Seventh FRAMEWORK PROGRAMME
FP7-ICT-2007-2 – ICT-2007-1.6
New Paradigms and Experimental Facilities

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Deliverable D3.6 - "Implementation of Techniques that Improve the Scalability and the Quality of the Routing System"

Project acronym: **ECODE**
Project full title: **Experimental Cognitive Distributed Engine**
Proposal/Contract no.: **223936**

Date of preparation of Deliverable: **October, 2009**

Document properties

Document Number	FP7-ICT–2007-2-1.6-223936-ECODE-D3.6
Document Title	Implementation of Techniques that Improve the Scalability and the Quality of the Routing System
Document responsible	Dimitri Papadimitriou (ALB)
Author(s)/editor(s)	Dimitri Papadimitriou (ALB), Grégory Detal (UCL), Olivier Bonaventure (UCL), Damien Saucez (UCL), Benoit Donnet (UCL)
Dissemination level	PU
Date of preparation	October 30, 2009
Version	Draft version 01

Table of Content

Executive Summary

The aim of the ECODE experimental research project is to introduce a new Internet architectural component realized by means of a machine learning component. This deliverable, belonging to WP3 (Cognitive network and system experimentation), is part of a series of 3 deliverables – one per technical objectives (TO) as defined in the ECODE project technical annex. It aims at describing the feasibility, benefits and applicability of introducing machine learning techniques to improve the scalability and the quality of the Internet routing system that is based on Border Gateway Protocol (BGP). More specifically, this deliverable addresses the implementation, development and first experiments performed in the framework of TO3 related to this use case. We are interested in finding appropriate machine learning technique for path exploration sequences detection and identification so as to mitigate its effects. By mitigation, we mean here enforcing the suppression of subsequent sequences of BGP routing update messages that are detrimental to the routing system convergence (delays, and transient instabilities).

After having formalized the fundamental cause deteriorating the routing system quality, i.e., uninformed path exploration, the machine learning techniques used and experimented for this use case are specified. In particular, we focus on two techniques: the decision trees and the hidden Markov model (HMM). A decision tree is a decision support tools using a tree-like graph of decisions. HMM is a statistical model in which the system being modelled is an embedded stochastic process with an underlying Markov process that is not observable (it is hidden) but can only be observed through another set of stochastic processes that produce the sequence of observations. Initial performance evaluation results of these techniques are detailed. At this stage, evaluation of the proposed machine learning has been performed by means of off-line experiments. These experiments have been conducted for realistic AS topologies, following several properties of today's Internet topology such as hierarchical structure, power-law degree distribution, strong clustering as well as a constant average path length. Further experiments are ongoing to refine these initial results and future work will consist in improving the learning model specified in this document.

Based on these results and acquired experience in processing BGP UPDATE messages, we describe how we currently consider the introduction of the mechanisms involved in this use case into the ECODE common architecture as detailed in Deliverable D2.1. In particular, we show how the functionalities of this use case are split among the Routing Engine (RE) and the Machine Learning Engine (MLE) of the architecture, and which message exchanges are necessary among these components.

1. Acronyms

AS	Autonomous System
BGP	Border Gateway Protocol
CRF	Conditional Random Field
CPU	Central Processing Unit
eBGP	External Border Gateway Protocol
FN	False Negative
FP	False Positive
HMM	Hidden Markov Model
iBGP	Internal Border Gateway Protocol
IP	Internet Protocol
ISP	Internet Service Provider
TN	True Negative
TP	True Positive
T1	Tier-1
T2	Tier-2
T3	Tier-3

2. Introduction

2.1 Scope of the Document

This document describes a proposal to reduce the BGP *path exploration* process duration using machine learning technique. The following describes the path exploration problem and the proposed solution.

The *Border Gateway Protocol* (BGP) [4] is a path vector routing protocol used to maintain connectivity among the collection of independently administered *Autonomous Systems* (ASes) in the Internet. Thanks to BGP, *Internet Service Providers* (ISPs) are able to connect to each other and end-users can connect to more than one ISP. The primary function of a BGP router is to exchange network reachability information with other BGP routers. Among others, this information includes a list of ASes that the reachability information goes across. This list allows routers to prune routing loops while some policies at the AS level can be enforced.

However, the slow convergence problem, inherent to path vector protocols, has shown significant impact on the performance of BGP [5]. In response to path failures or routing policy changes, BGP routers may try several transient paths before selecting a new best path or declaring a destination as unreachable. This process, called *Path exploration*, can occur due to topology changes, hardware failure, and new routers or sessions deployment. Also, session failures due to equipment failures, maintenance or due to congestion on the physical links can lead to routing changes. Finally, it can also be due to changes in routing policies after reconfiguration of preferences or route filters, or when policies in different ASes are conflicting.

Routers mainly exchange information through BGP UPDATE message. It is gives information about a specific route in the network. BGP routers use this information to construct the graph describing the relationships among all known ASes. An update message will *advertise* feasible routes sharing common path attributes, or *withdraw* multiples unfeasible routes. Note that each UPDATE message can contain both advertisement and withdraws.

Among all information contained in a router announcement, the AS-Path is of keen interest for us. It is a simple vector of ASes defining in BGP the path used to join a prefix. In this document, we denote an AS-Path as [A_n, A_{n-1}... A_1, A_0], where A_0 is the origin AS to which *d* belongs and A_n the local BGP router. *d* is the destination prefix, announced by AS_0.

Let us consider the topology in Fig. 2.1 and the following events: AS_0 announces a path to destination *d*, this announcement is received at its neighbours, and propagated hop by hop. Finally, when the network converges, AS_5 knows three paths to reach *d*, i.e., [3,2,1,0], [4,2,1,0], and [7,6,1,0] (preferred in that order).

Now, let us consider what happens if the link between AS_0 and AS_1 fails, making *d* unreachable at AS_1. This failure triggers the following sequence of events: AS_1 sends withdrawals to AS_2 and AS_6. In turn, each of them sends withdrawals to their own neighbours. Eventually, AS_5 receive withdrawals from each of AS_3, AS_4, and AS_7 (in some order). Let us suppose the first one send from AS_3. AS_5 then removes the path [3,2,1,0], selects [4,2,1,0] as the best path and sends it to its (other) neighbours. However, if the withdrawal from AS_4

arrives next; then this best route is invalidated and AS_5 selects (and announces) [7,6,1,0]. Finally, after AS_5 receives the withdrawal from AS_7, it invalidates the path announced earlier and sends a withdrawal.

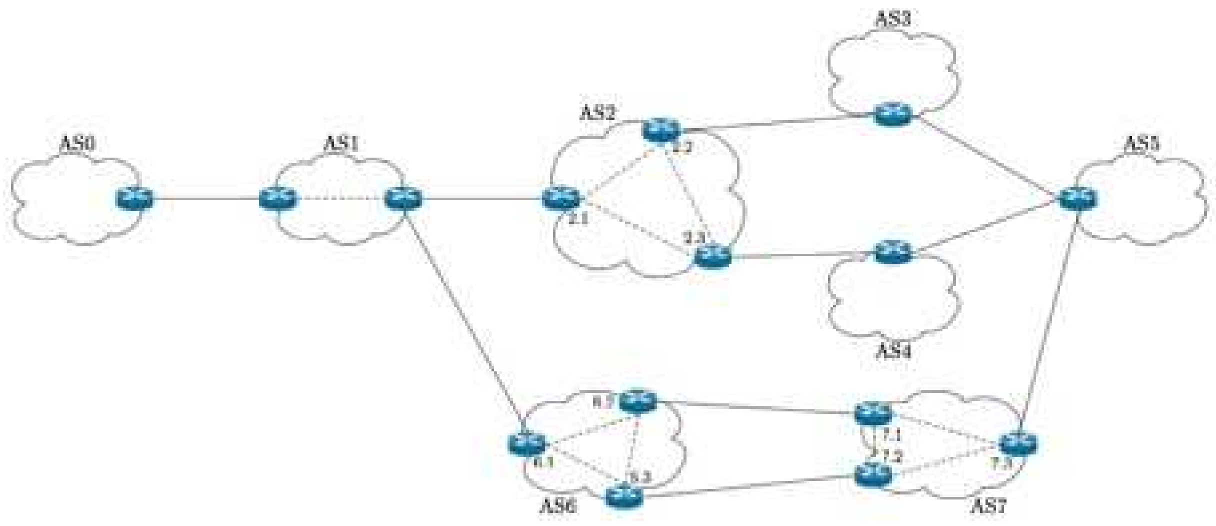


Fig. 2.1: BGP and path exploration. Solid/dashed lines represent eBGP/iBGP sessions

This cycle of selecting and propagating (invalid) paths is called the path exploration. The cycle stops after all the obsolete routes have been explored and invalidated.

In this document, we investigate machine learning techniques allowing to reduce the BGP convergence time due to path exploration. In particular, we focus on two techniques: the *decision trees* [12] and the *hidden Markov model* (HMM) [13]. A decision tree is a decision support tools using a tree-like graph of decisions while an HMM is a statistical model in which the system being modelled is assumed to be a Markov process with unobserved state.

2.2 Structure of the Document

The remainder of this document is organized as follows: Section 3. provides a formal model for the Path Exploration problem; Section 4 investigates the machine learning techniques for detecting Path Exploration and decreasing the BGP convergence time; Section 5 evaluates the performance of the techniques described earlier; Section 6 describes the introduction of the specified functional components into the ECODE common architecture as well as the interfaces and information exchanges between these components; finally, Section 7 concludes this deliverable.

