Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Earthquake Risk Induction Models with Evolutionary Computation

Yuri Cossich Lavinas

Monografia apresentada como requisito parcial
para conclusão da disciplina Estudos Em Inteligência Artificial

Orientador
Prof. Dr. Marcelo Ladeira

Coorientador
Prof. Dr. Claus de Castro Aranha

Brasília
2013

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Homero Luiz Piccolo

Banca examinadora composta por:

Prof. Dr. Marcelo Ladeira (Orientador) — CIC/UnB
Prof. Dr. Claus de Castro Aranha — Department of Computer Sciences/University of Tsukuba
Prof. Dr. Guilherme Novaes Ramos — CIC/UnB

CIP — Catalogação Internacional na Publicação

Lavinas, Yuri Cossich.

Earthquake Risk Induction Models with Evolutionary Computation / Yuri Cossich Lavinas. Brasília : UnB, 2013.
77 p. : il. ; 29,5 cm.

Monografia Parcial (Estudos Em Inteligência Artificial) — Universidade de Brasília, Brasília, 2013.

1. algoritmos genéticos, 2. sismos, 3. terremotos, 4. log-likelihood

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Earthquake Risk Induction Models with Evolutionary Computation

Yuri Cossich Lavinas

Monografia apresentada como requisito parcial
para conclusão de disciplina Estudos Em Inteligência Artificial

Prof. Dr. Marcelo Ladeira (Orientador)
CIC/UnB

Prof. Dr. Claus de Castro Aranha
Department of Computer Sciences/University of Tsukuba

Prof. Dr. Guilherme Novaes Ramos
CIC/UnB

Prof. Dr. Homero Luiz Piccolo
Coordenador do Bacharelado em Ciência da Computação

Brasília, 20 de dezembro de 2013

Agradecimentos

to be done

Resumo

To be done

Palavras-chave: algoritmos genéticos, sismos, terremotos, log-likelihood

Abstract

To understand the mechanisms and patterns of the earthquakes is very important to minimize its consequences. In this context, this projects aims to develop an earthquake prevision risk model using Evolutionary Computation (EC). The main goal is to define a method to estimate the probability of earthquake occurrences in Japan using historical earthquake data of a given geographical region. This work is established in the context of the "Collaboratory for the Study of Earthquake Predictability" (CSEP), which seeks to standardize the studies and tests of earthquake prevision models. The method is based in one application of Genetic Algorithms (GA) and aims to develop statistical methods of analysis of earthquake risk. The risk models generated by this application were analyzed by their log-likelihood values, as suggested by the Regional Earthquake Likelihood Model (RELM). They are compared with real data and the models generated by the application of the Relative Intensity Algorithm (RI). The data used was obtained from the Japan Metereological Angency (JMA) and are related with earthquake activity in Japan between the years of 2000 and 2010.

Keywords: genetic algorithms, earthquake, log-likelihood

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we present a general specification of the problem, its relevance and what are the goals of this study.

1.1 Earthquakes

Earthquakes may cause lots of damages environment and consequently may represent, directly or inderectly, a risk to human lives. They manifest themselves by shaking and sometimes displacement of the ground. They may also cause tsunamis, landslides, volcano activities, etc.

There are many examples that show how devastating one large earthquake can be. In April 2015,there was a magnitude 7.8 M_w earthquake in Nepal. It cause 8000 deaths and triggered two avalanches, one of them in the Mount Everest. Also, many people were made homeless, it destroyed UNESCO World Heritage sites and had many aftershocks, including a magnitude M_w 7.3 quake that caused 200 deaths. Another example happened in March 2011, Japan. It was a 9.0 M_w earthquake, and it is considered the most powerful earthquake to ever hit Japan and the 4th most powerful in the world. It caused tsunami waves that reached more than 40 meters, moved the main island in Japan more than 2 meters east and also changed the Earth axis. It was reported that it caused more than 15000 deaths, made more than 200000 people homeless and provoked a meltdown of the Fukushima Daiichi Nuclear Power Plant complex. (see Figure 1.1 for a picture of the devastation) (May i use this picture?). Also in 2011 a magnitude M_w 7.1 earthquake hit Van,Turkey and caused lots of deaths and great damages.These are only three very recent examples of large earthquake damages of how dangerous earthquakes can be.

Those earthquakes, and many others that hazard the human society, have some common characteristics. They not only are powerful quakes but they happened nearby populated areas, which increase the damaged provoked. To minimize as much as possible future earthquake disaster, a lot can be done. That include to developing goos urban planings, for example to build structures with techniques that can withstand the forces of earthquakes, to create earthquake warning systems, to create more precise civil enginnering codes, and such.

To be able to prevent as many casualty as possible, we need the patterns and mechanisms behind the occurrence of earthquakes. We need to know if there is any relationship between the earhquake locations and its time of occurrance, how they are related to each other, et cetera. With this information, it is possible to to create better seismic risk forecast models, indicating which regions show a higher probability of earthquake occurrence at certain periods in time.

Until by now, it has been difficult to clearly understand the many different seismic variables (hours, magnitude, local, depth,...) influences the quakes and either exists a mathemathical model capable of supplying detailed and precise information about the relations and ways to estimate them. Therefore, to develop a prediction earthquake model can prove itself very complex.

1.2 Earthquake Prediction

Earthquake prediction is a polemic subject. No research has even come close to suggesting that individual large scale earthquakes can be predicted [1] and many scientists think that earthquake prediction may not be fully impossible but that the resources needed for such a prediction may be out of reach [6].

In the context of this study, we do not aim to predict any individual earthquake and its major characteristics. Our goal relies on the fact that earthquakes do cluster in time and space. We want to use computer techniques to learn and to generate risk models. There is a lot of value behind the study of earthquake mechanisms, with the goal of generating statistical models of earthquake risk [35].

In [21], Koza says that Evolutionary Computation (EC) may find, by try trial and error, based on a great amount of data, better solutions for problems that human beings may not find it easy to solve. EC is a family of subfield of artificial intelligence that aim to extract patterns and to solve problems using a great amount of data, by trial and error.We may also say that without any domain knowl- edge about the problem to be controlled, the EC learns about it by trial and error [22].

EC are based on methaheuristic or stochastic optimization, and are mostly applied for black box problems. They are interesting to be used specially in cases that are difficult to understand and the knowledge available is not sufficiently available.

Based on these information and on the difficulty to understand how earthquakes behave, we want to explore historical earthquake data using EC. It is expected that it will help to find new ideas about earthquakes, their patterns and their mechanisms behind each earthquake occurrence. For doing so, we need first to outline the forecast problem, then verify the suitability of Evolutionary Computation to the problem of generating earthquake forecast models.

Next, we will study ways to improve the generated methods using both EC and other computer techniques and any siesmological knowledge. For this phase we want to propose

different representations aiming to refire the algorithm performance and to incorporate seismology methods to refine the models proposed.

Chapter 2

Theoretical foundation

This chapter focus on the teoretical concepts used as base for this study. The main topics are genetic algorithm, the CSEP framework and the siesmologic methods.

2.1 What are Genetic Algorithms

The main goal of a Genetic Algorithm (GA) is to find approximated solutions in problems of search and optimization. Based on Koza [21] , GA are mechanism of search based on natural selection and genetic. They explore historical data to find optimum search points with some performance increment, as said by Goldberg [9].

A GA uses those mechanisms to generate solutions to optimization and search problems. The first step is to create an initial population of possible solutions. Frequently, the initial population is randomly generated once it is common to ignore the main aspects that influence the algorithm performance.

Each possible solution of a population is called an individual. Every individual is a possible solution of a problem. Those individuals have its fitness value estimated by a fitness function. A fitness function should determine how suitable a individual is to a given problem. The most suitable individuals are graded with better values and the not so suitable ones have a lower value.

After measuring the population fitness value, some individuals are then selected by a process that takes into account each individual fitness value to influence the next population. The individuals with better values have a higher chance to be selected. The individuals selected take part in the varation process. This process may alter some of the individual characteristics using the crossover and mutation operators.

