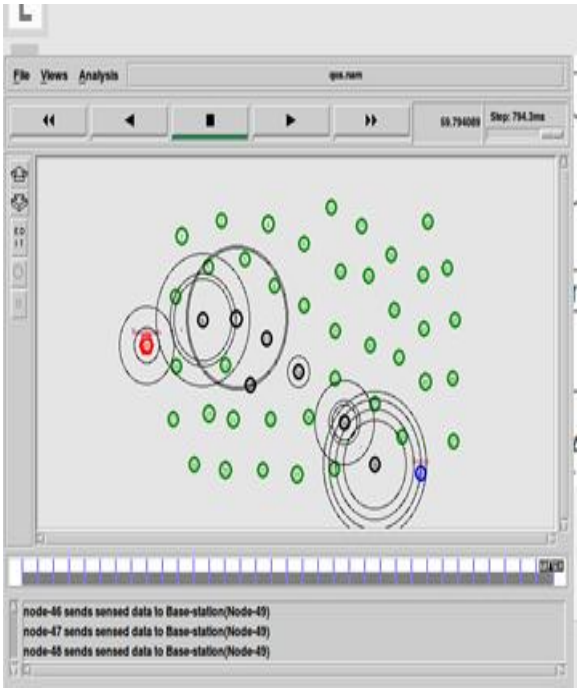


Fig 5 Simulation of nodes in NS2.

Simulated 50 nodes sensing each and every node and found the efficient route

4. Fault prediction on the network:
Used matplotlib and tensorflow libraries using a gaussian surface and found the mean, median and absolute mean deviation with the gaussian efficiency of 37% and scale estimator of gaussian efficiency 58%

Pseudocode for classification

```
def dist(x1,y1,x2,y2):
  return float(np.sqrt(((x2-x1)*(x2-x1))+((y2-y1)*(y2-y1))))

# Fixing random state for reproducibility
np.random.seed(0)

#node parameter
N =50     # total no. of nodes
n=10    # total faulty nodes in network
tr=56        # transmission range of each nodes
x1=24.4       # lower threshold
x2=26.3       # upper threshold
l1,l2=0,1000  #node lies between (0,1000)

# Sensor value assignment
sensor_val=np.array([round(random.uniform(x1,x2),2) for i in rang
e(0,N) ])

# (x,y) co-ordinate of sensor node
x = np.array([round(random.uniform(l1,l2),2) for i in range(0,N])
y =np.array([round(random.uniform(l1,l2),2) for i in range(0,N)])



distance=np.array([[0.0 for i in range(0,N)] for j in range(0,N])
no_of_neigh=np.array([0 for i in range(0,N)])
#array to store no. of neighbours of a sensor i
neigh_node_of_i=np.array([[0 for i in range(0,N)] for j in range(
0,N)])
#array to store the neighbours of sensor i
```

VII. EXPERIMENTAL ANALYSIS

1. Average delay on greedy and svm-Q:
The average delay on greedy forwarding takes more than the average delay on QOS which implies better results.

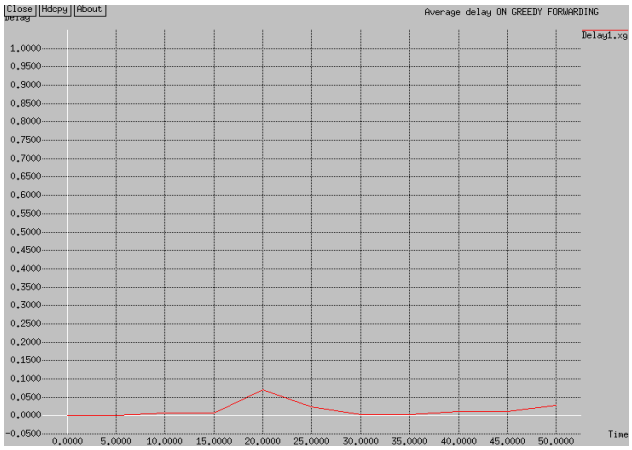


Fig 6 Average delay on greedy forwarding.

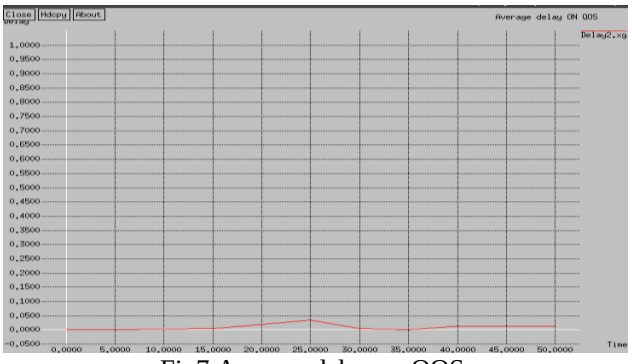


Fig7 Average delay on QOS.

