Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 467497, 9 pages
doi:10.1155/2012/467497

*Research Article*

# DI-GEP: A New Lifetime Extending Algorithm for Target Tracking in Wireless Sensor Networks

**Shucheng Dai,[1] Chuan Li,[1] and Chun Chen[2]**

[1] *College of Computer Science, Sichuan University, Sichuan 61065, China*
[2] *Business School, Sichuan Normal University, Sichuan 61065, China*

Correspondence should be addressed to Chuan Li, lcharles@scu.edu.cn

Received 25 May 2011; Accepted 4 January 2012

Academic Editor: Bahram Honary



Wireless sensor networks (WSNs) are widely used in detecting, locating, and tracking moving objects. The cheap, low-powered, and energy-limited sensors that are set up in large areas may consume large portion of energy and disable the whole network. In this paper, a new energy-efficient method based on Distributed Incremental Gene Expression Programming (DI-GEP) is proposed to collaboratively mine moving patterns of moving targets in order to turn on/off some sensor nodes at certain time to save energy further. Meanwhile, an adjustable sliding window is designed to quickly train the latest collected location data in order to improve the efficiency of DI-GEP. The simulation results show that the proposed method effectively prolongs the network lifetime by around 25% compared with the EKF and ECPA.

## 1. Introduction

Recent advances in low-power micro-electro-mechanical system (MEMS) technology, wireless communications, and digital electronics have made it possible to design and develop highly integrated, yet low-cost, low-power, multifunctional microsensor nodes, with the capabilities of sensing, processing, and wireless communications. Once deployed in a certain region, the wireless sensor networks (WSN), composed of thousands of sensor nodes, can work for several years. Through cooperative processing of these sensor nodes, WSNs work in many areas, for example, civil, military, health, and so on. For example, WSNs can be deployed in a hospital to track and monitor patients to remotely collect the physiological data of a patient continuously. Unlike traditional networks, WSNs are self-organized, application specific, and data centric [1].

A wireless sensor node is typically battery operated. Thus, the most important constraint in WSNs is the low energy consumption requirement among their sensor nodes. Sensor nodes carry limited, generally irreplaceable, battery-power sources. So, WSNs must focus primarily on power conservation and provide inbuilt trade-off mechanisms that give end users the chance of prolonging network lifetime at the cost of high quality of service (QoS). Sensor nodes may fail due to energy depletion and lead to network failure. So it is very important for WSN to operate energy efficiently. Raising research interest promotes us to develop energy-efficient protocols or algorithms for WSNs.

Target tracking is an important application in terms of WSNs [2]. Bayesian Network and Kalman filtering are two classical methods for achieving this task. One possible solution is as follows. The system state includes the position, direction, and velocity of the target. At each step, sensors near to the target form clusters and select a leader to perform the Kalman filtering, and the updated state is forwarded to cluster leaders chosen from the next step. The Kalman filtering implementation is straightforward in a centralized environment. But it is difficult in the extremely distributed environments such as WSNs due to the energy constraints and lower computation capability of sensor nodes.

The target tracking applications in WSNs are always limited by the inherent energy constraints of sensor nodes, aiming to improve the energy efficiency in target tracking applications in WSNs; the paper proposes a new scheme based on Gene Expression Programming (GEP) [3]. GEP is also adapted to fit for distributed environment. GEP works well in modeling the moving patterns of targets without

aprior knowledge. Based on the historical location information of the target, GEP automatically evolves a trajectory of a moving object. To handle the problem, this paper makes the following contributions.

(1) A new algorithm named Distributed Incremental Gene Expression Programming (DI-GEP) is proposed to mine moving patterns of targets. The basic idea is that DI-GEP runs at multiple collaboratively working sensor nodes to mine the trajectory of a target.

(2) An adjustable sliding window is adopted to ensure that distributed GEP can quickly train the latest collected location data. When new location data are received, old location data are discarded when prediction error exceeds a certain threshold, which is defined and can be calculated by (7). The policy ensures that succeeding evolutions can energy efficiently find latest moving patterns.

(3) Extensive simulations are conducted on OMNet++, a discrete event simulator, to show that new algorithms effectively prolong the network lifetime by about 25% in average when compared to other algorithms, that is, EKF and ECPA.

The rest of the paper is organized as follows. Section 2 presents the related work on energy-saving algorithms. Section 3 gives the preliminaries including GEP-related and target tracking. Section 4 introduces the target diction model. Section 5 formally defines the problems. Section 6 proposes the main algorithms in our scheme. Section 7 gives the experimental analysis. Section 8 concludes this paper and gives the future research directions.

## 2. Related Work

There are many research efforts on target detection and tracking in terms of WSNs, which describes several aspects of collaborative signal processing [2, 4, 5] and real-time application for biologists to find the presence of individuals [6]. A set of approaches presented in [7–9] were proposed recently to solve the target localization and tracking problem with proximity binary sensors, which transmit only one bit information to indicate whether a target is present. The information transmitted among sensor nodes was greatly reduced, while the localization error was increased. Shrivastava et al. [8] proved that the accuracy in tracking a target is of the order of $\rho * R$, where $R$ is the sensing radius and $\rho$ is the sensor density, which articulates the common intuition that, for a fixed sensing radius, the accuracy improves linearly with an increasing sensor density, which shows that, for a fixed number of sensors, the accuracy improves linearly with an increase in the sensing radius. Dai et al. present a light weight target tracking method based on densely distributed sensor networks [10, 11] and also propose a new node deployment policy for target tracing applications in WSNs [12] to further improve target tracking performance and quality including accuracy, network lifetime, energy consumption level, trends analysis, and so forth.

