Pós-Graduação em Ciência da Computação

Severino Mizael da Silva

USO DE ROBÔS EM BOLSAS DE BITCOIN

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
<www.cin.ufpe.br/~posgraduacao>

RECIFE
2017

Severino Mizael da Silva

USO DE ROBÔS EM BOLSAS DE BITCOIN

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Ruy J. Guerra B. de Queiroz*

RECIFE
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S586u Silva, Severino Mizael da
Uso de robôs em bolsas de Bitcoin / Severino Mizael da Silva. – 2017.
110 f.: il., fig., tab.

Orientador: Ruy José Guerra Barretto de Queiroz.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Ciência da computação. 2. Bitcoin. 3. Robôs. I. Queiroz, Ruy José Guerra Barretto de (orientador). II. Título.

004 CDD (23. ed.) UFPE- MEI 2018-033

Severino Mizael da Silva

Uso de Robôs em Bolsas de Bitcoin

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 28/08/2017

BANCA EXAMINADORA

_____

Prof. Dr. Carlos Alexandre Barros de Mello
Centro de Informática / UFPE

_____

Prof. Dr. Carlo Kleber da Silva Rodrigues
Faculdade de Tecnologia e Ciências Aplicadas / UniCEUB

_____

Prof. Dr. Ruy José Guerra Barretto de Queiroz
Centro de Informática
(Orientador)

Agradecimentos

Esta dissertação foi possível graças ao apoio, incentivo e auxílio de pessoas que estão sempre participando da minha vida acadêmica, desde antes do início até sua conclusão. Agradeço a Deus por tudo. A minha família especialmente minha mãe Josina, que sempre fez o melhor por mim. A minha namorada Gisele, que sempre está ao meu lado apoiando e incentivando. Ao meu orientador Ruy, que foi a pessoa que me orientou ao estudo do bitcoin, e me aceitou como orientando de mestrado. Ao Centro de Informática, que criou o ambiente para todo meu desenvolvimento acadêmico. Aos funcionários da portaria do CIN, que atendem nossas ligações e ajudam a encontrar os professores. Ao pessoal da secretaria, que sempre está trabalhando para permitir nossa formação acadêmica. A todos os professores da graduação e mestrado, que participaram ativamente na construção do meu conhecimento. Ao meu amigo Diógenes, que sempre me incentivou na vida acadêmica. Aos meus amigos que moravam comigo e aos que estudaram comigo durante todo esse período. A todos os integrantes da banca pela participação que irão ter na conclusão dessa dissertação.

Aprende que não importa até o ponto onde já chegamos, mas para onde estamos, de fato, indo

—WILLIAN SHAKESPEARE (Um dia você aprende)

Resumo

Apresentamos um experimento de criação de robôs, para atuar em algumas bolsas de bitcoins. Estas bolsas fornecem um ambiente online para negociar bitcoins, que funciona 24 horas todos os dias, o mercado é global e permite negociar moedas fiduciárias por moedas digitais, algumas delas são: real, dólar, euro, bitcoin, ethereum e litecoin. Esse mercado de câmbio tem alta volatilidade, grande volume de transações diárias e diferenças nos preços entre as bolsas de bitcoins. Esse conjunto de características do mercado de bitcoins cria um ambiente que pode ser explorado para gerar benefícios ao investidor. Para alcançar o nosso objetivo de criar e usar robôs, vamos estudar e descrever o funcionamento do mercado de bitcoins destacando algumas diferenças entre *exchanges* de bitcoin, e usar a interface de programação de aplicativos, disponível nas *exchanges* para criar robôs, com métodos para compra, venda e coleta de dados de ordens em aberto e histórico de transações entre outros. Os robôs devem atuar realizando coleta e tratamento de dados para extrair informações, e também negociar de forma autônoma baseado nos dados obtidos das *exchanges*. Como resultado vamos apresentar algumas diferenças detectadas entre bolsas de bitcoin, apresentar robôs criados para duas *exchanges*, Mercado Bitcoin e Bitfinex, apresentar e descrever a implementação dos robôs, discutir como o uso dos robôs traz beneficio para os investidores e apresentar os resultados obtidos com o uso dos robôs.

Palavras-chave: Bitcoin. Robôs. Bolsas. Câmbio. *Exchanges*.

Abstract

We present a robot-building experiment, to work on some exchange of bitcoins. These exchanges provide an online environment for trading bitcoins, which operates 24 hours every day, the market is global and allows you to trade fiduciary currencies for digital currencies, some of them are: real, dollar, euro, bitcoin, ethereum and litecoin. This exchange market has high volatility, a large volume of daily transactions and differences in prices between the bitcoins exchanges. This set of characteristics of the bitcoins market creates an environment that can be exploited to generate benefits for the investor. To achieve our goal of creating and using robots, we study and describe the behavior of the bitcoins market by highlighting some differences between bitcoin exchanges, and using the application programming interface, available in the exchanges to create robots, with methods to buying, sale and collect data of open orders and transaction history, among others. The robots must act collecting and processing data to extract information, and also to transact autonomously based on the data obtained from the exchanges. As a result we present some differences detected between bitcoin exchanges, present robots created for two exchanges, Mercado Bitcoin and Bitfinex, to present and to describe the implementation of the robots, to discuss how the use of robots brings benefit to investors and to present the results obtained with the use of robots.

Keywords: Bitcoin. Robot. Exchange houses.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Acrônimos

Sumário

1

INTRODUÇÃO

Neste capítulo apresentamos uma visão geral sobre o bitcoin, como é gerado e como pode ser usado. Citamos os problemas identificados que criam a necessidade dessa pesquisa, em seguida, apresentamos nossa motivação e nossa proposta para tratar o problema. Nossos objetivos traçados para a pesquisa. E por último, apresentamos a metodologia que será usada na elaboração da nossa pesquisa, descrevendo as etapas e os procedimentos para atingir os objetivos traçados.

1.1 BITCOIN

Bitcoin (BTC) é uma moeda digital, também conhecida como criptomoeda, foi criado em 2008, por Nakamoto (2008). Existem mais de 15.912.500 bitcoins em circulação e eles podem ser transferidos de uma pessoa para outra, sem passar por bancos ou instituições financeiras, através de um protocolo seguro que impede transações inválidas. Novas moedas são criadas gradualmente para pessoas que usam o seu poder computacional para validar transações e registrar no Blockchain, as pessoas que realizam esse trabalho são conhecidos como mineiros. O Blockchain é semelhante a um livro-razão que registra todas as transferências de bitcoins entre usuários.

Bitcoin pode ser utilizado para realizar pagamentos sem cobrança de taxas, porque pode ser transferido de uma pessoa para outra sem intermediários, e os mineiros envolvidos na confirmação da transação serão recompensados com moedas que serão criadas. O bitcoin tem oito dígitos decimais, permitindo a transferência de valores muito baixos, enquanto outros métodos tradicionais de pagamentos são inviáveis devido às taxas envolvidas.

Existem duas formas para uma pessoa obter bitcoins, ou ela recebe bitcoins que foram enviados por outro indivíduo, ou realiza a atividade de mineração para ganhar os bitcoins que são criados. O bitcoin também é usado como um ativo e é negociado em diversos websites de câmbio online.

O bitcoin é uma moeda escassa, e a atividade de mineração vai criar novos bitcoins somente até atingir o limite máximo de 21 milhões de BTC. Bitcoin é uma moeda global cujo preço vária de acordo com a demanda de pessoas comprando e vendendo bitcoins, nas diversas

bolsas de bitcoins existentes. Devido à facilidade de transferir bitcoins de uma bolsa para outra, o preço em cada região mantém uma correlação direta com o valor da moeda local. Um gráfico mostrando o valor do bitcoin em dólar é apresentado na Figura 1.1, onde podemos observar que o bitcoin é uma moeda valorizada e em 25 de agosto de 2016, estava cotado a 575,63 dólares uma unidade de bitcoin.

Figura 1.1: Apresenta o valor de mercado dos bitcoins ao longo do tempo, disponível em Coinmarketcap (2016).

O gráfico na Figura 1.1 apresenta o preço do bitcoin representado pela linha verde, o valor de mercado de todos os bitcoins existentes representado pela linha azul, e contém um gráfico de barras na parte inferior onde podemos observar que existe uma tendência de crescimento no volume negociado, ao longo do tempo. Os dados mostram que o mercado de bitcoin em 25 de agosto de 2016, representa um mercado de mais de 9 bilhões de dólares(linha azul), além disso, é mostrado que num intervalo das últimas 24 horas, foram negociados mais de 58 milhões de dólares em bitcoins.

