Anole: An Adaptive Neighbor Discovery Under Urban Environments

H. LING1, (Member, IEEE), S.-Q. YANG2, (Member, IEEE)
1School of Mathematical Sciences, East China Normal University, Shanghai 200241, China (e-mail: 52140601013@ecnu.cn)
2Department of Computer Science and Technology, Tongji University, Shanghai 200092, China (e-mail: youngfourkings@gmail.com)
Corresponding author: L. Hao (e-mail: 52140601013@ecnu.cn).

Abstract The fundamental operation in mobile wireless communication is to establish links between neighbor devices. The neighbor discovery problem (NDP) is non-trivial, especially under an urban environment. Various background scenarios, e.g., inside vehicles, at open-air squares, and in the supermarket, lead to complex discovery requirements. Discovery among fast moving devices requires an immediate exchange of emergency messages (minimum latency), while low-speed devices in crowded environments pay more attention to energy efficiency. Typical neighbor discovery protocols give solutions in a relatively stable scenario, which are not suited for the different environments in urban life. In this paper, we first proposed a non-integer framework to include all existing protocols. Then, a decentralized adaptive neighbor discovery protocol, named Anole, was designed under the framework. The protocol leveraged the genetic and similarity algorithms to be aware of and adapt to various scenarios with an appropriate discovery strategy. In the evaluation, we built different urban scenarios with real taxi and transportation card datasets in Shanghai. Meanwhile, an NS-3 simulator is applied to model device mobility and wireless communication. From the results, our protocol discovered 19% more links with similar energy consumption than typical protocols.

Index Terms chameleon, more mobile devices, neighbor discovery protocols, urban environment

I. INTRODUCTION
With the constant rapid increase of smart mobile devices, urban dwellers spend 80% of their usage time on mobile apps [1], e.g., games, social networking, and entertainment apps. Mobile ad hoc networks (MANET) give strong support to proximity-based and location-based mobile social networking applications [2] [3], e.g., social games and location services [4] [5].


Searchlight [14], as a variant of deterministic protocols, utilizes the same offset between two symmetric nodes (run in the same duty cycle) to improve the discovery efficiency. Especially, to further reduce the worst-case bound and to address the clock drift problem, Searchlight extends an active duration, over a complete time slot. Thus, the investigation direction towards non-integer aspects, which is closer to the real situation. Non-integer protocols are those that break the original assumption that the offset between any pair of mobile nodes is an integer number. The similarities and differences between integer and non-integer discovery algorithms were discussed in [15]. The authors proposed an abstract mathematical method, and proofed its optimality under the symmetric case. Moreover, Meng et al. [16] proposed an (A)Diff-Code discovery protocol to utilize a coding method to improve the discovery efficiency when the offset is a real number. Although it solves the failure of the traditional integer protocols under the non-integer situation, the protocol still uses a fixed active duration (the active length is a complete time slot), which cannot achieve a real non-integer strategy (flexible active duration).

Besides, these protocols are under the rendezvous situation (one-on-one) or a relatively stable environment (fixed neighbors). However, in the urban environment, there are various scenarios. For example, indoor, traveling, squares, and underground (as shown in Fig. 1). Under the indoor scenario, mobile nodes are crowded with low mobility, which results in high collisions. In contrast, when the mobile nodes are in...
the traveling scenario, high mobility and sparse density lead to a low collision ratio but high discovery missing probability. Due to this, a fixed discovery strategy is impossible to adapt to complex environments. More importantly, following the pace of urban life, surrounding scenario may change many times during a single day. Hence, a flexible discovery strategy is essential for a mobile device. This could not only suit the non-integer situation but also adapt to the scenario changing with time.

In this paper, we proposed an adaptive neighbor discovery protocol which is integrated with machine learning, named Anole. This protocol stands out when used in complex scenes found in urban environments, including fast, slow, and stationary scenes.