2. Packet drop on svm-q and throughput on svm-q:

The packet drop initially remains same and then it increases exponentially and then suddenly decreases and then it remains same, coming to the throughput, it increases exponentially and then it remains same

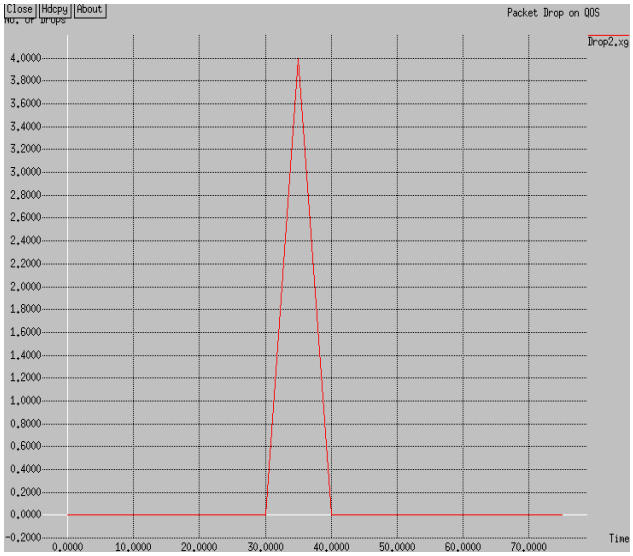


Fig 8 Packet drop on QOS.

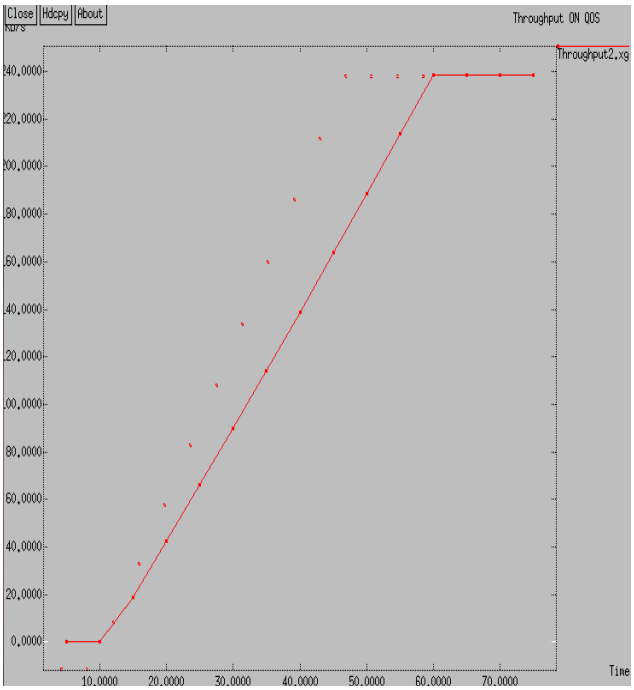


Fig 9 Throughput on QOS.

3. Energy consumption on svm-q and greedy forwarding

Here the energy consumption will be less using svm-q compared to the greedy forward method as the graph shows it is decreasing exponentially.

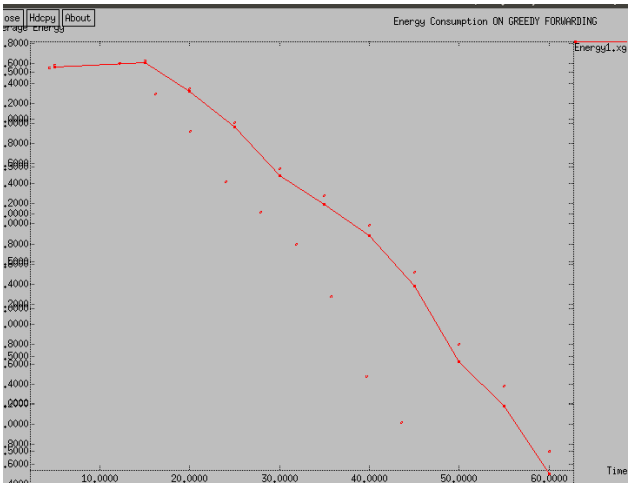


Fig 10 energy consumption on greedy.

