Developing an Immunity to Spam

Terri Oda[1] and Tony White[2]

[1] Carleton University
terri@zone12.com
[2] Carleton University
arpwhite@scs.carleton.ca

Abstract. Immune systems protect animals from pathogens, so why not apply a similar model to protect computers? Several researchers have investigated the use of an artificial immune system to protect computers from viruses and others have looked at using such a system to detect unauthorized computer intrusions. This paper describes the use of an artificial immune system for another kind of protection: protection from unsolicited email, or spam.

1 Introduction

The word "spam" is used to denote the electronic equivalent of junk mail. This typically includes advertisements (unsolicited commercial email or UCE) or other messages sent in bulk to many recipients (unsolicited bulk email or UBE). Although spam may also include viruses, typically the term is used to refer to the less destructive classes of email.

In small quantities, spam is simply an annoyance but easily discarded. In larger quantities, however, it can be time-consuming and costly. Unlike traditional junk mail, where the cost is borne by the sender, spam creates further costs for the recipient and for the service providers used to transmit mail. To make matters worse, it is difficult to detect all spam with the simple rule-based filters commonly available.

> Spam is similar to computer viruses because it keeps mutating in response to the latest "immune system" response. If we don't find a technological solution to spam, it will disable Internet email as a useful medium, just as viruses threatened to disable the PC revolution. [1]

Although many people would consider this statement a little over-dramatic, there is definitely real need for methods of controlling spam (unsolicited email).

This paper will look at a new mechanism for controlling spam: an artificial immune system (AIS). The authors of this paper have found no other research involving creation of a spam-detector based on the function of the mammalian immune system, although the immune system model has been applied to the similar problem of virus detection [2].

2 The Immune System

To understand how an artificial immune system functions, we need to consider the mammalian immune system upon which it is based. This is only a very general overview and simplification of the workings of the immune system which uses information from several sources [3], [4]. A more complete and accurate description of the immune system can be found in many biology texts.

In essence, the job of an immune system is to distinguish between self and potentially harmful non-self elements.

The harmful non-self elements of particular interest are the *pathogens*. These include viruses (e.g. Herpes simplex), bacteria (e.g. E. coli), multi-cellular parasites (e.g. Malaria) and fungi. From the point of view of the immune system, there are several features that can be used to identify a pathogen: the cell surface, and soluble proteins called *antigens*.

In order to better protect the body, an immune system has many layers of defence: the skin, physiological defences, the innate immune system and the acquired immune system. All of these layers are important in building a full viral defence system, but since the acquired immune system is the one that this spam immune system seeks to emulate, it is the only one that we will describe in more detail.

2.1 The Acquired Immune System

The acquired immune system is comprised mainly of *lymphocytes*, which are types of white blood cells that detect and destroy pathogens. The lymphocytes detect pathogens by binding to them.

There are around 10^{16} possible varieties of antigen, but the immune system has only 10^8 different antibody types in its *repertoire* at any given time. To increase the number of different antigens that the immune system can detect, the lymphocytes bind only approximately to the pathogens. By using this approximate binding, the immune system can respond to new pathogens as well as pathogens that are similar to those already encountered. The higher affinity the surface protein receptors (called *antibodies*) have for a given pathogen, the more likely that lymphocyte will bind to it. Lymphocytes are only activated when the bond reaches a threshold level, that may be different for different lymphocytes.

Creating the detectors. In order to create lymphocytes, the body uses a "library" of genes that are combined randomly to produce different antibodies. Lymphocytes are fairly short-lived, living less than 10 days, usually closer to 2 or 3. They are constantly replaced, with something on the order of 100 million new lymphocytes created daily.

Avoiding Auto-immune Reactions. An *auto-immune* reaction is one where the immune system attacks itself. Obviously this is not desirable, but if lymphocytes are created randomly, why doesn't the immune system detect self?

This is done by self-tolerization. In the thymus, where one class of lymphocytes matures, any lymphocyte that detects self will either be killed or simply not selected. These specially self-tolerized lymphocytes (known as T-helper cells) must then bind to a pathogen before the immune system can take any destructive action. This then *activates* the other lymphocytes (known as B-cells).

