Programação Genética Reforçada por CMA-ES aplicada a sinais de *Forex*

Introdução à Investigação

Bruno Costa **81451**

Mentor/Tutor: PhD Student Tiago França

27 de Novembro de 2018 - Instituto Superior Técnico

Resumo

Neste projeto de introdução à programação genética é implementado um algoritmo genético com o objetivo de alcançar um sofisticado indicador técnico capaz de prever movimentos nos preços do mercado financeiro das divisas. Para elevar o grau de sucesso e robustez do algoritmo implementa-se também um outro algoritmo de otimização, CMA-ES. Este último é utilizado para procurar a melhor combinação de parâmetros externos, por estes definirem um espaço de soluções demasiado amplo para serem testados pelo método de tentativa e erro, uma vez que se pretende maximizar a rapidez e eficiência da evolução. As estruturas genéticas são constituídas por funções triviais, tais como função soma, multiplicação, média móvel simples, etc., e após uma extensiva simulação os resultados obtidos apontam para uma insignificância estatística em termos dos parâmetros externos mais adequados. Portanto, mais poder computacional é necessário para poder tirar conclusões mais profundas. Não obstante, o *framework* para mais investigação futura está implementado.

I. Introdução Teórica

Por forma a obter vantagens sobre a competição, instituições financeiras com elevado capital e participantes singulares do mercado financeiro de transações têm devotado os seus esforços a tentar alcançar algoritmos dotados de capacidades de previsão de tendências de mercado de valores. Estas ferramentas de análise e execução incorporam avançados métodos, desde abordagens simples, a avançados métodos como algoritmos genéticos [1] e redes neurais [2], nas quais se destacam poderosos avanços como o NEAT [3] e HYPERNEAT [4].

Essencialmente, todo o processo se caracteriza pelo *input* de dados de *securities* como o preço atual, o preço mais elevado/diminuto da sessão, volume de títulos transacionados, notícias [5], etc. e, após a execução do programa, sinais são gerados, indicando se se deve abrir uma posição de compra ou venda de ações, podendo, inclusive, com recurso ao API de alguns *brokers*, ultimar a ação no mercado automaticamente (*algorithmic trading*).

Todo este projeto é motivado pela utilização de algoritmos genéticos, pois existe a possibilidade de abstração de estruturas em formato de árvores onde constam funções que podem representar indicadores de *performance* de mercado. Os nodos da árvore onde estão definidas estas funções matemáticas realizam operações de forma estruturada e as árvores evoluem segundo a teoria da evolução natural, pretendendo-se obter um indicador de complexidade arbitrária composto pelas funções que inicializam estas estruturas.

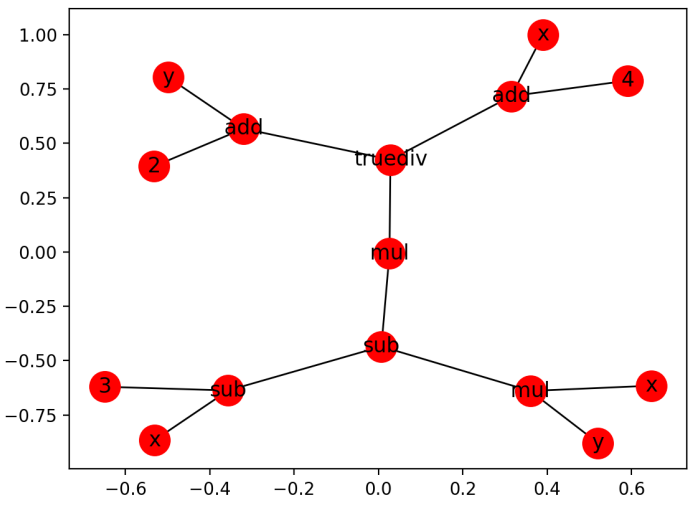

Figura 1: Estrutura de árvore composta por funções triviais.