In the protocol, a non-integer general framework was established based on the fact that time exists objectively, but the strategies were variable. In the framework, the strategy of a node was regarded as a combination of several genes. A Gene is a sequence of rational numbers in (-1, 1] to reflect the active duration. As we will explain in II, each number in a gene describes the active duration in the corresponding slot. A positive number means that the active duration is at the beginning of the corresponding slot. A negative number means that the active duration is at the end of the corresponding slot. The absolute value of the number means the proportion of active duration in the corresponding slot. All the ordered numbers fix a sequence, which is called the gene. Besides, each mobile node was equipped with a state set, including various scenario states and corresponding strategy genes. Scenario states give a classification of different scenarios under the urban environment. During each scenario, a genetic algorithm-based method was applied to the strategy genes under the framework to make the strategy more suitable for the current discovery scenario. Once a scenario state change was detected, the strategy initialized an appropriate one in the history. In addition, the state set was updated periodically. In the experiment, our protocol had a 19% improvement in link discovery compared with typical protocols, e.g., Birthday, Disco, U-connect, Searchlight, and (A)Diff-Codes. The assumptions are defined in the framework as follows:

1) We assume each mobile node has a unique ID, labeled as a MAC address. Moreover, each node is in one of two modes alternately: active and idle. When a node is in the active mode, it launches both discovery detection and neighbor discovery. Otherwise, it turns off the radio and goes into idle mode to save energy.

2) We let \( \delta \) denote the discovery time, which means the least time needed for two mobile nodes to discover each other without collisions.

3) We define a fix time interval for each node as strategy period, denoted as \( T \) (shown in Fig. 2). Each strategy period consists of several gene slots. We set up the length of each slot to \( \delta \). We let \( n \) denote the total number of slots in a strategy period.

4) We construct a strategy gene in each strategy period, denoted as \( G \), where \( G = \{g_1, ..., g_n\} \). The strategy gene is a set of numbers, and each element reflects the proportion that a mobile node is in active mode during a gene slot. We define the \( g_i \in [-1, 1] \), where \( i = 1, ..., n \). The \( \pm \) means the location...
of the active duration. + denotes that the mobile node acts actively in the beginning period of the slot, while − denotes the contrast. (Note that no mobile node will turn active for less than $\delta$ period, because it cannot discover neighbors but instead generates collisions.)

For example, Fig. 2 shows the discovery process in a strategy period between two mobile nodes $a$ and $b$. Each slot in the strategy period lasts for $\delta$. The strategy genes for these two nodes are $G_a = \{0, -0.5, 0.7, 0, \ldots, 0.5, 1, 0.8, 0\}$ and $G_b = \{0, -0.7, 0.9, \ldots, 0, -0.6, 1, 0.2\}$, respectively.

5) We assume that if there are two or more mobile nodes in the active state synchronously, and they are within the communication range of another active node by chance, the node will receive collision packets in the channel.

6) Discovery occurs when the strategy genes of two mobile nodes are overlapped at least lasting $\delta$ without collisions. We define two operations $\oplus$ and $\odot$ to describe whether two nodes are discovered each other.

The $\oplus$ operation between two strategy genes of node $a$ and node $b$ is defined in Eq. 1, and we let $\Delta D$ denote the result through the operation, where $\Delta D = \{\Delta d_1, \Delta d_2, \ldots, \Delta d_n\}$. In short, the $\oplus$ operation shows the continuous length and place where node $a$ and $b$ discover each other in each slot. In Fig. 2, we obtain $\Delta D_{ab} = G_a \oplus G_b = \{0, 0, 0, \ldots, 0, -0.6, 0.8, 0\}$ using the $\oplus$ operation between node $a$ and node $b$.

$$\Delta d_{i} = \min\left\{g_i(a), g_i(b)\right\} 0 < g_i(a), g_i(b) < 1$$

$$\max\left\{g_i(a), g_i(b)\right\} -1 < g_i(a), g_i(b) < 0$$

$$g_i(a) = 1, g_i(b) = 1$$

$$g_i(a) = 1, g_i(b) = 1$$

otherwise,

where $i = 1, \ldots, n$.

The $\odot$ operation works on each pair of two consecutive elements in the set $\Delta D$. We collect the results in $D$, where $D = \{d_1, 2, \ldots, d_{n-1}\}$. In other words, the $\odot$ operation shows whether the continuous length in two contiguous slot where node $a$ and $b$ discover each other is more than the discovery time $\delta$. Equation 2 gives a definition of the $\odot$ operation in detail.

$$d_{i,i+1} = \Delta d_i \odot \Delta d_{i+1} = \begin{cases} \Delta d_i < 0 & \Delta d_{i+1} > 0 \\ \& -\Delta d_i + \Delta d_{i+1} \geq 1 \\ \Delta d_i = 1 & \Delta d_{i+1} = 1 \\ 0 & \text{otherwise}, \end{cases}$$

where $i = 1, 2, \ldots, n - 1$.