The crossover operator is a operator that is used to vary the characteristics of a group of individuals. For that a number of parents, a group of individuals from the current population, are selected. In most of the cases, the parents are chosen to compose a pair that will exchange information that will take compose the child, a new individual that

will belong to the next generation.

Another important operator is called the mutation operator. It is a operator with the purpose of avoiding the loss of important information. It works by changing the characteristics of an individual, looking to add new information to the next population.

It is common to have a evolutionary operator that allows the fittest individual from the current generation to take part in the next generation. This operator is called Elitism and it is used to assure that the next generation best solution is at least as good as in the current generation.

2.2 Earthquake Likelihood Model Testing

We started our work by studying what are the good ways to build earthquake models, how to evaluate and compare them, as suggested by the Collaboratory for the Study of Earthquake Predictability (CSEP), an international partnership to promote rigorous study of the feasibility of earthquake forecasting and predictability [1].

For that, we created the theoretical foundation. Most of it is based on the paper *Earthquake Likelihood Model Testing* [38]. From this paper we gathered information that guided us into how to build, evaluate and compare earthquake forecast models efficiently.

A very difficult and yet very common problem when studying earthquake models is how to compare different kinds of models, that are based on different tests protocols. The CSEP proposes a methodology for rigorous scientific testing of these many different models. This group proposed an framework called The CSEP framework. It provides a method to compare earthquakes risk models in an objectively and consistently way [1].

All forecast models proposed in this study are based in the Collaboratory for the Study of Earthquake Predictability (CSEP) framework. In the CSEP framework, a forecast model uses a gridded rate forecast [45], one common format in the literature. For evaluate and compare these models we used the likelihood based tests.They are the L-test, the N-test and the R-test, as suggested by Regional Earthquake Likelihood Model (RELM) [38].

2.2.1 The Log-Likelihood Function

All the methods use the log-likelihood value, for the fitness function. The fittest individual among all the others, is preserved in the next generation, to make the solution of one generation as good as the its last generation. The bins, a gene of the genome representation, b_n, define the set β and n is the size of the set β:

$$\beta := b_1, b_2, ..., b_n, n = |\beta|. \tag{2.1}$$

The probability values of the model j, expressed by the symbol Λ, is made of expectations λ_i^j by bin b_i. The vector is define as:

$$\Lambda^j = \left(\lambda_1^j, \lambda_2^j, \ldots, \lambda_i^j\right); \lambda_i^j := \lambda_i^j(b_i), b_i \in \beta \tag{2.2}$$

The vector of earthquake quantity expectations is defined as: earthquake by time. The Ω vector is composed by observations ω_i per bin b_i, as the Λ vector:

$$\Omega = \left(\omega_1, \omega_2, \ldots, \omega_i\right); \omega_i = \omega_i(b_i), b_i \in \beta \tag{2.3}$$

The calculation of the log-likelihood value for the ω_i observation with a given expectation λ is defined as:

$$L(\omega_i|\lambda_i^j) = -\lambda_i^j + \omega_i \log \lambda_i^j - \log \omega_i! \tag{2.4}$$

The joint probability is the product of the likelihood of each bin, so the logarithm $L(\Omega|\Lambda^j)$ is the sum of for $L(\omega_i|\lambda_i^j)$ every bin b_i:

$$\begin{aligned} L^j = L(\Omega|\Lambda^j) &= \sum_{i=1}^{n} L(\omega_i|\lambda_i^j) \\ &= \sum_{i=1}^{n} -\lambda_i^j + \omega_i \log \lambda_i^j - \log \omega_i! \end{aligned} \tag{2.5}$$

The fitness function is a coded version of the equation 2.5. It uses the probabilities of the bins of each individual of model for the λ values.

2.2.2 Uncertainties in Earthquake Parameters

It is important to say that the earthquake parameters as the location, magnitude and focal time, can be estimated without uncertainties. Therefore, each parameter uncertainty is included in the testing [38]. Also, each observation must be treated as independent ones. This is not the case of the aftershocks, once they are directly related with another stronger earthquake.

In thi context, it is important to generated as much simulations as needed when using modified

Para levar-se em conta essas incertezas, são geradas S simulações usando observações modificadas e podendo considerar incertezas de seus parâmetros, se possível. Cada parâmetro possui um valor de incerteza e um número aleatório, vinculado à esse valor, é gerado para obter um evento modificado. Esse processo é repetido para todos os sismos. Também é gerado um número qualquer aleatório para decidir se o evento será considerado independente e, caso não o seja, é descartado. Dessa forma são obtidas S simulações representando as incertezas de cada observação. Posteriormente é calculado o *Log-likelihood*, quantidade de observações e a razão de *Log-likelihood* das simulações.

2.2.3 Vetor de expectativas e suas observações

Cada vetor de expectativas deve ter um vetor de observações simuladas relacionado, para que o modelo proposto possa ser analisado e comparado com outros modelos corretamente. Os valores para o vetor de observações são obtidos de acordo com as probabilidades de cada *bin* do vetor de expectativas. Dessa forma, os registros de sismos gerados serão consistentes com o modelo.

Após obtidos as expectativas e suas relativas observações, como referenciado acima, é possível calcular o *Log-likelihood* da simulação, a quantidade de observações e a razão *Log-likelihood* e, portanto, partir para a aplicação dos testes já citados.

2.2.4 L-test - Data-consistency test

O L-test, também chamado de testes de consistência dos dados observados, considera o modelo de previsão como verdadeiro e mostra se o valor de *likelihood* calculado do modelo é consistente com o valor obtido das simulações. A medida é realizada pela fórmula, onde $\widehat{L}_k$ é o valor do *Log-likelihood* do modelo j, no *bin* i e $\widetilde{L}$ é o valor do *Log-likelihood* da simulação j no *bin* q:

$$\gamma_q^j = \frac{\left|\left\{\widehat{L}_k^j | \widehat{L}_k^j \leq \widetilde{L}_q^j, \widehat{L}_k^j \in \widehat{L}^j, \widetilde{L}_q^j \in \widetilde{L}^j\right\}\right|}{|\widehat{L}^j|} \tag{2.6}$$

Caso o *Log-likelihood* observado seja menor que o *Log-likelihood* das observações simuladas e não é consistente com o modelo de previsão. Caso seja observado próximo ao valor referido das observações simuladas o modelo é consistente com os dados.

Surge um problema quando o resultado é elevado. Este resultado significa que o *likelihood* das observações reais é maior que os valores de *likelihood* das simulações baseadas no modelo. Um dos motivos é uma classificação com total de expectativas baixo, no qual a maioria das vezes ocorre zero eventos. Outro motivo pode ser que a soma total exceda 1 e, dessa forma, alguns eventos ocorrerão. Nesse caso, os *bins* com nenhum evento terão um *likelihood* com valor maior que a média calculada porque as simulações refletirão o número total de sismos esperados, distribuídos pelos *bins*. Por fim, uma previsão com expectativas que correspondam exatamente as observações terá um *likelihood* de valor maior quando comparado com os *likelihood* das simulações. Isso acontece porque cada simulação, geralmente, adiciona ruídos poissoniano para as expectativas gerando observações que não correspondem as expectativas. O resultado será *likelihoods* com valor baixo para as simulações.

Portanto, um modelo não deve ser rejeitado baseado somente no L-test, por se tratar de um teste unidirecional. Este teste rejeita modelos com baixos valores de *likelihood* quando comparado com as simulações. Entretanto, modelos com altos valores de *likelihood* também podem ser inconsistentes e para determinar se é o caso, deve-se aplicar o N-test.

2.2.5 N-test - Number test

O N-test também testa a consistência do modelo com a observação, porém compara o total de observações com o número de eventos calculados na simulação das incertezas.

A medida é calculada pela fração do número total de cada observação gerada aleatoriamente do modelo menores que o número por *bin* de observações das simulações das incertezas pelo módulo do número total das observações do modelo.

O N-test, portanto, busca testar a consistência do modelo gerado focando na quantidade de observações do modelo analisado. O teste analisa se o modelo não está deixando de prever uma quantidade adequada de eventos e, caso não esteja, será devidamente rejeitado.