3. Formalization of the Path Exploration Problem

In Border Gateway Protocol (BGP) [4], the Internet inter-domain/inter-AS (autonomous system) routing protocol, a route is defined as a unit of information that pairs a set of destinations (reachability information) with the attributes of a path to these destinations. Routes are advertised between BGP routers in Update messages. The actual path to this set of destinations is the information reported in the AS_Path attribute that enumerates the sequence of Autonomous Systems (AS) numbers that the reachability information has traversed through BGP Update message. This information is sufficient for constructing a graph of AS connectivity (AS routing topology), from which routing loops may be detected and avoided, and, at the AS level, some policy decisions may be enforced.

BGP is a policy-based shortest AS_Path vector routing protocol providing loop avoidance by detection, using the AS_Path information includes in the BGP update messages. This property of exchanging vectors of ASs (or AS_path) that prevent routing loops leads also to the path exploration phenomenon that is the root cause of the observed delays in BGP convergence time [8] [9]. In response to a topological change (e.g. failure) or a policy change (resulting in the advertisement or withdrawal of a given prefix associated to a previously selected AS_Path), path exploration is characterized by a sequence closely coupled in time of advertisements of the same prefix with a longer AS_Path (or a different AS_Path of the same length) followed either by a withdrawal for the same prefix (route to the destination prefix is declared unfeasible) or by stabilizing to a newly preferred AS_Path for the same prefix. The example depicted in Fig.3.1 (where A indicates an advertisement, and W a withdrawal) shows an example of BGP slow convergence due to uninformed path exploration. Note that the resulting effect on convergence time is theoretically: upper bound ~ O(N!) and lower bound = Ω[(N-3) x MRAI timer] where N is the number of AS. Nevertheless, in practice (due to the fact the Internet is not a complete graph), the convergence time can be estimated by the formula (Max_AS-Path - Min_AS-Path) x Minimum Route Advertisement Interval (MRAI) time interval.

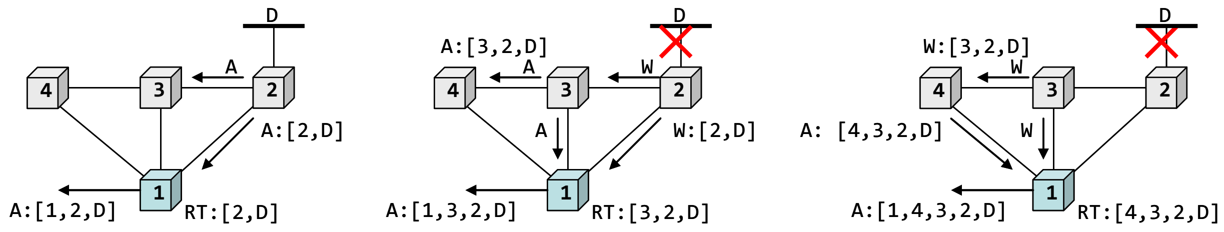


Fig.3.1: Example of Path Exploration event

The AS sequences included in the AS_Path attribute (or simply AS_Paths) of BGP UPDATE messages towards the same destination prefix are correlated in space. Topological correlation(s) in terms of ASs between different AS_Paths toward the same destination prefix result from the meshedness of the Internet AS topology. That is, an acyclic topology, e.g., a tree, does not exhibit path exploration, as there is only one feasible path to each destination. However, exchange of BGP UPDATE messages is spatially uncoordinated that is: there is no sequence number indicating to which instance of the topology the currently used AS_Path corresponds to. Uncoordinated BGP UPDATE message exchange coupled to the topological correlations between AS_Paths, results in a transient phase during which each AS_Path towards the same destination is explored upon topological change.

Path exploration phenomenon can occur both when a prefix is first advertised and when a prefix is withdrawn:

- At prefix advertisement time: resulting from the differential propagation/processing delay across the AS topology that may result into suboptimal selection until the optimal path is being received.
- At prefix withdrawn time: upon prefix withdrawal from a given BGP peer, transient selection by the local BGP speaker of a previously less preferred AS-path (learned from another BGP peer in a previously received route update announcement) that substitutes to the previously received and selected AS-path for the same prefix. This selection results in turn in a BGP UPDATE announcement for that prefix to its BGP peers. If this replacement path is itself subsequently withdrawn, the local BGP speaker will again select another route previously received from a different peer, if one exists. Hence, this process continues until no other transient path (usually of increasing length) in between the BGP router originating the BGP Update message and the local BGP router exist, i.e., is feasible for the same destination prefix.

In both cases, for a given address prefix, the dependency between the AS_Path announced/withdrawn at time t(i) for that prefix and the newly selected (and thus subsequently announced) AS_Path for the same prefix at some time t(i+j), $j > 0$, results potentially in a transient exploration of all the intermediate paths. Each step is characterized by the exploration of all intermediate AS_Paths (usually of increasing AS_Path length) until reaching next best preferred AS_Path. This transient path exploration results in delaying the BGP routing protocol convergence. Intermediate states existence is due to the cycles in the topology (between the failure and the local BGP speaker). Cycles of length n, C_n, are eliminated by the BGP protocol (loop avoidance mechanism) but local BGP speaker retains the sequence toward certain AS even if it has already an AS_Path to that AS. If the latter is the seed of the path exploration phenomenon the local AS will explore all states from C_2 (Min_AS-Path + 1) to C_{n-1} (Max_AS-Path). It is important to emphasize that in the example shown in Fig.3.1, exploring all intermediate state is "useless" in the sense that the address prefix D is unreachable. Hence, by directly detecting such event Router_1 can directly withdraw its route to prefix D after receiving the first BGP UPDATE message from Router_2 (indicating the withdrawal of its route to D).

In this context, we are interested in finding a technique for path exploration sequences detection so as to enforce suppression of sequences of inter-domain BGP routing updates that are detrimental to the routing system convergence. Formally, the objective is to decrease the local BGP convergence time form (Max_AS-Path - Min_AS-Path) x MRAI time interval to a probabilistic time (being the sum of the detection and the alternate best AS-path selection time).

4. Machine Learning Techniques to Detect and Mitigate Path Exploration

4.1. Decision Trees

The problem we are facing is the following: given a sequence of BGP UPDATE messages, we want to predict in advance the outcome. A sequence can either end with a new best path or with a withdrawal. This is thus a binary classification problem. Having information about the outcome of the path exploration process will then allow one to improve the global BGP convergence speed.

The model we develop is the entity in charge of predicting the outcome of a path exploration process. This model needs to fulfil two major constraints: it needs to be efficient and compliant with real-time constraints. Indeed, routers need to take quick decisions about whether aborting or continue the path exploration process. Also, routers may have limited CPU and memory resources.

Several assumptions must be defined in order to be able to define such a model precisely:

- A sequence of UPDATE messages related to a single prefix can be isolated.
- There exists a *path exploration detector* able to detect the beginning and the end of a path exploration process, based on BGP UPDATE messages.

The selected learning model is a tree-based learning. It matches our constraints as the complexity of the tree traversal is logarithmic to the number of nodes in the tree and the size of the tree depends on the number of rules contained in the tree. We chose to base our learning on received BGP UPDATE messages, which means the use of supervised technique to build the tree.

This section is decomposed as follows. Section 4.1.1 describes an overview of the supervised learning technique and details the algorithm used to build the tree. Decision trees, support tools using a tree-like graph of decisions, are detailed in Section 4.1.2. Next, we describe in Section 4.1.3, the selected the tree building algorithm: C4.5 [3]. Section 4.1.4 defines the measurements, i.e., the parameters used to describe the tree and defines the difference between a path exploration process ending with a withdrawal and an announcement. Finally, Section 4.1.5 provides an application to a real example of a path exploration process.

4.1.1. Supervised Learning

The supervised learning algorithm works as follows: given a set of possible outputs, the algorithm generates a function that maps any input to one of the outputs. This may correspond either to a classification problem where the inputs (typically vectors, but not necessarily) are mapped to one of several classes, or to a regression problem where the resulting function maps this time input to a continuous domain.

The data is called the *input* of the algorithm, while the prediction, i.e., the result of the algorithm application, is called the *output*. The learning process uses several input-output pairs as valid *instances* of the function, called *training sample*, and should return a function *h* of the inputs that approximates *at best* the output. This *h* can be used to predict the output for new inputs, as well as to understand the relationship between input and output.

When the output is symbolic, the supervised learning is a classification problem, while a numerical output refers to a regression problem. Being able to determine whether people will appreciate the time spend in a holiday centre, function of the time, is an example of classification problem. We have, for example, the following instances:

Sky	**Temperature**	**Humidity**	**Wind**	**Water**	**Forecast**	**Enjoy?**
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cold	Change	Yes

We may want to predict whether, for some new weather, people will enjoy the time:

Sky	**Temperature**	**Humidity**	**Wind**	**Water**	**Forecast**	**Enjoy?**
Sunny	Cold	Normal	Strong	Warm	Same	?
Rainy	Warm	Normal	Strong	Warm	Same	?

We could have instead sunlight, air and water temperature, humidity, wind, and forecast change measures, that would allow to predict which percentage of tourists will enjoy the time, in which case we would face a regression problem. The prediction space would not be limited to a set of symbolic values but would be of infinite size.

The function h will be restricted by a hypothesis space H defining a family of candidate models. We can take H, e.g., as the set of all linear functions on the inputs, or quadratic functions, or some very complex functions on the inputs.

A *learning phase* allows one to determine the best parameters for the functions. Best values do not mean that we search for the function that fits the best training sample. There comes the test set, which is a subset of the data set (for example 30%) used to estimate the classification performance of a model obtained on the rest of the data set (the 70% left). The performance of a model can be expressed in terms of error rate, which is the percentage of incorrectly classified instances in the data set. Several other measurements can be used as well. The confusion matrix reports predicted class facing the actual class, as shown in Table 4.1. A classification algorithm will be the best when its normalized confusion matrix tends to identity matrix. In a normalized confusion matrix, is composed of element: $Ncm_{ij} = cm_{ij} / N_i$ where cm_{ij} the elements of the confusion matrix and N_i the number of element from the true class i.

