

Distributed Genetic Algorithm for Energy-Efficient Resource Management in Sensor Networks

Qinru Qiu Qing Wu
Department of Electrical and Computer Engineering
Binghamton University
Binghamton, NY 13902
001-607-777-4918, 001-607-777-4536
{qqiu, qwu}@binghamton.edu

Daniel Burns Douglas Holzhauer
Air Force Research Laboratory, Rome Site
26 Electronic Parkway
Rome, NY 13441
001-315-330-2335, 001-315-330-4920
{Daniel.Burns, Douglas.Holzhauer}@rl.af.mil

ABSTRACT

In this work we consider energy-efficient resource management in an environment monitoring and hazard detection sensor network. Our goal is to allocate different detection methods to different sensor nodes in the way such that the required detection probability can be achieved while the network lifetime is maximized. The optimization algorithm is designed based on the Island multi-deme genetic algorithm (GA). The experimental results show that our algorithm increases the network lifetime by approximately 14.4% in average compared with the heuristic approaches. We also investigate the effect of the configuration parameters on the searching quality of the proposed distributed GA. A regression model is derived empirically that estimates the runtime of the distributed GA given the configuration parameters such as the sub-population size, parallelism, and migration rate. Once the model has been fit to a group of data, it can be utilized to find the efficient configurations of the proposed algorithm.

Categories and Subject Descriptors

J.7 [Computer Applications]: Sensors and Sensor Networks

General Terms

Experimentation

Keywords

Distributed Genetic Algorithm, Sensor Network, Energy Aware Design, Resource Management

1. INTRODUCTION

In this paper we focus on the management of the energy resource in an environment monitoring sensor network. A set of data acquisition and signal processing applications is available on each node. They provide the tradeoffs between detection quality and resource utilization. The resource management for the system is to determine which sensor nodes should be turned on to process which data acquisition and signal processing application such that the network lifetime can be maximized while meeting the required detection accuracy. This general assignment problem has been proven to be NP-complete [1]. The contribution of this paper is twofold. Firstly, the resource management problem is formulated as a constraint optimization problem and is solved using a distributed GA [2]. Secondly, empirical analysis results are provided that reveals the relationship between the configuration parameters, (i.e. the population size, the migration rate, and the parallelism) and the

quality of the search. A regression model is presented that estimates the runtime of the distributed GA given the configuration parameters. This model can be utilized to find efficient and practical configurations of the algorithm.

2. RESOURCE MANAGEMENT USING DISTRIBUTED GA

We consider the clustered sensor network that is deployed with a certain level of redundancy. Each cluster consists of p sensor nodes. Each sensor node is low cost and low quality however combined together they provide very accurate detection. The nodes in the same cluster have direct wireless communication with each other. A sensor node has n different detection methods, which will also be referred as *task* in the rest of the paper. Each task-processor pair (i, k) , $1 \leq i \leq n$ and $1 \leq k \leq p$, associates with two variables $pow_{i,k}$ and $prob_{i,k}$ which represent the power consumption and the detection probability of task i when it is running on processor k . The $prob_{i,k}$ is a function of the location and the environment of the sensor node. We assume that this function is pre-calibrated and evaluated during the runtime. The detection probability $Prob$ of a cluster with p nodes is calculated as $Prob = 1 - \prod_{1 \leq k \leq p} \prod_{i \in \Delta_k} (1 - prob_{i,k})$, where Δ_k is the set of tasks that are allocated to node k . The goal of resource management is to find the Δ_k for each processor k so that the combined detection probability of the cluster is larger than the user defined constraint while the network lifetime is maximized. In this work, we define the network lifetime as the time from the deployment of the sensor network to the time when the first node runs out of battery energy.

The Island multi-deme GA is used to optimize the resource management for a cluster of sensor nodes. Each individual solution is a chromosome with n symbols. If the j th task is allocated to node x then the j th entry of the chromosome is equal to x . If the j th entry of the chromosome is -1 then this task is not allocated to any of the processors. The fitness function is the expected lifetime of a cluster if the solution provides the required detection accuracy, otherwise the fitness is 0. Single point crossover mating function is used in our experiment. The mutation probability is set to 1%, and involves flipping bits in integer representations of the parameters stored in chromosomes.

The GA is running on np processors. Each sub-population is initialized randomly and its size is denoted as pop . The sub-population evolves independently for c generations, and then 5 of the best individuals are broadcast to all other processors. The three parameters, np , pop and c , will be referred as the *configuration parameters* in the rest of this paper.

3. CONFIGURATION PARAMETERS

Two sets of experiments have been carried out and the relation between the configuration parameters and the quality of search is derived empirically. We assume that there are 100 tasks and 10 sensor nodes in the cluster. The GA is running on np sensor nodes with $np \leq 10$. The detection probability and the power consumption of each task are uniformly distributed random variables whose range is 1% ~ 25% and 0.1Watt ~ 10Watt respectively. The battery of each sensor has the capacity of 5000 Ampere-hour. Because GA is a stochastic algorithm, we run each simulation 50 times and report the mean value.