2.2.6 R-test - Hypotheses Comparison

Para comparar o *Log-likelihood* conjunto de dois modelos, deve-se computar o R-test, também chamado *Log-likelihood_ratio*, definido como:

$$R = L(\Omega|\Lambda^0) - L(\Omega|\Lambda^1) = L^0 - L^1 \tag{2.7}$$

Onde Λ^0 é o vetor de expectativas do modelo H^0, Λ^1 é o vetor de expectativas do modelo H^1. L^0 e L^1 são os *likelihoods* conjuntos dos modelos H^0 e H^1, respectivamente. se o *Log-likelihood_ratio* R é menor que 0, o modelo H^1 oferece a melhor previsão, caso contrário, H^0 tem melhor performance.

O R-test compara dois modelos ao analisar qual desses prevê melhor possíveis ocorrências de sismos e mede a performance relativas dos modelos.

Neste trabalho, inicialmente o L-test foi utilizado para calcular a função de *fitness* do modelo e posteriormente substituído pelo cálculo do *Log-likelihood*. O N-test, que analisa a consistência do modelo ao comparar a quantidade de ocorrências do modelo com os dados reais, não foi utilizado porque a aplicação não calcula novos valores para ocorrências de sismos, logo os dados necessários para os devidos cálculos não estão disponíveis assim como R-test não foi utilizado por comparar dois modelos gerados e não o modelo gerado com os dados reais, como objetiva a aplicação. Nas próximas seções estão explicitados os detalhes sobre esses conceitos.

Chapter 3

Computação Evolutiva e Previsão de Sismos

O presente capítulo é dedicado a mostrar o que já foi explorado da relação entre Previsão de Sismos e Computação Evolutiva (EC) pela comunidade acadêmica.

Uma das abordagens utilizada é a hibridização entre as técnicas de Computação Evolutiva (EC). Zhang e Wang [47] utilizaram Algoritmo Genético (GA) para refinar uma Redes Neurais (ANN) e, a partir dessa aplicação, criar um modelo de previsão. Zhou and Zhu [48], para realizar uma previsão da magnitude de sismos, fizeram uma combinação entre ANN e EC.

Muitas das aplicações estimam características dos sismos ou de suas atividades, como por exemplo calcular o *Peak Ground Acceleration* (PGA), [3, 16, 19]. PGA é uma medida da aceleração do sismo no solo e pode ser utilizada, por exemplo, para projetar estruturas mais resistentes a abalos sísmicos, tendo importância elevada em áreas próximas ao cinturão sísmico [3].

GA já foi utilizado também para decidir a localização, baseado em atividades sísmicas, de estações de sensoriamento no México [32]. Ele foi utilizado como uma ferramenta de projeto para construir uma rede de estações em diferentes regiões do México, objetivando formar uma rede com estações em locais ótimos, afim de alertar a população o mais rápido possível para evitar maiores desastres.

Já Nicknam [25] e Kennett e Sambrigde [15] utilizaram EC para determinar parâmetros para modelos de falhas (como epicentro, localização, profundidade, etc.) de um dado sismo.

Huda e Santosa [12] recentemente publicaram um artigo em que buscam determinar, com algoritmos genéticos, a velocidade das ondas P e S no manto e na crosta terrestres. Ondas P são indicadas como a primeira falha encontrada em dados sismológicos e ondas S são as mudanças causadas na fase das ondas P [12]. Essa pesquisa busca obter a estrutura do subsolo japonês e, geograficamente, possui o mesmo foco que a presente pesquisa.

Chapter 4

Análise dos Dados

4.1 Dados de sismos

O foco dessa pesquisa é estudar padrões existentes nas ocorrências de sismos. Para isso é essencial que tenhamos acesso a dados confiáveis, seguros e ricos em detalhes. Pela página da *Japan Metereological Agency* fomos capazes de acessar dados fiéis aos nossos interesses. Os dados obtidos são compostos por longitude, latitude, data e horário da ocorrência, magnitude e profundidade.

Para a base de treino, foram separados os sismos considerados mais uniformes, formando o grupo cuja latitude e longitude na superfície representam abalos em áreas terrestres (terremotos), pouco profundos (acima de 20 km de profundidade) e com magnitude acima de 2.5 grau na escala Richter, durante os anos de 2000 a 2013.

Para selecionar terremotos, foi preciso extrapolar as informações contidas nos dados e buscar uma forma de, a partir dos dados disponíveis, deduzir quais sismos serão considerados terremotos e quais serão considerados maremotos. Para isso foi utilizamos *The Google Elevation API*[3], uma *Application Programming Interface* (API) que tem como objetivo fornecer informações sobre todos os locais sobre a superfície terrestre e oceânica.

A API foi utilizada por oferecer uma interface HTTP para consulta de dados de elevação territorial, que recebe via url os parâmetros latitude e longitude e retorna, em formato *JavaScript Object Notation* (JSON), dados como a altitude relativa à coordenadas fornecidas, dado este necessário para nossos estudos. Os abalos cuja coordenadas na superfície terrestre tivessem altitude relativa ao nível do mar acima de 0.0 foram selecionados.

Ao analisar a nova base de treino percebe-se um elevado aumento na quantidade de terremotos em 2011, ano que o terremoto de Tohoku, magnitude 9 na escala Ritcher, ocorreu, Figura **??**. Esse terremoto causou aumento desproporcional de sismos em todo o Japão, fato que levou a decisão de limitar a base de treino até o ano de 2010.

[3] https://developers.google.com/maps/documentation/elevation/?hl=pt-es

Ainda em relação a base de treino, ela foi alterada para gerar fatias anuais da base de treino anterior, baseadas nos dados cronológicos disponíveis nos dados da JMA. As fatias são definidas da seguinte forma: se a base refere-se a sismos ocorridos em um espaço de 10 anos, a base será divida por 10, gerando fatias anuais (por exemplo, de 2004 até 2005). Dessa forma, espera-se minimizar o *overfitting* (super ajustes) a base de dados.

Overfitting é o uso de modelos ou procedimentos que violam parcimônia, isto é, que incluem mais termos que o necessário ou usam abordagens mais complicadas do que necessárias [**?**]. Visar minimiza-lo, na aplicação GA, é importante para que tenhamos boas generalizações.

Outra limitação imposta a base de dados foi em relação a área analisada. Como buscamos entender os padrões dos sismos, escolhemos quatro regiões do Japão para focar o experimento, Kanto, Kansai, Touhoku e East Japan. A Figura **??** propõe uma visualização dessas regiões no Japão. A seguir descreveremos as quatro regiões.

Kanto Kanto é a região ao redor de Tóquio. É uma área com elevada atividade sísmica durante o período estudado. Essa região foi definida com coordenadas começando em 34.8 Norte, 138.8 Oeste, com 2025 *bins*. Cada *bin* corresponde a uma área de 25km2.

Kansai Kansai é a região que inclui cidades Kyoto, Osaka e outras. Essa área, ao contrário da região de Kanto, possui uma baixa atividade sísmica. Essa região foi definida com coordenadas começando em 34 Norte, 134.5 Oeste, com 1600 *bins*. Cada *bin* corresponde a uma área de 25km2.

Touhoku Touhoku é a região definida como a região ao norte da ilha principal japonesa. Ela possui alguns *clusters* de atividade sísmicas durante o período estudado. Essa região foi definida com coordenadas começando em 37.8 Norte, 139.8 Oeste, com 800 *bins*. Cada *bin* corresponde a uma área de 100km2.

Leste do Japão É a região corresponde a costa leste do Japão. Ela se diferencia da área anterior por incluir tanto áreas terrestres como não-terrestres. Foi nessa área que o sismo M9 de 2011 aconteceu. Essa região foi definida com coordenadas começando em 37 Norte, 140 Oeste, com 1600 *bins*. Cada *bin* corresponde a uma área de 100km2.

Chapter 5

Metodologia Proposta

5.1 Estrutura da solução

Antes de começar o desenvolvimento do código do algoritmo genético foi necessário criar estruturas, funções e cálculos, assim como estabelecer quais seriam os dados utilizados, como obtê-los e depois como definir qual grupo dentre os dados seriam utilizados para treino.