		Actual Class	
		X	Y
Predicted Class	X	True Positive (TP)	False Positive (FP)
	Y	False Negative (FN)	True Negative (TN)

Table 4.1: Confusion matrix in which each column represents the number of occurrences of a real class, while each row represents the number of occurrences of the estimated class. This table assumes that only two classes are used, while it could be build for any finite number of classes.

Alternative performance measurements are:

- Correct Classification Rate = 1 - error rate = $\frac{TP+TN}{TP+TN+FP+FN}$.
  We observe that this measure can be very high as soon as one class strongly dominates among samples. Indeed, the dominated class would not visibly influence the measure in case of misclassification.
- TP Rate = sensitivity = $\frac{TP}{TP+FN}$
- Precision = Positive Predictive Value = $\frac{TP}{TP+FP}$
- TN Rate = Specificity = 1 - FP Rate = $\frac{TN}{FP+TN}$

.
We say that some model *overfits* the data if the performance obtained on the learning set is almost maximal, while worst on the test set. Overfitting indicates that the model is too close to the training set, and cannot be generalized to any input.

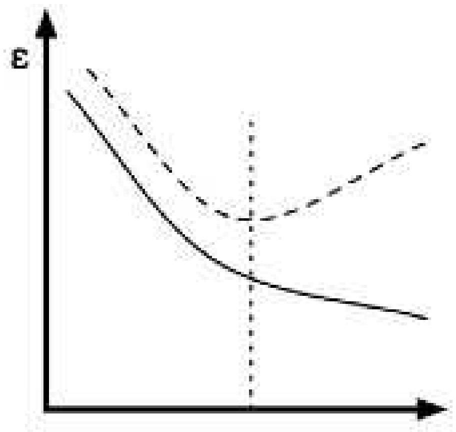


Fig. 4.1: Overfitting in supervised learning

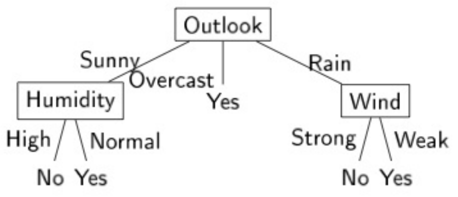
Fig. 4.1 shows an example of overfitting in supervised learning. The training error is plain line and the validation error (test set error) is the dashed line. The overfitting occurs when the training error decrease while the test set error increase, after the vertical dotted line on Fig. 4.1. This point is the optimal complexity of the model.

When not enough data is available to extract a test set, we use *k-fold cross validation*. The data set is partitioned into *k* subsets. For each subset, the model is learnt on data not contained in the subset, while the error rate is calculated on the data of the subset. The mean error rate is over the *k* subsets. A particular case of cross validation is when *k* equals the data size. This is called *leave-one-out cross validation*.

Several supervised learning algorithms exist. However, we only focus on the decision tree algorithm that is the one used to resolve the path exploration problem.

4.1.2 Decision Tree

Decision trees are decision support tools using a tree-like graph of decisions [12]. An example for the weather problem introduced in Section 4.1.1 could be the following:



1. **if** Outlook **is** Sunny **and** Humidity **is** High **then** Enjoy **is** No
2. **if** Outlook **is** Sunny **and** Humidity **is** Normal **then** Enjoy **is** Yes
3. **if** Outlook **is** Overcast **then** Enjoy **is** Yes
4. **if** Outlook **is** Rain **and** Wind **is** Strong **then** Enjoy **is** No
5. **if** Outlook **is** Rain **and** Wind **is** Weak **then** Enjoy **is** Yes

Fig. 4.2: Decision tree and equivalent set of rules directly translated from

This tree verifies several properties:

- Each internal node tests an attribute
- Each branch corresponds to an attribute value
- Each leaf node assigns a classification
- Each path from root to leaf represents a conjunction of attribute values
- The whole tree represents a disjunction of conjunctions

The tree classification for a previously unseen example is accomplished following the path corresponding to observed attribute values.

Fig. 4.2 shows an example of rules derivation from a decision tree. It is interesting to notice that those rules (and therefore the decision tree) offer good human readability and understanding properties thanks to their simplicity.

Trees are built iteratively. The best-known decision tree builder algorithms are ID3 [2] and C4.5 [3]. The next section focuses on the C.4.5 algorithm as it is the one we selected.

4.1.3 C.4.5 Algorithm

The ID3 algorithm, introduced by Quinlan [2], works as follows: for each node, the best decision attribute is chosen to maximize the expected reduction in entropy after sorting on this attribute, also called the *Gain*. The *C4.5* algorithm [3], also introduced by Quinlan, is an improvement of the ID3 algorithm using the information entropy. Compared to ID3, C4.5 introduces two new goals: the avoidance of overfitting to the data and the incorporation of continuous-valued attributes.

On the first side, C4.5 uses a validation set distinct from the training set used to detect overfitting. When increasing the tree depth, if the classification performance increases on the training set and decreases on the validation set, the hypothesis being built is overfitting the training data set. The growth is stopped when the data split is not statistically significant. *Post-pruning* is applied on the whole tree to remove sub-trees whose removals improve validation set accuracy. The post-pruning idea is to infer the tree as well as possible, convert the tree to an equivalent set of rules, and then prune each rule by removing any preconditions that result in improving its estimated accuracy. And finally, sort final rules by their estimated accuracy and consider them in this sequence when classifying.

Regarding continuous-valued attributes, C4.5 selects candidate thresholds midway between instances with distinct classifications. For example, let us consider whether a player will want to play tennis depending on the temperature. We have the following data:

Temperature	5	9	17	24	29	35
Play tennis?	No	No	Yes	Yes	Yes	No

In this case, candidate thresholds are 13 (between 9 and 17) and 32 (between 29 and 35). In the first case, the entropy gain will be *0.1887* and in the later case *0.0290*. C4.5 will then choose to split around value 13: all temperatures lower than 13 and those at least equal to 13.

4.1.4 Measurements

This section describes the measurements that are used to build the model. Those measurements allow us to abstract from the UPDATE messages sequences previously obtained per prefix, and this way to get rid of any reference to IP prefixes and AS numbers. Those measurements lead to build a decision tree to predict, with more accuracy, the binary outcomes of a path exploration process, i.e. it ends with a new path or a withdrawal.

We define five different measurements split into three different categories. *Intensity measurements* aim at detecting bursts of activity. They therefore include a time component in order to adapt differently their values depending on the occurrence frequency for events they are looking at in received UPDATE messages. *Categorical measurements* define several categories in which received UPDATE messages are classified. *Counting measurements* are numerical values taken from samples. In our model, those measurements adapt their values depending on BGP UDPATE messages types and attributes. In the remainder of this section, we deeply discuss each of those measurements.

The Intensity measurements aim at detecting bursts of activity. The measurements follow exponentially weighted sums, meaning that, whenever a new UPDATE message arrives, a measure takes a value according to the arrived UPDATE message and the previous value of this measure, multiplied by some decreasing exponential. Given the new UPDATE message *u* and any positive function *f* of the new UPDATE message to be defined depending on the measure itself, intensity measures are updated as follows:

$$Q_n = f(u) + Q_{n-1}\, 2^{-r\Delta t}$$

The new value of the measurement Q_n is thus function of its previous value Q_{n-1}, with $Q_{n-1} = 0$ when $n = 0$. Those measurements aim at detecting bursts of activity, they then need to contain a temporal component. Δt is the time elapsed since last received update. R is the decay factor, determining the half-life of the data used.

We define two intensity measurements:

- *BGP update arrival frequency (M1)*, where $f(u) = 1$.
- *Number of new AS-paths (M2)*, where $f(u) = N$, N is an indicator set to *1* if the UPDATE message contains an AS-path not recently observed, *0* otherwise. It intends to capture the variation in the number of AS-paths. Upon link/router failure occurrence, BGP experiences slow convergence problem. During this phase, BGP routers may receive a number of UPDATE messages with a previously unseen AS-Path, or seldom seen.

The Categorical measurements are based on the sampled message itself but also on the previous ones (from the same update sequence).

We define two categorical measurements:

- *BGP UPDATE type (M3)*, the idea is to classify the message into eight possible types. The measure is extracted from the decision tree depicted in Fig. 4.3 based on previous messages. Each leaf of the tree represents a class (values from *0* to *7*).
- *AS-path occurrence frequency (M4),* this measure intends to capture whether an AS-path is frequently observed or not. As AS-path diversity tends to increase during the path exploration process.

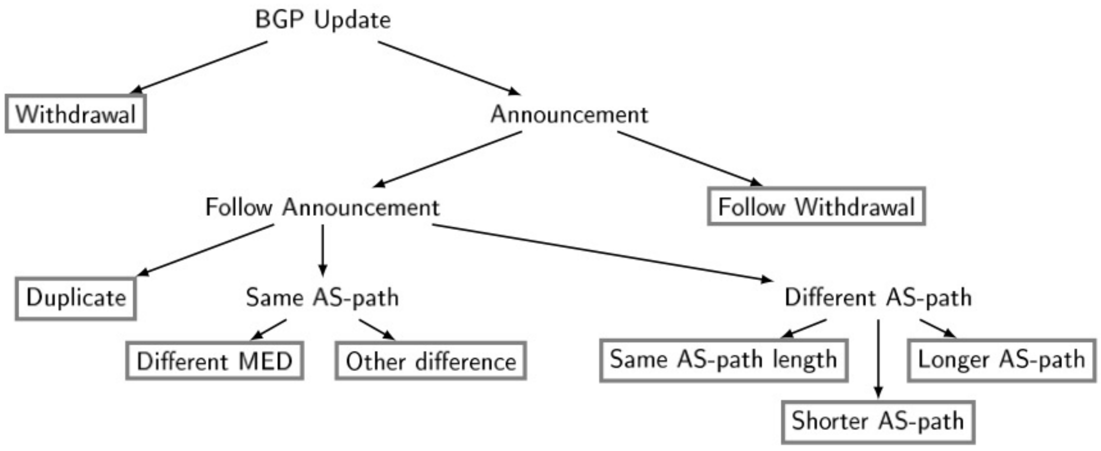


Figure 4.3.: BGP update type measure classification tree

We consider a single Counting measurement:

- *AS-path difference (M5)*, this measure intends to measure the difference between the AS-path received with the previous ones. The difference is calculated using the *Levenshtein distance* [6].