Finding the Best Fit. (Affinity maturation) Once lymphocytes have been activated, they undergo cloning with hypermutation. In hypermutation, the mutation rate is 10^9 times normal. Three types of mutations occur:

- point mutations,
- short deletions,
- and insertion of random gene sequences.

From the collection of mutated lymphocytes, those that bind most closely to the pathogen are selected. This hypermutation is thought to make the coverage of the antigen repertoire more complete. The end result is that a few of these mutated cells will have increased affinity for the given antigen.

3 Spam as The Common Cold

Receiving spam is generally less disastrous than receiving an email virus. To continue the immune system analogy, one might say spam is like the common cold of the virus world – it is more of an inconvenience than a major infection, and most people just deal with it. Unfortunately, like the common cold, spam also has so many variants that it is very difficult to detect reliably, and there are people working behind the scenes so the "mutations" are intelligently designed to work around existing defences.

Our immune systems do not detect and destroy every infection before it has a chance to make us feel miserable. They do learn from experience, though, remembering structures so that future responses to pathogens can be faster. Although fighting spam may always be a difficult battle, it seems logical to fight an adaptive "pathogen" with an adaptive system.

We are going to consider spam as a pathogen, or rather a vast set of varied pathogens with similar results, like the common cold. Although one could say that spam has a "surface" of headers, we will use the entire message (headers and body) as the antigen that can be matched.

4 Building a Defence

4.1 Layers revisited

Like the mammalian immune system, a digital immune system can benefit from layers of defence [5]. The layers of spam defence can be divided into two broad categories: social and technological. The proposed spam system is a technological defence, and would probably be expected to work alongside other defence strategies. Some well-known defences are outlined below.

Social Defences Many people are attempting to control spam through social methods, such as suing senders of spam [6], legislation prohibiting the sending of spam [7], or more grassroots methods [8].

Technological Defences To defend against spam, people will attempt to make it difficult for spam senders to obtain their real email address, or use clever filtering methods. These include two of particular interest for this paper:

SpamAssassin [9] uses a large set of heuristic rules.

Bayesian/Probabilistic Filtering [10] [11] uses "tokens" that are rated depending on how often they appear in spam or in real mail. Probabilistic filters are actually the closest to the proposed spam immune system, since they learn from input.

Some solutions, such as the Mail Abuse Prevention System (MAPS) Realtime Blackhole List (RBL) fall into both the social and the technological realms. RBL provides a solution to spam through blocking mail from networks known to be friendly or neutral to spam senders [12]. This helps from a technical perspective, but also from a social perspective since users, discovering that their mail is being blocked, will often petition their service providers to change their attitudes.

4.2 Regular Expressions as Antibodies

Like real lymphocytes, our digital lymphocytes have receptors that can bind to more than one email message. This is done by using *regular expressions* (patterns that match a variety of strings) as antibodies. This allows use of a smaller gene library than would otherwise be necessary, since we do not need to have all possible email patterns available. This has the added advantage that, given a carefully-chosen library, a digital immune system could be able to detect spam with only minimal training.

The library of gene sequences is represented by a library of regular expressions that are combined randomly to produce other regular expressions. Individual "genes" can be taken from a variety of sources:

- a set of heuristic filters (such as those used by SpamAssassin)
- an entire dictionary
- several entire dictionaries for different languages
- a set of strings used in code, such as HTML and Javascript, that appears in some messages
- a list of email addresses and URLs of known spam senders
- a list of words chosen by a trained or partially-trained Bayesian Filter

The combining itself can be done as a simple concatenation, or with wildcards placed between each "gene" to produce antibodies that match more general patterns.

Unfortunately, though this covers the one-to-many matching of antibodies to antigens, there is no clear way to choose which of our regular expression antibodies has the *best* match, since regular expressions are handled in a binary (matches/does not match) way. Although an arbitrary "best match" function could be applied, it is probably just as logical to treat all the matching antibodies equally.

4.3 Weights as Memory

Theories have proposed that there may be a longer-lived lymphocyte, called a memory B-cell, that allows the immune system to remember previous infections. In a digital immune system, it is simple enough to create a special subclass of lymphocytes that is very long-lived, but doing this may not give the desired behaviour.