A primeira estrutura definida foi relacionada aos *bins*, $b \in \beta$. Foi necessário definir qual seria o tamanho ideal para cada *bin*, pois, como foi dito no capítulo anterior, a resolução de uma previsão é inversamente proporcional ao tamanho do *bin*. Foi decidido que cada *bin* teria um valor escolhido para representar a variação geográfica de 5 quilômetros, 0.1 graus, utilizado para a construção de cada *bin*.

Em seguida, era necessário estruturar o vetor de expectativas, λ_i, baseado nos eventos reais, ω_i. Após obter os vetores de expectativas, a função do *log-likelihood* e a de L-test foram desenvolvidos como especificado na Seção 2.2.1. Para um maior entendimento do seu comportamento, o valor do *Log-likelihood* foi calculado tanto para os eventos quanto para um vetor de expectativas e observações pseudo-aleatório, e uma comparação entre eles foi feita. E para o entendimento do L-test, esses valores anteriores foram utilizados como os parâmetros da função.

5.2 Definição do Algoritmo Genético - Simple L-test Fitness Function

Inicialmente, para acelerar a prototipação, foi decidido que o Algoritmo Genético seria implementado utilizando *Distributed Evolutionary Algorithms in Python* - DEAP [4], um *framework* para prototipação rápida que almeja por deixar estruturas de dados e algoritmos transparentes [4]. Para começar o aprendizado de implementações foram utilizados os exemplos de algoritmo genético disponíveis no sub-diretório do framework como base de aprendizado.

Foi priorizado o desenvolvimento de um aplicação simples, para que fosse possível analisar o comportamento da função de *fitness*. Por esse motivo, por enquanto, o foco das atividades não foi direcionado para os parâmetros da evolução. Por exemplo, *crossover* e mutação ainda possuem valores fixos que não consideram a características da população ou dos indivíduos, porém são capazes de garantir uma evolução da população, não prejudicando o objetivo inicial da aplicação.

Para a função de *fitness* foi utilizado o L-test, comparando o *likelihood* dos eventos com o *likelihood* dos indivíduos que compõe a população. O indivíduo mais apto, aquele com maior valor do L-test, era mantido na próxima geração, garantindo que a solução de uma geração seja igual ou maior que a da geração anterior.

Janikow e Michalewicz, em [14], indicam que utilizar uma representação em ponto flutuante deve ter maior precisão, ser mais consistente, possuir acurácia em performance maior quando comparada a uma representação por bits. Portanto, a escolha de utilizar indivíduos definidos como vetores de números reais, iniciados pseudo-aleatoriamente, pode ser explicada tanto pelos motivos acima descritos tanto pela definição do próprio problema (calcular um modelo de probabilidades).

Uma vez que não foi investido muito esforço na análise dos valores escolhidos, o tamanho da população inicial (assim como os valores dos parâmetros) foi definido por tentativa e erros, até que um tempo de convergência aceitável fosse encontrado.

5.2.1 Crossover

Os resultados obtidos a partir de execuções do *Simple L-test Fitness Function* utilizaram dois operadores de *crossover*, o *Blend* e o *Two Points*, que serão descritos a seguir.

O *crossover Blend*, (BLX-α) é um operador para representações em ponto flutuante. Como foi definido em [42], a geração futura é gerada da seguinte forma:

1. Escolha dois pais, x^1 e x^2, aleatoriamente.
2. Um valor para cada elemento do filho x_i^c da próxima geração é escolhido aleatoriamente do intervalo $[X_i^1, X_i^2]$ da seguinte distribuição:

$$\begin{aligned} X_i^1 &= min(x_i^1, x_i^2) - \alpha d_i \\ X_i^2 &= max(x_i^1, x_i^2) + \alpha d_i \\ d_i &= |x_i^1, x_i^2| \end{aligned} \tag{5.1}$$

   onde x_i^1 e x_i^2 são o i-ésimo elementos de x_i^1 e x_i^2, respectivamente, e α é um parâmetro positivo.

Herrera, Lozano e Verdegay [11] afirmam que com valor de $\alpha = 0.5$ a relação entre convergência (*exploitation*) e divergência (*exploration*) alcança equilíbrio. A escolha do

valor deα da aplicação GA é de 0.5, baseada nesta afirmação e na relação de equilíbrio.

O *crossover Two Points* foi utilizado para efeito de comparação entre o operador para números reais e operadores tradicionais. Definido por Goldeberg em [9], pode seguir dois passos. Esse *crossover* é uma instância do *n-point crossover*, que por sua vez é uma generalização do *simple crossover*, [11].

Primeiro, dois pais são escolhidos para a reprodução aleatoriamente. Depois, cada pai é dividido em duas partes em uma posição k selecionada uniformemente e aleatoriamente entre 1 e o tamanho l dos indivíduos menos 1, $[1, l-1]$. O filho é, então, gerado ao trocar os dados das posições $k+1$ até o fim do indivíduo.

Em [11], os autores citam o trabalho de Schaffer, Caruana, Eshelman [37], para definir o *n-point crossover*. O *crossover* em questão, segue os mesmos princípios que o *simple crossover*, porém escolhe n posições aleatoriamente e esses segmentos criados que são trocados para gerar o filho. No caso do *Two Points*, $n = 2$.

5.2.2 Mutação

O operador de mutação escolhido foi o *FlipBit* que é desenvolvido para implementações binárias. Ele inverte o valor de um atributo escolhido aleatoriamente da entrada do indivíduo e usa uma probabilidade para decidir se um atributo sofrerá a mutação, novamente definido em [9].

5.2.3 Seleção

Para seleção foi utilizado Torneio Simples e Elitismo. O Torneio Simples foi utilizado para selecionar, baseado nos valores de *fitness* obtidos, os indivíduos para reprodução. O Elitismo foi empregado a fim de ter-se o melhor membro da população presente na geração seguinte, garantindo que a geração atual seja igual ou superior a anterior, quando analisados valores de *fitness*.

Os parâmetros utilizados pela aplicação *Simple L-test* estão descritos na tabela 6.2.

Tamanho da população	500
Número de gerações	100
cxBlend α	0.5
Mutação FlipBit	0.2
Mutação do atributo (indpb)	0.05
Tamanho do torneio	3

Table 5.1: Parâmetros utilizados.

5.3 Definição do Algoritmo Genético - Time-slice Log-Likelihood Fitness Function

Após análises realizadas a partir dos resultados obtidos pela metodologia anterior que demonstram a ocorrência de *overfitting* (situação descrita no Capítulo 6), algumas alterações foram realizadas.

A primeira mudança foi em relação a função de *fitness*. Ela foi modificada, passando-se a utilizar o valor do *Log-Likelihood*. Já a segunda mudança, foi em relação a base de treino. Ela foi alterada para gerar fatias anuais da base de treino anterior, baseadas nos dados cronológicos disponíveis nos dados da JMA. As fatias são definidas da seguinte forma: Se a base refere-se a sismos ocorridos em um espaço de 10 anos, a base será divida por 10, gerando fatias anuais (por exemplo, de 2004 até 2005). A última foi em relação a área analisada. Como buscamos entender os padrões dos sismos, escolhemos uma região do Japão para focar o experimento, a região de Kanto. Essa região foi escolhida por conter uma grande quantidade de dados terrestres.

5.3.1 Operadores Estudados

A escolha dos operadores do *Time-slice Log-Likelihood Fitness Function* será feita diferentemente do ocorrido com o *Simple L-test Fitness Function*. Para o segundo, as escolhas tiveram de caráter arbitrário e, em contrapartida, as escolhas para o primeiro serão feitas comparações entre o desempenho da aplicação para cada combinação de operadores.

A fim de obter resultados consistentes e gerais, foram feitas 50 rodadas de execuções de cada combinação de operadores visando analisar valores médios. Dessa forma, pontos dispersos que não representam o comportamento da aplicação terão pequena ou nenhuma influência nas análises finais.