The *Levenshtein distance L* returns the minimum number of operations needed to transform one vector into another. The possible operations are insertion, deletion, or substitution of a single element in the vector. For example:

$$L([34, 21, 19, 42],[78, 34, 15, 19]) = 3.$$

Because we need:

One addition:	[34, 21, 19, 42]	$\Rightarrow$ [78, 34, 21, 19, 42].
One deletion:	[78, 34, 21, 19, 42]	$\Rightarrow$ [78, 34, 21, 19].
One substitution:	[78, 34, 21, 19]	$\Rightarrow$ [78, 34, 15, 19].

The most stable AS-path is the one shared by most messages in the UPDATE message sequence. It is the one with the most *AS-path occurrence frequency*.

4.1.5. Application to a Real Example

Table 4.2 gives an example of a path exploration process (chronologically ordered). The topology that generated this sequence is represented in Fig. 4.4. Our model works on a per prefix basis, therefore the sequence always ends either with an announcement or a withdrawn.

Timestamp	Type	AS-path
1023	Announcement	[13, 11, 0, 58]
3537	Announcement	[13, 0, 58]
360029	Announcement	[13, 1, 0, 58]
362509	Announcement	[13, 18, 16, 0, 58]
365114	Announcement	[13, 11, 14, 1, 0, 58]
367542	Announcement	[13, 28, 15, 1, 12, 0, 58]
370225	Announcement	[13, 14, 11, 10, 15, 1, 12, 0, 58]
370932	Withdrawal	
378682	Announcement	[13, 16, 24, 26, 19, 11, 10, 15, 1, 12, 0, 58]
379157	Withdrawal	

Table 4.2: UPDATE messages received by an atomic AS part of a 30-Ases-clique topology for which a prefix is announced before being withdrawn.

Data from Table 4.2 is obtained upon simulation of a prefix announcement then withdrawn in the topology represented in Fig. 4.4. The table represents the whole path exploration process for a single prefix, showing the different needed parameters. The timestamp is the time at which the BGP UPDATE message has been received. The type is the BGP UPDATE type, whether an announcement or a withdrawal. The AS-Path is, as explain before, a vector of ASes.

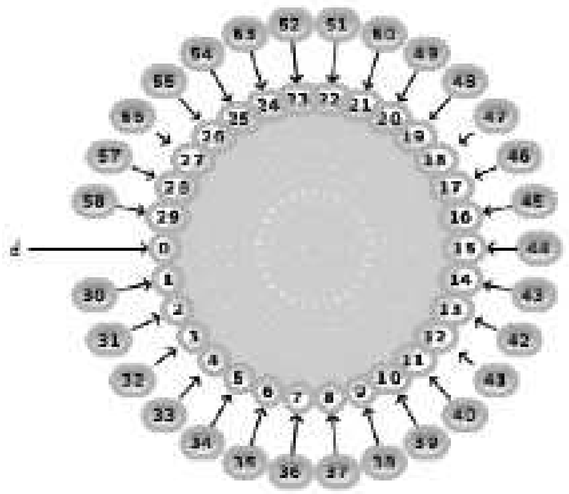


Figure 4.4: A 30-ASes-clique topology (theoretical topology).

The first step is the pre-processing. This is a direct application of measurements to UPDATE message sequence. For example, the first message of the UPDATE messages sequence corresponds to the first row measurements. The global table must then be stored.

Table 4.3 is the pre-processing result obtained with the update messages of Table 4.2.

M1 (r=0.01)	M1 (r=0.5)	M2 (r=0.0333)	M3	M4	M5
10	10	10	0	10	0
1,84008E+11	1,00016E+11	1,55974E+11	7	5	0
1E+11	10	10	6	5	1
1,84206E+11	1,00019E+11	15641536319	6	5	2
2,53775E+11	1,00012E+11	1,85733E+11	6	333333333333	2
3,14466E+11	1,00022E+11	2,06047E+11	6	25	0
3,61101E+11	10000915754	2,10921E+11	6	2	3
4,43831E+11	1,08628E+11	1,79163E+11	1	166666666667	8
3,5937E+11	10	1,29948E+11	0	2	0
4,4773E+11	1,19278E+11	1,16454E+11	1	4	0

Table 4.3: Measurements based on the sequence of UPDATE messages of Table 4.2. The pre-processing permitted to abstract from any AS number or prefix. The result is now ready for machine learning.

We cannot consider vectors of measurements as input in the machine learning process. We still have to extract samples from them. The objective of the learning is to be able to predict whether a sequence of update messages will end either in an announcement or a withdrawal. When we have enough data samples, the C4.5 algorithm is executed to generate a decision tree, as described in Section 4.1.3.

Considering the measurements related to the last update of a sequence to apply the machine learning process would be non-sense. Indeed, we want to predict the outcome of the sequence before it happens. It would be like trying to forecast today’s weather while we could simply look outdoors. We therefore do not consider updates after a threshold *k*. If this threshold is too small, we have only few recent historical data before taking our decision. The threshold *k* is considered from the precise time we know path exploration should have begun. Our sample related to the simulation is simply the *(k+1)th* update message, after that time. To have a trade-off between a small *k* to take fast decision and large enough *k* to have enough history and to fill time window, we chose for *k* the value *4*. This means that we predict the outcome after the fourth message of a sequence, applying this message to the decision tree.

4.2. Hidden Markov Model

We are interested in designing a technique for path exploration sequences detection so as to enforce suppression of sequences of inter-domain BGP routing updates that are detrimental to the routing system convergence. These sequences are responsible for the detrimental effects on BGP convergence time (since local BGP speaker propagates transient routing states to its downstream neighbours). The objective is to decrease the local BGP convergence time form (Max_AS-Path - Min_AS-Path) x MRAI time interval to a probabilistic time being the sum of the sequence detection and the alternate best AS-path selection time.

To detect path exploration phenomenon in BGP update sequences (modelled as sequences of AS-path sequences), we can design the solution around the following procedure:

- Step_1: path exploration event detection, i.e., determine whether an incoming AS-Path sequence is actually associated to a path exploration event or not.

- Step_2: identification of the largest AS-Path sub-sequence associated to this event before reaching end of the path exploration event to expectedly minimize the impact of the corresponding BGP UPDATEs on downstream neighbours.

- Step_3: anticipate BGP decision of the route selection process upon path exploration event identification. This last step comprises the selection of the alternate best AS_Path that will be advertised to BGP peers.

4.2.1 BGP communication channels as Markov chains

Consider a system which may be described at any time as being in one of a set of N distinct states (S_1, S_2,…, S_N). At regularly spaced discrete times, the system undergoes a change of state (possibly back to the same state) according to a set of probabilities associated with the state. We denote the time instants associated with state changes as t = 1, 2,. . . , and we denote the actual state at time t as q_t. In the particular case of a discrete, first order, Markov chain, this probabilistic description is truncated such that the probability of the current state (at time t) depends only on the preceding state (at time t-1).

$$P[q_t=S_j \mid q_{t-1}=S_i, q_{t-2}=S_k, \ldots, q_1=S_1] = P[q_t=S_j \mid q_{t-1}=S_i] = P[q_{j,t} \mid q_{i,t-1}] \qquad \text{(Eq.1)}$$

If we only consider those processes in which the right-hand side of (1) is independent of time, thereby leading to the set of state transition probabilities a_{ij} = P(q_j at time t+1|q_i at time t) = P($q_{j,t+1}$|$q_{i,t}$) with $1 \leq i,j \leq N$ such that $\sum_{j=1}^{N} a_{ij} = 1$ (Eq.2). This stochastic process can be referred to an *observable Markov model* since the output of the process is the set of states at each instant of time, where each state corresponds to a physical (observable) event.

During path exploration events the BGP routing system transitions between different states. The transition between the different states is governed by a Markov chain. However, this Markov chain is not directly observable -it is hidden- as the Path exploration event result from uninformed BGP UPDATE processing. Nevertheless, the received sequences of AS-Paths (in the locally processed BGP UPDATE messages) provide probabilistic information about the current state of the BGP routing system (at least the upstream part of the network that is under interest). The BGP routing system state is an abstract variable, representative of the effect of all BGP UPDATE message exchanges on the incoming BGP communication channel and thus on the local BGP decision process. As the local router decision process has no direct access/knowledge of the global routing system state, the routing system state is modelled as a hidden variable. This hidden variable that can only be perceived by the local router through the sequence of observations as produced by the BGP communication channel (modelled as a stochastic process) whose sequential output represents the sequence of AS-paths as received by the local route selection process from the incoming BGP UPDATE messages. In summary, the idea is that the observed BGP AS-Path sequence at a given time can be explained as a random function of an unobserved global state of the network (undergoing topological change), which follows a Markov chain. Note that the proposed model does not limit applicability per destination prefix but per cluster of destination prefixes undergoing the same state transitions.