A Programação Genética

Algoritmos genéticos pertencem a uma classe de *machine learning*, caraterizada pela procura, adaptação e técnicas de otimização baseadas nos princípios da seleção natural. Um algoritmo evolucionário, portanto, processa-se iterativamente do seguinte modo: **1)** é gerada uma população de indivíduos aleatoriamente desenhados para solucionar um problema especificado; **2)** cada um desses indivíduos é prontamente exposto ao problema e através de uma função que mede a sua *performance*, comummente designada de função de *fitness*, o seu sucesso é avaliado e uma classificação atribuída à qualidade da solução correspondente à estrutura genética; **3)** avaliada a população, entra a fase evolucionária, selecionando por diversos critérios um conjunto de indivíduos para a reprodução e recombinando-os através de operadores como o acasalamento, mutação, etc.; **4)** por fim, a próxima geração é criada com a ascendência e o processo repetido até um determinado critério de paragem (até um certo número de gerações, por exemplo), sendo que essa população final fornece um conjunto de candidatos-solução prontos a serem expostos ao problema original, que, se usada uma população inicial geneticamente diversificada, deverão ser maximizantes da performance do mesmo.

O acasalamento envolve dois indivíduos e a sua recombinação dá-se através de um operador de *crossover* que divide as duas estruturas genéticas em pontos aleatórios e as combina para criar dois 'filhos' ou *offsprings*, reproduzindo o processo natural de *crossover* de cromossomas no ADN.

Um teorema designado por *Schema Theorem* demonstra que os algoritmos genéticos alocam automaticamente um número exponencial de tentativas para a estrutura genética mais favorável, levando a um bom compromisso entre as direções mais promissoras do espaço das soluções e a exploração de regiões menos frequentes desse mesmo espaço. Contudo, não existe prova geral que garanta a convergência de um algoritmo genético para um extremo global.

Uma vantagem de extrema relevância na programação genética é a possibilidade de paralelização do processo, pois tarefas como procurar a solução no espaço das soluções com uma população ou avaliar o *fitness* de estruturas genéticas podem ser paralelizadas.

Estes algoritmos podem ser aplicados a problemas com funções de *fitness* não diferenciáveis ou descontínuas e com diversos extremos locais, devido à natureza estocástica dos operadores de seleção e recombinação, sendo menos provável convergir para extremos locais do que métodos do tipo de gradiente. [6]

No entanto, algumas desvantagens podem ser evidenciadas. Manter uma volumosa população de estruturas genéticas de indivíduos leva a um enorme tempo de computação, por ser necessário proceder à avaliação do *fitness* de cada uma dessas estruturas. Ou-

tro aspeto negativo é o facto de estes algoritmos poderem ser menos eficientes do que outros algoritmos desenhados especificamente para o problema em domínios mais bem percebidos, dado que pouco conhecimento acerca do problema pode ser inserido nestas estruturas. E uma fraqueza desvantajosa é a falta de prova teórica para a convergência do resultado. Por fim, apesar das desvantagens, os algoritmos de programação genética podem ser utilizados numa panóplia de problemas, encontrando boas soluções relativamente rapidamente.

B CMA-ES

Uma técnica computacional poderosíssima que pode ser utilizada para identificar extremos locais de uma função num espaço de soluções de um problema de otimização é a técnica da *Covariance Matrix Adaptation-Evolution Strategy*. Resultados em competições internacionais dedicadas ao assunto e outras comparações sistemáticas têm provado que este algoritmo supera amplamente outros métodos de otimização multidimensional. [7] [8]

No âmbito deste artigo, o uso do CMA-ES decorre da necessidade de parametrizar corretamente a dinâmica da evolução, traduzida em fatores como a probabilidade de ocorrência de mutação, de *crossover* e outros que serão descritos em diante. Por as condições fronteira, ou o domínio dos espaços de soluções, estarem bem definidos, a aplicabilidade desta técnica é válida. [9]

C Mercado de Divisas (*Forex*)

O mercado de divisas é um mercado no qual os seus participantes podem comprar, vender, trocar ou especular em moedas. Este mercado é considerado o maior mercado financeiro, com cerca de 5 triliões de dólares transacionados diariamente. Optou-se por incidir o estudo no *forex* pela completude dos dados em minha posse, provenientes da *Bloomberg*, ou seja, existe um acesso generalizado a todos os pares de moedas listados no terminal da *Bloomberg* para os períodos de dias úteis. Adicionalmente, objetiva-se realizar uma análise técnica ao invés de uma análise fundamental. Numa análise fundamental analisar-se-iam aspetos intrínsecos à empresa ou país por detrás da *security* (balanços anuais, *cash flow statements*, etc.). Numa análise técnica admite-se que o preço do *stock* inclui inerentemente todas essas informações, pelo que as técnicas de análise utilizadas circundam indicadores matemáticos, como por exemplo médias simples, médias exponencias, valor mais alto/baixo num determinado período temporal, osciladores estocásticos, índices de força relativa, volatilidade, etc. [10]