The lifetime of a WSN depends greatly on power consumption from each sensor node. Energy-efficient algorithms, protocols, and node hardware and software designing technologies can help prolong the lifetime of the network. Several approaches have been proposed at hardware and software levels to design energy efficient CPU, OS, algorithms, and communication protocols [1]. Dynamic power management (DPM) schemes have been proposed in [13–15] to reduce the power consumption by selectively turning off idle components, such as radio frequency (RF) transmitter, RF receiver, sensing device, A/D converter, and the sensor node.

Target tracking applications are special and have their own characteristics. It is unnecessary to turn on all sensor nodes because an object only appears at certain time and place. It is feasible to turn off some idle nodes if we can predict the time and place where the object will appear. Classical target tracking algorithms such as Bayesian Network and Kalman filtering cannot be directly used to predict moving patterns of targets in WSNs due to resource limitations.

To track moving targets energy efficiently, Allegretti et al. [16] proposed a solution based on CA (Cellular Automata) to reduce long distance communications among nodes because of its locally data exchanging scheme, but a higher power consumption is introduced because it cannot turn off those nodes that are far away from the moving object. Qing et al. [17] proposed ECPA (Enhanced Closest Point Approach) to predict the location of targets during the phase of moving, but the velocity and direction calculation algorithms with regard to the targets are computation intensive for sensor nodes, which often have low power and computation capability.

## 3. Preliminary

*3.1. Introduction of Gene Expression Programming.* Gene Expression Programming (GEP) was proposed by Ferreira in 2006 [3]. As a new member of Evolutionary Algorithm (EA) family, GEP is widely applied in data mining areas, that is, function finding, classification, association rule mining, time series prediction, parameter optimization, and digital circuit design, and so forth. In GEP, Genotype (Chromosome) and Phenotype (Expression Tree (ET)) are separated. Without prior knowledge, GEP automatically evolves over training data and discovers knowledge as mathematical formula depicting movement patterns of moving objects in WSNs.

In GEP, an individual, that is, a solution corresponding to a problem, is represented as linear fixed-length string named chromosome. It contains one or more genes. Each gene is decoded into a nonlinear expression tree (ET). Decoded ETs are linked together by prespecified linking function symbols such as plus (+) and minus (−). One chromosome represents one formula that is, the solution to a specific problem. Genetic operations are applied on chromosomes, that is, genotype and selection operations are performed on ETs.

A gene in GEP consists of head and tail. The head contains symbols from either function symbol set ($F$) or terminal symbol set ($T$) and the tail only contains symbols

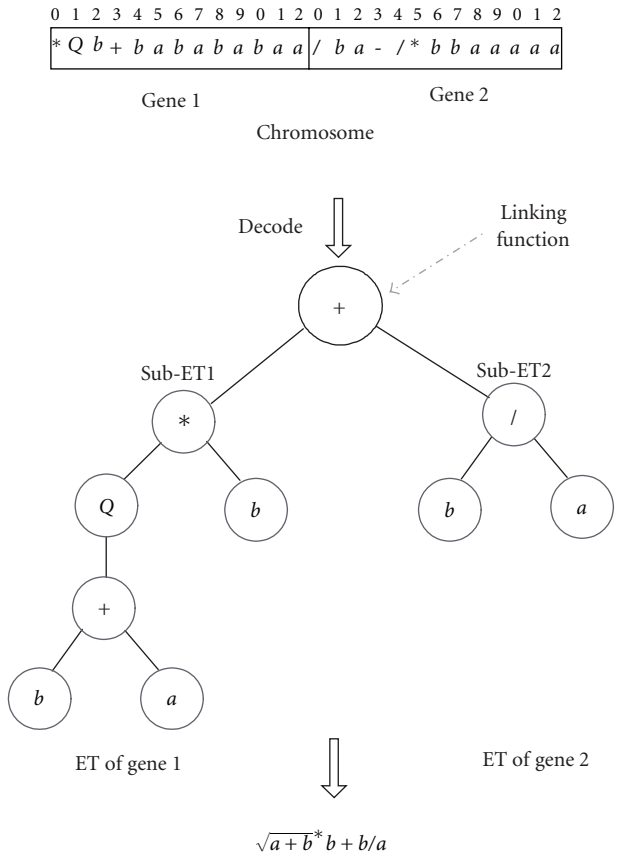


Figure 1: Individual representation in GEP: 2-gene chromosome, 2 expression trees, and corresponding one mathematical expression.

from terminal symbol set. The tail length satisfies (1), which guarantees that a gene can be decoded into a valid ET

$$t = h(n-1)+1. \tag{1}$$

In (1), $n$ is the maximum parametric number of function in $F$, $h$ is head length, and $t$ is tail length. Example 1 below shows a 2-gene chromosome in GEP and its ETs.