The first set of experiments is designed to find out the effect of the configuration parameters on the quality of the solution. We swept the np from 2 to 8, the pop from 25 to 350, and the c from 1 to 35. For each configuration, the distributed GA is simulated. The GA will stop when the fitness of the best individual does not improve for 2000 generations. Our results show that increasing both the pop and the np improves the quality of the solution. However, varying c has very little impact on it.

The second set of experiments is designed to find out the effect of the configuration parameters on the runtime of the GA. The value of np , pop and c are swept in the same way as the first set of experiments. The GA stops when the fitness of the best individual exceeds the threshold which is set to be 5 times of the expected fitness of a random individual. The number of generations that the GA has iterated is reported. The following trends are shown in our experiments. 1) When the size of the sub-population increases, the runtime increases linearly. Combined with the results from the first experiment we can see that a large population should be used to find the best possible solution, while a small population should be used to find a good solution in a short time. 2) Reducing the migration rate will result an almost linear increase in solution time. The slope is the same for different sub population size. 3) Increasing the number of processors will reduce runtime, and this effect is more dominant when the sub-population is small.

In order to consider the combined effect of all of the three configuration parameters, we introduce a new variable called *effective population* ($Epop$). Its size is equal to $Epop = pop + pop \cdot (np - 1) / c$. Let G denote the number of generations that the GA has iterated before it finds the solution with the required fitness. We observed that G is a continuous and differentiable function of $Epop$. The following model is developed to predict the value of G $G = a + a_0 \sqrt{Epop} + \sum_{i=1}^5 a_i / Epop^i$. The coefficients a, a_0, \dots, a_5 are obtained using regression analysis. This model provides very accurate prediction of runtime of the GA.

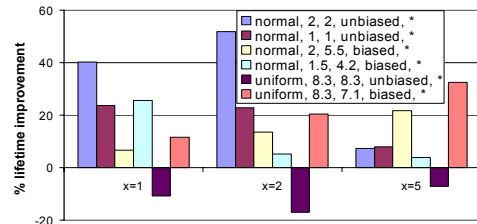
4. EXPERIMENTAL RESULTS

Experiments are setup to evaluate the performance of the GA based resource management scheme. The cluster has the same configuration as described in section 3. The detection probability and the power consumption of each task are randomly generated. Different distributions with different variances are tested in the experiment. Furthermore, to emulate the behavior of the real sensor network which is deployed in a dynamic environment, the detection probability of the sensors is constantly changing. Every 1000 hours, for a set of x sensor nodes, their detection probability $prob_i, 1 \leq i \leq$

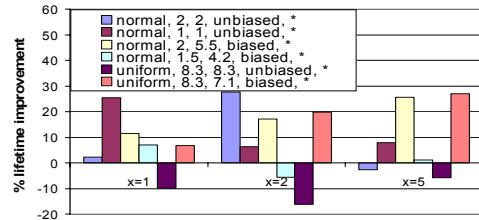
100 will be regenerated and reapplied to model the change of their environment.

The environment setup is named by a quintuplet (*distribution, prob variance, power variance, biased/unbiased, x*). The first field specifies the type of distribution that is used to generate the detection probability and power consumption of each task. The second and third field specifies the variance of the detection probability and the power consumption respectively. The fourth field is either biased or unbiased. When an environment setup is biased, half of the sensor nodes have lower power consumption than the others. The final field specifies the value of x .

Our distributed GA algorithm is denoted as *GA-lifetime*. The configuration parameters np, pop and c are equal to 5, 100 and 5 respectively. We designed two algorithms to compare with the *GA-lifetime*. The first one is also a distributed GA whose objective is to minimize the total power consumption of the cluster and it is denoted as *GA-power*. The second one is a heuristic algorithm which selects and allocates task based on the power versus detection probability ratio. For all of the algorithms, the detection accuracy constraint is set to 99.9%. Figure 1 shows the percent lifetime improvement of *GA-lifetime* relative to the heuristic algorithm and *GA-power*. Due to the space limitation, the average improvement for different x is reported in the chart.



(a) GA-lifetime outperforms heuristic algorithm by 14% on average



(b) GA-lifetime outperforms GA-power by 6.5% on average

Figure 1 GA-lifetime vs. heuristic algorithm

5. REFERENCES

- [1] H. Feltl and G. R. Raidl, "Evolutionary computation and optimization (ECO): An improved hybrid genetic algorithm for the generalized assignment problem," *Proceedings of the 2004 ACM symposium on Applied computing*, March 2004.
- [2] E. Cantu-Paz, "A Survey of Parallel Genetic Algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, Vol. 10, No. 2.