Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 10 (2012) 1100 – 1105

Procedia Computer Science

The International Workshop on Cooperative Robots and Sensor Networks

# An Evolutionary Approach to Improve Connectivity Prediction in Mobile Wireless Sensor Networks

Gustavo Medeiros de Araújo[a,c], A.R. Pinto[b], Jörg Kaiser[c], Leandro Buss Becker[a]

[a]*Department of Automation and Control Systems, Federal University of Santa Catarina, Florianópolis, Brazil*
[b]*Department of Computer Science and Statistics, Paulista State University (UNESP), São Paulo, Brazil*
[c]*Department of Distributed Systems, Otto-Von-Guericke-Univesität Magdeburg, Germany*

## Abstract

Connectivity is the basic factor for the proper operation of any wireless network. In a mobile wireless sensor network it is a challenge for applications and protocols to deal with connectivity problems, as links might get up and down frequently. In these scenarios, having knowledge of the node remaining connectivity time could both improve the performance of the protocols (e.g. handoff mechanisms) and save possible scarce nodes resources (CPU, bandwidth, and energy) by preventing unfruitful transmissions. The current paper provides a solution called *Genetic Machine Learning Algorithm* (GMLA) to forecast the remainder connectivity time in mobile environments. It consists in combining Classifier Systems with a Markov chain model of the RF link quality. The main advantage of using an evolutionary approach is that the Markov model parameters can be discovered on-the-fly, making it possible to cope with unknown environments and mobility patterns. Simulation results show that the proposal is a very suitable solution, as it overcomes the performance obtained by similar approaches.

*Keywords:* Wireless sensor networks; Mobility; Connectivity prediction; Genetic Algorithm; Classifier Systems.

## 1. Introduction

Recently, it emerged an eminent need for using Wireless Sensor Networks (WSN) in scenarios dealing with mobility. A typical scenario could be robots moving through a factory while communicating with other robots and also with fixed nodes to perform collaborative work. However, deploying mobile-WSN bring several challenges [1]. For instance, the communication protocols should attempt to topology changes due to mobility, as it brings uncertainty about the connectivity between neighbors nodes. Indeed, the awareness of the remaining connectivity time is a key information for designing more reliable network protocols.

Some related work address the problem of connectivity prediction [2]. A GPS is used in [2], where authors provide a Markov Chain to predict the connectivity between the mobile nodes and some fixed base stations. These works mostly intended for mobile ad hoc networks (MANETs) applications and, most of the time, rely on positioning information obtained by means of a GPS device. Such solutions are not well suited for indoor applications and neither

*Email addresses:* `araujo@das.ufsc.br` (Gustavo Medeiros de Araújo), `arpinto@ibilce.unesp.br` (A.R. Pinto), `kaiser@ivs.cs.uni-magdeburg.de` (Jörg Kaiser), `lbecker@das.ufsc.br` (Leandro Buss Becker)

doi:10.1016/j.procs.2012.06.156

for low-cost mobile-WSNs. Moreover, applying GPS-free solutions to determine location would be too costly for the WSN in terms of processing and communication resources.

Differently, the proposals presented in [3], [4] and [5] use link quality information instead of positioning to predict the link connection states. In [3] it was presented the Birth-death (BD) Markov Model, which uses *Signal-to-Noise Ratio* (SNR) information to characterize and predict link quality. MTCP [4] also modeled the RSSI as a Markov Model (MTCP) in order to predict the signal variation. In [5] authors used a time series to model the changes on link quality due to node mobility.

A drawback of using Markov Chain or Time series is that a previous history is needed to fill up the model, so that it could be out of date if the mobility pattern changes. Moreover it requires a large amount of memory. Despite that, MTCP model has being successfully applied to improve network protocols, like for example to improve the transmission power control mechanisms and to support more efficient back-off within clusters.

This paper presents a new proposal called *Genetic Machine Learning Algorithm* (GMLA) to predict the remaining connection time between neighbor nodes. It consists of a modified version of the BD model, with two main differences: (i) GMLA introduces the notion of orientation to represent a tendency in the link future state, and (ii) it makes use of an evolutionary approach to help finding the best transition matrix for a given situation. As advantage, the proposed solution can cope with environment variations, which may avoid nodes re-programming after such variations. Using such mechanism a considerable prediction improvement can be obtained.

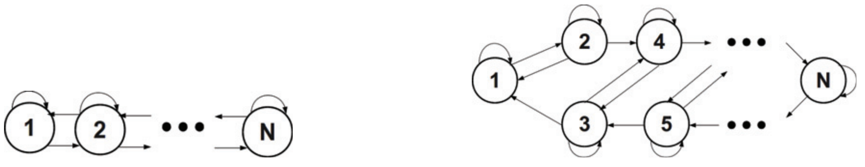
The rest of this paper is organized as follows: section 2 presents the proposed model; section 3 describes its evaluation while discussing the obtained results; section 4 concludes the paper.