*Example 1.* Let $C_1$ be a chromosome with 2 genes in GEP. Let $\{Q, *, /, -, +\}$ be the function set $F$, and $T = \{a, b\}$ the terminal set, $n = 2$. If head length is 6, then tail length is 7 by (1). The length of gene is 13 (the sum of head length and tail length). Figure 1 shows the 2-gene chromosome, two sub-ETs, which are linked by the linking function "plus," denoted as "+" and the mathematical function obtained from the linked expression tree.

*3.2. Target Tracking in WSNs.* There are several target tracking methods that adopt distinct models. Kalman filtering method requires that the velocity, direction, and acceleration of a moving object are given to predict the next location of a moving object. But it is impossible for WSNs to equip each node with these devices due to its lower cost and lower power supply. It is critically important to design energy-efficient target tracking algorithms for WSNs.

The tracking model is described in Figure 2. The monitored region is covered with sensor nodes that are distributed manually or randomly. A target moves along a trajectory that is unknown before and is detected by some sensor nodes that are depicted as black solid circle nodes. Suppose that the location data the moving object passed are known at time $t_0, t_1, t_2, \ldots, t_i$. A question arises: where will it appear at time $t_{i+1}$, $t_{i+2}, \ldots$?

Once the trajectory of a moving object is found and expressed as a mathematical function, then the following tasks are straightforward.

(i) Predicting the locations that the moving object will appear at time $t_{i+1}$, $t_{i+2}, \ldots$.

(ii) Activating the necessary sensor nodes and turning off unrelated sensors to save energy.

Target-tracking applications need careful consideration of trade-off between tracking error and energy consumption. The tracking error is defined by the average target location estimation error of sensor nodes. The better trajectory leads to a better target location estimation.

To achieve the above goals, we propose three policies.

(1) DI-GEP, that is, a trajectory discovery algorithm based on distributed incremental gene expression programming. It mines the trajectory in order to reduce tracking errors.

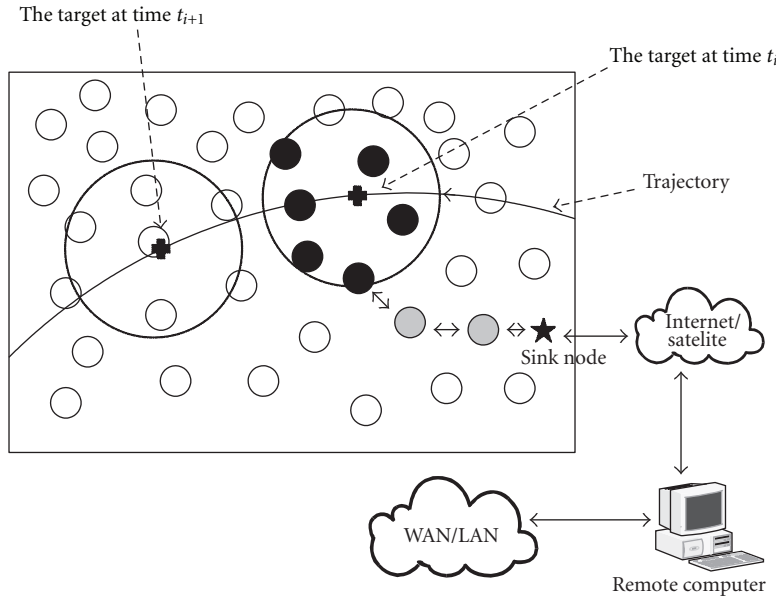


Figure 2: Target tracking model.

(2) Node scheduling algorithm. It activates the necessary nodes and turn off unrelated sensors at certain time in future to save energy.

(3) Sliding window strategy is used to improve performance of DI-GEP.

The experiments in Section 7 will demonstrate the effectiveness and efficiency of proposed methods.

## 4. Target Detection Model

Sensor nodes receive the physical signal and convert it to electrical signal. Based on the variation of electrical signal, sensors can detect the existence of target.

To describe the model formally, we make the following three assumptions.

$A_1$: There are $N$ sensors $s_i \in S$ distributed randomly or manually across the monitored areas, where $S$ is the set of all sensors. Each sensor detects targets by its reading $x_i$, where $i = 1, 2, \ldots, N$.

$A_2$: There is a single target anytime.
Note that, by Assumption $(A_2)$, a present target is depicted by $(H_1)$, and an absent target is represented by $(H_0)$. The criterion is based on the following formulations [18]:

$$\begin{aligned} H_0 &: x_i = n_i, \\ H_1 &: x_i = w_i + n_i, \end{aligned} \tag{2}$$

where $w_i$ is the obtained signal by sensor $s_i$. Several physical signals such as sound and electromagnetic wave have signal strength decaying according to the power law, and the noise is represented by $n_i$.

$A_3$ (borrowed from [19]): Le $w_t$ is the power emitted by target

$$w_i = \begin{cases} w_t, & d_i < d_0, \\ \dfrac{w_t}{(d_i/d_0)^k}, & d_i \geq d_0, \end{cases} \tag{3}$$

where $d_0$ is determined by the target shape and size which is set to be small enough and satisfies $d_i > d_0$, where $d_i$ is the distance between the target and sensor node $x_i$ and $k$ is the decaying factor which is set to 2–5 according to different physical signals and its environment.