1.1.1 Bolsas de bitcoin

Grandes volumes de bitcoins são negociados diariamente em diversos websites que funcionam como bolsas de criptomoedas também chamadas de bolsas de bitcoin ou *exchanges* de bitcoin, estes websites criam um ambiente onde usuários podem negociar bitcoins por moedas, fiduciárias ou virtuais. Isso é parte de um mercado global de criptomoedas, onde usuários podem transferir e negociar fundos em tempo real. Diferentemente de bolsas que negociam ações, o mercado de criptomoedas funciona por 24 horas, todos os dias.

Existem diversas bolsas de bitcoin, com grandes diferenças em relação ao preço do bitcoin, o volume negociado, as taxas cobradas e outros fatores. Além das diferenças, existem outras características nas bolsas de bitcoins, que dificultam a atuação de investidores operando manualmente, uma delas é a existência de robôs atuando no mercado que se apresentam como

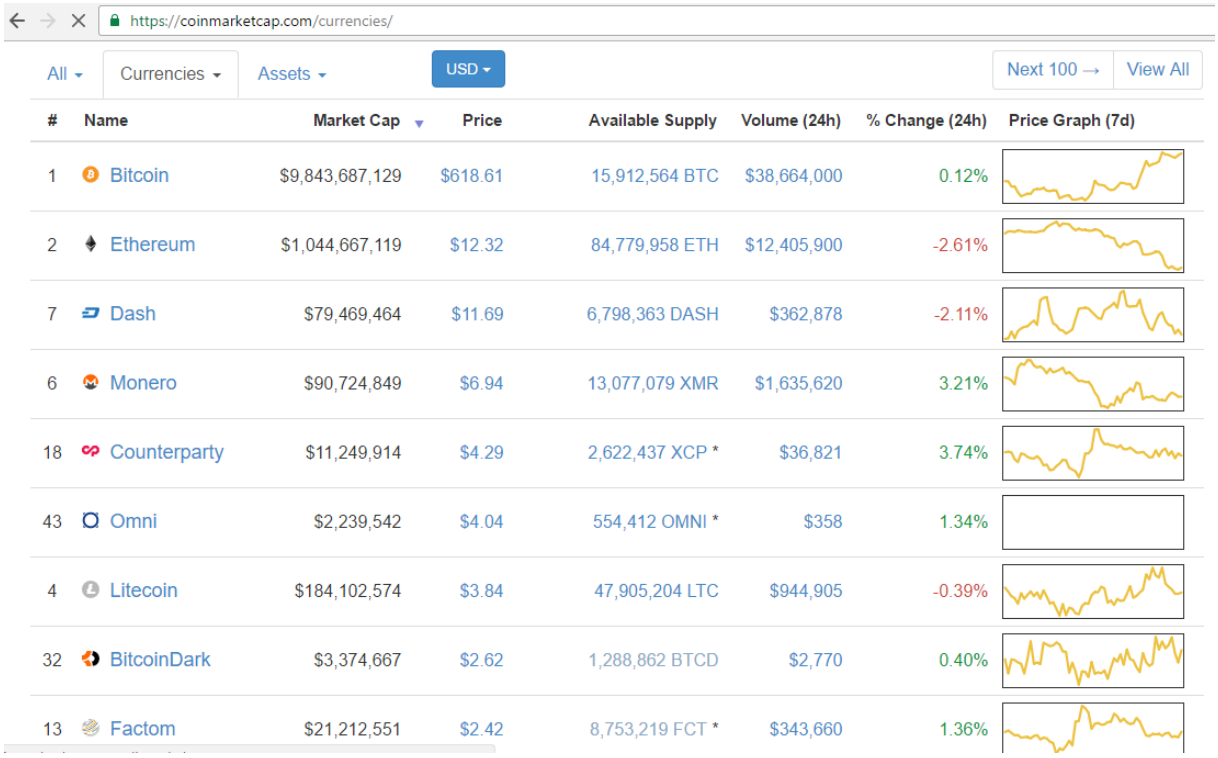
concorrentes ao usuário. O funcionamento 24 horas do mercado também dificulta a atividade do usuário, porque pode ocorrer variações de preço em horários que o usuário não está ativo no mercado e podem causar perda.

Diversas bolsas de bitcoin disponibilizam Interface de Programação de Aplicativos (API), que pode ser usada por pessoas com conhecimento em programação para criar robôs.

1.1.2 Outras criptomoedas

Além do bitcoin existem mais de mil outras criptomoedas segundo Coinmarketcap (2016), algumas delas são o litecoin e etherium, porém o bitcoin é a moeda mais valorizada.

O litecoin tem uma forte correlação com o bitcoin e o preço sobe e desce numa proporção semelhante ao bitcoin. A quantidade de litecoins em circulação é maior que a de bitcoins, porém o preço da unidade é bem menor como pode ser visto na Figura 1.2.



#	Name	Market Cap	Price	Available Supply	Volume (24h)	% Change (24h)	Price Graph (7d)
1	Bitcoin	$9,843,687,129	$618.61	15,912,564 BTC	$38,664,000	0.12%	
2	Ethereum	$1,044,667,119	$12.32	84,779,958 ETH	$12,405,900	-2.61%	
7	Dash	$79,469,464	$11.69	6,798,363 DASH	$362,878	-2.11%	
6	Monero	$90,724,849	$6.94	13,077,079 XMR	$1,635,620	3.21%	
18	Counterparty	$11,249,914	$4.29	2,622,437 XCP *	$36,821	3.74%	
43	Omni	$2,239,542	$4.04	554,412 OMNI *	$358	1.34%	
4	Litecoin	$184,102,574	$3.84	47,905,204 LTC	$944,905	-0.39%	
32	BitcoinDark	$3,374,667	$2.62	1,288,862 BTCD	$2,770	0.40%	
13	Factom	$21,212,551	$2.42	8,753,219 FCT *	$343,660	1.36%	

Figura 1.2: Lista de criptomoedas ordenada pelo preço da unidade, obtida de Coinmarketcap (2016).

Da Figura 1.2 podemos observar a uma grande diferença no preço entre o bitcoin e o etherium que é a segunda moeda mais valorizada, e temos que o bitcoin lidera o mercado de critomoedas. A rede de bitcoin também tem o maior valor de mercado *Market Cap*.

1.2 PROBLEMA

Como obter benefícios para o investidor que negocia em bolsas de bitcoin, que funcionam 24 horas todos os dias, vulnerável a variações de preço a qualquer momento, com diversas bolsas que criam mercados diferenciados para negociação de bitcoins, que apresentam: preços, volume negociado, taxas e comportamento do usuário distintos.

- **Hipótese de solução:** O uso de robôs, supervisionado pelo investidor, com capacidade de obter informações do mercado de bitcoin, e negociar de forma autônoma deve beneficiar o usuário com: aumento no lucro, redução da carga de trabalho do investidor, negociar durante período de tempo maiores que o usuário humano e de forma mais rápida.

1.3 MOTIVAÇÃO

Nossa pesquisa foi motivada pela ideia de criar robôs com capacidade de atuar em *exchanges* de bitcoin de forma autônoma. Com o uso de robôs nas bolsas de bitcoin pretendemos auxiliar o investidor que está negociando de diversas formas, isso inclui a coleta de dados para análises; criação de alertas para monitorar os preços do bitcoin em diversas bolsas e compra e venda de bitcoins a preços melhores. Portanto, com o uso de um algoritmo eficaz é viável manter um robô negociando por tempo indeterminado.

Um robô eficaz pode aumentar os lucros do investidor, reduzir a quantidade de tempo que o investidor fica diante do computador realizando as operações de compra e venda, realizar operações com maior velocidade, e também pode ser capaz de operar por períodos de tempo longos, que seriam inviáveis para um humano.

1.4 PROPOSTA

A partir do estudo das APIs e suas respectivas documentações, serão implementados robôs para atuar separadamente em algumas bolsas de bitcoin brasileiras e outros em bolsas do exterior. E também serão analisadas e discutidas a viabilidade de criar robôs que atuam de forma integrada. Os robôs devem coletar dados do histórico de transações já executadas, com volume negociado e variação de preços, dados do livro de ordens, que representam as ordens em aberto no mercado com preços e volumes. Além disso, devem manipular dados do usuário como saldos e históricos de transações, criar novas ordens, cancelar ordens em aberto, entre outras. Por fim, descrever e discutir alguns cenários onde o uso dos robôs apresenta resultados satisfatórios.