Descrição dos Operadores do pacote DEAP

Uma vez que operadores específicos ainda não foram definidos, todos os operadores disponíveis no pacote DEAP possíveis de serem testados serão explicados a seguir (com exceção para o *Two Points*, o *Blend*, *Flip Bit* e o *Tournament* (já descritos anteriormente). Todas as descrições a seguir foram adaptadas da página do DEAP[4].

Os parâmetros utilizados pela aplicação *Time-slice Log-Likelihood* estão descritos na tabela 5.2.

Crossover Os operadores de *crossover* testados foram: *One Point*, *Uniform*, *Two Points*, *Partially Matched*, *Ordered* e *Simulated Binary*.

[4] http://deap.gel.ulaval.ca/doc/default/api/tools.html

Tamanho da população	500
Número de gerações	100
Operador de *crossover*	0.9
Operador de mutação	0.1

Table 5.2: Parâmetros utilizados.

One Point Executa o *crossover* de um ponto. Os dois indivíduos de entrada (pais) são modificados e os indivíduos resultantes (filhos) tem o mesmo tamanho dos pais, respectivamente.

Parâmetro (1)	O primeiro indivíduo participante da operação
Parâmetro (2)	O segundo indivíduo participante da operação
Retorno	2 indivíduos modificados

Table 5.3: Parâmetros e retorno do *One Point*

Uniform Executa um *crossover* uniforme que modifica os dois pais. Os atributos são trocados de acordo com uma probabilidade, usualmente com valor 0.5.

Parâmetro (1)	O primeiro indivíduo participante da operação
Parâmetro (2)	O segundo indivíduo participante da operação
Parâmetro (3)	Probabilidade independente para troca de cada atributo
Retorno	2 indivíduos modificados

Table 5.4: Parâmetros e retorno do *Uniform*

Partially Matched Executa, nos pais, um *crossover* parcialmente correspondido. Os indivíduos gerados são criados pela correspondência de pares de índices dos pais em um dado intervalo e pela troca dos valores de seus índices.

Parâmetro (1)	O primeiro indivíduo participante da operação
Parâmetro (2)	O segundo indivíduo participante da operação
Retorno	2 indivíduos modificados

Table 5.5: Parâmetros e retorno do *Partially Matched*

Uniform and Partially Matched Executa uma combinação entre o *Uniform* e o *Partially Matched*. Segue o mesmo comportamento desse último, porém o pareamento dos índices é feito aleatoriamente a partir de uma probabilidade, assim como o *Uniform*.

Parâmetro (1)	O primeiro indivíduo participante da operação
Parâmetro (2)	O segundo indivíduo participante da operação
Retorno	2 indivíduos modificados

Table 5.6: Parâmetros e retorno do *Uniform and Partialy Matched*

Ordered Gera "buracos" nos indivíduos de entrada. São criados quando um atributo de um indivíduo está entre 2 pontos de outro indivíduo. O elemento é rotacionado de tal forma que todos os elementos entre os pontos de *crossover* são preenchidos com os elementos removidos, em ordem.

Parâmetro (1)	O primeiro indivíduo participante da operação
Parâmetro (2)	O segundo indivíduo participante da operação
Retorno	2 indivíduos modificados

Table 5.7: Parâmetros e retorno do *Ordered*

Ordęred

Simulated Binary Faz um *crossover* por simulação binária que modifica os indivíduos de entrada. Espera uma seqüência de indivíduos em ponto flutuante. Além de dois indivíduos, recebe também um valor para β (Um β alto produz filhos parecidos com os pais e um β baixo produz filhos mais diferentes).

Mutação Os operadores de mutação testados foram: *Shuffle Indexes* e *Uniform Integer*.

Shuffle Indexes Embaralha os atributos do indivíduo de entrada. Geralmente aplicada em vetor de índices.

Parâmetro (1)	O primeiro indivíduo participante da operação
Parâmetro (2)	Prob. independente para troca de cada atributo para outra posição
Retorno	1 indivíduo modificado

Table 5.8: Parâmetros e retorno do *Shuffle Indexes*

Uniform Integer Aplica mutação no indivíduo ao substituir alguns de seus atributos por um inteiro uniformemente retirado de um intervalo definido.

Parâmetro (1)	O primeiro indivíduo participante da operação
Parâmetro (2)	Prob. independente para troca de cada atributo para outra posição
Retorno	1 indivíduo modificado

Table 5.9: Parâmetros e retorno do *Uniform Integer*

Seleção Os operadores de seleção testados foram: *Roulette* e *Random*, *Best* e *Worst*.

Roulette Seleciona indivíduos a partir de giros da roleta. A seleção é feita levando-se em conta espaços que representam a aptidão relativa de cada indivíduo.

Parâmetro (1)	1 lista de indivíduos a serem selecionados
Parâmetro (2)	K, número de indivíduos a selecionar
Retorno	1 lista dos indivíduos selecionados

Table 5.10: Parâmetros e retorno do *Roulette*

Random Seleciona indivíduos aleatoriamente da entrada, com substituição.

Parâmetro (1)	1 lista de indivíduos a serem selecionados
Parâmetro (2)	K, número de indivíduos a selecionar
Retorno	1 lista dos indivíduos selecionados

Table 5.11: Parâmetros e retorno do *Random*

Best Seleciona os melhores indivíduos.

Parâmetro (1)	1 lista de indivíduos a serem selecionados
Parâmetro (2)	K, número de indivíduos a selecionar
Retorno	1 lista dos indivíduos selecionados

Table 5.12: Parâmetros e retorno do *Best*

Worst Seleciona os piores indivíduos.

Parâmetro (1)	1 lista de indivíduos a serem selecionados
Parâmetro (2)	K, número de indivíduos a selecionar
Retorno	1 lista dos indivíduos selecionados

Table 5.13: Parâmetros e retorno do *Worst*

5.4 Estudo dos Operadores - Funções CEC'13

Com objetivo único de aprimorar os conhecimentos sobres os operadores do DEAP e entender melhor o comportamento de Algoritimos Genéticos em problemas de otimização, as primeiras análises foram feitos sobre a suíte de testes do CEC'13 - Congress on Evolutionary Computation - que inclui 28 funções de referência. Dentre as 28 funções, oito representavam um grupo interessante para nossos objetivos. Todas as oitos são funções compostas que combinam as propriedades das sub-funções além de terem comportamento contínuo próximo ao ótimos locais e global. Para maiores informações acerca das funções, recomendamos a leitura do artigo [**?**].

O CEC, além de fornecer todas as informações necessárias para o entendimento das funções e suas características, ainda disponibiliza códigos da suíte em C, Java e Matlab.

Uma vez que a integração entre Python e C é facilitada pelo uso da biblioteca *ctypes*, não foi necessário implementar nenhuma das oito funções. Somente foi necessário criar essa integração entre o código em C disponibilizado e a aplicação de GA em python. O método utilizado pode ser encontrado junto ao repositório Git do Peabox [6].

5.5 Análises GA - Time-slice Log-Likelihood Fitness Function

Posteriormente, estendemos os estudos para o contexto da aplicação. Seguimos duas direções para termos objetivos comparativos: a primeira, testar a GA *Time-slice Log Likelihood Fitness Function* com as diferentes combinações de operadores; a segunda, testar o comportamento da mesma GA juntamente da técnica de pesos adaptativos dos operadores.

A técnica de pesos adaptativos utiliza valores variáveis para os operadores ao longo dos ensaios, valores que periodicamente são ajustados para refletirem a performance recente do operador. A maior justificativa para seu uso é: operadores podem mudar de importância ao longo do ensaio, e, portanto, sua influência deve ser alterada de acordo com o grau de importância adequado.