II. Implementação, Resultados e Discussão

A Aplicação do algoritmo genético

As fundações computacionais de toda esta investigação centram-se na biblioteca DEAP do *python* que, genialmente pensada e concebida, permitiu uma implementação relativamente célere de algo de extrema complexidade. [11]

Primeiramente, definiu-se a estrutura do indivíduo (gene) em formato de árvore, especificando o conjunto de funções possíveis de serem inicializadas e inseridas na estrutura aquando da formação aleatória dos indivíduos. Cada árvore é do tipo *strongly typed*, significando que as primitivas (nodos intermédios) e os terminais (nodos finais) se interligam por compatibilidade entre *input* e *output*, *i.e.*, as primitivas e os terminais se interligam apenas se tiverem nodos de *input* e *output* com tipos de dados compatíveis, por forma a evitar a construção de árvores onde, por exemplo, uma primitiva que espera receber uma lista de números receba, ao invés, um booleano.

Deste modo, definiu-se o seguinte conjunto de primitivas:

- Simple Moving Average (SMA): média aritmética móvel de um conjunto de dias de início e fim bem definidos;
- Soma: operador de soma simples;
- Lag: função para aceder a posições absolutas de um vetor;
- Subtração: operador de subtração simples;
- Multiplicação: operador de multiplicação simples;
- Divisão: operador de divisão com proteção contra divisões por 0 (retorna grandes números nesse caso);
- IF: função que recebe 4 argumentos e compara os dois primeiros, retornando o terceiro se o primeiro for maior do que o segundo e retorna o quarto no caso oposto.

Relativamente aos terminais, estes são dois tipos de constantes que sempre que são inicializadas

ora retornam um inteiro, com o objetivo de permitir às primitivas SMA e Lag determinar a que índices dos vetores recebidos aceder, ora retornam um *float* no intervalo $[-1,1]$ com probabilidade uniforme, para uso na escrita das fórmulas em si. Por outro lado, o terminal pode ser um objeto na forma de um vetor, que neste caso só pode receber os dados do mercado de valores.

Optou-se por cingir o estudo a este grupo inicial de funções para facilitar a interpretação dos resultados e potenciar um tempo de computação mais diminuto.

Definiu-se também que em cada árvore a profundidade dos ramos era variável entre si até 15 níveis de nodos, onde o tipo de nodos a utilizar é indistintamente escolhido, retornando ao final um *float*.

Ultimada a descrição dos elementos individuais, basta inserir capacidades de evolução. Começa-se por iterar a produção de indivíduos aleatoriamente até obter uma população de tamanho pré-definido. Consecutivamente, define-se a seleção do subconjunto de indivíduos para acasalamento e mutação. Definiu-se que o processo de seleção seria: de uma população de N indivíduos são extraídos N conjuntos aleatórios (denominados torneios) de tamanho **tournsize** onde é feita uma comparação entre os indivíduos, resultando em que o melhor desses conjuntos segue para a próxima geração.

O acasalamento escolhido foi o de aleatoriamente selecionar nodos em cada uma das duas árvores, resultantes do processo anterior, e trocar as subárvores entre si. Dois parâmetros importantes neste processo são o da probabilidade de ocorrer acasalamento, **cxpb**, e o da probabilidade de se selecionar um ponto terminal ou um ponto intermédio da árvore, doravante designado de **termpb**. Novamente, mais tarde serão abordados todos os parâmetros a negrito. Como a árvore é da forma *strongly typed*, a compatibilidade entre *input-output* destas novas conexões está garantida.