1.5 OBJETIVOS

A seguir apresentamos os objetivos traçados para cumprir a proposta da dissertação.

1.5.1 Objetivo Geral

- Analisar o mercado de bitcoins e apresentar um experimento de criação de robôs para atuar negociando de forma autônoma.

1.5.2 Objetivos Específicos

- Pesquisar bolsas de bitcoin no Brasil, que negociem bitcoin por real.
- Pesquisar bolsas de bitcoin no exterior, que negociem bitcoin por dólar.
- Buscar documentação das APIs oferecidas pelas bolsas de bitcoins estudadas.
- Realizar experimentos com contas em bolsas de bitcoins que possam ser usadas pelos robôs.
- Definir linguagem de programação que será adequada a criação de dos robôs.
- Buscar exemplos de códigos que descrevem o uso dos métodos das APIs.
- Testar a chamada de métodos através das APIs.
- Apresentar robô que coleta dados de transações passadas.
- Realizar experimentos com robô que monitora preços em tempo real de algumas bolsas de bitcoin.
- Testar limites das APIs relacionados à velocidade de acesso e quantidades de chamadas possíveis.
- Testar execução de robôs por períodos significativos.
- Pesquisar padrões nas variações de preços para criação de um robô.
- Descrever a criação de robôs que realizam operações de compra e venda.
- Analisar dados de históricos de execução dos robôs.
- Apresentar resultados obtidos.
- Discutir futuras melhorias.

1.6 METODOLOGIA

Esta dissertação foi construída como pesquisa exploratória, e nossa coleta de dados se deu de forma experimental, com a intenção de estudar e discutir o uso de robôs no mercado de câmbio de bitcoins. Os robôs foram testados em ambiente real e os dados coletados serão discutidos. As empresas estudadas criam um ambiente para o investidor negociar bitcoin e outras moedas virtuais, e foram escolhidas por sua representatividade, dado que têm apresentado um grande volume de negociações diárias, e por oferecerem APIs que serão usadas para criar os robôs.

1.6.1 Ambiente de pesquisa

Para iniciar a criação deste trabalho foi realizada uma pesquisa sobre o mercado de câmbio de bitcoin, que iniciou com buscas na internet, que resultaram na descoberta de nossa primeira bolsa de bitcoins estudada: MercadoBitcoin (2016). Com um estudo mais detalhado sobre o funcionamento dessa bolsa, foi constatado que ela oferecia aos usuários uma API com capacidade de coletar dados e realizar operações. A segunda bolsa de bitcoins brasileira estudada foi Foxbit (2016), que chamou a atenção por movimentar um volume de bitcoins negociados um pouco superior à primeira. Também foi constatado a existência de API. Os estudos comparativos em relação as diversas bolsas de bitcoins, são apresentados no Capítulo 4, mostrando algumas das diferenças entre elas. Atualmente, existem serviços online, como o encontrado em BitValor (2016), que integra dados de diversas bolsas de bitcoins brasileiras, e apresenta um índice de preços do bitcoin, além de relatórios e dados das últimas negociações em diversas bolsas brasileiras. Os trabalhos relacionados a pesquisa de APIs e criação de robôs foram iniciados desde a descoberta da primeira bolsa de bitcoin Mercado Bitcoin.

As bolsas de bitcoin no exterior foram encontradas a partir de pesquisas na internet com o objetivo inicial de observar o mercado global. Algumas bolsas como Bitstamp (2016), Bitfinex (2016) e OKCoin (2016) foram encontradas, e constatamos que oferecem um ambiente desenvolvido para negociação de bitcoins por dólar, todas fornecem suas próprias APIs, que foram respectivamente investigadas para o desenvolvimento de robôs que possam atuar nesses mercados.

1.6.2 Recursos necessários

Para iniciar o desenvolvimento dos robôs, foram criados cadastros nas respectivas bolsas de bitcoins que seriam estudadas, iniciando pela *exchange* Mercado Bitcoin que foi a porta de entrada para os primeiros testes com robôs. Para utilizar todos os serviços do website foi necessário realizar uma autenticação de identidade do usuário, esse processo aumenta limite de saque de R$500 para até R$20.000,00 e depósitos de R$100 para ilimitado. Na Foxbit, também foi criado um cadastro e realizada a respectiva autenticação de identidade. Em bolsas de bitcoin

no exterior não foi necessário realizar autenticação de identidade, porque foram realizados saques e depósitos apenas em bitcoins e, portanto, o cadastro simples foi suficiente.

A linguagem de programação utilizada para implementar os robôs apresentados nesse trabalho foi Python v2.7 e v2.7.12. Essa escolha foi para otimizar a curva de aprendizado dado que existiam alguns exemplos de códigos em Python na primeira bolsa estudada Mercado Bitcoin. A versão inicial foi atualizada de 2.7 para 2.7.12 para utilizar algumas bibliotecas que apresentaram problemas na versão anterior.

Os principais métodos utilizados através das APIs, que são comuns, são para consultar o livro de ordens, consultar o saldo do usuário, criar ordem, obter a lista de ordens em aberto e para cancelar uma ordem aberta. Outros métodos mais sofisticados são suportados em algumas das bolsas. Os métodos de consulta a dados da própria bolsa como preços, volume diário e livro de ordens, podem ser chamados sem informação do usuário, portanto não é necessário cadastro para utilizar essas funções. Métodos que envolvem dados do usuário são autenticados através de chaves, que são geradas pelo próprio usuário concedendo permissão para manipulação os tipos de dados desejados, que podem ser desde simples leitura de dados até a manipulação de fundos.

Os bitcoins utilizados foram obtidos a partir de depósitos e transferências em reais para o Mercado Bitcoin seguidas da compra de bitcoins. Para desenvolver o estudo em outras bolsas de bitcoins foram realizadas transferências de bitcoins. A transferência de bitcoins de uma bolsa para outra pode ser gratuita, cobrar uma taxa opcional ou em alguns casos uma taxa obrigatória dependendo da política da bolsa de onde os bitcoins estão sendo enviados.

1.6.3 Procedimento para criação e uso de robôs

Para criação dos robôs usamos APIs fornecidas pelas bolsas de bitcoins. As APIs de cada website podem ser encontradas através de links em suas páginas iniciais. A *exchange* Mercado Bitcoin tem uma API bem documentada em português e com exemplos de códigos em Python. Quanto à Foxbit, não foi encontrada nenhuma documentação em português, mas a documentação em inglês foi suficiente para a sua utilização. As APIs de bolsas estrangeiras são bem documentadas e com maior quantidade de métodos, portanto fornecem mais recursos para criação de robôs.

Os robôs foram criados e executam através da Shell do Python. Na criação de todo o código, os métodos foram implementados e testados individualmente, observando os resultados de suas execuções na bolsa em tempo real. Em seguida, foram criados robôs que realizam apenas coleta de dados, a fim de observar o comportamento de robôs executando em tempo real. Para concluir foram implementados métodos para criação de ordens de compra e venda, baseados no histórico de transações e no livro de ordens, com isso os robôs passaram a funcionar com capacidade de negociação, de forma autônoma.

Os algoritmos utilizados para decidir o momento da compra e da venda foram elaborados e, muitas vezes aperfeiçoados, durante o desenvolvimento da pesquisa, através de tentativa e

erro; nenhum algoritmo com intuito de previsão dos preços foi implementado. Contudo, diversas estratégias apresentaram lucro durante determinados períodos de tempo, que são discutidas com detalhes nos próximos capítulos.

1.7 ESTRUTURA DA DISSERTAÇÃO

Neste capítulo apresentamos uma introdução ao tema de nossa pesquisa, o problema identificado, nossa motivação para resolver o problema, nossa proposta de solução e os objetivos trassados para a pesquisa e apresentamos a metodologia que será usada.

No Capítulo 2, apresentamos o referencial teórico usado como fonte de pesquisa, citando algumas pesquisas relevantes na área.