Para a nossa aplicação, utilizamos operadores mais tradicionais, a fim de facilitar futuras análises e comparações. O operador de *crossover* utilizado foi o *Two Points*, o de mutação, *Shuffle Indexes* e o de seleção, a roleta. A probabilidade inicial para o *crossover* foi de 0.9 e para a mutação, 0.1, valores arbitrários. Como inicialmente queremos abranger o espaço de busca o máximo possível, priorizamos o operador *crossover*. Ao final da execução, o espaço de busca já está bem definido, e, nesse caso, queremos especificar a busca, e, portanto, elevamos a prioridade de ocorrência de mutação para 30% mais provável (conseqüentemente, precisamos diminuir a probabilidade do *crossover* em 30%). Essa variação nos valores dos operadores acontece suavemente durante os ensaios, quando, a cada ro-

[6] https://github.com/stromatolith/peabox/tree/master/cec2013_testfuncs_via_ctypes

dada, os valores dos operadores é alterado pelo valor = [30%/(número de gerações)].

Algumas adversidades foram encontradas durante as execuções, que contribuíram para o atraso da definição dos operadores. Entre elas, a de maior destaque foi a fraca performance do cálculo do fatorial, necessário para a função de *fitness*. Para melhor compreender o comportamento da aplicação e descobrir e confirmar quais eram gargalos da aplicação, foi utilizado um *profiler*[7].

Um *profiler* é uma ferramenta para análise dinâmica da execução de programas. A motivação é analisar o quanto de recurso computacional cada parte do código consome. Em específico, o *profiler* verifica a freqüência e a duração das chamadas do código. Após a descoberta dos pontos críticos e a confirmação de que o função de cálculo do fatorial realmente era um ponto crítico, substituímos a função por uma tabela em memória dos valores do fatorial.

As análises realizadas com as funções CEC'13, ao contrário do esperado, não nos direcionaram na escolha de um grupo de operadores capaz de oferecer uma desempenho significativamente maior, seja ele em termos de convergência mais rápida ou em termos de soluções mais robustas.

[7] https://pythonhosted.org/line_profiler/

Chapter 6

Resultados Obtidos

6.1 Simple L-test Fitness Function

O operador escolhido como parâmetro de *crossover* foi o *Blend*, e a preferência por ele dentre os demais implementados pelo pacote DEAP foi pela observação empírica de crescimento dos valores de L-test e pelo fato de o operador ser específico para indivíduos formados por números reais, situação encontrada na aplicação. A média do resultado de 10 execuções:

Primeira geração (s)	145.6644
Última geração (s)	113.4796
Valor do L-test da primeira geração	-0.635949214
Valor do L-test da última geração	-0.009772415

Table 6.1: Tempo gasto e valor do L-test na média de 10 execuções com Blend.

Operador de *crossover Two Points* foi utilizado por motivos de comparação. A média do resultado de 10 execuções:

Primeira geração (s)	129.03775
Última geração (s)	97.00832
Valor do L-test da primeira geração	-0.635856248
Valor do L-test da última geração	0.042102222

Table 6.2: Tempo gasto e valor do L-test na média de 10 execuções com Two Points.

Tanto o modelo preferido quanto o modelo comparativo foram capazes de evoluírem, obtendo valores finais médios superiores aos valores médios aleatórios iniciais.

Surpreendemente, por [14], era esperado que representações em ponto flutuante tivessem um desempenho de maior acurácia. Porém, o desempenho do operador específico para números reais, o *Blend*, foi menor quando comparado a um operador de uso mais geral, o

Two Points, sendo mais lento e obtendo valores de L-test menores, na média.

Todos os dados foram obtidos após a execução do algoritmo em um computador Apple MacBook Pro com processador 2.9 GHz Intel Core i7, memória RAM 8 GB 1600 MHz DDR3 com sistema operacional OS X 10.9.1 (13B42).

Para melhor visualização e aumentar o poder de comparação, quatro figuras podem ser analisadas a seguir. As Figuras **??** e **??** mostram as médias do valores do L-test para a todas as populações enquanto que as figuras **??** e **??** mostram as médias valores do tempo também para todas as populações. Figuras **??** e **??** são referentes a execuções com o *crossover Blend* e as figuras**??** e **??**, com o *crossover Two Points*.

Por observação empírica das diversas execuções realizadas o maior tempo gasto é com cálculos de L-test, sendo influenciado principalmente pela quantidade de observações e pelo tamanho escolhido para *bins*. Quanto menor for o tamanho escolhido para o *bin* maior será a resolução do terreno analisado e, conseqüentemente, mais informações sobre ele teremos e maior será o espaço de busca, aumentando o tempo total gasto.

Há um grande aumento do valor do L-test entre a vigésima geração e a quadragésima geração. Isso significa que a função de *fitness* resulta em um excessivo valor de *overfitting*, um super ajuste a base de dados.

6.2 Time-slice Log Likelihood Fitness Function

Os resultados da aplicação GA it Log Likelihood Fitness Function serão demonstrados a seguir.

Foram feitas comparações entre os resultados dos valores de *fitness* dos melhores indivíduos e o desvio padrão da população a que ele pertence. Esses resultados comparados são referentes ao estudo dos grupos de operadores e a técnica de pesos adaptativos. A seguir, além dos resultados obtidos, algumas Figuras utilizadas para comparações serão mostradas.

As Figuras **??** e **??** demonstram o resultado das 50 execuções, mostrando dados referentes a execuções do conjunto de operadores *One Point, Worst* e *Shuffle Indexes* para os anos de 2000 (quando há um ganho de desempenho ao utilizar-se a tabela) e de 2010 (quando há uma perda de desempenho). As Figuras mostram o melhor indivíduo e o desvio padrão de sua população. Está claro que existe os resultados não são consistentes e que existe alguma falha na aplicação.

Essa falha ficou clara após a mudança do cálculo de fatorial pela tabela em memória. Até onde x = 27 (x se refere ao número da execução), os valores obtidos vieram de cálculos do fatorial, a partir disso, foi utilizada a tabela. Porém, devido a essa falha, não é possível qualificar se algum grupo gerou indivíduos mais aptos. Portanto é essencial corrigi-la para

novas execuções sejam geradas e as comparações possam ser refeitas.

Foram duas as causas levantadas para essa falha. A primeira, refere-se ao valor limitante para a tabela do fatorial. Uma vez que a tabela possui valores até o fatorial de 100, quaisquer valores acima desse limitante terão seus fatoriais arredondados a 100. Há casos que essa aproximação foi vantajosa e a aplicação respondeu com modelos de melhor desempenho, mas em alguns casos, a aproximação foi desvantajosa.

A segunda, refere-se a um erro no código da execução. Após uma análise superficial do código, é provável que o mapeamento dos dados coletados esteja propagando algum erro. Esforços futuros serão direcionados para a correção deste erro.

Ainda assim, a tabela em memória do valores do fatorial resultou em um avanço em termos de performance temporal. Uma vez que a segunda possível causa seja o verdadeiro erro da aplicação, acredita-se que o uso da tabela será capaz de facilitar futuras execuções.

Os resultados obtidos pela técnica de pesos adaptativos não foram tão elevadas quanto o esperado. Para efeitos comparativos dois grupos de figuras são mostrados a seguir. As Figuras **??** e **??** mostram os valores dos melhores indivíduos e o desvio padrão da população para o ano de 2000 e compõem o primeiro grupo, já o segundo grupo é composto pelas figuras **??** e **??** e segue o mesmo princípio, porém para o ano de 2010. No primeiro grupo, a Figura **??** refere-se aos dados da técnica em questão, enquanto que a outra Figura refere-se ao mesmo conjunto de operadores utilizados (*crossover*, *Two Points* e *Shuffle Indexes*), mas com pesos fixos. O segundo grupo é descrito da mesma maneira.

Não fica claro, pelos dados mostrados, quais são os benefícios que a técnica introduz para a aplicação em questão. Após os mais diversos tipos de análises, análises sobre a influência dos grupos de operadores, seja na própria aplicação GA ou mesmo nas funções da CEC'13 e análises sobre diferentes técnicas para estruturação da abordagem GA, não ficou claro qualquer tipo de variação substancial capaz de direcionar os estudos e definir uma abordagem única para os experimentos. Portanto, deve-se fazer a devida ponderação: será que as variações realmente influem pouco nos resultados da aplicação, ou a aplicação deve passar por refinamentos, que levem a resultados mais consistentes?