4.2.2 Hidden routing state as Hidden Markov Model

To include the case where the observation is a probabilistic function of the state, i.e., the resulting model is a doubly embedded stochastic process with an underlying stochastic process that is not observable (it is hidden) but can only be observed through another set of stochastic processes that produce the sequence of observations, the model is extended to so-called Hidden Markov model (HMM). Such model represents stochastic sequences as Markov chains where the states are not directly observed but are associated with a probability density function (pdf).

The generation of a random sequence is then the result of a random walk in the chain (i.e. the browsing of a random sequence of states and of a draw (called an emission) at each visit of a state. The sequence of states, which is the quantity of interest can be observed only through the stochastic processes defined into each state (i.e. you must know the parameters of the pdfs of each state before being able to associate a sequence of states $Q = q_1, q_2, \ldots, q_T$ to a sequence of observations $O = O_1, O_2, \ldots, O_T$ where T is the number of observations in the sequence. The true sequence of states is therefore hidden by a first layer of stochastic processes. HMMs are dynamic models, in the sense that they are specifically designed to account for some macroscopic structure of the random sequences.

The hidden Markov Model (HMM) with N hidden states and M distinct observation symbols per state that correspond to the physical output of the system being modelled, is defined by three probability distributions:

- *(State) Transition probability distribution*: probability a_{ij} to go from a state i (q_i) to a state j (q_j) is given by $P(q_j \text{ at time } t+1 | q_i \text{ at time } t) = P(q_{j,t+1} | q_{i,t})$ with $1 \leq i,j \leq N$. They are stored in matrix $\mathbf{A} = \{a_{ij}\}$ where each term a_{ij} denotes a state transition probability $P(q_{j,t+1} | q_{i,t})$.

- *Observation probability distribution*: the probability distribution of emitting an observation vector O in state j, is given by $\mathbf{B} = \{b_j(O)\}$, where $b_j(O) = P(O \text{ at time } t | q_j \text{ at time } t) = P(O_t | q_{j,t})$. In the discrete case, each observation O_t takes its values from the library of M observation symbols corresponding to the physical output of the system being modelled. The emission probabilities are the pdfs that characterize each state q_j.

- *Initial state distribution:* $\mathbf{\Pi} = \{\pi i\}$ where $\pi_i = P(q_i \text{ at } t = 1) = P(q_{i,1})$ gives the initial probabilities.

There are three basic problems that must be solved for the model $\lambda = (A,B,\Pi)$ to be useful in real-world applications.

Problem 1: Given the observation sequence $O = O_1, O_2, \ldots, O_T$, and the model $\lambda = (A,B,\Pi)$, the problem is to compute $P(O|\lambda)$, the probability of the observation sequence O, given the model l. This problem is solved by the Forward and Backward algorithms.

Problem 2: Given the observation sequence $O = O_1, O_2, \ldots, O_T$, and the model $\lambda = (A,B,\Pi)$, find the optimal state sequence $Q = q_1, q_2, \ldots, q_T$ associated with the given observation sequence. A formal technique for finding the single best state sequence (that maximizes $P(Q|O)$ given the model λ) exists based on dynamic programming methods, called the Viterbi algorithm [14], [15].

Problem 3: Determine a method to adjust the model parameters A, B, and Π to maximize the probability of the observation sequence O given the model. There is no known way to analytically solve for the model which maximizes the probability of the observation sequence. In fact, given any finite observation sequence as training data, there is no optimal way of estimating the model parameters. One can, however, choose $\lambda = (A,B,\Pi)$ such that $P(O|\lambda)$ is locally maximized using an iterative procedure such as the Baum-Welch method.

4.2.3 Learning model

A classifier is a function h that maps observed AS-paths (O) to BGP state event classes. The goal of the learning process is to find a function h that correctly predicts the class h(O) of new AS-path(s). This is accomplished by searching some space H of possible classifiers for a classifier that gives good results on the training data without overfitting.

A training example is a pair (O,q), where q is the label associated to state q and we will refer to a set of N such examples as the training data. The training data actually consist of sequences of (O,q) pairs. These sequences exhibit sequential correlation. That is, nearby O and q values are likely to be related to each other. For example, before occurrence of a topological change, all of the q label values will be “no AS-Path change”. Afterwards, all of the q label values will be “AS-Path increase”. Such patterns are important because they can be exploited to improve the prediction accuracy of our classifier. In our case, it is only possible to explore sequences by looking at the distribution of typical (legitimate) sequences and then to see that this distribution changes when the BGP routing state changes. The goal is to construct a classifier h that can correctly predict a new label sequence q = h(O) given an input sequence O.

HMM models the probability of the simultaneous occurrence of the observations O and routing system events Q, that is, it is a representation of the joint distribution P(O,Q). Our problem consists in classifying observed AS-Path sequences with as purpose the accelerated detection of path exploration sequences and subsequent selection (or generation) of the adequate AS-Path after a minimum number of path exploration hits have been reached.

Each state of the BGP routing system is modelled as HMM state. Four hidden states are defined for the HMM. The output of the HMM populates the Loc-RIB. Note that the model can be applied per destination prefixes or cluster of prefixes. Observations O are modelled as AS-Paths as received by the BGP route selection process thus, before execution of the BGP route selection process. Note that AS-Path attribute information is extracted on a per-destination prefix basis. If we defined A as the announcement of a prefix (with no change in AS-path or attributes), and

- A^+ : announcement of a prefix with an increasing AS-path length (update to longer AS-Path)
- A^*: announcement of a prefix with the same AS-path but different attributes (update of attributes)
- A^0: announcement of a prefix with a different path of the same length (update to a different AS-Path of same length)
- $W(A)$: withdraw of an AS-Path A for a given prefix

Then, a prevalent form of path exploration is the sequence of increasing AS path length (for an already announced prefix), followed by a withdrawal W, all closely coupled in time: A, $\{A^+, A^0, A^*\}$, W. The A^+, A^0 and A^* updates are intermediate updates representing transient routing states. The general sequence of announcement/withdrawal of interest, i.e., that is characteristic of path exploration event, can be represented as follows:

$$A_0, W(A_0), \{A^+, A^0, A^*\}_1, W(A_1), \{A^+, A^0, A^*\}_2, W(A_2), \ldots$$

This sequence can either terminate with a withdrawal or stabilization to a newly preferred best path defined as A, $\{A^+, A^0, A^*\}^c$ where the minimum value of the integer c = 1 and maximum value of the integer c = n-1. Indeed, the seed of the path exploration phenomenon the local AS may explore all states from cycles of length C_2 (Min_AS-Path + 1) to C_{n-1} (Max_AS-Path).

At this point in time it is important to emphasize that such sequences of AS-paths are the result of the BGP route selection process that populates the Loc-RIB. The actual sequence of incoming BGP routing updates as maintained in the Adj-RIB-In is, e.g., of the form

$$\text{Adj-RIB-In: } A_0, \{A^+, A^0, A^*\}_1,\ldots,\{A^+, A^0, A^*\}_m, W(A_0), W(A_1), \ldots, W(A_{m-1}), W(A_m)$$

The corresponding AS-Path as selected by the BGP route selection process is:

$$\text{Loc-RIB: } A_0, \{A_0\}_1,\ldots,\{A_0\}_m, A_1, A_2,\ldots, A_m, \text{none}$$

The sequence of announcement/withdrawal as advertised by the local BGP router to its downstream neighbours is actually (where $A_1 = \{A^+, A^0, A^*\}_1,\ldots, A_m = \{A^+, A^0, A^*\}_m$)

$$A_0, \{A_0\}_1,\ldots,\{A_0\}_m, W(A_0), A_1, W(A_1), \ldots, W(A_{m-1}), A_m, W(A_m)$$

Hence, once the Adj-RIB-In pattern is known, the purpose is to detect as part of the flow of incoming BGP UPDATEs, the AS-path (sequence) that will lead to such pattern of outgoing BGP UPDATEs. As such, we want to determine the most probable state sequence for reaching a “path exploration hit” given a certain observation sequence O: $p(q|O)$. The corresponding observation sequence is then removed from the BGP route selection process such as to directly lead to a BGP UPDATE that does not expose the transient local decisions. The actual sequence of incoming BGP routing updates as maintained in the Adj-RIB-In is, for instance of the form

$$\text{Adj-RIB-In: } A_0, \{A^+, A^0, A^*\}_1,\ldots,\{A^+, A^0, A^*\}_m, W(A_0), W(A_1), \ldots, W(A_{m-1}), W(A_m)$$

By detecting that the observation sub-sequence $\{A^+, A^0, A^*\}_1,\ldots,\{A^+, A^0, A^*\}_m$, $W(A_0)$, …, $W(A_m)$ is part of an exploration sequence, the corresponding AS-Path as selected by the BGP route selection process is the one given by the following sequence of announcements:

Loc-RIB: A_0, $\{A_0\}_1,\ldots,\{A_0\}_m$, none

The sequence of announcement/withdrawal as advertised by the local BGP router to its downstream neighbours would then become

A_0, $\{A_0\}_1,\ldots,\{A_0\}_m$, $W(A_0)$

The HMM proposed to models the routing system states per (set of) destination prefix (undergoing the same state transitions) is characterized as follows:

- **N** (the number of hidden states in the model): 4
  These four states are defined as follows: State_1 (S1): No AS_Path change, State_2 (S2): Re-initialization, State_3 (S3): AS_Path Increase, State_4 (S4): Path Exploration Hit. These states actually characterise hidden global routing states that are not directly observable at the local BGP router. The term "Global" shall be understood here as the part of the routing system that is "upstream" to local router wrt to the ASes involved in the path exploration event.
- **M** (the number of distinct observation symbols per state): observation symbols correspond to the output of the system being modelled. In the present case, these symbols correspond to the AS sequences as received in BGP UPDATE messages and processed by the BGP route selection process that populates the Loc-RIB.
- **A**: the state transition probability distribution $a_{ij} = P(q_{j,t+1}|q_{i,t})$ with $1 \leq i,j \leq N$ correspond to the individual state transition of the routing system state.
- **B**: the observation probability distribution in state j, $b_j(O) = P(O_t \,|q_{j,t})$.
- **Π**: the initial state distribution