No Capítulo 3, descrevemos o ambiente de negociação de bitcoins, que envolve algumas *exchanges* de bitcoins. Descrevemos também como o bitcoin é negociado e destacamos diferenças entre as *exchanges* citadas. Apresentando e discutindo pontos chaves que diferenciam o ambiente criado por *exchanges* distintas.

No Capítulo 4, apresenta os resultados obtidos em nossa pesquisa, onde apresentamos exemplos de robôs criados para bolsas de bitcoin por real e bitcoin por dólar, vamos discutir vantagens e riscos no uso dos robôs. Também apresentamos alguns resultados obtidos com a execução dos robôs nas respectivas *exchanges*.

No Capítulo 5, concluímos a pesquisa, discutindo trabalhos futuros.

2

REFERENCIAL TEÓRICO

Este capítulo apresenta citações de pesquisas e artigos de autores que citam o bitcoin em contextos relacionados ao seu uso como moeda de troca, e está dividido em secções que tratam de referências a indicadores de preço, negociação algorítmica e previsão de preços.

2.1 INDICADORES DE PREÇO

Conforme Bell (2015), o mercado de bitcoins, semelhante ao mercado de câmbio de moedas estrangeiras, é um mercado aberto que permite consumidores e investidores negociarem bitcoins. O preço ao qual o bitcoin é negociado neste mercado está relacionado ao seu valor percebido pelo investidor. Os indicadores que afetam o valor das moedas fiduciárias emitidas pelo governo, como a quantidade de produtos e mercadorias importados e exportados usando uma determinada moeda, são diferentes para o bitcoin devido à sua utilização global e descentralizada. Bell (2015) também afirma que os condutores de mineração do bitcoin foram comparados a uma *commoditie* e indicam que sua oferta constante e estável faz com que a demanda seja aliviada, mas durante os intervalos de datas dos dados utilizados, o bitcoin aprecia que a oferta é incapaz de acompanhar a demanda.

Segundo Ciaian, Rajcaniova e Kancs (2014), os resultados da modelagem de funções de média móvel autorregressiva, mostram que os valores de bitcoin reagem ao índice de volatilidade Chicago Board Options Exchange (CBOE), sugerindo que uma força preliminar que move atualmente os preços do bitcoin é a especulação pelos investidores que olham fora dos mercados tradicionais. MacDonell (2014) postula que os investidores procuram a maior volatilidade da bitcoin para maximizar seu retorno potencial sobre o investimento, em vez de manter os ativos em um mercado de baixa volatilidade, onde há menos capacidade para maiores retornos. Segundo Kristoufek (2014), a especulação dos investidores é um importante impulsionador do valor bitcoin. Segundo Ciaian at al. (2014), as descobertas de seu estudo de cointegração não contradizem a hipótese de Kristoufek de que as especulações realmente afetam o preço de bitcoin.

Segundo Badev e Chen (2014), a medição nas flutuações da taxa de câmbio como uma porcentagem do preço médio diário, identificou as variações no preço diário do bitcoin abaixo

de 12-15%, com algumas poucas exceções. Segundos os autores, que também consideraram um grande crescimento no valor do bitcoin ao longo do período, afirmam que do ponto de vista do proprietário do bitcoin, isso indica que o risco de manter bitcoin por períodos relativamente curtos é baixo, tornando-o adequado como moeda para a transferência de fundos. Comerciantes que aceitam bitcoins como forma de pagamento podem converter bitcoins para moeda local imediatamente, reduzindo os riscos da volatilidade, e o uso de robôs é adequado para tal cenário.

Kristoufek (2014) faz a hipótese de que as teorias econômicas padrão da oferta e da demanda não podem ser usadas para explicar a formação de preços da bitcoin, devido à natureza descentralizada da bitcoin. Segundo Ciaian at al. (2014), aumentos no estoque bitcoin podem levar à diminuição dos preços e os aumentos no tamanho e velocidade da economia bitcoin podem levar ao aumento dos preços.

Bell (2015) afirma que a oferta de bitcoin se comporta muito mais como uma mercadoria do que como uma moeda. Bitcoin é criptograficamente extraído, limitando sua oferta e tornando-a mais previsível. A oferta é limitada pela capacidade computacional que é exposta à mineração de bitcoin, análoga à atividade das minas de *commodities*. E conclui que a oferta depende do incentivo para investir em equipamentos de mineração. À medida que o preço aumenta, mais é investido na mineração, o que aumenta a oferta e, por sua vez, reduz o preço. Este circuito de feedback acrescenta à estabilidade do preço do bitcoin e é uma dinâmica importante a considerar ao modelar os movimentos de preços.

Bell (2015) apresenta um comparativo entre moedas de estado e bitcoin, e comenta que embora o fornecimento de moedas emitidas pelo Estado seja controlado por bancos e regulado por políticas de uma autoridade central, o fornecimento da bitcoin é intrinsecamente controlado pela taxa de extração de bitcoins. Enquanto a demanda por moedas emitidas pelo Estado é, em grande parte, impulsionada pela quantidade de exportações do estado-nação governante para compradores estrangeiros, a demanda por bitcoin é impulsionada por incentivos globais para adotar o bitcoin como facilidade de uso, utilidade do bitcoin como moeda legítima para uso em uma ampla gama de mercados, benefícios de segurança, custos de transação reduzidos, estabilidade, incentivos filosóficos e sentimentos especulativos como uma oportunidade de investimento de curto ou longo prazo. Bell (2015) conclui que há espaço para muito mais pesquisas da utilização de sentimento de mídia social como um conjunto de dados com capacidade de previsão.

Kaminski (2016) usou tweets diários para produzir índices de tweets positivos, negativos e emocionais, bem como aqueles que expressam incerteza (esperança, medo e preocupação). O autor cita que 1-Emoções emoções negativas e emoções que aparentam incerteza se relacionam com o volume negociado. 2-A soma das emoções e sinais de incerteza alimentam a volatilidade diária refletindo na maior diferença entre o maior preço e o menor preço diário. 3-Emoções negativas e sinais de incerteza são mais prováveis em dias com preço de fechamento baixo, um misto de emoções negativas e incertezas pode ser visto como sinal de insatisfação e pessimismo pelos negociantes.

Wijk (2013) conclui que vários indicadores financeiros, incluindo o valor do Dow Jones, a taxa de câmbio euro-dólar e o preço do petróleo WTI, têm um efeito significativo no valor do Bitcoin a longo prazo. O valor do Índice Dow Jones também afeta significativamente o valor do Bitcoin no curto prazo.

Bell (2015) discursa como a oferta tem um impacto indireto sobre o preço do bitcoin e como a oferta é controlada e mantida estável. A demanda por bitcoin é o fator direcionador mais direto do preço do bitcoin. Não importa o quanto ou o quão poucos bitcoins estão disponíveis em circulação, é a demanda que define o preço que os investidores estão dispostos a pagar por eles.

Bell (2015) expõe que a chave para as descobertas de Ciaian et al. tem sido que a demanda do interesse dos investidores pelos bitcoins é co-integrada com seus fundamentos de oferta e demanda. Suas descobertas sugerem que as visualizações da Wikipédia e a atividade de mídia social são estatisticamente significativas na descrição do preço do bitcoin. Isto é suportado pela análise de ondulações de Kristoufek, em que as pesquisas do Google e da Wikipedia estão correlacionadas com os movimentos de preços do bitcoin.

Segundo Garcia et al. (2014), ao usar a autorregressão vetorial, identificou dois laços de *feedback* positivos que levam a bolhas de preços na ausência de estímulos externos: um conduzido pelo boca a boca e o outro por novos adeptos do bitcoin. O autor cita que os picos de pesquisa de informações, presumivelmente vinculados a eventos externos, precedem a queda drástica de preços

2.2 NEGOCIAÇÃO ALGORÍTMICA

Bell (2015) apresenta uma pesquisa sobre a aplicação de Regressão Bayesiana, para criação de estratégia de compra e venda de bitcoins, onde ficou constatado que a estratégia executa mais eficazmente quando há alta volatilidade e ainda é rentável quando o preço real do bitcoin está diminuindo. As pesquisas apontaram possíveis ganhos de 89% em aproximadamente dois meses.

Rowlands (2014) cita que o uso de algoritmos pode auxiliar humanos a negociar de forma mais segura. Ao definir preços de comprar, vender ou preços sair do mercado, os comerciantes são capazes de entrar, sair e evitar perdas excessivas em suas "posições"com segurança. Isso leva os comerciantes humanos a trabalhar significativamente de forma mais eficiente e segura dentro do mercado.