This study adopts a practical target detection model shown in Figure 3, which satisfies

$$p_i = \begin{cases} 1, & d_i \leq r_l, \\ \left(\dfrac{d_i}{r_0}\right)^k, & r_l \leq d_i \leq r_u, \\ 0, & d_i > r_u, \end{cases} \tag{4}$$

where $r_l$ is the lower bound (LB) of sensing range, $p_i$ is the probability of a target detected by the sensor node $s_i$, and $r_u$ is the upper bound (UB) sensing range.

## 5. Problem Formulation

A trajectory of a moving object is treated as a sequence of time-stamped locations that are collected by sensor nodes around the target. It is described as follows.

*Definition 2* (Trajectory). A trajectory of a moving object is a time sequence with time interval $\Delta t$:

$$P(t) = [X(t), Y(t)] = \{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}, \tag{5}$$

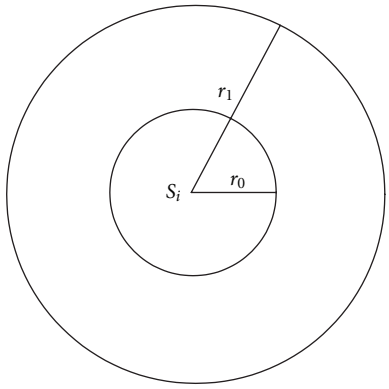


Figure 3: Target detection model.

where for all $i \in [0, n]$, $t_i < t_{i+1}$, $t_{i+1} = t_i + \Delta t$ $(x_i, y_i)$ is 2D points that represent locations of the target appeared at time $t_i$ and $x(t_i) = x_i$, $y(t_i) = y_i$, $P(i) = [x_i, y_i]$. $\Delta t$ is used to sample the locations collected by sensors to improve our algorithms energy efficiency as well as performance. Because the location data may be of large scale, which will put great burden on sensors and exhausts a great of energy because of huge amount of computations and communications.

$P(t)$ describes a varying kinds of trajectories, that is, line segments, quadric curves, cubic curves, and splines. Once $P(t)$ is obtained, it is easy to achieve single-step or multiple-step location predictions.

$P(t)$ can be obtained by trajectory mining algorithm. In terms of target tracking applications, there are several unnecessary historical location data during evolving process in distributed GEP. To deal with this problem, we adopt a sliding window prediction method (SWP) to load the latest historical data to train trajectories. The basic idea of SWP is given below.

Given the historical location data $P(0), P(1), \ldots, P(n)$ with length $n + 1$. The sliding window size is denoted as $h$ $(h \leqslant n + 1)$.

(a) Find a formula $\hat{P}(t) = [f(t), g(t)]$ from $h$ samples and predict the location at time instant $m$, $(m > n-1)$ by (6). Example 3 illustrates the phases of evolutions of trajectories.

$$\hat{P}(m) = \left[\hat{X}(m), \hat{Y}(m)\right] = [f(t_m), g(t_m)]. \quad (6)$$

(b) During evolving process, the size of sliding window $h$ determines how many historical location data are used. The smaller $h$ leads to less energy consumption and faster convergence speed. $h$ is adjusted based on the location prediction error $\varepsilon$, geometric distance between the prediction value and the real measurement. $\varepsilon$ should be as small as possible and is calculated by

$$\varepsilon = \sqrt{(\hat{x}(m) - x(m))^2 + (\hat{y}(m) - y(m))^2}. \quad (7)$$

Table 1: Location data obtained.

| $t_i$ | $x$ | $y$ |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 2 | 5 |
| 2 | 3 | 11 |
| 3 | 4 | 19 |

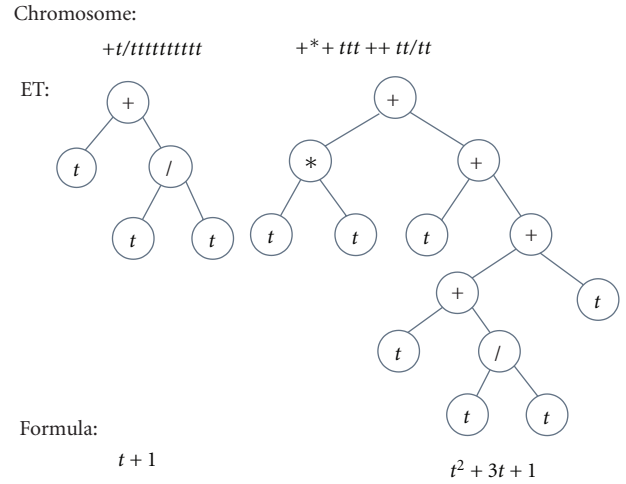


Figure 4: An example of a tracking trajectory.

In target-tracking applications, the trade-off between the energy consumption and the prediction accuracy is balanced by adjusting $\varepsilon$ to satisfy different application requirements. In densely distributed sensor networks, tracking-tolerant environment or fast response time tracking areas, $\varepsilon$ can be set to a bigger value to save energy. But, in some areas with higher tracking accuracy with slow moving targets environment, $\varepsilon$ can be set to a smaller one. In sum, $\varepsilon$ cannot be set to zero since the estimated trajectory would always deviate from the actual path targets passed.