It is easy to know that $D$ is a binary sequence. If there is 1 in the sequence, it means that the two mobile nodes have more than $\delta$ overlapped duration in the active state without considering the collisions, which leads to discovery. We define that when $\sum_{i} d_{i,i+1} > 0$, discovery occurs. Otherwise, the discovery fails. For instance, in Fig. 2, the result after the $\odot$ operation among the elements in $\Delta d_{AB}$ is $\{0, 0, \ldots, 0, 1, 0\}$ (note that $d_{n-2,n-1} = 1$ because $|\Delta d_{n-2}| \geq 1$).

7) An operation $\odot$ is defined to represent the residual strategy that a target node could use to discover other neighbors in a slot. For example, Eq. 3 illustrates the $\odot$ operation between node $a$ and node $b$.

$$g_i(a) \odot g_i(b) = \begin{cases} g_i(a) \oplus (1 - g_i(b)) & g_i(b) \geq 0 \\ g_i(a) \oplus (1 + g_i(b)) & g_i(b) < 0 \end{cases}$$

where $i = 1, 2, \ldots, n - 1$.

For instance, the residual strategy after node $a$ discovered node $b$ is $G_a \odot G_b = \{0, 0, \ldots, 0, -0.5, 1, 0.4, 0, 0\}$. Obviously, $G_a \odot G_b = G_a \oplus (I - G_b)$, where $I$ is a vector of ones and its size is $n$.

8) When the discovery occurs at the joint place between two consecutive strategy periods, there lies a problem that the protocol cannot realize the discovery because the two genes are discrete. To solve the problem, we improved a strategy gene with one more gene slot $g_0$. $g_0$ denotes the gene value of the last slot in the previous strategy period.

**Theorem 1**: According to the assumptions, the strategy gene framework is suited for typical protocols, e.g., Birthday, Disco, U-connect, Searchlight and (A)Diff-Codes, in both symmetric and asymmetric cases.

Proof: We divide the time into several strategy periods, which is the same for each mobile node. According to the third item in the assumptions, we have $T = n \cdot \delta$. We assume that the duration of the time slot in the typical protocols is $\tau$. Obviously, $\tau \geq \delta$.

There exists a mapping between the strategies of the typical protocols and the strategy gene in each period, denoted as $F : \pi \mapsto G$, where $\pi$ denotes the strategies in the typical protocols and $G$ denotes the strategy gene in the framework during a strategy period.

Let the strategy period $G$ be from $t_s$ to $t_s + T$, where $t_s$ is a randomly selected time moment. Besides, we find that the duration in the active state for a mobile node consists of several discrete time slots in the typical protocols, and each one lasts $\tau$. Let $k$ denote the number of time slots. Due to this
characteristic, the strategy $\pi$ for each typical protocol from $t_s$ to $t_s + T$ can be expressed as follows:

$$\pi = \{(t_{s+1}, t_{s+1} + \tau), (t_{s+2}, t_{s+2} + \tau), \ldots, (t_{s+k}, t_{s+k} + \tau)\}.$$ (4)

Note that $t_{s+1}, \ldots, t_{s+k}$ are time moments between $t_s$ and $t_s + T$, and $t_s + \tau \leq t_{s+1}$ ($i = s + 1, s + 2, \ldots, s + k - 1$).

Since $\tau \geq \delta$, an active time slot lasts $\tau$ can be expressed as a series of gene slots as $(t_{r_f}, 1, \ldots, t_{r_e})$ ($0 \leq r_f, r_e \leq 1$) based on the assumptions, where the omitted part consists of 1, and $r_f$ and $r_e$ denote the connected parts between the time slot and the gene slots (as shown in Fig. 3). Therefore, the elements in strategy $\pi$ could be represented by gene slots.

Besides, to deal with the junction parts (heads and tails) between traditional strategy $\pi$ and strategy gene $G$, we confirm the values of the first and last gene slot under different cases. In a strategy period, there are three situations for the value of the first gene slot, and four situations for that of the last gene slot. We obtain $G = \{(t_{s+1} + \delta - t_s)/(t_s + \delta - t_{s+1})/0, 1/-(t_s + T - t_{s+k})/(t_{s+k} + \tau - t_s - T + \delta)/0\}$, where the different values are separated by the symbol $\delta$.

Obviously, the mapping takes both asynchronous and asymmetric situations into account.

In addition, according to the different genes, we give a definition about two types of genes: isomorphic genes and heterogeneous genes. We define genes with the same pattern as isomorphic genes. There is only an offset between a pair of isomorphic genes. In contrast, heterogeneous genes are defined as those with different patterns. For example, assuming $T = 5$, \{-0.5, 0.7, 0, 0.1\} and \{0, 0.1, -0.5, 0.3\} are isomorphic genes while \{-0.5, 0.7, 0, 0.1\} and \{-0.3, 0.8, 0, 1, 0.1\} are heterogeneous genes.