Redman (2017) Afirma que o comércio de bot (robô) pode não ser para todos, pois o software pode ser difícil para os comerciantes inexperientes entenderem. Além disso, os comerciantes devem confiar na eficiência e confiabilidade das empresas ou no software livre que oferece operações de *cryptocurrency* algorítmicas. Existem muitas empresas diferentes que oferecem serviços de bot, e algumas delas podem não ser legítimas. Juntamente com isso, os programas gratuitos de comércio de bot podem ser encontrados em sites como o Sourceforge, mas as pessoas devem pesquisar diligentemente antes de confiar em qualquer software livre. No

entanto, os bots de negociação respeitáveis e funcionais podem aumentar os lucros comerciais, se utilizados corretamente.

2.3 PREVISÃO DE PREÇOS

Bell (2015) afirma que em suas pesquisas foi demonstrado que há um grande espaço para usar o preço bitcoin e outras séries de tempo como dados que contém informações sobre bitcoin que é poderoso para prever seu preço futuro. Uma gama de técnicas matemáticas e computacionais existentes foram utilizadas com uma gama de conjuntos de dados e mostram que o bom desempenho de um agente de negociação bitcoin automático pode ser alcançado.

Madan, Saluja e Zhao (2015) apresentou um trabalho no qual buscavam compreender e identificar tendências diárias no mercado de bitcoins, a partir de um conjunto de dados com mais de 25 características relacionadas à rede bitcoin de preços e pagamentos ao longo de cinco anos, registrados diariamente. Com esta informação foi possível prever o sinal de mudança diária com uma precisão de 98,7%.

Segundo McNally (2016), métodos de previsão de series temporais transicionais como Holt-Winters, não são efetivos para prever os movimentos de preços no mercado de bitcoins. Ele cita que alguns dos motivos que dificultam a previsão é a falta de sazonalidade e a alta volatilidade nos preços do bitcoin. E sugere que métodos que usam aprendizado de máquina são mais adequados para prever os movimentos de preços do bitcoin.

Segundo McNally (2016), a precisão de seus experimentos foi de 50,25% para Recurrent Neural Network (RNN) e 52,78% para Long Short Term Memory (LSTM), e afirma que a precisão apresentada, é uma melhoria marginal em relação às probabilidades que tem em uma tarefa de classificação binária, isto é, 50%.

Madan at al. (2015) sugerem que faz sentido que uma mesma metodologia de previsão aplicada em mercados de ações, seja replicada no mundo da bitcoin, à medida que a rede ganha maior liquidez e mais pessoas desenvolvem um interesse em investimentos rentáveis no sistema. E aponta que para isso, é necessário alavancar a tecnologia de aprendizado de máquina para prever o preço do bitcoin.

Żbikowski (2015) apresentou em sua pesquisa, estratégias baseadas em aprendizado de máquina para simular um agente autônomo negociando na *exchange* Bitstamp, onde usou a API fornecida pela própria *exchange* para coletar os dados usados em sua pesquisa. Żbikowski alcançou resultados de lucro de 33,52% usando a estratégia VW-SVM (*Volume-Weighted support vector machines*), enquanto que no mesmo período a estratégia de comprar e segurar os bitcoins representava um lucro de 4,86%. Żbikowski (2015) conclui que os resultados são promissores, e presume que os ganhos possíveis em um ano, excederiam níveis razoáveis de lucro em outros mercados financeiros.

3

MERCADOS DE BITCOINS

Este capítulo tem como objetivos: descrever o processo de compra e venda de bitcoins dentro de *exchanges*, apresentar uma descrição dos mercados de bitcoin por real, bitcoin por dólar e bitcoin por yuan, descrever o mercado de financiamento em *exchanges* de bitcoin e apresentar diferenças entre os mercados criados em cada região devido as diferenças das *exchanges*.

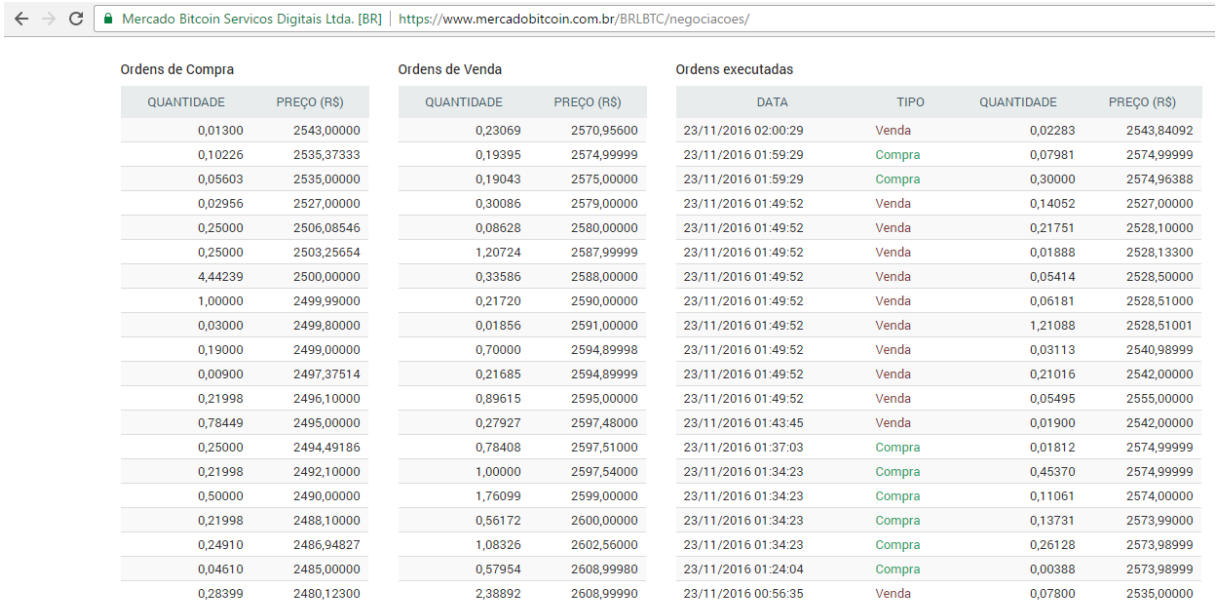
3.1 COMPRA E VENDA DE BITCOINS

Para auxiliar o mercado de compra e venda de moedas digitais, existem diversos websites que permitem a compra de bitcoins, e alguns deles oferecem uma plataforma profissional de negociação, que permitem comprar e vender bitcoins instantaneamente.

As bolsas de bitcoins fornecem uma plataforma que possibilita ao usuário atuar manualmente criando ordens de compra e venda, e também usar APIs, específicas de cada website, para criar robôs que operam automaticamente, realizando transações e/ou coleta de dados. Este mercado funciona 24 horas todos os dias e tem características diferentes para cada website.

Algumas bolsas de câmbio de bitcoin online são: Mercado Bitcoin, Bitstamp, Bitfinex, OKCoin entre outras. Elas funcionam de forma semelhante: têm um livro de ordens, com duas listas, uma com ordens de compra, ordenadas pelo maior preço, e outra com ordens de venda, ordenadas pelo menor preço; as duas listas contêm ordens com o preço e a quantidade de bitcoins, ordens com preço idêntico são agrupadas, com prioridade para executar a ordem mais antiga. Todas as ordens no livro, foram criadas por usuários que têm saldo suficiente reservado para executar as respetivas ordens. As ordens de compra no livro têm o preço menor que as ordens de venda. Se uma ordem de compra for criada com valor igual ou superior a ordem de venda mais barata, então ela será imediatamente executada, e os bitcoins transferidos entre os usuários envolvidos. Ordens podem ser executadas parcialmente, e o livro de ordens irá apresentar o montante ainda não executado. A Figura 3.1 apresenta parte do livro de ordens no website www.mercadobitcoin.com.br.