Como última capacidade evolucionária, temos a mutação. Esta ocorre com probabilidade **mutpb**, onde num nodo aleatório na árvore se substitui a subárvore seccionada por uma aleatoriamente gerada, seguindo as regras impostas anteriores à introdução da evolução. É de notar que outros tipos de mutação poderiam ter sido implementados, tais como aleatoriamente alterar os números presentes na árvore.

É de notar que tanto no processo de acasalamento como no de mutação a altura máxima que a árvore (indivíduo) resultante poderá ter é de 17 nodos ligados consecutivamente, como sugerido por Koza. Em caso de insucesso, os pais são retornados como filhos sem efetuar o acasalamento. [12]

B Medida de *Fitness*

Para este estudo, e uma vez que o objetivo é, afinal de contas, sinalizar a compra ou venda de títulos de ações do mercado de divisas, definiu-se que o sucesso de cada indivíduo se mede com base na assertividade da previsão da tendência do valor da ação para o dia seguinte. Num estudo mais completo incluir-se-ia a previsão precisa do preço do dia seguinte, mas para o âmbito deste projeto foi decidido medir simplesmente de forma binária a assertividade da previsão da tendência do valor.

Entende-se como previsão correta se o *output* gerado for maior do que 0 quando o preço do dia seguinte é superior ao do último

dia analisado (e vice-versa se for inferior). Como *fitness*, avalia-se o número de vezes em que, após processar os dados até um determinado dia, a árvore retorna um *output* correto para o dia seguinte.

C Dados

Por limitações de tempo, apesar de se estar na posse de dados diários históricos de todos os pares de divisas desde que começaram a ser cambiados no mercado financeiro, escolheram-se os dados *EUR/USD* pelo sua relevância no mercado económico mundial.

O *input* destes dados para a árvores era, não os dados em bruto, mas sim a diferença relativa entre preços consecutivos, $\frac{x_i - x_{i-1}}{x_{i-1}}$, onde x_i é o preço no dia i. Assim, os dados ficam normalizados e adimensionais, um critério usual (tal como em preços logarítmicos) em análise estatística multidimensional e técnicas de *machine learning*. [13]

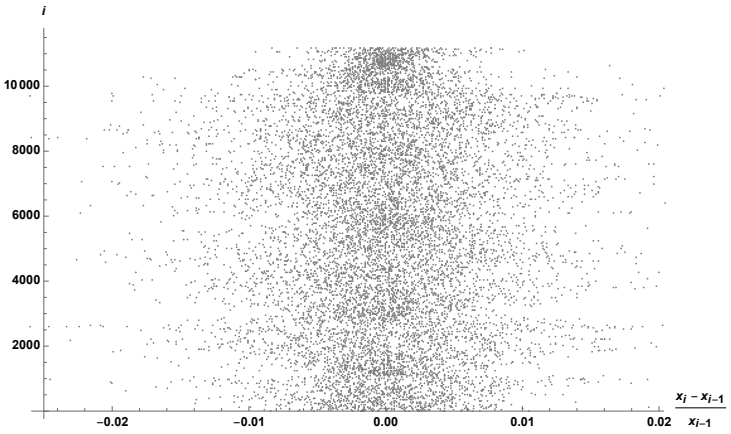

Figura 2: Diferença relativa dos preços *EURUSD*.

Repare-se na arbitrariedade dos dados, visualmente é clara a dificuldade do problema a resolver.

Como método de aprendizagem, julgou-se melhor segmentar/-seccionar estes vetores em subvetores de tamanho **n** e, deste modo, os indivíduos tentariam prever a posição $n+1$, recebendo como input os **n** dias de preços que se antecederam, determinando-se o *fitness* de cada individuo ao varrer todas as posições.

D Dinâmica e Otimização dos parâmetros

Executou-se a simulação uma vez para visualizar a evolução do *fitness* e do tamanho das estruturas em função do número de gerações. Obtiveram-se os seguintes resultados:

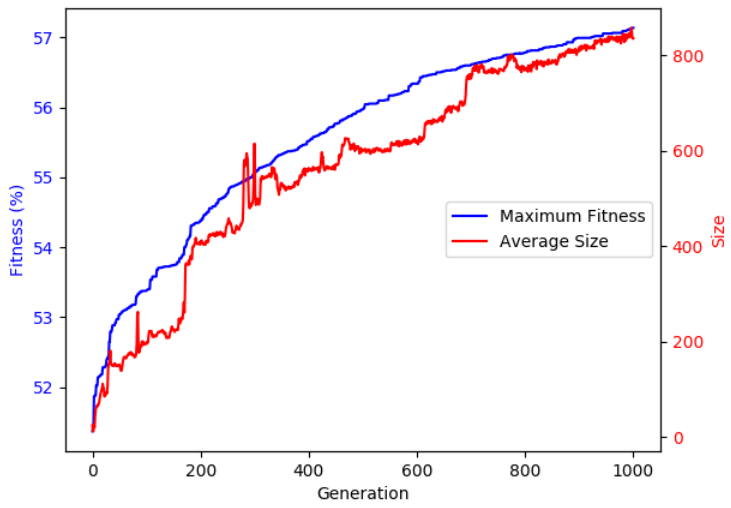

Figura 3: Evolução de um algoritmo genético com o *fitness* máximo e tamanho médio das árvores em função do número de gerações (15h de simulação) - população de 250 indivíduos e 1000 gerações

Concluída a especificação principal do algoritmo, segue-se a fase da simulação para obtenção do indivíduo mais apto a resolver o problema proposto. Não obstante, resta ainda definir o espaço de parâmetros $\mathscr{P}$={**n, tournsize, cxpb, termpb, mutpb**}, bem como o número de população e de gerações, e é aqui que entra o algoritmo de otimização CMA-ES. Uma vez que não é imediato qual a combinação de parâmetros que minimiza o tempo de computação necessário para obter os melhores resultados, incorporou-se este mecanismo na simulação. Esta dificuldade decorre também do facto de não der um problema muito explorado na literatura, *i.e.*, é um problema demasiado específico.

Classicamente, o que se faria era definir o número de população inicial bem como o número de gerações de acordo com o poder computacional disponível e, para um dado espaço de soluções de $\mathscr{P}$, foi o que se fez inicialmente, tendo sido definidas populações e gerações na ordem da centena. Contudo, o espaço de soluções de soluções de $\mathscr{P}$ é vasto demais para sequer considerar o método de tentativa e erro, pelo que se avançou efetivamente com o CMA-ES. Este é um passo essencial para determinar os parâmetros corretos que maximizarão futuras investigações.

Inicialmente, os parâmetros a otimizar são os do conjunto $\mathscr{P}$, ou seja, um espaço a 5 dimensões. De seguida, com o objetivo de obter significância estatística, programou-se o algoritmo do CMA-ES a iterar *reps_CMAES* vezes e, para cada uma destas repetições, o algoritmo genético iterou *reps_GA* vezes. Em detalhe, isto justifica-se do seguinte modo: 1) a população inicial do algoritmo genético é muito diversa, e, não sendo possível incluir uma população inicial demasiado vasta, foi, ao invés, decidido repetir a evolução genética para diversas populações iniciais e obter uma média do melhor *fitness* conseguido por cada; 2) esta abordagem de criar várias populações segue a linha de algoritmos genéticos mais complexos em evolução por ilhas, tomando apenas a simplicidade de não permitir migrações; 3) ainda assim, flutuações estatísticas no algoritmo genético não são possíveis de eliminar. Logo, para levar isto em conta na apresentação final dos resultados de minimização do CMA-ES, este é por si corrido *reps_CMAES* vezes de forma a ter uma noção da distribuição e significância estatística do resultado.

Portanto, com as diversas repetições do CMA-ES, obtiveram-se dados para estimar a média e variância de cada um dos 5 parâmetros. Note-se que o domínio dos parâmetros que definem probabilidades é [0,1], o domínio do **n** varia de 0 ao tamanho do vetor das diferenças relativas dos dados e o domínio do **tournsize** varia de 0 até ao tamanho total da população da geração a ser iterada. Todavia, normalizou-se o **n** e o **tournsize** aos respetivos tamanhos máximos, escolha esta para facilitar o processo estocástico do CMA-ES a partir da escolha usual de parâmetros iniciais (matriz de correlação igual à identidade).