## 2. The GMLA Model

The proposed *Genetic Machine Learning Algorithm* (GMLA) model attempts to predict the link state (connectivity) between neighbor nodes after a certain time ahead. The key point in GMLA is the fact that it does not require any location information to compute the connectivity prediction. Instead, it uses as input some form of link quality (LQ) information such as SNR, RSSI (*Received Signal Strength Indicator*), etc.

GMLA is based in the BD model [3], as it also uses a Markov Chain model to predict the LQ in the future. Markov models have the advantage that they do not need to keep a previous history to predict the next value, as the history is already embedded on its states transitions probability (it is a fixed-size matrix). Figure 1(a) illustrates the BD Automaton, where states represent LQ levels. The states on the extreme represent the best and worst possible LQ levels. The main difference between GMLA and BD while modeling the Markov Chain relies on the fact the former takes the nodes movement direction into consideration. To represent such issue, all intermediate states of the automaton from the BD model are duplicated, as shown in figure 1(b). The states on top represent nodes moving in one direction, and the states on the bottom represent nodes moving in the opposite direction. This does not mean that a node cannot change its direction, but means that this has a very low probability to occur.

Although the model could use any number of states, as discussed in [3] it is necessary to have at least three states to represent the possible LQ levels. If the model adopts a larger number of states it should be possible to achieve more fine grained results. However, we noticed by simulation that as the number of states grows, the prediction precision drops. This means that finding the proper number of states for the automaton is not straight forward. For our model we used 5 states and assumed that the LQ value has a smooth variation in time, just like in [3].



(a) BD Automaton (b) GMLA Automaton

Figure 1: Automaton models.

### 2.1. Discrete Markov Chain

The knowledge provided by the model is helpful to allow forecasting the next state. Therefore we used the discrete Markov model theory, where each transition happens within a second. The discrete Markov chain could be described as a triple $< S, T, \pi >$. $S$ represents the set of states, as illustrated in figure 1(b), and $T$ is the transition matrix that represents the probability to go from a certain state $i$ to the state $j$. It is related to each possible scenario (mobility pattern), and it can be obtained by means of simulation or practical experimentation. Finally, $\pi$ is the initial probability distribution of the set $S$. To choose the next state we use the vector state probability, as shown in equation 1. $T^n$ is the transition matrix powered by $n$ (number of steps ahead). After this calculation, it is possible to obtain the new probability vector $\pi^n$ and finally choose the next state using equation 2.

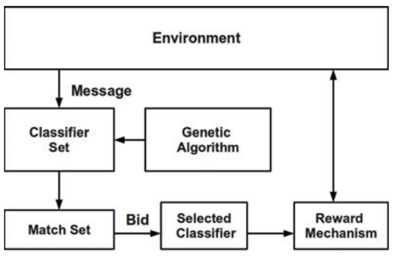
$$\pi^n = \pi * T^n \qquad (1) \qquad\qquad s^n = Max[s_1^n, s_2^n, ..., s_k^n, ] \qquad (2)$$

### 2.2. Evolutionary Approach

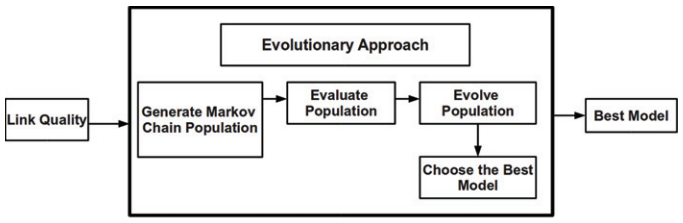
As mentioned, the transition matrix $T$ used by the Markov models is dependent on the environment characteristics, like for example the mobility pattern. To overcome this issue and allow the same protocol-stack to be used in unknown mobility conditions, we developed an evolutionary algorithm that is capable to find the best $T$ on the fly. For $\pi$, we assumed that each state has the probability *1/m*, where $m$ is the number of states of the model.

A node may begin to communicate at any signal range. The time needed for training the model and finding the best $T$ varies with the amount of LQ samples available. Of course this depends on each application scenario. As the LQ samples arrive with higher frequency, the faster the model can be built. The opposite is also true. However, it is not always the case that this learning phase is required. For instance, nodes could receive $T$ from other nodes that already learned, or even from the environment itself.