Para responder essa ponderação devemos analisar as escolhas feitas para a criação de nosso modelo. Pela própria estrutura definida pelo modelo proposto, cada *bin* é tratado independentemente dos seus vizinhos, não considerando a influência de sismos próximos. Porém, pela características inerentes aos sismos é claro que poucos são os casos de sismos completamente independentes entre si. Podemos considerar, logo, que a influência da independência influi consideravelmente e é ela a principal causa da mínima variação pré-citada.

Por outro lado, as funções CEC'13 são funções unicamente matemáticas e por isso a questão levantada anteriormente não é suficiente para sugerir que a resposta a nossa ponderação deva ser direcionada somente para o segundo ponto da pergunta, "a aplicação deve passar por refinamentos". Devemos deixar claro, entretanto, que essa independência

entre os *bins* deve ser melhor explorada a fim de encontrar uma solução que possibilite considerar a dependência dos sismos. Nesse ponto, percebemos que a complexidade da questão é elevada.

Apesar da utilização da técnica de pesos adaptativos não ter contribuído para uma performance mais significativa, fomos capazes de observar que os valores médios do melhores indivíduos durante os anos foram muito mais coesos do que quando comparados aos valores similares dos ensaios com pesos fixos. Isso nos leva a crer que utilizar a técnica indiretamente influiu positivamente em nossos resultados. Consequentemente, a idéia de abranger uma área maior de busca e posteriormente especifica-la, mostrou-se, como esperado, bastante adequada.

Ainda em relação a técnica anterior, é possível que outras contribuições não tenham sido percebidas e que, de acordo com as devidas mudanças, sejamos capazes de percebê-las. Essas mudanças tanto podem ser algumas das já citadas como também na porcentagem de variação dos pesos dos operadores, que é de 30% (escolhido arbitrariamente), nos valores desses pesos na situação inicial, entre outras, ou seja, variações na estrutura utilizada juntamente com a técnica.

Chapter 7

Conclusão

Este projeto apresentou uma proposta de desenvolvimento de um modelo de previsão de probabilidades elaborado a partir de uma implementação simples de algoritmos genéticos. Foi possível perceber uma evolução do modelo em relação ao modelo completamente aleatório, caracterizado pela população inicial.

Foram muitos os desafios enfrentados ao utilizar os testes propostos pelo RELM como função de *fitness* pelo algoritmo genético. O primeiro está relacionada ao tempo de execução e o uso desses a cada geração como função de *fitness*. Por se tratar de testes com uma grande quantidade de cálculos em grande quantidades de observações, o tempo gasto para analisar as informações poderia ser demasiado grande para ser viável continuar a utilizar os testes junto ao algoritmo. O segundo problema estava vinculado ao comportamento desses testes em uma aplicação de algoritmos genéticos, pois não havia conhecimento anterior sobre a aplicabilidade desses com Computação Evolutiva e se aplicável, se o resultado seria promissor.

Pelos resultados finais do L-test, foi verificado que tanto o primeiro desafio quanto o segundo, não foram suficientes para impossibilitar o uso dos testes e que as gerações de populações resultaram em indivíduos mais aptos para a previsão de sismos. Pelos resultados mostrados o trabalho mostra que aplicar Computação Evolutiva para prever ocorrências de sismos é minimamente promissor, uma vez que há uma evolução do modelo em relação ao modelo pseudo-aleatório.

Foi possível, portanto, vincular os testes proposto pelo RELM com algoritmos genéticos. A partir disso, agora, deve-se explorar as características da aplicação, tornar os testes implementados mais completos e estruturados, definir os operadores genéticos visando um comportamento adequado em relação aos indivíduos como para as populações evoluídas.

Posteriormente, o primeiro desafio foi minimizado, pelo uso de uma tabela em memória do fatorial. Alguns estudos deverão ser realizados para compreendermos melhor o porquê da alteração dos valores após a introdução da tabela.

O uso de operadores de peso adaptativos mostrou-se interessante e é provável que, sua inclusão na GA estudada traga ainda mais benefícios assim que outros problemas forem

resolvidos. Problemas, tais quais, a independência dos *bins*, que influenciaram negativamente a performance da aplicação e serão alvos de maiores esforços.

Algumas propostas de melhorias e trabalhos futuros podem ser listadas:

• Compará-lo com um outro modelo de previsão de terremotos, como por exemplo, o *Relative Intensity* (RI). Para melhores efeitos de comparação, é interessante acrescentar cálculos de confiabilidade estatísca, como *p-value*, e criar um mapa demonstrativo da previsão relativa de cada área representada pelos *bins*;

• Implementar o operador de mutação específico para números reais e que tenha um comportamento direcionado a aplicação, capaz de alterar seus valores no decorrer das gerações capaz de equilibrar *explotaition* e *exploration*;

• Aplicar alguns testes sugeridos por Zechar[**?**] afim de definir a qualidade do modelo desenvolvido;

• Fazer analises de complexidade do algoritmo executado;

• Criar múltiplas observações para cada indivíduo para que o cálculo de incertezas possa ser aplicado e, assim, aumentarmos a qualidade dos modelos;

• Realizar experimentos em outras áreas do Japão, além de Kanto;

• Especializar a abordagem de algoritmo genético (*crossover* mais apropriado, mutação específica, ...) ou utilizar soluções híbridas, entre algoritmo genéticos e outras técnicas de aprendizado de máquina;

• Incrementar os cálculos estatísticos com as ferramentas adequadas, desvio padrão, variância, etc, que compõe os cálculos dos testes demonstrados;

• A fim de evitar o *over fitting*, podemos inserir dados sismológicos, como sobre de magnitude e hora da ocorrência, aumentando a área de busca da aplicação;

• Cada *bin* é trato e considerado individualmente. Portanto a ocorrência de sismos em um *bin* não influência ocorrências de sismos em seus vizinhos, o que sabemos não ser verdade;

• Contrariando expectativas, execuções com operadores reais obtiveram desempenho pior que com operadores tradicionais pode estar relacionado ao fato de não existir influência entre os *bins*, uma vez que pode haver áreas de altíssima probabilidade próximas a áreas de baixa probabilidade.

Referências

[1] Claus Aranha, Yuri Cossich Lavinas, Marcelo Ladeira, and Bogdan Enescu. Is it possible to generate good earthquake risk models using genetic algorithms? In *Proceedings of the International Conference on Evolutionary Computation Theory and Applications*, pages 49–58, 2014. 2, 5

[2] Claus Aranha, Ryoji Tanabe, Romain Chassagne, and Alex Fukunaga. Optimization of oil reservoir models using tuned evolutionary algorithms and adaptive differential evolution. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 877–884. IEEE, 2015.

[3] Ali Firat Cabalar and Abdulkadir Cevik. Genetic programming-based attenuation relationship: An application of recent earthquakes in turkey. *Computers and Geosciences*, 35:1884–1896, October 2009. 9

[4] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. Deap: A python framework for evolutionary algorithms. In *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion*, GECCO Companion '12, pages 85–92, New York, NY, USA, 2012. ACM. 12

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[6] David Eberhard. Multiscale seismicity analysis and forecasting: Examples from the western pacific and iceland. 2014. 2

[7] Agoston E Eiben and Selmar K Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[8] Aguston E Eiben and Marc Schoenauer. Evolutionary computing. *Information Processing Letters*, 82(1):1–6, 2002.

[9] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. 4, 14

[10] Agnes Helmstetter and Didier Sornette. Predictability in the epidemic-type aftershock sequence model of interacting triggered seismicity. *Journal of Geophysical Research: Solid Earth (1978–2012)*, 108(B10), 2003.

[11] Francisco Herrera, Manuel Lozano, and Jose L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial intelligence review*, 12(4):265–319, 1998. 13, 14

[12] A. M. Huda and Bagus Santosa. Subsurface structure in japan based on p and s waves travel time analysis using genetic algorithm in japan seismological network. *International Journal of Science and Engineering*, 6(1), 2014. 9

[13] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration (extended version). Technical report, Technical Report TR-2010-10, University of British Columbia, Computer Science, 2010.