While a biological immune system has access to all possible self-proteins, a spam immune system cannot be completely sure that a given lymphocyte will not match legitimate messages in the future. Suppose the user of the spam immune system buys a printer for the first time. Previously, any message with the phrase "inkjet cartridges" was spam (e.g. "CHEAP INKJET CARTRIDGES ONLINE – BUY NOW!!!"), but she now emails a friend to discuss finding a store with the best price for replacement cartridges. If her spam immune system had long-lived memory B-cells, these would continue to match not only spam, but also the legitimate responses from her friend that contain that phrase.

In order to avoid this, we need a slightly more adaptive memory system in that it can *unlearn* as well as learn things. A simple way to model this is to use weights for each lymphocyte.

In the mammalian immune system, pathogens are detected partially because many lymphocytes will bind to a single pathogen. This could easily be duplicated, but matching multiple copies of a regular expression antibody is needlessly computationally intensive. As such, we use the weights as a representation of the number of lymphocytes that would bind to a given pathogen.

When a lymphocyte matches a message that the user has designated as spam, the lymphocyte's weight is then incremented (e.g. by a set amount or a multiple of current weight) Similarly, when a lymphocyte matches something that the user indicates is not spam, then the weight is decremented.

Although the lymphocyte weights can be said to represent numbers of lymphocytes, it is important to note that these weights can be negative, representing lymphocytes which, effectively, detect self. Taking a cue from SpamAssassin, we use the sum of the positive and negative weights as the final weight of the message. If the final weight is larger than a chosen threshold, it can be declared as spam. (Similarly, messages with weights smaller than a chosen threshold can be designated non-spam.)

The system can be set to learn on its own from existing lymphocytes. If a new lymphocyte matches a message that the immune system has designated spam, then the weight of the new lymphocyte could be incremented. This increment would probably be less than it would have been with a human-confirmed spam message, since it is less certain to be correct. Similarly, if it matches a message designated as non-spam, its weight is decremented.

When a false positive or negative is detected, the user can force the system to re-evaluate the message and update all the lymphocytes that match that message. These incorrect choices are handled using larger increments and decrements so that the automatic increment or decrement is overridden by new weightings based on the correction. Thus, the human feedback can override the adaptive learning process if necessary.

In this way, we create an adaptive system that learns from a combination of human input and automated learning.

An Algorithm for Aging and Cell Death. Lymphocytes “die” (or rather, are deleted) if they fall below a given weight and a given age (e.g. a given number of days or a given number of messages tested). This simulates not only the short lifespan of real lymphocytes, but also the negative selection found in the biological immune system.

We benefit here from being less directly related to the real world. Since there is no good way to be absolutely sure that a given lymphocyte will not react to the wrong messages, co-stimulation by lymphocytes that are guaranteed not to match legitimate messages would be difficult. Attempting to simulate this behaviour might even be counter-productive with a changing ”self.” For this prototype, we chose to keep the negatively-weighted, self-detecting lymphocytes in this prototype to help balance the system without co-stimulation as it occurs in nature. Thus, cell death occurs only if the absolute value of the weight falls below a threshold. It should be possible to create a system which ”kills” off the self-matching lymphocytes as the self changes, but this was not attempted for this prototype.

How legitimate is removing those with weights with small absolute values?

Consider a antibody that never matches any messages (e.g. **antidisestablishmentarianism.* aperient.* kakistocracy**). It will have a weight of 0, and there is no harm in removing it since it does not affect detection. Even a lymphocyte with a small absolute weight is not terribly useful, since small absolute weights mean that the lymphocyte has only a small effect on the final total. It is not a useful indicator of spam or non-spam, and keeping it does not benefit the system.

A simple algorithm for artificial lymphocyte death would be:

```
if (cell is past “expiry date”) {
  decrement weight magnitude
  if (abs(cell weight) < threshold) {
       kill cell
  } else {
       increment expiry date
  }
}
```

The decrement of the weight is to simulate forgetfulness, so that if a lymphocyte has not had a match in a very long time, it can eventually be recycled. This decrement should be very small or could even be none, depending on how strong a memory is desired.