III. ADAPTIVE PROTOCOL DESIGN

In this section, we demonstrate the procedure of the Anole protocol under the framework mentioned in II. The main object for each mobile node is to find the optimal strategy when the urban environment changes.

A. PROBLEM IN EXISTING PROTOCOLS

From Thm. 1, we could use a series of genes to display the strategies in existing ND protocols. However, fixed strategies have an inherent drawback: unstable performance under various environments. Especially in modern life, mobile device users will move into different scenarios even for a single minute. Figure 1 gives examples of four scenarios: indoor, traveling, walking and underground. Both the density and the mobility of the nodes at each scenario are totally different.

The genetic algorithm usually focuses on optimization by generating high-quality solutions in the repeated iterative. Since the environment is complex and we cannot tell the direct factors that influence the discovery, we use a genetic based algorithm to find the suitable strategy under each scenario.

B. GENETIC ALGORITHM-BASED STRATEGY

For each mobile node, a gene set is equipped with finite initializing heterogeneous genes. Let $m$ denote the number, and $G$ denote the set. According to the assumptions in II, we have $G = \{G_1, \ldots, G_m\}$. After initializing, there is an evolution step for the mobile node to seek the optimal gene under the current scenario.

1) Initialization

The initialization of each gene slot should obey the following rules:

1) When $g_i < 0, g_i + g_{i+1} \geq 1$;
2) When $0 \leq g_i < 1, g_{i+1}$ should between $(-1, 0]$;
3) When $g_i = 1$, $g_{i+1}$ chooses a random value in $(-1, 1]$;
4) The value of the initial gene slot in $G_1 \in G$ is chosen in $(-1, 0] \cup \{1\}$.

Note that $i = 0, 1, 2, \ldots, n - 1$.

With appropriate probabilities of the random numbers, we could control the duty cycle of a mobile node to satisfy the following equation:

$$dc \approx \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} |g_i, j|}{m \cdot n},$$ (5)

where $g_i, j$ denotes the value of the $j$th gene slot in $G_i$, and $n$ denotes the number of gene slots in a strategy period.

2) Evolution

Then, we describe the evolution of the gene set. The mobile node will apply the strategy genes in the set in turn. We name this process strategy round. After a strategy round, the mobile node will evolve its genes set. The evolution contains three operators: select, crossover and mutation.

1) Select Two genes will be selected randomly to do the crossover and mutation.
2) Crossover We utilize the single-node crossover. A single point, denoted as $l$, is chosen for the two selected genes. The two genes will trade the values of the first $l$ gene slots with a crossover probability $p_c$. For example, Fig. 4 shows a crossover between $G_i$ and $G_j$, while $l = 3$. The duty cycle of the mobile node is guaranteed, since the crossover operation does not change the total active duration in a strategy round.
3) Mutation After the crossover operation, two created offspring may mutate with a certain probability. Let
utilization during the strategy period, which means determining the ratio between the number of discovered links and the active duration. We define the fitness function of gene $G$ and link list $L$ as follows:

$$f(G, L) = \begin{cases} 0 & |L| = 0 \lor \|G\|_1 = 0 \\ \frac{|L|}{\|G\|_1} & \text{Otherwise} \end{cases}$$

(7)

where $|L|$ means the number of links, and $\|G\|_1$ denotes the 1-norm of $G$.

Obviously, $f(G, L) \in [0, 1]$. More discovered links and a less active duration result in a bigger fitness degree.

2) Gene Elimination

After a strategy round, we ranked the fitness results of the $m$ groups in descending order, and collected them into a set. The algorithm eliminated two genes with the worst performances in the set. Then, two genes were selected in the set to do the crossover and the mutation operations. In the future, the number of the eliminated genes can be increased into a integer lower than $\lfloor m/2 \rfloor$. Moreover, when there is only one element in the set, the gene takes the crossover step by swapping its own gene codes before and after a single point. Algorithm 1 shows the evolution process.