[14] Cezary Z Janikow and Zbigniew Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In *ICGA*, pages 31–36, 1991. 13, 22

[15] B. L. N. Kennet and M. S. Sambridge. Earthquake location — genetic algorithms for teleseisms. *Physics of the Earth and Planetary Interiors*, 75(1–3):103–110, December 1992. 9

[16] Tienfuan Kerh, David Gunaratnam, and Yaling Chan. Neural computing with genetic algorithm in evaluating potentially hazardous metropolitan areas result from earthquake. *Neural Computing and Applications*, 19(4):521–529, 2010. 9

[17] Tienfuan Kerh, David Gunaratnam, and Yaling Chan. Neural computing with genetic algorithm in evaluating potentially hazardous metropolitan areas result from earthquake. *Neural Comput. Appl.*, 19(4):521–529, June 2010.

[18] Tienfuan Kerh, Yu-Hsiang Su, and Ayman Mosallam. Incorporating global search capability of a genetic algorithm into neural computing to model seismic records and soil test data. *Neural Computing and Applications*, pages 1–12, 2015.

[19] E. Kermani, Y. Jafarian, and M. H. Baziar. New predictive models for the v_{max}/a_{max} ratio of strong ground motions using genetic programming. *International Journal of Civil Engineering*, 7(4):236–247, December 2009. 9

[20] John R Koza, Martin A Keane, and Matthew J Streeter. Genetic programming's human-competitive results. *IEEE Intelligent Systems*, pages 25–31, 2003.

[21] John R. Koza, Martin A. Keane, and Matthew J. Streeter. What's ai done for me lately? genetic programming's human-competitive results. *IEEE Intelligent Systems*, 18(3):25–31, 2003. 2, 4

[22] D. Michie, D. J. Spiegelhalter, and C.C. Taylor. Machine learning, neural and statistical classification, 1994. 2

[23] K. Z. Nanjo. Earthquake forecasts for the csep japan experiment based on the ri algorithm. *Earth Planets Space*, 63:261–274, 2011.

[24] K. Z. Nanjo, H. Tsuruoka, N. Hirata, and T. H. Jordan. Overview of the first earthquake forecast testing experiment in japan. *Earth Planets Space*, 63:159–169, 2011.

[25] Ahmad Nicknam, Reza Abbasnia, Yasser Eslamian, Mohsen Bozorgnasab, and Ehsan Adeli Mosabbeb. Source parameters estimation of 2003 bam earthquake mw 6.5 using empirical green's function method, based on an evolutionary approach. *J. Earth Syst. Sci.*, 119(3):383–396, June 2010. 9

[26] Obsis. O que é um terremoto? http://www.obsis.unb.br/index.php?option=com_content&view=article&id=54&Itemid=65&lang=pt-br, 2015. [Online; acessado: 07-07-2015].

[27] Yosihiko Ogata. Statistical analysis of seismicity: updated version (sa-seis2006). *Computer Science Monographs*, 33:29, 2006.

[28] Yosihiko Ogata and Jiancang Zhuang. Space–time etas models and an improved extension. *Tectonophysics*, 413(1):13–23, 2006.

[29] Fusakichi Omori. On the after-shocks of earthquakes. 1895.

[30] William H. Press, Saul A. Teukolsky, Willian T. Vetterling, and Brian P. Flannery. *Numerical Recipes, The Art of Scientific Computing*. Cambridge University Press, third edition, 2007.

[31] Deap Project. Evolutionary tool. http://deap.gel.ulaval.ca/doc/default/api/tools.html#deap.tools.mutPolynomialBounded, July 2015. [Online; acessado: 08-07-2015].

[32] Josafath I. Espinosa Ramos and Roberto A. Vázques. Locating seismic-sense stations through genetic algorithms. In *Proceedings of the GECCO'11*, pages 941–948, Dublin, Ireland, July 2011. ACM. 9

[33] Paul Reasenberg. Second-order moment of central california seismicity, 1969–1982. *Journal of Geophysical Research: Solid Earth (1978–2012)*, 90(B7):5479–5495, 1985.

[34] Negar Sadat, Soleimani Zakeri, and Saeid Pashazadeh. Application of neural network based on genetic algorithm in predicting magnitude of earthquake in north tabriz fault (nw iran). *Current Science (00113891)*, 109(9), 2015.

[35] A. Saegusa. Japan tries to understand quakes, not predict them. Nature 397, 284, 1999. 2

[36] Bahram Saeidian, Mohammad Saadi Mesgari, and Mostafa Ghodousi. Evaluation and comparison of genetic algorithm and bees algorithm for location-allocation of earthquake relief centers. *International Journal of Disaster Risk Reduction*, 2016.

[37] J David Schaffer, Richard A Caruana, Larry J Eshelman, and Rajarshi Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the third international conference on Genetic algorithms*, pages 51–60. Morgan Kaufmann Publishers Inc., 1989. 14

[38] D. Schorlemmer, M. Gerstenberger, S. Wiemer, D. Jackson, and D. A. Rhoades. Earthquake likelihood model testing. *Seismological Research Letters*, 78(1):17–29, 2007. 5, 6

[39] D. Schorlemmer, M. Gerstenberger, S. Wiemer, D. Jackson, and D. A. Rhoades. Earthquake likelihood model testing. *Seismological Research Letters*, 78(1):17–29, 2007.

[40] Danijel Schorlemmer, J Douglas Zechar, Maximilian J Werner, Edward H Field, David D Jackson, Thomas H Jordan, and RELM Working Group. First results of the regional earthquake likelihood models experiment. *Pure and Applied Geophysics*, 167(8-9):859–876, 2010.

[41] Hiroshi Someya. Striking a mean- and parent-centric balance in real-valued crossover operators. *Transactions on Evolutionary Computation*, 17(6):737–754, December 2013.

[42] Masato Takahashi and Hajime Kita. A crossover operator using independent component analysis for real-coded genetic algorithms. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 643–649. IEEE, 2001. 13

[43] Tokuji Utsu and Yosihiko Ogata. The centenary of the omori formula for a decay law of aftershock activity. *Journal of Physics of the Earth*, 43(1):1–33, 1995.

[44] Yoshiko Yamanaka and Kunihiko Shimazaki. Scaling relationship between the number of aftershocks and the size of the main shock. *Journal of Physics of the Earth*, 38(4):305–324, 1990.

[45] J Douglas Zechar. Evaluating earthquake predictions and earthquake forecasts: A guide for students and new researchers. *Community Online Resource for Statistical Seismicity Analysis*, pages 1–26, 2010. 5

[46] J. Douglas Zechar and Thomas H. Jordan. The area skill score statistic for evaluating earthquake predictability experiments. *Pure and Applied Geophysics*, 167(8–9):893–906, August 2010.

[47] Qiuen Zhang and Cheng Wang. Using genetic algorithms to optimize artificial neural network: a case study on earthquake prediction. In *Second International Conference on Genetic and Evolutionary Computing*, pages 128–131. IEEE, 2012. 9

[48] Feiyan Zhou and Xiaofeng Zhu. Earthquake prediction based on lm-bp neural network. In *Proceedings of the 9th International Symposium on Linear Drives for Industry Applications, Volume 1*, pages 13–20. Springer, 2014. 9

[49] Feiyan Zhou and Xiaofeng Zhu. Earthquake prediction based on lm-bp neural network. In Xiaozhu Liu and Yunyue Ye, editors, *Proceedings of the 9th International Symposium on Linear Drives for Industry Applications, Volume 1*, volume 270 of *Lecture Notes in Electrical Engineering*, pages 13–20. Springer Berlin Heidelberg, 2014.

[50] Qing Zhou and Hong Ye. Demarcation of potential seismic sources on integration of genetic algorithm and bp algorithm. *Acta Seismologica Sinica*, 15(6):677–682, 2002.

[51] Jiancang Zhuang. Gambling scores for earthquake predictions and forecasts. *Geophysical Journal International*, 181(1):382–390, 2010.

[52] Jiancang Zhuang, Yosihiko Ogata, and David Vere-Jones. Analyzing earthquake clustering features by using stochastic reconstruction. *Journal of Geophysical Research: Solid Earth (1978–2012)*, 109(B5), 2004.