4.4 Mutations?

Since we have no algorithm defined to say that one regular expression is a better match than another, we cannot use mutation easily to find matches that are more accurate. Despite this, there could still be a benefit to mutating the antibodies of a digital immune system, since it would be possible (although perhaps unlikely) that some of the new antibodies created would match *more* spam, even if there was no clear way to define a *better* match with the current message. Mutations could be useful for catching words that spam senders have hyphenated, misspelled intentionally, or otherwise altered to avoid other filters. At the very least, mutations would have a higher chance of matching with similar messages than lymphocytes created by random combinations from the gene library.

Mutations could occur in two ways:

1. They could be completely random, in which case some of the mutated regular expressions will not parse correctly and will not be usable.
2. They could be mutated according to a scheme similar to that of Automatically Defined Functions (ADF) in genetic programming [13]. This would leave the syntax intact so that the result is a legitimate regular expression.

It would be simpler to write code that would do random mutations, but then harder to check the syntax of the mutated regular expressions if we wanted to avoid program crashing when lymphocytes with invalid antibodies try to bind to a message. These lymphocytes would simply die through negative selection during the hypermutation process, since they are not capable of matching with anything. Conversely, it would be harder to code the second type, but it would not require any further syntax-checking.

Another variation on mutation is an adaptive library. In some cases, no lymphocytes will match a given message. If this message is tagged as spam by the user, then the system will be unable to “learn” more about the message because no weights will be updated. To avoid this situation, the system could generate new gene sequences based upon the message. These could be “tokens” as described by Graham [11], or random sections of the email. These new sequences, now entered into the gene pool, will be able to match and learn about future messages.

5 Prototype Implementation

Our implementation has been done in Perl because of its great flexibility when it comes to working with strings.

The gene library and lymphocytes are stored in simple text files. Figure 1 shows the contents of a short library file. In the library, each line is a regular expression. Each “gene” is on a separate line.

Figure 2 shows the contents of a short lymphocytes file. For the lymphocytes, each line contains the weight, the cell expiry date and the antibody regular expression. The format uses the string ”###” (that does not occur in the library)

```
remove.{1,15}subject
Bill.{0,10}1618.{0,10}TITLE.{0,10}(III|\#3)
check or money order
\s+href=['"]?www\.
money mak(?:ing|er)
(?:100%|completely|totally|absolutely) (?-i:F)ree
```

Fig. 1. Sample Library Entries

```
-5###1040659390###result of
10###1040659390###\<BODY.*bgcolor="#?[^f]
-98###1040659390###(?:\$|US\$|usd?).?\d{2,3}(?:\.\d)?
52###1040661344##trial version of CommuniGate.*opt-in
33###1040661344###click .{0,30}(?:here|below)
```

Fig. 2. Sample Lymphocytes

as a separator, so each lymphocyte definition string has <weight>###<expiry time>###<regular expression antibody>.

Our AIS spam detector has three phases:

1. Generation of Lymphocytes
2. Application of Lymphocytes
3. Culling of Lymphocytes

Figure 3 gives an overview of the life cycle of a spam lymphocyte. The generation and culling scripts are run on a regular schedule, and the application script is called with appropriate arguments to increment, decrement, or simply detect, as appropriate. For example, if the target message has been tagged as spam by the user, the application script is called with a large increment so that all the lymphocytes that match that message are incremented. The target message could fall into six potential classes: Tagged as spam or legitimate by the user, tagged as spam or legitimate by the immune system, or tagged as a false positive or a false negative by the user.

5.1 Generation

This script reads all the "genes" in the library and combines them randomly to produce the number of antibodies specified as the first argument.

The antibodies are created by choosing a gene randomly, then effectively flipping a coin to see if another gene should be appended to the resulting regular expression antibody. The probability of appending to an antibody can be changed relatively easily. For the purposes of testing, we chose 50% as a the likelihood for each gene addition. (Although an antibody using fewer genes will tend to match more frequently, we wished to to see if some longer antibodies matched spam more exclusively.) To add another gene, another gene is chosen

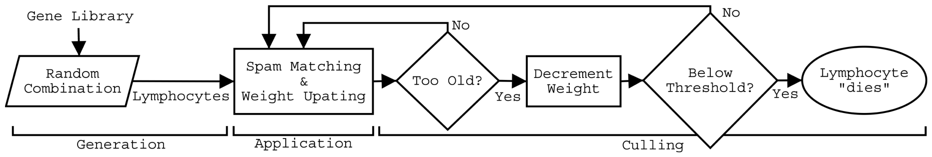

Fig. 3. Life cycle of a digital lymphocyte

randomly and the original string is concatenated with the string .* (match any number of characters) followed by the new gene sequence. This intermediary .* is used to increase the number of possible matches.