Algorithm 1 Genetic Algorithm-Based Strategy Algorithm

Require: Original Strategy Gene Set $G_o$, and Neighbor list $L$;
Ensure: New Strategy Gene Set $G_n$;
1: for each gene $G_o$ in $G_o$ do
2: Calculate the fitness function $f(G_o, L)$, where $L \in L$;
3: Insert $G_o$ into a set $G_s$ in the order of $f(G_o, L)$;
4: end for
5: Rank the elements in $G_s$ from large to small;
6: Eliminate the two with worst performances, and reconstruct $G'_s$;
7: Randomly select two genes in $G'_s$ and do crossover and mutate operations;
8: Add springs into $G'_s$;
9: return $G_n = G'_s$;

D. ADAPTING URBAN ENVIRONMENT

In urban life, neighbor discovery among mobile devices may occur under various scenarios. Common classification methods include inside versus outside nodes, and traveling versus static nodes. The main means to distinguish different surroundings of a mobile device in our protocol is the velocity. A high velocity implies the mobile device is traveling on the road or on the subway; meanwhile, the velocity is low when the user works at the office or walks across a square. However, it is not easy to obtain the instantaneous velocity from a mobile device. Hence, we used acceleration (which can be easily gained from mobile sensors) to replace the velocity.
For each mobile device, we established a scenario vector to describe the surroundings in each strategy round, denoted as \( |V| = (\mu_a, \sigma_a^2, |N|) \). \( \mu_a \) and \( \sigma_a^2 \) represent the mean and variance of the acceleration, respectively. Once the scenario changes, the variation of the velocity could be reflected through the mean and the variance of the acceleration.

Besides, a mobile node keeps a state set \( S = \{(s_1, c_1, G_1), (s_2, c_2, G_2), \ldots \} \). Each state has a central vector \( s \), its size \( c \) (number of vectors), and a binding gene set \( G_s \). When a mobile node finds that the state changed, it replaces the corresponding gene set with the previous one. Each state is described as a scenario vector. In the initial phase, there are three elements in \( S \), which are obtained from the clustering results of the historical scenario vectors. Here, we use the \( k \)-means algorithm to do the clustering step.

When a mobile node perceives a scenario vector \( e \) after a strategy round, it calculates the similarity between \( e \) and all \( s \) (\( s \) belongs to \( S \)), and transforms to the state with the biggest similarity. The cosine similarity is used to measure the similarity, as follows:

\[
sim(e, s) = \frac{e \cdot s}{\|e\| \cdot \|s\|}, \tag{8}
\]

where \( \| \cdot \| \) denotes the Euclidean norm.

**E. STATE UPDATE**

Obviously, three states are not enough for complex urban scenarios. Hence, it is necessary to update the state set and the corresponding gene set.

1) State Update

The update of the state set includes two aspects: adding a new state and adjusting the current state. A threshold is provided in this step: \( e \), which denotes the threshold of the similarity.

If the current scenario vector \( e \) is different from all states in the set (lower than \( e \)), a new state will be created with a initial value \( e \). Besides, for each state, if the current scenario vector \( e \) is similar to that of another existing state (biggest one that greater than or equal to \( e \)), the value of the state will be updated as follows:

\[
s_{\text{new}} = \frac{1}{e + 1}[e + c \cdot s_{\text{old}}], \tag{9}
\]

where \( s_{\text{old}} \) and \( s_{\text{new}} \) denote the previous and the updated central states, respectively.

2) Gene Update

By the same logic, the corresponding gene set should be changed along with the update of state set. There still lies two cases. When a new state is created, the genes in the set are randomly selected with a certain duty cycle, which is decided by the remaining energy of the mobile node. Otherwise, the gene set will be updated by the previous one so that all its genes perform well in the fitness function.

**IV. ANALYSIS**

In this section, we describe the suitability of the GA-based protocol in NDP under the urban environment. We start the analysis from the two-node case, and then extend to the multi-node case.

**A. TWO-NODE CASE**

We first analyze a simple situation where there are only two nodes in the region. Suppose node \( a \) and \( b \) are equipped with gene sets \( G_a \) and \( G_b \), respectively. As defined in II, we apply operation \( \oplus \) between two gene sets, and get the result, denoted as \( \Delta D_{ab} \). Then, we obtain \( D_{ab} \) by using operation \( \odot \). After an evolution, we assume the gene sets become \( G_a' \) and \( G_b' \). The corresponding sets under \( \oplus \) and \( \odot \) operators are denoted as \( \Delta D_{ab}' \) and \( D_{ab}' \).

**FIGURE 5: The Discovery of Two Consecutive Gene Slots**

We first consider the probability that two nodes are discovered in a pair of consecutive gene slots, as shown in Fig. 5. Discovery succeeds when the overlapping part is bigger than \( \delta \).