The evolutionary technique used in our GMLA is based on Classifier Systems [6], a machine-learning technique based on genetic algorithms (GA) that is capable of learning syntactically simple rules. The adopted classifier system scheme is shown in Figure 2(a) and works as follows. A message received from the environment can activate one or more classifiers. As classifiers are selected, they perform their rules. Afterwards, the selected classifiers are rewarded based on their performance. GAs consider a population of classifiers for some optimization problem. In this case there are individuals (classifiers) representing their genotypes, which are usually a set of bits or characters (in our case the genotype is the Markov model). This population is evolved by the GA after a predetermined number of consults. At each generation of answers, a new set of artificial creatures (classifiers) is generated. The answers are based on fragments of the most adapted previous individuals.



(a) Classifier System Scheme



(b) Genetic Machine Learning Algorithm

Figure 2: Evolutionary Schemes.

The main focus of a GA is its robustness. If the system is more robust, it requires a smaller number of interventions or redefinitions. Moreover, it will achieve higher levels of adaptation and will be able to perform better and last longer. The main difference between a classical GA approach and a classifier system is that the latter just evolves its population after some consults to the classifier set. Thus, it will perform the evolution process while it is running.

It should be highlighted that the main goal of our evolutionary approach is to generate the most suitable transition matrix $T$ to represent the LQ variation. Thereby, our approach is capable to predict the connectivity pattern of the mobile nodes even in a very dynamic environment. The structure of our evolutionary approach is represented by a set of individuals, with each individual having its own set of genes. Each individual represents a Markov model following the same structure of the model shown in figure 1(b). Each genotype is represented by a state (each state is composed by N probability transition). Thereby, each transition probability is randomly generated. It should be observed that the transitions that outcome from a state must sum 100%.

During startup each individual is randomly generated, then it evolves according to the phases shown in figure 2(b). After sampling the LQ, the individuals compare the LQ value with the current state and try to guess the next state for $N$ transitions ahead by computing equations 1 and 2. If the individual guessed correctly the next state, it receives a reward. After a certain number of consults, the populations is evaluated according to the fitness function (eq. 3). The accuracy percentage ($AP$) is the number of correct guesses $P_t$ divided by the number of consults $C$. The next step is the population evolution, where a crossover is applied to the two best individuals (those with highest $AP$) and their genes (states) are crossed. For example, in an 8 state model, states 1 to 3 are chosen from individual A and states 4 to 8 are chosen from individual B, generating two new individuals. After the crossover, all individuals that have an $AP$ value below a certain threshold are mutated. The mutation is applied in a state randomly chosen. Transitions are also randomly chosen by applying algorithm 1, so new probabilities are obtained. The algorithm 1 has the property to keep the sum of the outcome transitions in 100%. Therefore, the worst two individuals are replaced by new individuals. These steps are known as the learning phase. They are repeated until an individual with $AP$ above the threshold is obtained. Finally, the best individual is chosen (the one with the highest $AP$).

```
Input: StateTrasitions,numbertransitions
$N = numbertransitions$;
foreach $StateTrasitions\ i$ do
    if $i == N$ then
        $P_i = 100 - (\sum_{i=0}^{N-1} P_i)$;
    end
    else
        $P_i = random(0, 100 - \sum_{i=0}^{j-1})$;
    end
end
```

**Algorithm 1**: Random selection of transitions probabilities.

$$AP = \frac{\sum P_t}{C} \quad (3)$$

## 3. Evaluation

The main goal of the performed evaluation was to measure the prediction accuracy of the proposed GMLA model. This accuracy represents the ability of the model to correctly predict the next LQ state. In fact, it does not predict the next LQ state just for the next point in time, but for after $n$ seconds ahead (recall that in the model it is assumed at most one state transition per second). The prediction accuracy was expressed here as a metric called rate of accuracy. In the evaluation presented here SNR is used as link quality metric. For a matter of comparison, the original Birth-Death (BD) model and the MTCP [4] were also implemented in our study.

The evaluation was performed by means of simulations using OMNET++[1] with the INETMANET framework[2]. Wireless nodes communicate by means of the IEEE 802.15.4 standard, using as MAC protocol the non-beacon mode. The GMLA model was implemented here on top of the MAC protocol (network layer).

To perform the simulations it was observed the need for using different mobility models. Therefore we selected three protocols related with a factory automation scenario where mobile robots collaborate with each other and also with fixed sensor nodes. As discussed in [7], such scenarios attempt to represent realistic behavior.