No algoritmo do CMA-ES define-se também o desvio padrão para os quais os valores vão variar e a sua média. Realizou-se a simulação com todos os valores inicias da média iguais a metade do valor do seu domínio (de forma a não introduzir nenhum viés), excetuando a probabilidade de mutação, **mutpb**, cuja média se definiu igual a 0.4, dado que existe uma relação direta entre a probabilidade de mutação e o tempo de computação, uma vez que só são reavaliadas árvores com nodos não presentes na geração anterior e as mutações introduzem constantemente novos nodos. Já o desvio padrão ficou estabelecido em 0.25 para os 5 parâmetros, visto que o domínio de cada parâmetro é $[0,1]$ (valor este próximo do desvio padrão de uma distribuição uniforme (0.29)).

E Dificuldades

O computador ao dispor para desempenhar o processo de simulação possui 6 núcleos, ou seja, virtualmente tem-se 12 núcleos, pelo que se paralelizou o programa por todos os núcleos para tirar vantagem da paralelização inerente ao processo algorítmico, como notado inicialmente.

Contudo, após alguns teste de *benchmark*, concluiu-se ser ne-

cessário selecionar cuidadosamente os parâmetros dos dois algoritmos e racionar o tempo de computação para cada sub-tarefa (de modo a poder entregar os resultados a tempo), nomeadamente as populações do algoritmo genético e do CMA-ES, o número de gerações para ambos e quantas vezes seriam repetidos. Detalhadamente, e de forma a escalar os testes feitos para um alvo de 84h de simulação, definiu-se os parâmetros do seguinte modo:

pop_GA	**pop_CMAES**	**ngen_GA**	**ngen_CMAES**	**reps_GA**	**reps_CMAES**
50	9	180	12	4	2

Tabela 1: Parâmetros iniciais da simulação do CMA-ES

Note-se que o valor para a população inicial do CMA-ES, ***pop_CMAES***, segue a fórmula $\lambda = [4 + \text{Floor}(3 \cdot \ln(D))] + 1$, onde D é a dimensão do problema a solucionar (neste caso 5). [8]

F Resultados Finais

Decorrida a simulação, obtiveram-se os valores finais:

cxpb	**mutpb**	**n** (dias)	**tournsize**	**termpb**
0.29 ± 0.17	0.29 ± 0.08	56 ± 34	19 ± 13	0.35 ± 0.19

Tabela 2: Espaço de soluções $\mathscr{P}$ computado (84h de simulação)

A probabilidade de acasalamento é ligeiramente reduzida face ao esperado, e isto pode-se explicar porque provavelmente foi dada maior relevância à ocorrência de mutações mais frequentemente. Em termos absolutos, o valor de probabilidade de mutação é relativamente alto. Este direcionamento de relevo é consequência do facto do número de gerações ser baixo, logo, em termos evolutivos, as mutações funcionam melhor a curto prazo e o *crossover* funciona melhor a longo prazo.

O tamanho do vetor n é algo incerto (1 a 3 meses de dados), no entanto o baixo valor é positivo, significando isto que a árvore mais forte foi construída com pequenas frações de toda a informação.

Por sua vez, o valor do *tournsize* é de difícil justificação por apresentar um intervalo de incerteza bastante largo, com um desvio padrão equivalente ao que resultaria de uma distribuição uniforme.

Por fim, o valor do parâmetro *termpb* demonstra ser adequado à proporção que existe entre nodos intermédios e nodos terminais, com o algoritmo a dar primazia às primitivas (nodos intermédios). Uma vez que existem funções que recebem vetores e o único vetor presente na árvore nasce do *input* dos dados, torna-se imperativo aceder mais vezes aos terminais, daí o elevado valor deste parâmetro. No entanto, este valor ultrapassa o valor sugerido por Koza, 0.1. [13]

Alguns dos valores apresentam um intervalo de incerteza muito amplo, e isto acontece porque não existe uma amostra estatística suficientemente boa para poder aferir quais os valores e intervalos corretos. Já que não é possível repetir o CMA-ES um número suficiente de vezes, nem é possível tomar grandes populações e extensas gerações no algoritmo genético, a generabilização destes parâmetros fica comprometida quando considerados maiores tempos de computação.