4.2.4 Classifying AS-Path Sequences

Assume that the cost function L(i,j) gives the cost of assigning state label value i to an example whose true label is j. The goal is to find a classifier h with minimum expected cost. In our case, the cost function translates the impact of missed path exploration events and impact of events not timely detected. One approach for developing such a classifier is to learn a conditional probability density estimator P(Q|O) and then classify a new observation sequence O according to the formula which chooses the class whose expected cost is minimum.

$$q = \arg\min_i \sum_j P(j|O)L(i,j) \qquad \text{(Eq.3)}$$

Incorporating the cost functions into our classifying task of AS-Path sequences consists in predicting the (conditional) joint distribution of all of the labels in the output sequence: $P(q_i|O_i)$. If this joint distribution can be accurately predicted, then the cost function can be evaluated, and the optimal decisions can be chosen. As we are only interested in classifying the entire sequence correctly, the objective consists in predicting $\text{argmax}_{q_i} P(q_i|O_i)$ correctly. Predicting q_i given an observed sequence O_i depends on the nature of the cost function. Because the HMM is a representation of the joint probability distribution P(O,Q), it can be applied to compute the probability of any particular q_i given any particular O_i: $P(q_i|Oi)$.

Hence, for the cost function $L(q'_i,q_i)$, the optimal prediction is given by Eq.3. However, as the length T of the sequences can be very long, the direct evaluation of this equation requires $O(N^T)$ probability evaluations (where N is the number of labels) can be impractical. As the cost function depends on the entire sequence, this computation can be performed in $O(N^2L)$ time. In this case, finding the q'_i with the highest probability consists in computing:

$$q'_i = \mathrm{argmax}_{qi}\, P(q_i|O_i) \qquad \text{(Eq.4)}$$

This expression can be computed by means of the Viterbi algorithm [14], [15]. This dynamic programming computes, for each class label l and each time step t, the probability of the most likely path starting at time 0 end ending at time t with class l. When the algorithm reaches the end of the sequence, it has computed the most likely path from time 0 to time t_i and its probability.

5. Experimentation

5.1. Performance Objectives and Evaluation Criteria

The different directions we experiment are depicted in Fig 5.1 that defines a space that we call the experimental space. This space suggests three possible deviations from a *study set*. A study set is a well-defined environment, specifying the topologies we are considering, their sizes, and from where we are observing them. In the following subsections, we explain in details how we use the various directions depicted in Fig. 5.1 to evaluate our model.

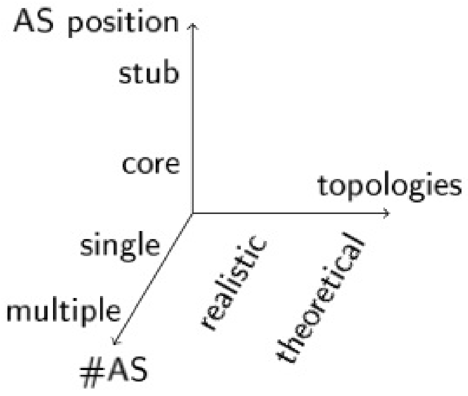


Figure 5.1: Different model experimentation directions

5.2. Methodology: Scenarios and Tools

5.2.1. Topologies

To evaluate our decision tree model, we define topologies used in the evaluation process, in order to explore a number of scenarios and infer information on the model quality.

We define two types of topologies: *theoretical* and *realistic*, each of them being a network interconnecting ASes (limiting to a single router per AS) through BGP.

In the following, we refer to *d* as the prefix to be announced and withdrawn. We also refer to *listeners* as the routers collecting BGP UPDATE message.

Theoretical topologies are defined by mathematical attributes. They all depend on a scale factor *n*: the larger *n*, the larger the topologies in terms of ASes number.

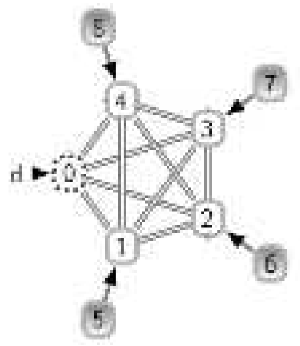
In the subsequent figures, we define:

- $\boxed{0}\!-\!\!-\!\!-\!\!-\boxed{1}$ to be a peer relationship between AS_0 and AS_1.
- $\boxed{0}\longleftarrow\boxed{1}$ to be a provider to customer relationship. The provider and customer being respectively AS_0 and AS_1.
- $\boxed{0}$ to be a core AS.
- $\boxed{1}$ to be a stub AS.
- $\boxed{1}$ to be other AS.

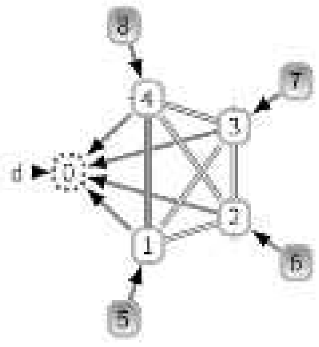
The AS announcing prefix *d* is called the *announcer* and sets the advertised prefix origin to its router identifier.

We define two theoretical topologies:

- ***n*-nodes clique**, *n* ASes are in a full mesh, so that any two different ASes are neighbors. Route selection is based on the shortest AS-path. After a single prefix withdrawal, each router can potentially enumerate a large number of alternate paths to the origin.



- ***n*-nodes p-clique**, For all AS_i with $i \leq n$, we chose AS_0 to be considered as a provider network, while all other ASes are considered as peer networks. Peers pass on route advertisements learned from their provider, AS_0, to all other peers.



The Simula Research Laboratory [10] define a baseline model for ASes realistic topologies, following several properties of today's Internet topology such as hierarchical structure, power-law degree distribution, strong clustering as well as a constant average path length.

From this model, they propose different deviations to be compared with the baseline model such as:

- **Dense core**: multi-homing in the core of the network is much stronger
- **Dense edge**: edge ASes (Tier-3) have a higher multi-homing degree
- **Strong core peering**: core ASes (Tier-1) share more peering links
- **Strong edge peering**: edge ASes share more peering links

Usually, we refer to core and stub ASes as, respectively, ASes with no provider belonging to Tier-1 (T1) and the rest. However, we observe that 10,000 ASes topologies contain only, on average, six core ASes. We, therefore, choose to pick the core ASes into the first 10% of all ASes. Those 10% ASes always contains all T1 ASes, but also Tier-2 (T2).

For stub ASes, we select only 50% of All ASes, to reduce the number of real stub ASes. All those ASes belong to Tier-3 (T3).

5.2.2. Data Generation

To generate and observe a path exploration, we need to provoke it. Two kinds of scenarios have been defined, to provoke path exploration in simulated topologies:

- **W scenario**, a sequence of advertisements leading to the withdrawal of the concerned prefix.
- **A scenario**, a sequence ending with a prefix still available.

Table 5.1 summarizes the scenarios and defines times in order to get network convergence before a new event occurs.

Data are generated using the *SimBGP* BGP simulator [11]. BGP UPDATE messages are then collected and sorted per router, and then per prefix. The number and the location of listeners vary and define the amount of available data to train and test our model. Each study set groups one or more right cuboids in the experimental space that can contain several points with different characteristics.

Time (sec)	W scenario	A scenario
10	Announce	Announce
3600	Withdraw	Withdraw
7200	-	Announce

Table 5.1: Scenarios definition

5.3. Experimental Results

First simulations on theoretical and realistic topologies show good results as most of their classification success rates are above 95% [1]. We do not detail those results here. However, in topologies giving the worst classification success rates (*n*-nodes p-clique, 82%), the model is unable to distinct precisely the outcome of a path exploration. This is due to the fact that the path exploration after a withdrawal and after the announcement looks very similar in those topologies.

5.3.1. Listener(s) Impact

Routers can base their data samples on their own local collected sequence of UPDATE messages. They can also use other samples in order to maybe build a stronger and more accurate model.

In this section, we look at whether it is interesting to use other information, how much it is needed to increase the accuracy and which is more interesting in terms of location, either core or stub.

Fig. 5.2 and Fig. 5.3 show respectively the number of listener impact in the core ASes and in the stub ASes. These simulations are done on the realistic topologies.

As a first observation, as expected, we note that increasing the number of information increases the average model performance. The location of the listeners (either stub or core) has only an impact on the performance when more than 1000 listeners are used. After this point, performance of stub listeners does not increase, as stub listeners have less information compared to core listeners.

To obtain good performance, based on these simulations on realistic topologies, one needs to use more than 200 routers information. This seems difficult to implement in the real world.