The first adopted mobility model is the *Gauss-Markov*, created to overcome the too high randomness behavior of the well-known Random Way Point Mobility, which has the problem of generating unrealistic movements with suddenly stop and restarts. Gauss-Markov adds an $\alpha$ randomness parameter, where variables like direction and velocity follow a Gaussian distribution. Its value range from 0 (Brownian Motion) and 1 (linear motion). It is our understanding that in a factory it does not exist a completely Brownian motion, as robots or people must follows some mobility pattern with lower variation of direction and speed in order to have a safe and predicted behavior.

[1]http://www.omnetpp.org/
[2]http://wiki.github.com/inetmanet/inetmanet/

*Reference Point Group Mobility Model* (RPGM) is the other adopted mobility model. RPGM was designed to represent the behavior of nodes that must move in groups to work together in order to achieve a common goal. In RPGM a leader is randomly chosen and the other nodes must follow it.

The third and last adopted mobility model is the *Manhattan Grid Mobility Model* (MGM), which was mainly proposed to represent the movement in an urban area. In MGM nodes move in the horizontal or vertical directions, i.e., they can only chose to go straight, turn left, or turn right. This can be quite similar to the mobility pattern inside the factory plant facility. cleaning section.

As already mentioned, our approach needs to learn the transition matrix for each individual scenario. In the simulations performed, it was needed the 500 initial seconds of GMLA simulation to complete the so-called learning phase. The other approaches used in the comparison needed twice this time.

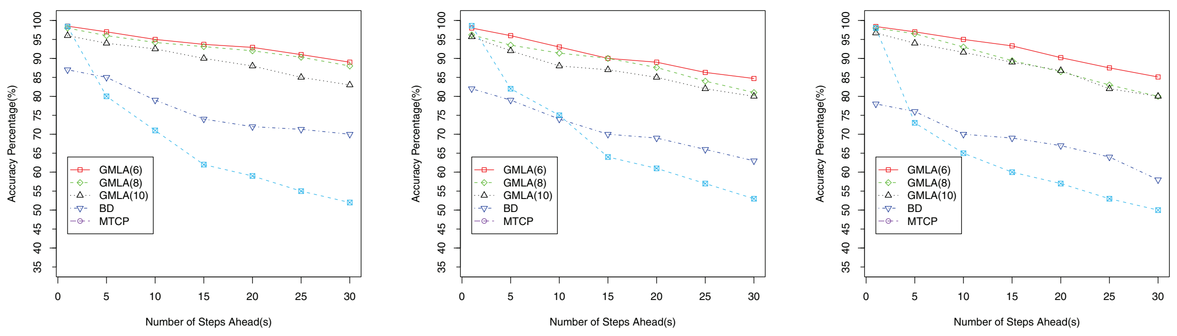
Each experiment included 100 nodes sending messages using a random interval between 1 and 5 seconds. 10% of the nodes are static (e.g. fixed sensors spread in the factory floor) and the other 90% of the nodes are mobile (e.g. robots that move around the factory), following one of the three different mobility models presented in the previous section. Additionally, 10% of the mobile nodes were randomly chosen to be responsible for predicting the connectivity between their neighbors (an uniform distribution was used). This is different for the RPGM model, as the group leader was chosen to predict the connectivity between neighbors. In the Gauss-Markov model the $\alpha$ parameter of the mobile nodes was randomly adjusted between 0.5 and 0.9 to avoid Brownian and Linear motions.

The prediction accuracy of the models was evaluated considering $n$ steps (seconds) ahead. More specifically, for each mobility pattern we have conducted experiments using 1, 5, 10, 15, 20, 25, and 30 steps. Additionally, in order to obtain a better understanding about the influence of the number of states, three different versions were implemented, containing 6, 8, and 10 states. The original BD model was implemented using 5 states, as this number was suggested in [3] as the one that best represents the LQ variation.

As general simulation parameters, the nodes transmission power was set to 1.0 mW, the attenuation model used was *Path Loss*, the playground size was $1000m^2$, and the speed was randomly chosen between 1 and 10 mps. For the evolutionary algorithm configuration it was used a population size of 16, a mutation threshold of 70%, and a number of consults of 100. The number of individuals was chosen to cope with the restrictions of a typical sensor node.

## 3.1. Obtained Results

The results of the comparison between the three different GMLA models (6,8 an 10 states), the BD and MTCP models are presented in figures 3(c), 3(b), and 3(a). The models were compared by means of their prediction accuracies, considering $n$ steps (seconds) ahead. It must be also highlighted that the intention of the experiments was not just to evaluate the models in a quantitative basis. Indeed, it intended to observe the feasibility of using link quality as a way to predict connectivity between mobile nodes in a WSN. Note that this is a very different approach if compared to what is under use in MANETs, as discussed in section 1.