Na Figura 3.1, podemos observar a coluna **Ordens executadas** com uma sequencia de ordens de venda executadas no mesmo instante, data 23/11/2016 às 01:49:52 horas, isso reflete possivelmente uma ordem de venda com volume alto o bastante para consumir diversas ordens



Ordens de Compra

QUANTIDADE	PREÇO (R$)
0,01300	2543,00000
0,10226	2535,37333
0,05603	2535,00000
0,02956	2527,00000
0,25000	2506,08546
0,25000	2503,25654
4,44239	2500,00000
1,00000	2499,99000
0,03000	2499,80000
0,19000	2499,00000
0,00900	2497,37514
0,21998	2496,10000
0,78449	2495,00000
0,25000	2494,49186
0,21998	2492,10000
0,50000	2490,00000
0,21998	2488,10000
0,24910	2486,94827
0,04610	2485,00000
0,28399	2480,12300

Ordens de Venda

QUANTIDADE	PREÇO (R$)
0,23069	2570,95600
0,19395	2574,99999
0,19043	2575,00000
0,30086	2579,00000
0,08628	2580,00000
1,20724	2587,99999
0,33586	2588,00000
0,21720	2590,00000
0,01856	2591,00000
0,70000	2594,89998
0,21685	2594,89999
0,89615	2595,00000
0,27927	2597,48000
0,78408	2597,51000
1,00000	2597,54000
1,76099	2599,00000
0,56172	2600,00000
1,08326	2602,56000
0,57954	2608,99980
2,38892	2608,99990

Ordens executadas

DATA	TIPO	QUANTIDADE	PREÇO (R$)
23/11/2016 02:00:29	Venda	0,02283	2543,84092
23/11/2016 01:59:29	Compra	0,07981	2574,99999
23/11/2016 01:59:29	Compra	0,30000	2574,96388
23/11/2016 01:49:52	Venda	0,14052	2527,00000
23/11/2016 01:49:52	Venda	0,21751	2528,10000
23/11/2016 01:49:52	Venda	0,01888	2528,13300
23/11/2016 01:49:52	Venda	0,05414	2528,50000
23/11/2016 01:49:52	Venda	0,06181	2528,51000
23/11/2016 01:49:52	Venda	1,21088	2528,51001
23/11/2016 01:49:52	Venda	0,03113	2540,98999
23/11/2016 01:49:52	Venda	0,21016	2542,00000
23/11/2016 01:49:52	Venda	0,05495	2555,00000
23/11/2016 01:43:45	Venda	0,01900	2542,00000
23/11/2016 01:37:03	Compra	0,01812	2574,99999
23/11/2016 01:34:23	Compra	0,45370	2574,99999
23/11/2016 01:34:23	Compra	0,11061	2574,00000
23/11/2016 01:34:23	Compra	0,13731	2573,99000
23/11/2016 01:34:23	Compra	0,26128	2573,98999
23/11/2016 01:24:04	Compra	0,00388	2573,98999
23/11/2016 00:56:35	Venda	0,07800	2535,00000

Figura 3.1: Livro de ordens em Mercado Bitcoin, com histórico das últimas ordens executadas.

de compra do livro, e por isso, a venda foi realizada em diversas partes a preços distintos.

O procedimento para negociar em bolsas de bitcoin online inicia com o usuário criando um cadastro no site onde vai negociar, em seguida, deve transferir fundos, que podem ser bitcoins, dólar, real, euro etc. de acordo com as moedas aceitas pela *exchange*. Para ter acesso a todas as ferramentas será necessário algum tipo de verificação de identidade. Cada website tem tipos específicos de moedas com as quais é possível negociar. A partir do momento que o usuário tem saldo, ele pode realizar compra ou venda de bitcoins, instantâneas, no valor do livro de ordens ou criar ordens no valor desejado, que serão colocadas em espera no livro de ordens.

Grandes volumes de bitcoins são negociados diariamente, em websites de câmbio que facilitam a compra e venda de moedas digitais, isso pode ser visto na batela da Figura 3.2, que mostra uma lista de *exchanges* que negociam bitcoin, algumas delas se destacam por negociar diariamente volumes superiores a 1 milhão de bitcoins.

A Figura 3.2 apresenta uma tabela comparativa de *exchanges* que negociam bitcoin por dólar ou bitcoin por yuan. A tabela apresenta colunas que indicam: nome da *exchange*, ultimo preço estimado ao qual o bitcoin foi negociado, maior preço de compra no livro, menor preço de venda no livro e volume negociado em bitcoins nas ultimas 24 horas. O principal destaque da Figura 3.2 é a diferença de volume negociados entre *exchanges* que negociam por dólar para *exchanges* que negociam com yuan.

Usando os websites de câmbio é possível converter real para dólar usando o bitcoin como intermediário, com os seguintes passos: primeiro a pessoa transfere fundos para um website que permite negociar real por bitcoin, em seguida realiza a compra de bitcoins, depois transfere os bitcoins para outro website que negocia com dólar e bitcoin, então realiza a venda dos bitcoins, e pode sacar seu saldo em dólar para sua conta bancária. As operações de depósito ou saque

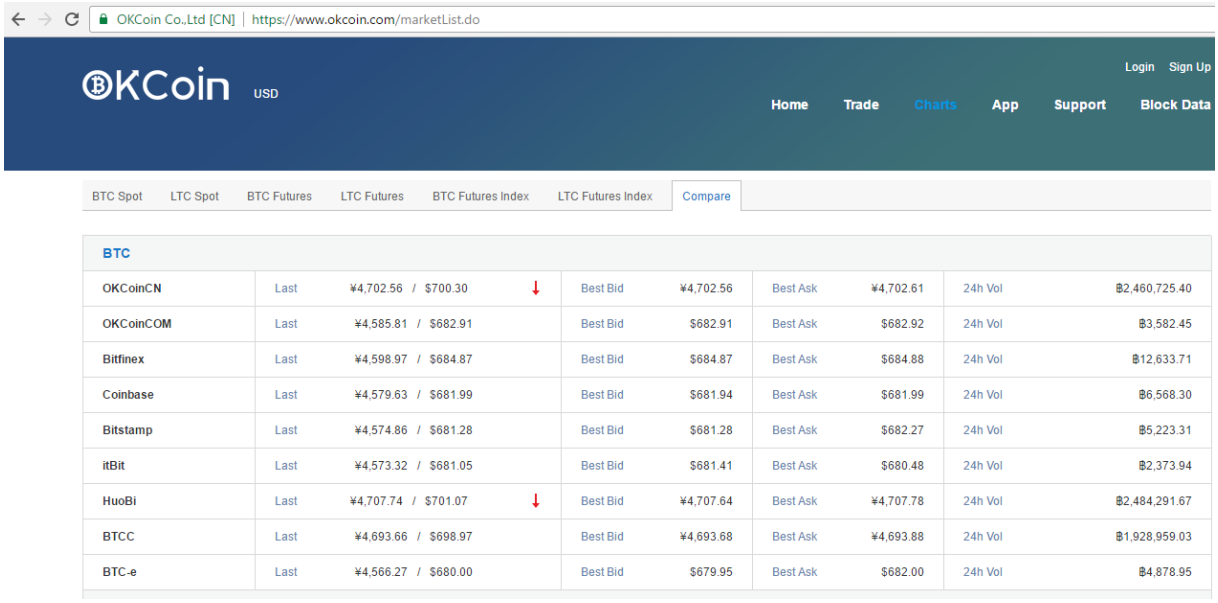


Figura 3.2: Tabela de volume de bitcoins movimentados em 27 de outubro de 2016.

de moeda fiduciária normalmente envolvem pagamento de taxas, enquanto a transferência de bitcoins entre websites, em alguns casos, não tem custo. Porém com a dificuldade crescente de mineração, que reduz a quantidade de bitcoins paga ao minerador, é cada vez mais comum a cobrança de taxas para recompensar os mineiros, alguns casos as taxas são variáveis e podem ter relação com a velocidade de confirmação, pois os mineiros dão prioridades a transferências com maiores taxas.

3.2 MERCADOS DE CÂMBIO

O mercado de câmbio de bitcoin é composto de diversas empresas, que oferecem um ambiente onde os usuários podem negociar com moedas digitais por moedas fiduciárias. Bitcoins podem ser transferidos entre bolsas de bitcoins, até o atual momento não é permitida a transferência de moedas fiduciárias entre bolsas de bitcoins, mas o usuário pode sacar seus fundos para uma conta bancária. Em seguida, fazer um depósito ou transferência para outra bolsa de bitcoins. Contudo esse processo é lento comparado a transferência de bitcoins, porque envolve instituições financeiras como intermediárias. A transferência de bitcoins entre websites pode levar alguns minutos, porém a transferência que acontece no momento de compra e venda dentro em uma *exchange* é um processo instantâneo, permitindo que o usuário compre e venda tão rápido quanto ele queira, com isso é possível aproveitar as oscilações de preço.