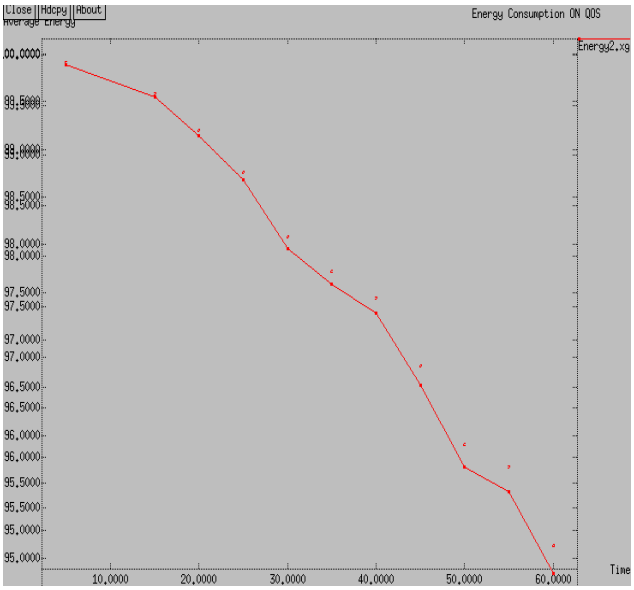
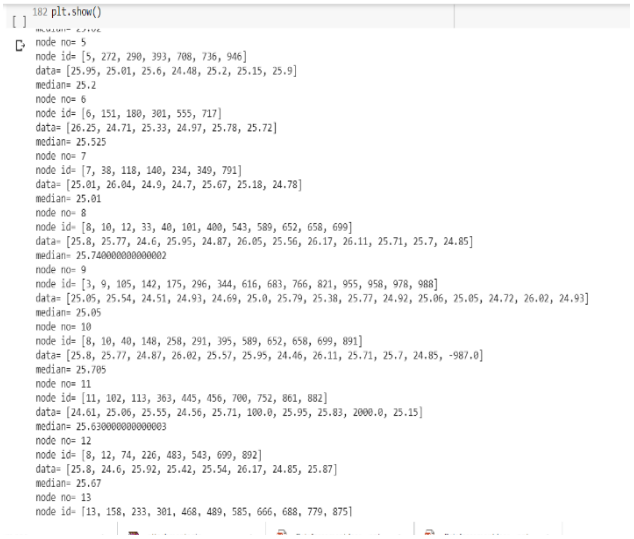


Fig 11 energy consumption on QOS.



```
182 plt.show()
[ ]
node no= 5
node id= [5, 272, 290, 393, 708, 736, 946]
data= [25.95, 25.01, 25.6, 24.48, 25.2, 25.15, 25.9]
median= 25.2
node no= 6
node id= [6, 151, 180, 301, 555, 717]
data= [26.25, 24.71, 25.33, 24.97, 25.78, 25.72]
median= 25.525
node no= 7
node id= [7, 38, 118, 140, 234, 349, 791]
data= [25.01, 26.04, 24.9, 24.7, 25.67, 25.18, 24.78]
median= 25.01
node no= 8
node id= [8, 10, 12, 33, 40, 101, 400, 543, 589, 652, 658, 699]
data= [25.8, 25.77, 24.6, 25.95, 24.87, 26.05, 25.56, 26.17, 26.11, 25.71, 25.7, 24.85]
median= 25.740000000000002
node no= 9
node id= [3, 9, 105, 142, 175, 296, 344, 616, 683, 766, 821, 955, 958, 978, 988]
data= [25.05, 25.54, 24.51, 24.93, 24.69, 25.0, 25.79, 25.38, 25.77, 24.92, 25.06, 25.05, 24.72, 26.02, 24.93]
median= 25.05
node no= 10
node id= [8, 10, 40, 148, 258, 291, 395, 589, 652, 658, 699, 891]
data= [25.8, 25.77, 24.87, 26.02, 25.57, 25.95, 24.46, 26.11, 25.71, 25.7, 24.85, -987.0]
median= 25.705
node no= 11
node id= [11, 102, 113, 363, 445, 456, 700, 752, 861, 882]
data= [24.61, 25.06, 25.55, 24.56, 25.71, 100.0, 25.95, 25.83, 2000.0, 25.15]
median= 25.630000000000003
node no= 12
node id= [8, 12, 74, 226, 483, 543, 699, 892]
data= [25.8, 24.6, 25.92, 25.42, 25.54, 26.17, 24.85, 25.87]
median= 25.67
node no= 13
node id= [13, 158, 233, 301, 468, 489, 585, 666, 688, 779, 875]
```

Fig 12 Results of fault prediction using ISVM-Q.