**Theorem 2**: Based on the probability theory and set theory, we infer the probability that no discovery occurs at two consecutive gene slots is

\[
p_{ab} = \left\{ \begin{array}{ll}
g_{i,i+1}^{(a)} + g_{i,i+1}^{(b)} - 2 & 2 \leq g_{i,i+1}^{(a)} + g_{i,i+1}^{(b)} < 3 \\
1 & 3 \leq g_{i,i+1}^{(a)} + g_{i,i+1}^{(b)} \leq 4
\end{array} \right., \tag{10}
\]

where \( g_{i,i+1} = |g_i| + |g_{i+1}| \).

\[
P(|D_{ab}| = 0) = (1 - E[p_{ab}])^{(n-1)m}, \tag{11}
\]

where \( p_a \) and \( p_b \) denote the probabilities that node \( a \) and \( b \) are in the active mode in the pair of gene slots, respectively. Note that the calculation of \( E[p_{ab}] \) is:

\[
E[p_{ab}] = \int_D l_a l_b p_{ab} dl_a dl_b. \tag{12}
\]

Besides, \( l_a, l_b \in [1, 2] \).

If two nodes are not discovered during the last strategy round, the discovery probability in this round compares the conditional probability \( P(\|D_{ab}\| = 0) \) with \( P(\|D_{ab}'\| = 0) \| D_{ab}\| = 0 \), where \( \|D_{ab}\| \) denotes the sum of all child elements. Obviously, \( P(\|D_{ab}\| = 0) \| D_{ab}\| = 0 \) = 0, which means two undiscovered nodes still cannot discover each other if they do not change their strategies.
The influence of the mutation operation can be omitted, since the occurrence probability is low and it has little influence on the mobile node in a strategy round. In the protocol, the crossover operation changes the order of gene slots among the genes of a mobile node. Let \( l_{ab} \) denote the total number of different gene slots between \( \Delta D_{ab} \) and \( \Delta D'_{ab} \). Hence, we can infer the probability that two nodes are discovered in this strategy round as:

\[
P(\|D_{ab}'\| \neq 0 | \|D_{ab}\| = 0) = \left[ 1 - \frac{p_{l_{ab}}}{(nd)^2 \cdot dc_a dc_b} \right]^{l_{ab} - q}.
\]  

(13)

Note that \( q \) is a constant, which denotes the number of deleted discontinuous gene slot parts between \( \Delta D_{ab} \) and \( \Delta D'_{ab} \). \( dc_a \) and \( dc_b \) denote the duty cycle of the two nodes, respectively.

### B. MULTI-NODE CASE

Now, we consider the multi-node situation. Here, \( \Delta D_{ab} \) becomes \((G_a \oplus G_{i \in N_a}) \oplus (G_b \oplus G_{j \in N_b})\), where \( N_a \) collects the nodes in the discovery range of node \( a \), and \( N_b \) collects the nodes in the discovery range of node \( b \). Clearly, as in the two-node case, in a relatively stable environment (having fixed nodes), if the mobile nodes do not have an evolution operation, the undiscovered nodes still cannot discover each other whether there are collisions or not. With the evolution operation, the mobile nodes will adjust the genes with bad fitness performances, which improves the discovery probability of the strange mobile nodes (as shown in Eq. (13)). It is easy to add a discount factor parameter into Eq. (13) to estimate the collision probability. Besides, in a dynamic environment, mobile nodes randomly move in and out of the discovery range of each mobile node in a strategy round. Therefore, there is equal chance of discovering a new-coming neighbor for mobile nodes with and without an evolution operation.

### V. EVALUATION

In this section, we evaluate our protocol under an urban environment constructed by real traffic datasets and the NS-3 network simulator. The environment contains three different scenarios, as illustrated in V-A. Then, we validate our protocol on discovery latency and energy consumption, and compare it with typical protocols (Birthday, Disco, U-connect, Searchlight).

#### A. URBAN ENVIRONMENT BUILD

Here we give an overview of the real datasets, and describe the construction of the scenarios with the help of the NS-3 simulator.

1) Dataset

We used the real traffic datasets collected in Shanghai on April 1st, 2015, to build a compound urban environment. The traffic datasets replaceconsistconsists of two aspects: check-in data from the transportation card at subway stations and GPS data from the taxicabs. As the background of Fig. 6 shows, the discovery region is selected around People Square (including a elevated road segment, two subway stations, and 120 urban roads), which is about 1 square kilometer in area. The total data lines are 263,558 and 556,635 in the transportation card dataset and taxicabs dataset, respectively.