III. Conclusão

Evidentemente, o presente estudo está numa fase muito embrionária. Dada a complexidade de todo o processo, houve alguns aspetos muitíssimo relevantes que tiveram de se reservar para a posterioridade. Por exemplo, cada transação no mercado de valores é acompanhada de uma taxa, logo nem sempre é vantajoso mudar a posição no mercado diariamente, pelo que previsões menos frequentes têm muita relevância quando se consideram esses custos. Para além do mais, imagine-se que se compram/vendem ações no primeiro dia de avaliação dos dados e se fecha a posição no último dia. Os lucros resultantes deste simples e comum cenário têm evidentemente de ser comparados com o cenário programado que é o da atualização da posição no mercado diariamente, considerando-se muitas vezes o primeiro cenário (estratégia de *buy-and-hold*) como *benchmark* principal.

Adicionalmente, a biblioteca DEAP fornece inúmeras outras ferramentas, funcionalidades e seleção de parâmetros que possibilitam um estudo rico e diversificado no campo da programação genética, cuja exploração total extravasava temporalmente o presente âmbito.

Por último, refere-se o grande relevo do estudo efetivado do espaço de parâmetros antes de prosseguir cegamente com métodos de otimização e *machine learning*, algo incomum na comunidade.

Independentemente da qualidade dos resultados, ficou demonstrado que melhorias neste aspeto são consequência direta de menos pressão/limitação temporal nos valores da tabela 1.

Tendo tudo em conta, pode-se afirmar assertivamente que o *framework* para futura investigação foi implementado com sucesso!

Trabalho Futuro

Futuramente, ambiciona-se transferir toda a biblioteca DEAP do *python* para uma versão de C++, já que em termos de eficiência o C++ é superior e problemas desta natureza provam ser extremamente exigentes em termos de tempo de computação.

Pretende-se também inserir mais indicadores técnicos, explorar a novidade/curiosidade como critério de evolução, estudar evolução isolada em 'ilhas', testar novas funções de *fitness* e tentar eliminar redundâncias nas árvores.

Referências

[1] "Introduction to Genetic Algorithms", Vijini Mallawaarachchi, https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3, acedido em Novembro 2018

[2] *Tan P., Steinbach M., Kumar V., "Introduction to Data Mining"*, Pearson, ISBN-10: 0321321367, 2006

[3] *Stanley K., Miikkulainen R,. "Evolving Neural Networks Through Augmenting Topologies"*, Evolutionary Computation, 10(2):99-127, 2002

[4] *"The Hybercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT)"*, Kenneth Stanley, `https://eplex.cs.ucf.edu/hyperNEATpage/HyperNEAT.html`, acedido em Novembro 2018

[5] "What is NLP?", The Empowerment Partnership, `http://www.nlp.com/what-is-nlp/`, acedido em Novembro 2018

[6] *Allen F., Karjalainen R., "Using genetic algorithms to find technical trading rules"*, Journal of Financial Economics, Volume 51, Issue 2, February 1999, Pages 245-271

[7] *Muraro D., Dilão, R., "A parallel multi-objective optimization algorithm for the calibration of mathematical models"*, Swarm and Evolutionary Computation, Volume 8, February 2013, Pages 13-25

[8] *Carapuço J., "Volatility Models in Option Pricing"*, Instituto Superior Técnico, 2018, Pages 95-97

[9] *Araújo R., Oliveira A., Soares S., "A Covariance Matrix Adaptation based Evolutionary Methodology for Phase Adjustment in Financial Time Series Forecasting"*, 2010, Pages 1315-1316

[10] *Badawy F., Abdelazim H., Darwish G., "Genetic Algorithms for Predicting the Egyptian Stock Market"*, International Conference on Information and Communication Technology, 2005, Pages 109-122

[11] *Fortin F., De Rainville F., Gardner M., Parizeau M., Gagné C., "DEAP: Evolutionary Algorithms Made Easy"*, Journal of Machine Learning Research, Volume 13, July 2012, Pages 2171-2175

[12] *Tan P., Steinbach M., Kumar V., "Genetic Programming: On the Programming of Computers by Means of Natural Selection "*, The MIT Press, ISBN-10: 9780262111706, 1992

[13] "Why Log Returns", Quantivity, `https://quantivity.wordpress.com/2011/02/21/why-log-returns/`, acedido em Novembro 2018