*Example 3.* The historical locations are listed in Table 1.

Thus, $f$ may be $P(t) = [t + 1, t^2 + 3t + 0.5]$, $P(t) = [t + 2, t^2 + 2t + 1]$ or even perfect approximation $P(t) = [t + 1, t^2 + 3t + 1]$ given in Figure 4. All these approximations are suitable in environments with different prediction accuracy requirements.

## 6. DI-GEP Scheme

*6.1. Fitness Evaluation of Individual.* In evolutionary computations, fitness functions and selection environments are the two very important faces of fitness and are, therefore, intricately connected. When we speak of the fitness of an individual, on the one hand, it is always relative to a particular environment and, on the other, it is also relative to the measure (the fitness function) we are using to evaluate

them. Consequently, the success of a problem not only depends on the way the fitness function is designed but also on the quality of the selection environment [3].

Combining the fitness evaluation and prediction error, DI-GEP calculates the fitness of each individual in distinct populations by

$$E_i = \frac{1}{h}\sum_{j=1}^{h}(\varepsilon_i^2), \tag{8}$$

where $\varepsilon_i$ is the evaluation error of the $i$th location data and $E_i$ is the fitness value of the $i$th individual.

*6.2. Trajectory Mining Algorithm.* The main steps of trajectory search algorithms are given below.

(1) Sensor nodes are activated based on the node scheduling algorithm.

(2) Communications occur among sensor nodes when one node succeeds in obtaining a trajectory and notifies other nodes.

(3) Other nodes stop running their algorithms and obtain the trajectory to predict future location of the moving object.

Figure 5 details the flowchart of DI-GEP.

DI-GEP stops if one of these stopping criteria is satisfied.

(1) The maximum number of generations is reached.

(2) DI-GEP exceeds the specified runtime.

(3) One or more other nodes send stopping signal to the node.

(4) The node succeeds in obtaining a trajectory.

The implementation of DI-GEP is described in Algorithm 1. It mines a trajectory represented as individual in DI-GEP.

*6.3. Location Prediction and Node Scheduling.* Once a trajectory is found, the model uses it to predict the location where the target will appear as at time $t_{i+j}$ by (6), where $0 < j \le L$ and $L$ is the prediction length that is used in single-step or multistep predictions.

To reduce computational cost, we do not use a circle but a square to select nodes around $\hat{P}(t_{i+1})$. If sensor node $s_k(x_k, y_k)$ satisfies (9), then it should be selected and activated at time $t_{i+j}$ to detect the target

$$\begin{aligned} f\left(t_{i+j}\right) - r_0 \le x_k \le f\left(t_{i+j}\right) + r_0, \quad 0 < j \le L, \\ g\left(t_{i+j}\right) - r_0 \le y_k \le g\left(t_{i+j}\right) + r_0, \quad 0 < j \le L. \end{aligned} \tag{9}$$

The node scheduling algorithm is given in Algorithm 2.

*6.4. Incremental Evolution Strategy.* In real-world practice, a trajectory of a target is very complex and variable. Thus, to improve the performance of DI-GEP, the key steps of our policy are as follows.

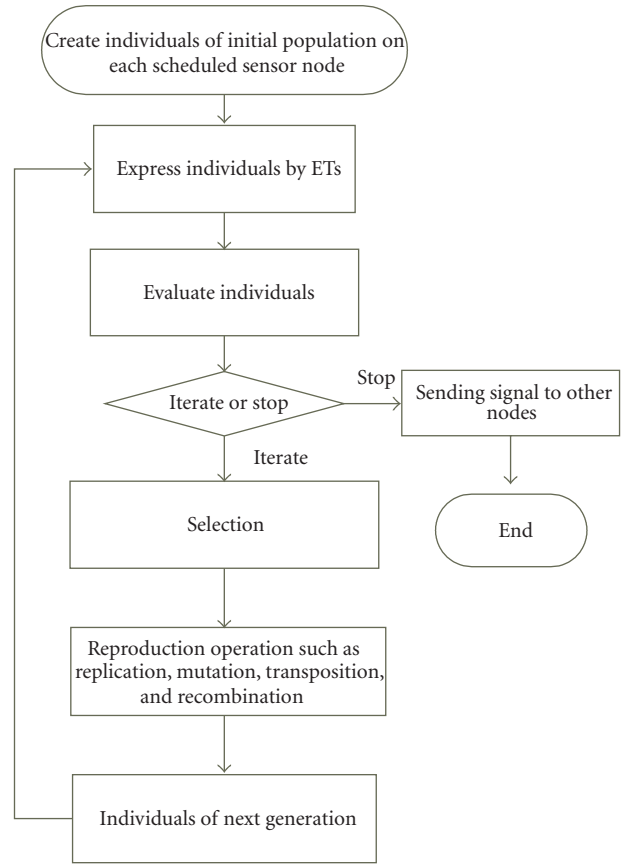


Figure 5: The workflow of distributed GEP.