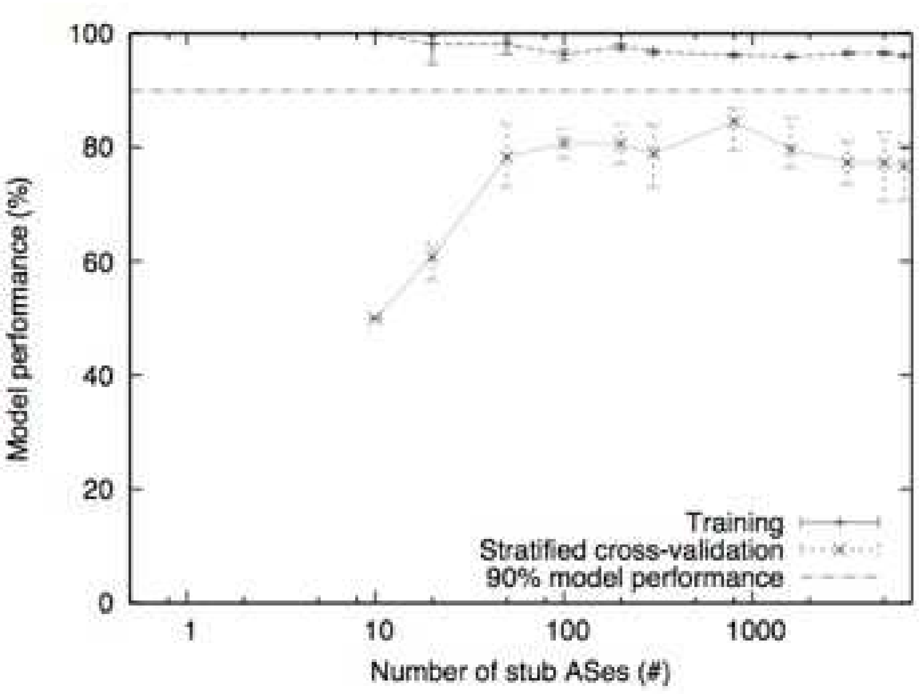


Figure 5.2: Impact of the number of listeners on model performance when listeners are fist chosen among core ASes.

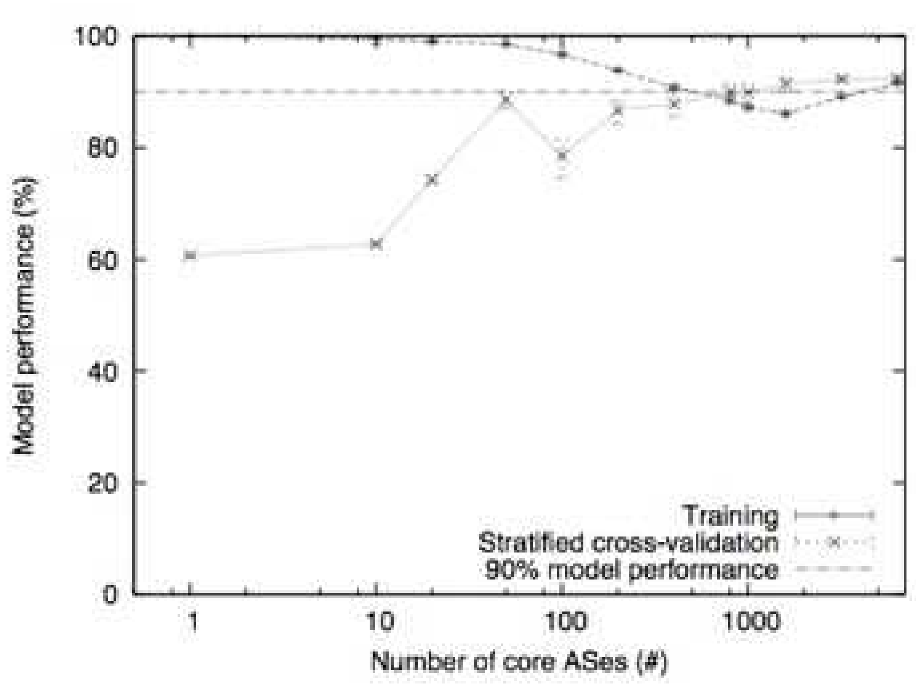


Figure 5.3: Impact of the number of listeners on model performance when listeners are first chosen among stub ASes.

We now evaluate the listeners location. We are interested by the influence of listeners location on the model performance. In the following we give results in terms of performances of the models. Simulations are performed 25 times, the tables giving the mean, the minimum and the maximum performance on those 25 runs.

- Training set with samples from core ASes
  - Test set with samples from core ASes:

	Mean	Min	Max
Train	93.12%	92.88%	93.34%
Test	92.91%	92.71%	93.1%

  - Test set with samples from stub ASes:

	Mean	Min	Max
Train	93.09%	92.84%	93.34%
Test	91.93%	89.9%	93.92%

- Training set with samples from stub ASes:
  - Test set with samples from core ASes:

	Mean	Min	Max
Train	95.59%	95.33%	95.9%
Test	89.3%	88.2%	89.77%

  - Test set with samples from stub ASes:

	Mean	Min	Max
Train	95.47%	95.2ç%	95.42%
Test	95.28%	95.06%	95.42%

Those results show that using the model in the same environment as the one it was built in gives the best classification performance. The best performance for a stub AS is obtained with a model learnt in other stub ASes, while the best performance for a core AS is obtained using a model learnt with samples from other core ASes. We also observe that samples from core ASes generalize more to samples from stub ASes than the opposite.

5.3.2. *Changing Environment*

In this section, we evaluate the impact of small modifications to topologies on the model performance. The idea of the simulation is the following: from a training set composed of all instances collected in all realistic topologies except one we built a model. We, then, test the model on the latter. Plotted results can be found in [1].

The extracted observations are the following:

- We obtain better performances with realistic topologies than with theoretical ones. It looks like models being built on those topologies generalize better for other topologies of the same kind.
- Baseline and strong-core peering topologies offer the best test performance, indicating that those topologies are well decided thanks to our model.

- Learning our model on all theoretical topologies but the p-clique one does not provide good results when testing this model on p-clique topologies.

We are also interested in the accuracy of the learning when a model is placed in a completely different environment. Mixing all topologies, all sizes together and listeners both at core and stub ASes, we build one study set containing theoretical topologies, and one for realistic ones. Results are:

	Test at realistic	Test at theoretical
Training at realistic	95.62%	50.09%
Training at theoretical	69.53%	96.41%

These results show that it is not a good idea to mix tree learnt in realistic topologies for theoretical ones, and vice-versa. Therefore, the model is not able to abstract enough from topologies. It means, that implementing it on the real Internet could need some learning time, as we cannot use simulated tree, to abstract its model.

5.4. Future Work

Future work will be conducted along the following directions:

1. *Include Explicit State Duration Density*: in a Markov chain, the (discrete) probability density function of duration in state i is an exponential function. Thus, in conventional HMMs, the inherent duration probability density $p_i(d)$ associated with state q_i, with self transition coefficient a_{ii}, that is, the probability of d consecutive observations in state q_i, is an exponential function. Such function is inappropriate as the BGP UPDATE message exchanges is governed by the MRAI timer that regulates by rate limiting (in fact introduces a temporal correlation) exchange BGP UPDATEs. However, there is a need to associate an explicitly model duration density in an analytic form to each state. The differences between HMMs without and with explicit duration density is that in the former case, the states have exponential duration densities based on self-transition coefficients whereas in the latter case, the self-transition coefficients are set to zero, and an explicit duration density is specified. In the latter case, a transition is made only after the appropriate number of observations occurred in the state (as specified by the duration density).

2. *Explorationless withdraws*: not all withdraws shall be considered as part of a state sequence leading to a path exploration hit. Indeed, the HMM shall account that only $W(A_0)$ is a trigger for exploration hit, withdraws of intermediate states associated to announcements A_1, …, A_m occurring before $W(A_0)$ shall not be considered part of an exploration sequence. Taking this effect into account would lead to introduce a new state in our model.

3. *HMM structure*: offers a limited model of the true process producing the data. Part of the problem stems from the Markov property as any relationship between two separated q values (q_1 and q_4) must be communicated via the intervening q's, i.e., q_3 and q_2. A first-order Markov model, i.e., where $P(q_t)$ only depends on q_{t-1}, cannot in general capture these kinds of relationships. The second issue with the HMM model is that it generates each O_t only from the corresponding q_t. Several directions have been explored in the scientific literature to overcome the limitations of the HMM: Maximum Entropy Markov models (MEMMs), Input-Output HMMs (IOHMMs), and conditional random fields (CRFs). All of these are

conditional models that represent P(Q|O) rather than the joint probability distribution P(O,Q). They do not try to explain how the observation sequences O are generated but instead try to predict the q values given the observation sequences. This permits them to use arbitrary features of the observations including global features, features describing non-local interactions, and sliding windows. Nevertheless, MEMM and IOHMM models suffer from the so-called label bias problem (note that HMM does not suffer from the label-bias problem). Conditional Random Field [16] overcomes the label bias problem. In CRF, the relationship among adjacent pairs (q_{t-1}, q_t) is modelled as a Markov Random Field conditioned on the observation sequences (used as input), i.e., the influence between adjacent q values is determined by the input. Hence, we will consider CRF to determine if we can reduce the error rate observed with HMMs.

6. Recommendation for integration into common ECODE architecture

After exposing a summary of the ECODE common architecture; we explain the way the procedures and the information exchanges they involve, as introduced by this document, fit within the ECODE common architecture described in Deliverable D2.1.

A standard router comprises a forwarding engine (as part of its forwarding plane) and a routing engine (as part of its control plane). The forwarding engine includes a packet processor and a Forwarding Information Base (FIB). The routing engine includes a routing information processor and Routing Information Base (RIB). The RIB stores the routes and (in some cases) the metrics associated with those routes to particular network destination prefixes. This information contains the topology of the network immediately around the router. The FIB is used to find the proper interface to which the input interface should send a packet to be transmitted by the router. The FIB is constructed based on the RIB and according to policies defined by the operator. It is optimized for fast lookup of destination addresses.