Diferente da bolsa de valores que tem horário de funcionamento relacionado com o horário comercial, os websites que negociam bitcoins funcionam 24 horas todos os dias. O funcionamento constante cria um cenário totalmente diferente, onde podem acontecer grandes variações de preço a qualquer momento do dia ou da noite.

O mercado de bitcoins tem características diferentes de acordo com as regiões onde o

bitcoin é mais usado, e as diferenças particulares de cada website são um diferencial que podem atrair o usuário. Algumas diferenças entre bolsas de bitcoins são: o volume de bitcoins negociado diariamente, as taxas cobradas por transações de compra/venda e a volatilidade. Alguns websites brasileiros como www.mercadobitcoin.com e www.foxbit.com.br movimentam diariamente algumas centenas de bitcoins em suas negociações, outros websites estrangeiros como www.bitstamp.net e www.bitfinex.com, movimenta diariamente milhares de bitcoins. Websites como www.OKCoin.cn, que negociam bitcoin por yuan, com sede na China, movimentavam milhões de bitcoins diariamente em suas transações de compra e venda devido a sua política de taxa zero, que foi alterada no início de 2017 e desde então o volume de bitcoins negociado caiu para valores equivalentes ao mercado de bitcoin por dólar.

Um brasileiro que pretende negociar no mercado de bitcoins da China, inicialmente pode comprar bitcoins no Brasil com a moeda local, em seguida transferir para um website chinês e permanecer negociando o tempo desejado, no momento que desejar pode transferir os bitcoins novamente para um website brasileiro, depois vender pela moeda local do Brasil e então sacar o seu saldo para uma conta bancária normalmente. A facilidade de transferência de bitcoins permite ao usuário movimentar seus bitcoins para os locais mais adequados aos seus objetivos.

3.2.1 Câmbio de bitcoin por real

No Brasil, existem alguns websites para câmbio de moedas digitais que oferecem a possibilidade de negociar bitcoins por reais. Esses websites que chamamos de bolsas de bitcoin ou *exchanges* são similares a bolsa de valores, e criam um ambiente onde os usuários podem negociar bitcoins por real, criando ordens de compra e venda ou executando ordens criadas de outros usuários.

O cenário de mercado criado no Brasil é diferente das outras bolsas de bitcoins do exterior em vários aspectos, entre eles, o volume de bitcoins negociados diariamente, as taxas envolvidas nas transações, a variação de preço e a quantidade de transações.

Duas das maiores bolsas de bitcoins no Brasil são Mercado Bitcoin e Foxbit, o volume de transações gira em torno de centenas de bitcoins negociados por dia, o mercado funciona 24 horas todos os dias, com uma tendência a reduzir o volume de transações nos finais de semanas. As taxas cobradas por transações de compra, venda, depósito e saque nestas duas empresas serão descritas a seguir.

Como exemplo para descrever o mercado de bitcoins no Brasil vamos usar as *exchanges* Mercado Bitcoin e Foxbit. A seguir, vamos apresentar algumas diferenças relacionadas a: taxas cobradas por transações de compra, venda, depósito e saque.

Mercado Bitcoin tem um sistema de cadastro que diferencia a conta do usuário entre normal, vip e gold. O processo de update da conta do usuário depende da autenticação de identidade e não tem custo, os benefícios são: maiores limites nas transações de depósito e saques. O usuário pode manter saldo em reais ou moedas digitais em sua conta, e não serão

cobradas nenhuma taxa de mensalidade ou custódia, Figura 3.3. As comissões cobradas pela empresa, limites e prazos relativos a depósitos e saques em reais são apresentadas na Figura 3.4, enquanto as operações de depósito e saques de reais têm taxa de 2,90 + 1,99%, e envolvem bancos, as operações de saque e depósito de bitcoins são gratuitas e não geram custo para a empresa, são rápidas envolvendo apenas o tempo para confirmação da transação pelos mineiros, que serão recompensados com novas moedas criadas. As taxas cobradas nas operações de compra e venda são detalhadas na Figura 3.5; toda negociação envolve pelo menos uma ordem de compra e uma ordem de venda, para a negociação ocorrer o preço da ordem de compra deve ser maior ou igual ao preço da ordem de venda. O momento de criação de cada ordem define qual taxa cada indivíduo irá pagar pela negociação, a ordem que foi criada primeiro e estava no livro de ordens em espera é considera uma ordem executada, e paga uma taxa de 0,3%, enquanto a ordem que foi criada depois será considerada uma ordem executora, e paga uma taxa de 0,7%. A negociação ocorre no momento que a ordem executora é criada, então para o usuário que a cria, é uma negociação instantânea. Uma ordem pode ser executada parcialmente e o montante que resta após a negociação é colocado automaticamente no livro de ordens, a taxa de negociação só é cobrada do valor negociado, ordens em espera podem ser canceladas e recriadas sem nenhuma cobrança de taxas, enquanto ordens executadas não podem ser canceladas, exemplos são encontrados em https://www.mercadobitcoin.com.br/info/execucao-ordem/.

Comissões, prazos e limites

CONTA

TIPO DE CONTA:	NORMAL	VIP	GOLD
CADASTRO	- CPF/CNPJ ativo - Validação de e-mail	**Pessoa Física:** - Documento colorido com foto **Pessoa Jurídica:** - Contrato social - Documento colorido com foto (do responsável)	- Documentação autenticada
MENSALIDADE	ZERO		
CUSTÓDIA	ZERO		

Conta GRÁTIS

O Mercado Bitcoin não tem custo de mensalidade ou custódia em nenhuma das contas. As taxas são cobradas apenas para as operações realizadas.

Figura 3.3: Tipo da conta em Mercado Bitcoin (data 29/11/2016).

A Figura 3.3 indica a documentação necessária para o usuário apresentar a fim de obter o nível de vip ou gold para sua conta. Com relação a custodia e mensalidade mostra que não são cobradas nenhuma taxa em qualquer nível de conta, o que permite ao usuário armazenar bitcoins ou até mesmo saldo em reais sem custos.

Mercado Bitcoin Servicos Digitais Ltda. [BR] | https://www.mercadobitcoin.com.br/comissoes-prazos-limites/

REAIS

				TIPO DE CONTA:	NORMAL	VIP	GOLD
REAIS	DEPÓSITO	COMISSÃO			R$ 2,90 + 1,99%		
		LIMITE 24HS			R$ 100,00	livre	
		PRAZO			5min até 24hs		
	SAQUE	COMISSÃO			R$ 2,90 + 1,99%		
		LIMITE 24HS	C/C		R$ 500,00	R$ 20.000,00	Flexível
			POUPANÇA			R$ 5.000,00	
		PRAZO			5min até 24hs		

* dentro do nosso horário operacional, o período normal de aprovação de depósitos é de 5 a 30 minutos, contados a partir da validação do comprovante de depósito e do banco disponibilizar o crédito em nossa conta. Entretanto, ressaltamos que esse prazo pode se estender em casos de exceções relacionadas à operação do sistema bancário.

Operações bancárias

Os depósitos e saques são feitos com uso das contas bancárias do Mercado Bitcoin. Atualmente, possuímos contas nos seguintes bancos: Banco do Brasil, Santander, Caixa Econômica Federal e Itaú.