**Input**: settings and historic location data
**Output**: one individual representing a trajectory or null if no trajectory is found
(1) load historic location data of size $h$ and initial configuration
(2) randomly create an initial population
(3) decode each chromosome into one ET
(4) calculate each chromosome's fitness by (8).
(5) while (stopping criteria are not satisfied){
(6) select individuals, generate next generation
(7) apply genetic operations sequentially on the new generation
(8) decode each chromosome into ET.
(9) calculate each chromosome's fitness
(10)}
(11) return an individual with best fitness.

Algorithm 1: Distributed GEP trajectory mining.

(1) Trajectory is described as a curve. Any complex curve can be spitted into multiple simpler curves that are described as line segments, quadric curves, or cubic curves. This method can not only ensure the flexibility of modeling the trajectory but also guarantee less computation cost.

(2) To capture the variation of a trajectory and accelerate the evolving process, sliding window policy is

**Input**: functions $\hat{P}(t)$ found in Algorithm 1 and prediction length $L$.
**Output**: activated sensor nodes
(1) for $(k = 0; k < L; k++)$ {
(2) for each (sensor node $s_j(x_j, y_j)$ in WSN) {
(3) if $(f(t_{i+k+1}) - R \leqslant x_j$ and $x_j \leqslant f(t_{i+k+1}) + R$
(4) and $g(t_{i+k+1}) - R \leqslant y_j$ and $y_j \leqslant g(t_{i+k+1}) + R)$
(5) $S_i$ is scheduled awake at $t_{i+k+1}$
(6) else
(7) $S_i$ is scheduled asleep at $t_{i+k+1}$
(8) }}

Algorithm 2: Node scheduling.

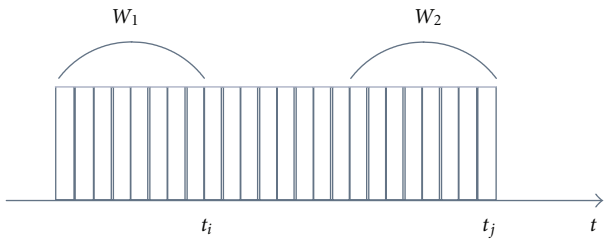


Figure 6: Sliding window.

proposed to keep a certain number of latest historical data during individual evolving.

(3) When the obtained function cannot represent the current moving behaviors, that is, the prediction error is greater than a certain threshold; distributed GEP and the node scheduling algorithm should run again with location data in sliding window.

We assume that a trajectory function $P(t)$ is obtained through historical location data sampled at $t_i, t_{i-1}, \ldots, t_{i-h-1}$. These data fall in the sliding window $w_1$ described in Figure 6.

$P(t)$ works well in predicting the future location during the time interval $[t_{i+1}, t_{j-1}]$, that is, $\varepsilon \leq \tau$. Suppose that at time $t_j$, $P(t)$ does not work well because $\varepsilon > \tau$, the trajectory should be recalculated as follows.

(1) The sliding window moves to $w_2$ to include the latest location data. The process can be simplified by $w_1$ sliding right when prediction is performed to reduce memory usage.

(2) DI-GEP and the node scheduling algorithm run again.

The performance of these algorithms is evaluated on OMNet ++ and Castalia.

In order to compare the performance with other target tracking algorithms, we evaluate DI-GEP, ECPA, and extended Kalman filtering (EKF). The target cannot be found until the network fails, and the tracking task stops simultaneously.

Table 2: Parameters setting in DI-GEP.

| Parameters | values |
|---|---|
| Function set | $\{+, -, *, /\}$ |
| Basic terminal set | $\{t\}$ |
| Number of generations | 1000 |
| Population size | 100 |
| Head length | 7 |
| Mutation rate | 0.044 |
| Inversion rate | 0.1 |
| IS rate | 0.1 |
| RIS rate | 0.1 |
| Length of insertion sequence | $\{1, 2, 3\}$ |
| One-point recombination rate | 0.3 |
| Two-point recombination rate | 0.3 |
| Gene recombination rate | 0.1 |

## 7. Experiments

*7.1. Sensor Networks Setting.* Suppose that hundreds of sensor nodes are uniformly distributed in a square of $100 \times 100$ meters. A target can present at a random place in the WSNs and sends signal with the strength of $w_t$. The signal decays according to (3), with parameters $d_0 = 0.15$ and $w_t = 15d_0^{-k}$. The prediction accuracy $\tau$ is 2 meters and $h = 7$.

The sensing range $r_o$ of sensor nodes is set to 7 and $r_1$ is set to 10. The sensing energy $e_s$ is 100 uJ, and transmission (receiving and sending) energy for one packet is $e_t = e_r = 100$ uJ. The initial energy of each sensor node is 100 mJ. Parameters used in distributed GEP are listed in Table 2.

*7.2. Network Lifetime.* Suppose that different number of sensor nodes are uniformly distributed in the grid network, this experiment analyses the impact of the number of sensor nodes on the network lifetime. The results are given in Figure 7. It shows that the three algorithms often obtain longer network lifetime when the number of sensor nodes gets bigger. The network lifetime of DI-GEP is averagely 35% longer than EKF and ECPA.

*7.3. Energy Consumption.* In this experiment, four hundred sensor nodes are used to monitor the grid network. The sensor nodes are manually distributed at cross points in the grid network.