The lymphocyte is then given a weight of 0 and an expiry date. The resulting lymphocytes are written to disk in the order of generation.

There is no attempt made to avoid duplicates since doing so could be costly and having duplicates should not overly adversely affect results, only giving some antibodies a stronger effect by duplicating. This effect should be balanced by other antibodies.

5.2 Application

The application script reads in all the lymphocytes and applies them one by one to each message in the file specified as first argument. The total weighted sums are outputted for each message, along with the Subject: and From: lines to give some context.

The antibodies are applied in a case-insensitive manner. This was chosen so that more matches would be found, although a quick comparison using the same lymphocytes showed that matching in a case-sensitive manner did not make a significant difference in the weightings.

This script takes a second argument that indicates the increment or decrement to be applied to each lymphocyte that matches messages in this file. The updated lymphocytes are then re-written to the lymphocytes file.

5.3 Culling

The culling script reads in all the lymphocytes and kills off cells based on the algorithm described earlier in this paper. Ideally, the culling script would call the generation script to replace those lymphocytes removed, but for the moment this is done manually.

6 Results

6.1 Initial Parameters and Data

We initially used a set of 150 regular expression ”genes” as a library. Many of these were simplified SpamAssassin heuristics. SpamAssassin contains over 700 very complex heuristics, but our gene library contained only pieces of those, typically pieces that matched only a few shorter phrases. These heuristics were used to reduce the training time required for the lymphocytes, since many people would prefer a solution which required little training time. (If individual users were not interested in training their own artificial immune systems at all, it could be possible give them spam-trained lymphocytes, much like users currently download known virus patterns for their computer virus scanners.)

To test and train the lymphocytes, a set of mailbox files (in mbox format) were used with messages compiled from personal mail, Grant Taylor's collection of spam [14], and mailing list archives of lists known to contain both legitimate posts and spam.

In a quick test with 100 lymphocytes generated, it was soon apparent that either this size of library was insufficient to detect the selection of spam being used to train the lymphocytes or more lymphocytes needed to be generated. Over 40% of the messages known to be spam were not detected by any of the lymphocytes, and thus the immune system could not learn about these messages. This may indicate a need for a process akin to affinity maturation where, knowing a message to be spam, the digital immune system produces lymphocytes that match it.

With a larger library of around 300 genes and set of around 1000 lymphocytes generated, the tests take much longer to run, so only a few iterations were run after the initial testing phase.

Increment and decrement values were chosen so that a false positive or negative was given a more significant weight than either training values or automated learning values. Because missed messages are more irritating to users, we wanted to encourage faster learning from these messages so that users would not have to keep reporting similar missed messages. The following values were used:

Determined by	User (training)	AIS (automated)	User (false positive/negative)
Spam/non-spam weight adjustment	5/-5	1/-1	10/-10

6.2 Output

The initial spam training gave weights ranging from 0 to over 10^5. As such, some lymphocytes had a much larger ability than others to affect the final sum. Some of these were balanced after initial "self-tolerization" training against legitimate messages.

After the initial training phase of over 1600 spam messages and 1000 legitimate messages was done, we forced a cell death on lymphocytes with absolute weights less than 4. Out of the 1000 lymphocytes originally generated, less than 100 were saved, leading credence to the idea that a larger lymphocyte population would be necessary to do proper tests.

These remaining 100 lymphocytes were used as a quick test against a heterogeneous (spam and non-spam) mailbox with approximately 1200 messages.

The results were very promising. We set the threshold at 10, but even with this small threshold, only two false positives occurred (where a legitimate message was found as spam).

– One was a message sent by a random stranger and was vastly different in content and style from the messages used initially to set the weights.
– The other was a message from a current spam control system. It contained a listing of the Subject: and From: lines of the messages blocked that day.