(a) Acurracy Percentage for Manhattan Mobility Model (b) Acurracy Percentage for Reference Point Group Mobility Model (c) Acurracy Percentage for Gauss-Markov Mobility Model

Figure 3: Acurracy Percentage to choose the correct next state in 1,5,10,15,20,25 and 30 seconds ahead for (a), (b) and (c)

In all experiments GMLA(n) overcomes BD and MTCP in terms of prediction accuracy. The measures between the different GMLA versions were slightly different. It confirms the fact that as the number of states grows, it gets more difficult to predict. The results also show that GMLA ability (for 6, 8, and 10 states) to foresee the correct next state *j* from current state *i* has a respective accuracy average of 98.5%, 98.0%, and 96.7% for 1 step ahead in the Manhattan mobility model (best case). For a 30 seconds ahead prediction running the Gauss-Markov model (worst case) it achieves 80%, 79%, and 77%. This means that overall the model is able to forecast correctly the next states and, even with 30 seconds ahead, the result was satisfactory.

The results might also be analyzed by two other perspectives: (i) the influence of the number of states and (ii) the influence of the mobility models. In the first aspect it is noticed that as the number of states grows the accuracy decreases. This is due to fact that more states means more options to choose, suggesting a finer grain prediction. As the measurements itself are not so accurate, it is better to stay with a more general view (smaller number of states). However, there must be a compromise in the protocol design, because if the number of states is too low there might be not enough precision for the protocols to behave properly.

In regarding the mobility pattern, despite the differences in the simulation results we noticed that for Manhattan and RPGM mobility models the results for each GMLA version was quite similar. However, for the Gauss-Markov model, the prediction sharps fast after 5 steps ahead. The randomness characteristic of this model creates more difficulty to create a mobility pattern, therefore leading to estimation mistakes. However, the model could work very well with the results of 98% for GMLA (6, 8) and 96% from GMLA (10) (in Manhattan mobility) and 80%, 79%, and 77% of accuracy in the worst case (Gauss-Markov). For the DB model, the results ranged between 87% for Manhattan mobility (1 step ahead) and 58% for Gauss-Markov (30 steps ahead) in the best and worst case respectively. The MTCP model had a good result, about 98%, for 1 step ahead in all mobility models. However, the results drop fast after 1 step ahead reaching 50% in worst case for 30 steps ahead in Gauss-Markov Mobility model. The reason for such decrease is that the MTCP is almost a full connected automaton. It means that each state has more outcome transitions probabilities, then it has more possible final states destinations. The more probabilities to choose it has, more difficult is to hit the correct state. Finally, the results also show that it is possible to use the knowledge of link quality within ranges to predict the connectivity with a suitable precision.

## 4. Conclusions

This work was motivated by the need to improve the existing mechanisms that deal with connectivity prediction, since most existing solutions are either not so accurate or too costly for WSNs. Our proposed *Genetic Machine Learning Algorithm* (GMLA) consisted of modifying the Birth-Death (BD) model to add the notion of orientation, representing a tendency in the link variation, and also to allow learning on-the-fly the mobility pattern.

Conducted experiments show that in all different mobility scenarios GMLA presents a considerable prediction improvement in comparison to BD and MTCP. Despite the fact that we used a synthetic scenario to perform the evaluation, given that MTCP was already used to improve existing network protocols, it is possible to conclude that our GMLA can also bring further improvements in such protocols. Therefore, as future work, we plan to combine GMLA with some existing network protocol and then provide a comparative performance analysis.

[1] S. Munir, B. Ren, W. Jiao, B. Wang, D. Xie, M. Ma, Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing, in: Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on, Vol. 2, IEEE, 2007, pp. 113–120.

[2] A. Nicholson, B. Noble, Breadcrumbs: Forecasting mobile connectivity, in: Proceedings of the 14th ACM international conference on Mobile computing and networking, ACM, 2008, pp. 46–57.

[3] R. Guha, S. Sarkar, Characterizing temporal snr variation in 802.11 networks, Vehicular Technology, IEEE Transactions on 57 (4) (2008) 2002–2013.

[4] R. Sabitha, T. Thangavelu, Performance enhancement of fuzzy logic based transmission power control in wireless sensor networks using markov based rssi prediction, European Journal of Scientific Research (EJSR) 59 (1) (2011) 68–84.

[5] K. Farkas, T. Hossmann, F. Legendre, B. Plattner, S. Das, Link quality prediction in mesh networks, Computer Communications 31 (8) (2008) 1497–1512.

[6] D. Goldberg, Genetic algorithms in search, optimization, and machine learning, Addison-wesley, 1989.

[7] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wireless communications and mobile computing 2 (5) (2002) 483–502.