This experiment analyzes the energy consumption of three algorithms. The results are given in Figure 8. The results show that the total left energy decreases when the time passes. The total left energy cannot be zero because all these algorithms are invalid when the network fails. Note that, some sensor nodes consume their energy and cannot communicate with other nodes any more. Meanwhile, DI-GEP performs better than EKF and ECPA. This is because it consumes less energy than EKF and ECPA after running the same time.

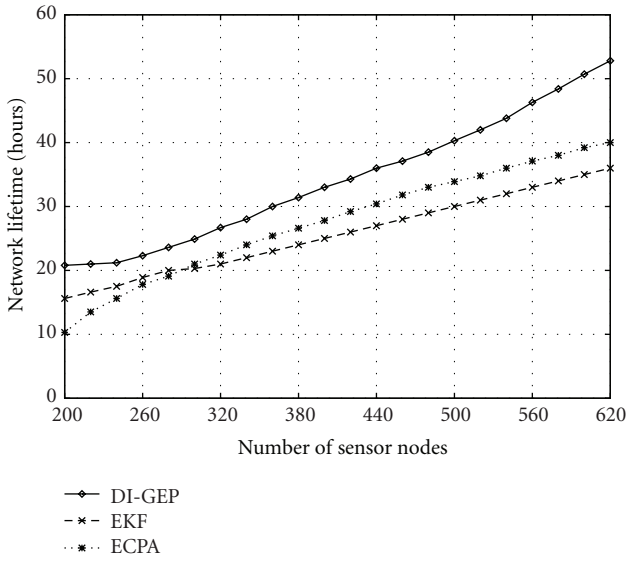


Figure 7: Network lifetime and number of sensor nodes.

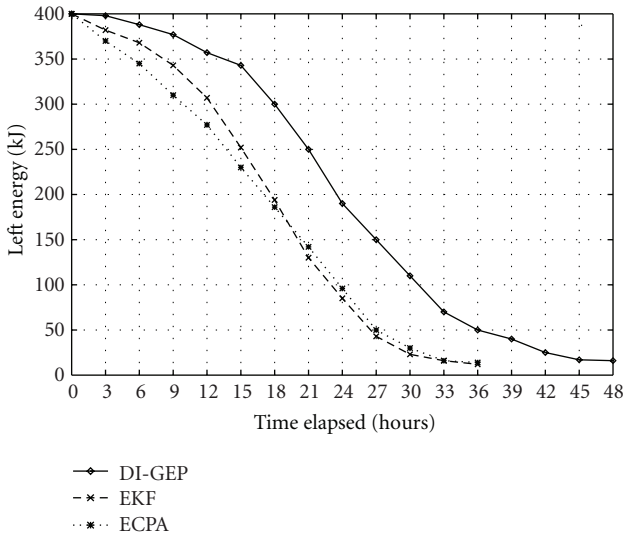


Figure 8: Energy consumption and time elapsed.

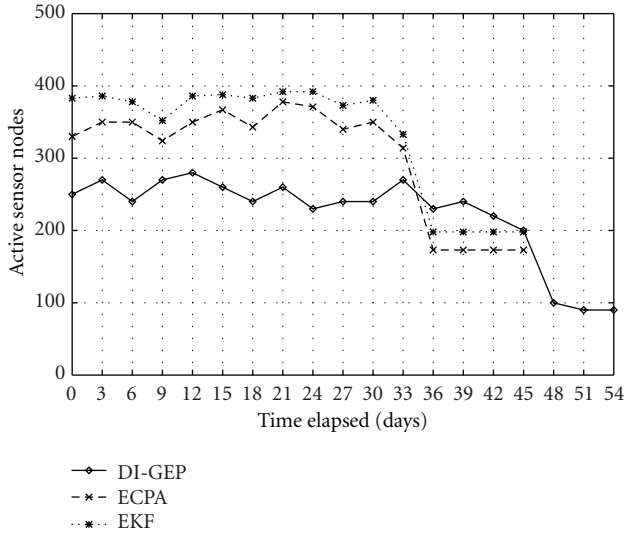


Figure 9: Active nodes and time elapsed.

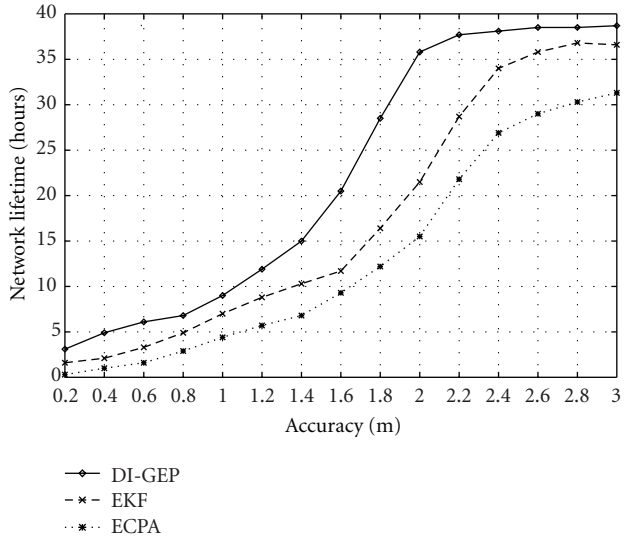


Figure 10: Network lifetime and prediction accuracy.