Although the second false positive seems reasonably explicable, this behaviour is problematic for layering spam defences. Some sort of special exception may have to be found so that messages of this sort from a reporting system will not be caught. This could be through manual addition of an overriding lymphocyte (one with a very large negative weight), for example. Training the system normally should be possible, but could neutralise otherwise-useful lymphocytes.

Similarly, the spam messages not tagged by the system were unusual spams, unlike those that had been seen earlier by the system.

There were some negatively-valued messages, but most of the messages were given weights of 0, as would be expected given that the library was drawn from spam-matching regular expressions used by SpamAssassin. If a more broad-based library were used, then there would be more negatively-weighted lymphocytes, and thus it would be possible to have more negatively-valued messages.

Further tests after more iterations and re-running of false positives yielded similar results to those initial ones: approximately 1% of the messages scanned were false positives. Overall, the 1000 lymphocyte system was correctly identifying approximately 90% of the spam messages.

7 Conclusions

This model has some potential: even with empirically-chosen values for increments and decrements and a relatively small library of regular expressions, emergent behaviour has been observed. It is detecting spam with very few false positives in the test mailboxes. Due to the small size of the library space, it is still missing approximately 10% of the spam messages entirely. While false negatives are less damaging than false positives [15], they are still an indicator of a detector or library space that is still too small to cover the range of potential spam messages.

This initial 90% accuracy is not as good as Graham's probabilistic filter, which can work at over 99% accuracy [11]. However, while we looked at 1000 lymphocytes, Graham's system recognises over 23,000 tokens, and his algorithm has had significantly more time to mature.

7.1 Future Work

Several directions for future research are planned. A mutation model is to be included, perhaps with a function to determine better matching or preferential weightings of lymphocytes (e.g. One could use the length of the string matched by a given regular expression as an indicator of better matching[16]). Larger and alternative gene libraries are to be created and tested. A variation self-tolerization could be used to generate only lymphocytes which match spam rather than relying upon weight-balancing through self-detecting lymphocytes. Finally, alternative weighting schemes are to be designed and compared to the performance of the algorithm described in this paper.

References

1. Hellweg, E.: What price spam? Business 2.0 (1999)
2. White, S.R., Swimmer, M., Pring, E.J., Arnold, W.C., Chess, D.M., Morar, J.F.: Anatomy of a commercial-grade immune system. Technical report, IBM Thomas J. Watson Research Center (2002)
3. Nunn, I.: Immunocomputing: The natural immune system and its computational metaphors. (2002) `http://www.scs.carleton.ca/~arpwhite/courses/95590Y/notes/Immunocomputi%ng.ppt`.
4. Hofmeyr, S.A.: An overview of the immune system (1997) `http://www.cs.unm.edu/~immsec/html-imm/immune-system.html`.
5. Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C.D., Stamatopoulos, P.: Stacking classifiers for anti-spam filtering of E-mail. In: Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing, Pittsburgh, US, Association for Computational Linguistics, Morristown, US (2001)
6. Jesdanun, A.: AOL wins $7 million spam lawsuit. Salon.com (2002) `http://www.salon.com/tech/wire/2002/12/17/aol_spam/`.
7. Email, C.A.U.: Pending legislation (2002) `http://www.cauce.org/legislation`.
8. Wendland, M.: Internet spammer can't take what he dishes out. Detroit Free Press (2002)
9. SpamAssassin: Spamassassin website (2002) `http://spamassassin.org/`.
10. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approach to filtering junk E-mail. In: Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, AAAI Technical Report WS-98-05 (1998)
11. Graham, P.: A plan for spam (2002) `http://www.paulgraham.com/spam.html`.
12. Vixie, P.: MAPS RBL rationale (2000) `http://mail-abuse.org/rbl/rationale.html`.
13. Poli, R.: Introduction to evolutionary computation: Automatically defined functions in genetic programming (1996) `http://www.cs.bham.ac.uk/~rmp/slide_book/node7.html`.
14. Taylor, G.: A collection of spam (2002) `http://www2.picante.com:81/~gtaylor/download/spam.tar.gz`.
15. Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G., Spyropoulos, C.D.: An evaluation of naive bayesian anti-spam filtering. In Potamias, G., Moustakis, V., van Someren, M., eds.: Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000), Barcelona, Spain (2000) 9–17
16. O'Byrne, K.: Private communication (2002)