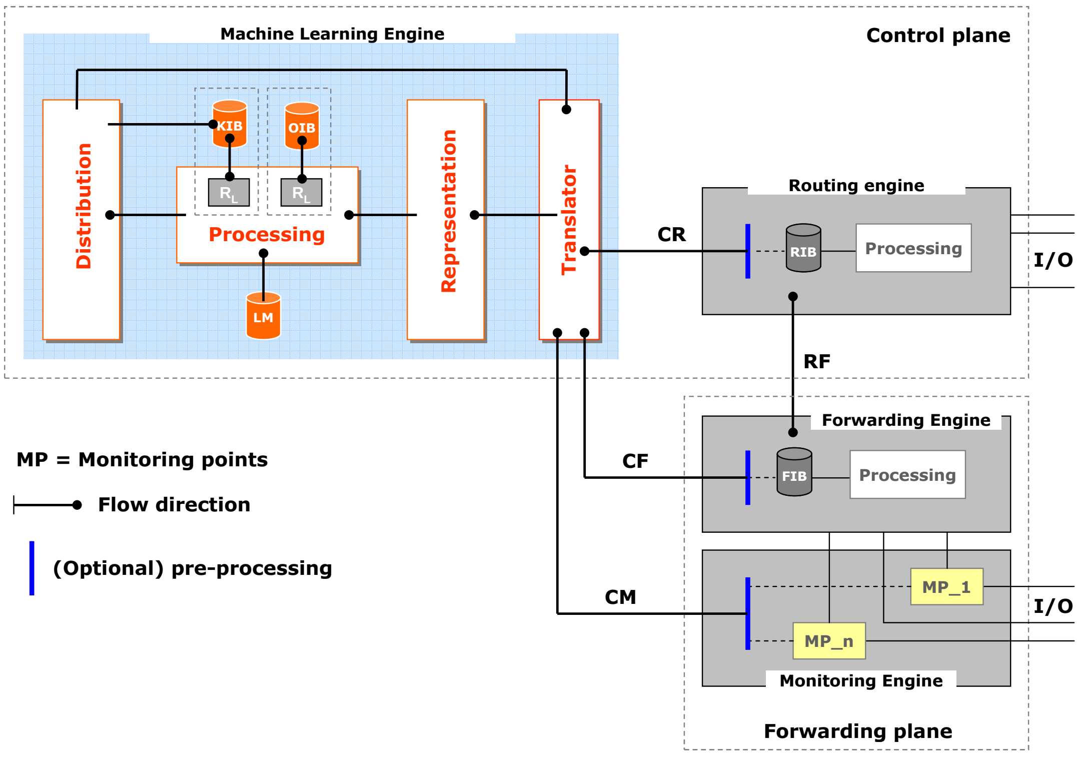


Fig.6.1: ECODE Common Architecture

The ECODE architecture introduces in addition to the forwarding, and routing engines:

- The Machine Learning Engine (MLE), part of the control plane, aims at processing by means of learning methods, the input from the network (obtained via forwarding and control components) to subsequently decide on forwarding and routing execution. The MLE provides the means to propagate the corresponding decisions to the routing and forwarding engines. The MLE comprises four different functional components: the Translator (syntax function), the Representation (semantic function), the Processing (learner and performer and associated register), and the Distribution. The MLE includes also the following data structures:

- o The Knowledge Information Base (KIB): stores learned models and (past) decisions. Both constitute the so-called prior knowledge.
- o The Observation Information Base (OIB) stores bounded sequences of observations that can be accessed by means of the Register (RL) or loaded (on-demand) by the Processor.
- o The Learning Methods (LM) base: stores the learning algorithms

The interaction between the various engines is performed through dedicated interfaces:

1. RF (for Routing – Forwarding): through this interface, the routing and forwarding engines can communicate with each other and exchange information if requires.
2. CR (for Cognitive – Routing): through this interface, the MLE may retrieve data from the routing engine and communicate to the routing engine the decision it takes.
3. CF (for Cognitive – Forwarding): similarly to the CR, the MLE may retrieve data from the forwarding engine and communicate to the forwarding engine the decision it takes.
4. CM (for Cognitive - Monitoring): through this interface, the MLE may retrieve path performance information from the ME.

Monitoring, routing, and/or forwarding engines provide raw and/or pre-processed data to the Machine Learning Engine (MLE) through the CM, CR, and CF interface, respectively. The MLE takes decisions and directs them to the routing, forwarding, and monitoring engines.

- The Monitoring Engine (ME): part of the forwarding plane, its description can be found in Deliverable D2.1 (Section 5.3).

In this use case, the ME is not involved in the mechanisms specified in this document, neither is the FE. Processing of incoming BGP related information is limited to the MLE. More precisely, the data sent from the RE to the MLE comprises i) temporal information: timestamp, and optionally, the inter-arrival time (IAT) between the previous and current BGP UPDATE message, and ii) spatial information extracted from the BGP UPDATE message itself: <Address Prefix, AS-Path, [AS originator], Peer>. The Peer attribute identifies the BGP speaker from which the BGP UPDATE message is received. The strings exchange from the RE to the MLE comprise are thus of the form: (<Time, [IAT]>; <Address Prefix, AS-Path, [AS originator], Peer>).

The MLE builds up its model using this spatio-temporal data: data is first processed by the translator (format translation) and then semantically processed by the representation module before being processed as observations by the MLE processor. The output of the learned model comprises the sequence of announcements identified as associated to the detected path exploration event and the replacing announcement for that destination prefix. This information is directed to the RE. At the RE, it is used by the local BGP route selection process to determine which announcements should be eliminated after the execution of this process and which announcements shall replace it. This output is not distributed to the router’s neighbours or other nodes in the system. However, the router's output (i.e., the BGP UPDATE messages that are propagated to its downstream neighbours) will influence the route selection of the BGP router's neighbour.

7. Conclusion

In this deliverable, we have described the research work achieved so far in the task dedicated to the experimentation on the technical objective 3 (TO3) addressing machine learning techniques for improving routing system scalability and quality. The problem addressed by this use case has been formalized. Then, the developed machine learning techniques, decision trees and Hidden Markov Model (HMM) have been presented as used to provide appropriate solution to this networking change. The proposed machine-learning-based algorithms are evaluated by off-line experimentation on representative Internet topologies. The simulated codes used for this purpose constitute first high-level prototypes. Finally we explain how this use case fits in the ECODE common architecture specified in deliverable D2.1.

Detecting path exploration events, identifying path exploration announcement/withdrawal sequence, and anticipating BGP route selection process decision to mitigate the effects of a phenomenon inherently associated to the routing protocol of the Internet routing system, i.e., BGP, is the fundamental problem addressed by this use case. At this stage, the proposed solutions (i.e. machine learning algorithms and associated procedures) applied on theoretical and realistic topologies show good results as most of their classification success rates are above 95%. However, in *n*-nodes p-clique topologies giving the worst classification success rates (82%), the learned model is unable to distinct precisely the outcome of a path exploration. The introduction of HMM is as a mean to infer a “global” routing state from the observed sequence of BGP UPDATE messages. The results obtained with the HMM model needs to be complemented as detailed in Section 5.4 in order to prevent effects of explorationless withdraws. We aim also at introducing explicit state duration density as part of our model.

The next objective is to experiment the proposed techniques on the ILAB platform by developing prototype codes, perform the full integration as part of the ECODE common architecture. When these implementations will be up and running, a second stage of tests and validations including performance evaluation and functionality benchmarking will be carried out to check whether the results promised by simulations are confirmed.

Reference

[1] V.Lefèvre, *Applying Machine Learning Techniques to the BGP Path Exploration Process,* Master Thesis, UCL. June 2009.

[2] J.R.Quinlan, *Induction of Decision Trees*. In Machine Learning, 1(1), pp. 81-106. 1986.

[3] J.R.Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers. 302 pages, 1993.

[4] Y.Rekhter, T. Li, and S. Hares. *A Border Gateway Protocol (BGP 4)*. Internet Engineering Task Force, RFC 4271. January 2006.

[5] R.Oliveira, B.Zhang, and R.Izhak-Ratzin, *Quantifying Path Exploration in the Internet*. In Proc. ACM Internet Measurement Conference (IMC). October 2006.

[6] V.Levenstein, *Binary Codes Capable of Correcting Deletions, Insertions, and Reversals*. In Soviet Physics Doklady, 10(8), pp. 707-710. 1966.

[7] S.Deshpande, and B.Sikdar, *On the Impact of Route Processing and MRAI Timers on BGP Convergence Times,* Proceedings of IEEE Globecom 2004, November 2004.

[8] C.Labovitz, A.Ahuja, A.Bose, and F.Jahanian, Delayed Internet Routing Convergence, IEEE/ACM Transactions on Networking, vol.9, no.1, pp. 293-306, June 2001.

[9] T.G.Griffin, and B.J.Premore, *An Experimental Analysis of BGP Convergence Time*. In Proc. IEEE International Conference on Network Protocols (ICNP). November 2001.

[10] A.Elmokash, A.Kyalbein and, C.Dovrolis, *On the Scalability of BGP: the Roles of Topology Growth and Update Rate-Limiting*. In Proc. ACM CoNEXT. December 2008.

[11] J.Qiu. *SimBGP: Simple BGP Simulator,* http://www.bgpvista.com/simbgp.php

[12] A.V.Aho, J.E.Hopcroft, and J.D.Ullman, *Data Structures and Algorithms*, Addison Wesley. 427 pages. 1983

[13] L.E.Baum and, T.Petrie, *Statistical Inference for Probabilistic Functions of Finite State Markov Chains*. In Annals of Mathematical Statistics, 37(6), pp. 1554-1563. 1966.

[14] A.J.Viterbi, *Error bounds for convolutional codes and an asymptotically optimal decoding algorithm,* IEEE Trans. Information Theory, vol. IT-13, pp. 260-269, April 1967.

[15] G.D.Forney, *The Viterbi algorithm*, Proc. IEEE, vol. l. 61, pp. 268-278, March 1973.

[16] Y.LeCun, L.Bottou, Y.Bengio, and P.Haffner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278{2324, 1998.