![FIGURE 6: Example of simulation environment](image-url)
TABLE 1: Scenario Initial Data

<table>
<thead>
<tr>
<th>Typical Scenario</th>
<th>8 am</th>
<th>12 am</th>
<th>8 pm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urban Road</td>
<td>48</td>
<td>113</td>
<td>106</td>
</tr>
<tr>
<td>People Square Station</td>
<td>2484</td>
<td>1455</td>
<td>1700</td>
</tr>
<tr>
<td>Great World Station</td>
<td>657</td>
<td>260</td>
<td>346</td>
</tr>
<tr>
<td>Subway Carriage</td>
<td>200</td>
<td>117</td>
<td>136</td>
</tr>
</tbody>
</table>

and updated the nodes every 5 min. The update accorded with the check-in data at the subway station. This scenario was relatively stable.

B. PARAMETER SETTING

Before we display the evaluation results, we will describe the settings of the parameters in our protocol and typical protocols.

1) The duration of a time slot in typical protocols is set to 10 ms. Specially, the time slot of (A)Diff-Code protocol is set to 20 ms.
2) The duty cycle in the symmetric case is 5 %, while the range of the duty cycle in the asymmetric case is from 1 % to 5 %.
3) In Anole, the crossover probability is set to 0.5 and the mutation probability is set to 0.01. The length of a gene (strategy period) lasts 100 time slots. Each gene set contains 5 genes for a mobile node. The threshold parameter $\epsilon$ is obtained by the $k$-means clustering method while training the historical data. In the experiment, we let $\epsilon = 0.5$.
4) In the typical protocols, we let the offset between each pair of nodes be no longer a integer (this is the same as the assumption in the ADiff-Code protocol). The detailed parameters in the typical protocols are shown in the figures.
5) We apply the Anole and typical protocols at three specific time moments (8 am, 12 am, and 8 pm) in a day. Each moment lasts 1 min, and Table 1 shows the initial number of nodes in the region. In Fig. 7, we indicate the variations of the mobile nodes at the urban road and subway station. Fig. 7(a) gives the number of taxicabs in the region every 10 seconds for the moments, while Fig. 7(b) indicates the number of check-in nodes at the two stations. Note that “PS” and “GW” in Fig. 7(b) denote the People Square Station and Great World Station, respectively. Furthermore, in the subway carriage scenario, the number of mobile nodes could be a constant at each moment.

C. DISCOVERY EFFICIENCY

Since the mobile nodes join and leave the urban road and subway station continuously, the discovery efficiency is regraded as the number of links discovered in a certain time period under these two scenarios. In the subway carriage scenario, we used the discovery fraction (the percentage of discovered neighbors in the region for a mobile node) to reflect the efficiency.

According to Table 1, we set corresponding nodes in different scenarios, and simulated the performances of various protocols in NS-3. The results of discovery efficiency are displayed in Fig. 8. Figures 8(a) 8(b) show the discovery latency under the urban road situation.

Figures 8(e) and 8(f) illustrate the number of discovered links during each second in the simulated period with symmetric and asymmetric mobile nodes at the Great World Railway Station, respectively. In the first few seconds of both cases, more links were established by our protocol. As time goes by, the number of the discovered links decreased and gradually stabilized due to the majority nodes having discovered each other. Anole still performed a little better when nodes joined or left the region.

The subway carriage scenario is a typical instance of indoor scenes since the mobile nodes are relatively static. In Fig. 8(g), Anole had a 19 % improvement over the typical protocols in the discovery latency under the symmetric case, and had a great performance under the asymmetric case.

In addition, to evaluate the performances under the scenario transitions, we let 10 nodes first travel by taxies, then come into the subway station (5 for each station) and take the subway around each moment.
situation, that the energy consumption of our protocol was in the middle among existing protocols. Under the asymmetric situation, although there were vibrations among all the protocols, our protocol still performed at the average level. Therefore, we concluded that our protocol performed better than existing protocols at a similar energy consumption.

### D. ENERGY CONSUMPTION

The energy consumption of a mobile device is shown in Table 2 [19].