*7.4. Active Nodes.* This experiment uses the similar setting as given in the previous section and testifies the influence of the number of the active nodes as shown in Figure 9. DI-GEP outperforms EKF and ECPA in node scheduling because of its better trajectory prediction, so the number of active nodes in DI-GEP is about 25, 30% less than that in EKF and ECPA, separately.

*7.5. Prediction Accuracy.* This experiment uses the same setting as given in previous section and will testify that the prediction accuracy can heavily affect the network lifetime. The results are shown in Figure 10. Distributed GEP outperforms EKF and ECPA because of its better trajectory prediction, so at the same prediction accuracy, the network lifetime is averagely 28% longer than that in EKF and ECPA.

## 8. Conclusions

In order to track targets energy-efficiently in WSNs, we presented a distributed incremental algorithm based on GEP for target tracking applications in WSNs, proposed sliding window policy for distributed GEP to improve evolution process, proposed a new target tracking model, and give extensive experimental results to show the good performance of our method.

The future work includes (a) optimize DI-GEP to capture abrupt moving behaviors, (b) optimize DI-GEP to suit randomly distributed wireless sensor networks, and (c) consider border intrusion detection to save more energy in the initial state.

## Acknowledgments

The work in this paper was partially supported by the National Science Foundation of China under Grant no. 61103043, and Youth Fund of Sichuan University (project No. 2030404134011 and 2010SCU11049). The authors would like to express their gratitude to these two entities for their financial and technical support.

## References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.

[2] D. Li, K. D. Wong, Y. H. Hu et al., "Detection, classification and tracking of targets in distributed sensor networks," *IEEE Signal Processing Magazine*, pp. 17–29, 2002.

[3] C. Ferreira, *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*, Springer, Berlin, Germany, 2006.

[4] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 61–72, 2002.

[5] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163–1171, 2003.

[6] A. M. Ali, K. Yao, T. C. Collier, C. E. Taylor, D. T. Blumstein, and L. Girod, "An empirical study of collaborative acoustic source localization," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 41–50, New York, NY, USA, April 2007.

[7] J. Singh, U. Madhow, R. Kumar, S. Suri, and R. Cagley, "Tracking multiple targets using binary proximity sensors," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 529–538, New York, NY, USA, April 2007.

[8] N. Shrivastava, R. Mudumbai, U. Madhow, and S. Suri, "Target tracking with binary proximity sensors: fundamental limits, minimal descriptions, and algorithms," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*, pp. 251–264, New York, NY, USA, November 2006.

[9] W. Kim, K. Mechitov, J. Y. Choi, and S. Ham, "On target tracking with binary proximity sensors," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pp. 301–308, April 2005.

[10] S. Dai, C. Chen, C. Tang et al., "Light-weight target tracking in dense wireless sensor networks," in *Proceedings of the 5th International Conference on Mobile Ad-hoc and Sensor Networks*, IEEE Computer Society, pp. 480–487, 2009.

[11] S. Dai, C. Tang, S. Qiao, Y. Wang, H. Li, and C. Li, "An energy-efficient tracking algorithm based on gene expression programming in wireless sensor networks," in *Proceedings of the 1st International Conference on Information Science and Engineering (ICISE '09)*, IEEE Computer Society, pp. 774–777, Nanjing, China, December 2009.

[12] S. Dai, C. Tang, S. Qiao, K. Xu, H. Li, and J. Zhu, "Optimal multiple sink nodes deployment in wireless sensor networks based on gene expression programming," in *Proceedings of the 2nd International Conference on Communication Software and Networks (ICCSN '10)*, IEEE Computer Society, pp. 355–359, Singapore, February 2010.

[13] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *IEEE Design and Test of Computers*, vol. 18, no. 2, pp. 62–74, 2001.

[14] C. Lin, Y. X. He, and N. Xiong, "An energy-efficient dynamic power management in wireless sensor networks," in *Proceedings of the 5th International Symposium on Parallel and Distributed Computing (ISPDC '06)*, pp. 148–154, Timisoara, Romania, July 2006.

[15] X. Wang, J. Ma, and S. Wang, "Collaborative deployment optimization and dynamic power management in wireless sensor networks," in *Proceedings of the 5th International Conference on Grid and Cooperative Computing (GCC '06)*, pp. 121–128, Hunan, China, October 2006.

[16] D. G. Allegretti, G. T. Kenyon, and W. C. Priedhorsky, "Cellular automata for distributed sensor networks," *International Journal of High Performance Computing Applications*, vol. 22, no. 2, pp. 167–176, 2008.

[17] Y. Qing, A. Lim, K. Casey, and R. K. Neelisetti, "Real-time target tracking with CPA algorithm in wireless sensor networks," in *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '08)*, pp. 305–313, San Francisco, Calif, USA, June 2008.

[18] W. Wang, V. Srinivasan, K. C. Chua, and B. Wang, "Energy-efficient coverage for target detection in wireless sensor networks," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 313–322, April 2007.

[19] T. Clouqueur, K. K. Saluja, and P. Ramanathan, "Fault tolerance in collaborative sensor networks for target detection," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 320–333, 2004.