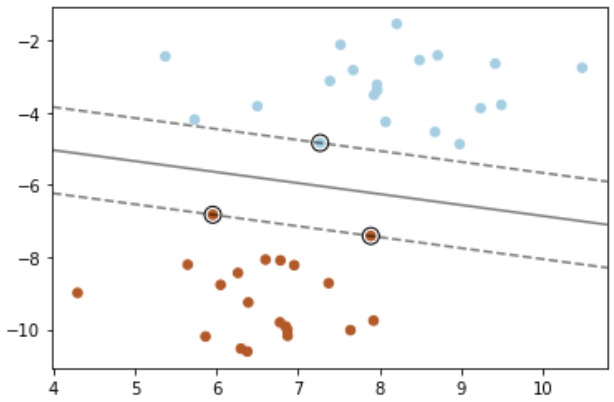


Fig 13 Using ISVM-Q classification after removing noise in gaussian surface.

VIII. CONCLUSION

- we advocate the use of Reinforcement Learning for task scheduling in WSNs. We proposed a Q-learning algorithm for Task Scheduling Based on Improved SVM (ISVM-Q) in WSNs. ISVM-Q has intrinsic support to handle dynamics and uncertainty prevailing in WSNs
- The experiments show that ISVM-Q can schedule the current task reasonably according to the environment. Compared with the traditional task scheduling algorithm, and the classic reinforcement learning scheduling algorithm, ISVM-Q can make the system obtains less energy consumption and better application performance.

Besides, ISVM-Q has better stability than BP-Q while the neural network has the same numbers of layers. It is important to emphasize that the determination of reward function needs to be based on practical application, but it is not the goal of this article to determine the reward function based on a particular adaptation of the WSN application. Note that although different learning factors and discount factors used in the classic literature are adopted in this paper, the selection of the best learning factor and the dis- count factor is not the focus of this article. Finally, although this paper verified the ISVM-Q for data collection application model has good practicability, but in fact, the data collection application of WSN just is a typical example to the algorithm. Fault prediction is improved using ISVM-Q classification

REFERENCES

[1] I.F. Akyildiz, W.Su, Y. Sankarasubramaniam, and E. Cayirci; ―Wireless sensor networks: a survey‖, Computer Networks 38 (2002) 393–422, 2002 Elsevier Science B.V

[2] The SmartDetect Project Team, Indian Institute of Science, Bangalore, India; ―Wireless Sensor Networks for Human Intruder Detection‖ Report, May 2010.

[3] Tian He, Sudha Krishnamurthy, John A. Stankovic, etc. ―VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance‖

[4] Anna F¨orster, ―Machine Learning Techniques Applied to Wireless Ad-Hoc Networks: ‖ (2017) 244–258.

[5] Ma Di, Er Meng Joo, ―A Survey of Machine Learning in Wireless Sensor Networks - From Networking and Application Perspectives

[6] Z.X. Lu, W.W. Li , M. Pan ,Maximum lifetime scheduling for target coverage and data collection in wireless sensor networks, IEEE Trans. Veh. Technol. 64 (2) (2015) 714–727 .

[7] M.I. Khan, B. Rinner, Performance analysis of resource-aware task schedul- ing methods in wireless sensor networks, Int. J. Distrib. Sens.Netw. 2014 (10) (2014) 1155–1167.

[8] Z.C. Wei , Y. Zhang , X.W. Xu , A task scheduling algorithm based on q-learning and shared value function for WSNs, Comput. Netw. 126 (1) (2017) 141–149 .

[9] M.I. Khan, K. Xia, A. Ali , N. Aslam , Energy-aware task scheduling by a true online reinforcement learning in wireless sensor networks, IJSNet 25 (4) (2017) 244–258 .

[10] K. Shan, M. Kumar, Distributed independent reinforcement learning (DIRL) ap- proach to resource management in wireless sensor networks, in: IEEE Interna- tional Conference on Mobile Adhoc and Sensor Systems, 2008, pp. 1–9 .

[11] M.I. Khan, B. Rinner, C.S. Regazzoni, Resource coordination in wireless sensor networks by combinatorial auction based method, in: IEEE International Con- ference on Networked Embedded Systems for Every Application, 2013, pp. 1–6 .

[12] M.I. Khan, B. Rinner, Energy-aware task scheduling in wireless sensor networks based on cooperative reinforcement learning, in: IEEE International Conference on Communications Workshops (ICC), 2014, pp. 871–877.

[13] M.I. Khan, K.B. Rinner, Resource coordination in wireless sensor networks based on cooperative reinforcement learning, in: IEEE International Conference on Pervasive Computing and Communications Workshops, 2012, pp. 895–900 .

[14] S. Satyanarayana, P.S. Kumar, G. Sridevi,Improved process scheduling in real–time operating systems using support vector machines, in: International Con- ference on Micro-Electronics, Electromagnetics and Telecommunications, 2018, pp. 603–611 .

[15] H.Z. Tong , M.K. Ng , Calibration of insensitive loss in support vector machines regression, J. Frankl. Inst. 356 (4) (2019) 2111–2129 .