We give the average energy consumptions of our protocols and typical protocols on each node under the urban scenario. Since the energy consumption of the three situations were similar, we only considered the differences between symmetric and asymmetric cases, which is shown in 9. Figure 9(a) shows the symmetric situation. Since in deterministic protocols all nodes run in the same duty cycle, the average energy consumption is relatively stable. Conversely, the energy consumption of the Birthday protocol is unstable due to the stochastic schedule. We found under the symmetric

![Figure 8: Discovery Efficiency](image)

![Figure 9: Average Energy Consumption in Both Symmetric and Asymmetric Cases](image)

### TABLE 2: Energy Consumption of the Mobile Device

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Radio (ad hoc)</td>
<td>Active</td>
<td>1502 mW</td>
</tr>
<tr>
<td></td>
<td>Idle</td>
<td>979 mW</td>
</tr>
</tbody>
</table>

### VI. RELATED WORK

Neighbor discovery is the basic process in Mobile Ad-hoc Networks (MANET) to support proximity-based services and location-based services (e.g., LocalSocial [4], and Cheers [5]).

Early neighbor discovery schedules are embedded in MAC protocols (e.g., SMAC [7], and BMAC [8]). These MAC protocols use overhead packets or GPS to keep two nodes synchronized and discovered. McGlynn and Borbash [6] proposed a probabilistic discovery protocol, named “Birthday”, which first made neighbor discovery an independent problem. Vasudevan et. al. [20] further investigated the probabilistic schedule by converting the neighbor discovery problem (NDP) as an example of the coupon collector’s problem.

However, the worst-case bound of these protocols cannot be guaranteed. Hence, deterministic protocols appeared to give the bound. These kind of protocols are mainly divided into two types: Quorum-based protocols [9] [10], and
Prime-based protocols (e.g., Disco [11], and U-connect [12]). Quorum-based protocols assume time as a two-dimensional mesh, and the node selects one row and one column in the active state. Meanwhile, Prime-based protocols depend on the Chinese Remainder Theorem [21]. Discovery occurs when a pair of nodes has active slots overlapped. Compared with probabilistic protocols, the typical deterministic protocols sacrifice their average performance for obtaining the worst-case bound.

To improve the discovery performance and the worst bound, several variant deterministic protocols (e.g., Searchlight [14], and BlindDate [13]) have been proposed. Searchlight utilizes the rule that the offset between two discovery periods is constant in the symmetric situation, and achieves a lower discovery latency than typical protocols. BlindDate [13] improved Searchlight, and could guarantee a lower worst-case performance. The main weakness of these two protocols is that they cannot handle the asymmetric situation well. Zhang et al. [22] proposed Acc, an on-demand discovery accelerating middleware based on Disco protocol, leveraging the direct and indirect neighbors to locate optimal additional active slots. As a summary of the typical deterministic protocols, a general framework Hello [23] was proposed. In the strategy of the framework, all deterministic protocols (e.g., Quorum, Disco, U-connect and Searchlight) were considered. However, the authors did not take energy consumption into account.

Recently, the trend of investigations break the inherent assumption that time is slotted. Authors in [15] discussed the integer and non-integer discovery algorithm. They gave a reduction that transforms any integer-based schedule into a corresponding non-integer-based schedule. In addition, they provided a novel family of lower bounds for the non-integer model. Meng et al. [16] proposed an (A)Diff-Code discovery protocol to improve energy efficiency under the non-integer case. The protocol gave the tight worst-case bound in the symmetric situation, and found the optimal Diff-Code according to a perfect difference set.

NPD investigation has been extended to several additional factors (e.g., multi-channel [24], multi-hop [25], and multi-packet [26]). Furthermore, some other researchers studied NPD using a directional antenna [27] [28] or in a cognitive network radio [29]. To handle special situations, Yang et al. [30] discussed the discovery under a crowded place.

To the best of our knowledge, all existing protocols focus on discovery under a rendezvous situation, without considering complex and dynamic urban environments.

VII. CONCLUSION

In this paper, we proposed a non-integer neighbor discovery, named Anole. The protocol leveraged the genetic algorithm to adapt to the dynamic urban environment during discovery. In the protocol, each mobile node was equipped with a state set, which collected a classification of the environments and their corresponding gene sets. During a strategy round, the mobile node evaluated the genes and evolved those with bad performances to improve the efficiency. Moreover, once the environment changed, the mobile node replaced its gene set with a suitable one from the previous rounds under this state. The state set was updated with newly created state or a modified old state. We evaluated our protocol under an urban environment, which was established from real traffic datasets in Shanghai and the NS-3 network simulator.

In future work, we would like to let the discovery time \( \delta \) follow a Gaussian distribution since it affected by various factors (e.g., transmission delay and broadcast range). We will also give a strict proof on the convergence of our algorithm under different urban scenarios and conduct further experiments on test bed implementation on phones or sensors.

REFERENCES