HORÁRIO OPERACIONAL:

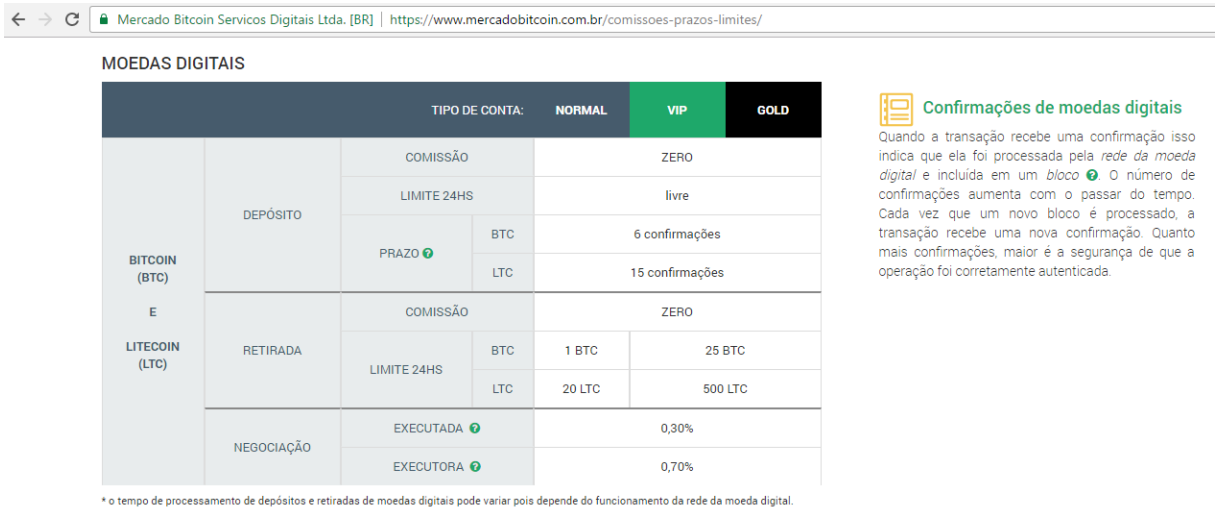
Aprovação de depósitos de reais:

dias úteis: 8hs às 19hs
finais de semana e feriados: 3 vezes ao dia (em regime de plantão)

Aprovação de saques de reais:
dias úteis: 8hs às 19hs

Figura 3.4: Comissões, limites e prazos em Mercado Bitcoin (data 29/11/2016).

A Figura 3.4 apresenta uma tabela informando as comissões, limites e prazos, relativos a saques e depósitos em reais, para usuários do tipo normal, vip e gold. Onde podemos observar que a comissão e o prazo são idênticos para saques e depósitos independente do tipo da conta do usuário. O limite para depósito e saques são influenciados diretamente pelo tipo de conta do usuário, restringindo usuários que não apresentaram nenhuma identificação de identidade, a realizar operações com limites diários reduzidos de no maximo R$ 100 para depósitos e R$ 500 para saques, usuário vip ou gold pode realizar depósitos ilimitados e saques com valores de até 20 mil reais para conta corrente no caso de usuário vip e limite flexível para usuário gold.



Mercado Bitcoin Servicos Digitais Ltda. [BR] | https://www.mercadobitcoin.com.br/comissoes-prazos-limites/

MOEDAS DIGITAIS

				TIPO DE CONTA:	NORMAL	VIP	GOLD
BITCOIN (BTC) E LITECOIN (LTC)	DEPÓSITO	COMISSÃO			ZERO		
		LIMITE 24HS			livre		
		PRAZO	BTC		6 confirmações		
			LTC		15 confirmações		
	RETIRADA	COMISSÃO			ZERO		
		LIMITE 24HS	BTC		1 BTC	25 BTC	
			LTC		20 LTC	500 LTC	
	NEGOCIAÇÃO	EXECUTADA			0,30%		
		EXECUTORA			0,70%		

* o tempo de processamento de depósitos e retiradas de moedas digitais pode variar pois depende do funcionamento da rede da moeda digital.

Confirmações de moedas digitais

Quando a transação recebe uma confirmação isso indica que ela foi processada pela *rede da moeda digital* e incluída em um *bloco*. O número de confirmações aumenta com o passar do tempo. Cada vez que um novo bloco é processado, a transação recebe uma nova confirmação. Quanto mais confirmações, maior é a segurança de que a operação foi corretamente autenticada.

Figura 3.5: Taxas por transações de compra e venda de bitcoins em Mercado Bitcoin (data 29/11/2016).

A Figura 3.5 apresenta uma tabela informando as comissões, limites e prazos, relativos a retirada e depósitos em bitcoins, e taxa de negociação de bitcoins no mercado, para contas do tipo normal, vip e gold. Podemos observar que o tipo de conta influencia apenas no limite para retiradas de moedas digitais aumentando os limites do usuário.

Foxbit é uma bolsa de bitcoins que cria um ambiente onde usuários podem comprar e vender bitcoins. Também tem uma política para verificação da identidade do usuário, que aumenta os limites para saques e depósitos entre outros benefícios. A tabela na Figura 3.6 apresenta as taxas, prazos e limites para transações. Depósitos em reais ou bitcoins não têm nenhuma cobrança de taxa, enquanto saques em reais têm taxas de 1,39% para bancos conveniados a Foxbit e uma taxa adicional de R$ 8,80 para outros bancos, uma particularidade nesta bolsa é uma pequena cobrança de taxa para saques em bitcoins, e apresenta uma das menores taxas para negociação de compra e venda de bitcoins do mercado brasileiro, 0,25% para ordem passiva (executada) e 0,5% para ordem ativa (executora), mais detalhes são descritos no website da empresa https://foxbit.com.br/taxas-prazos.

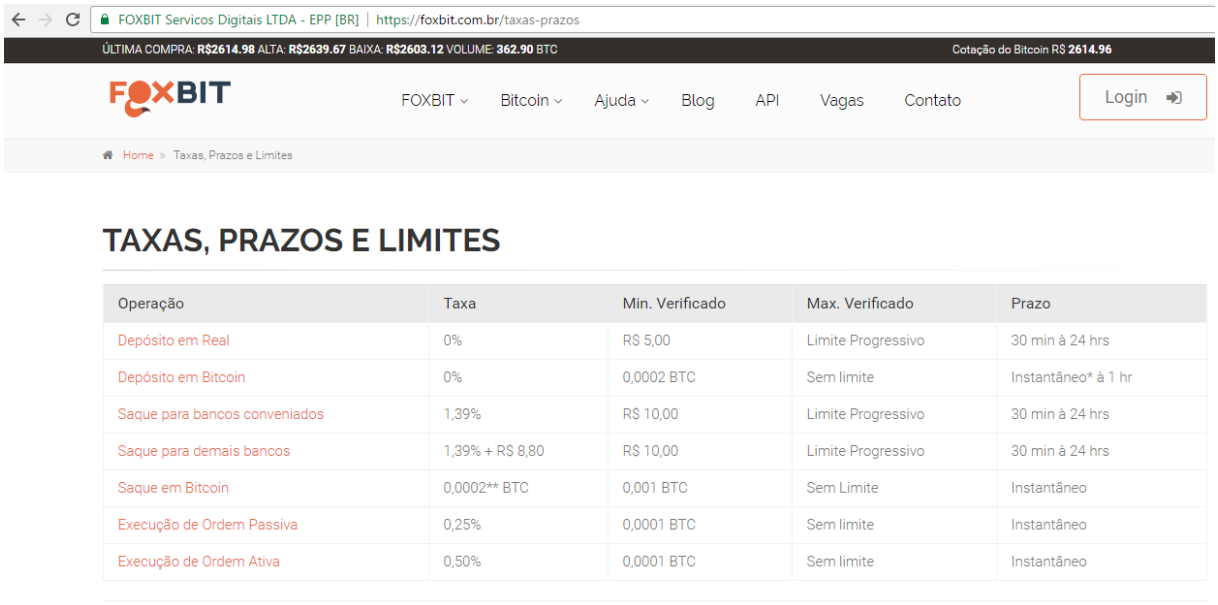


Figura 3.6: Taxas, prazos e limites no Foxbit (data 30/11/2016).

A Figura 3.6 apresenta a tabela de taxas, prazos e limites na *exchange* Foxbit, onde podemos observar que a taxa zero para deposito em reais ou bitcoins beneficia usuários que estão entrando no mercado, isso cria um ambiente convidativo ao investidor.

O volume de transações em ambas as bolsas de bitcoins, Mercado Bitcoin e Foxbit são da ordem de centenas de bitcoins diariamente. O preço do bitcoin em cada bolsa é determinado pelos usuários que estão criando ordens de compra e venda, isso cria uma leve diferença de preço entre ambas as bolsas, e a facilidade de transferir bitcoins entre elas contribuem para um equilíbrio de preço. Ambas também oferecem APIs com métodos para coletar dados e criar transações, que podem ser usadas para elaborar robôs, que realizam operações de compra e venda automaticamente, entre outras. O uso de robôs é frequente e facilita a atividade de usuários que atual como *day-trader* (usuário que realiza diversas operações de compra e venda diariamente), devido ao funcionamento 24 horas das bolsas, uma pessoa atuando manualmente perde diversas oportunidades de negociações em momentos de ausência por sua limitação física.

O volume movimentado no Brasil é considerado baixo comparado a outras bolsas

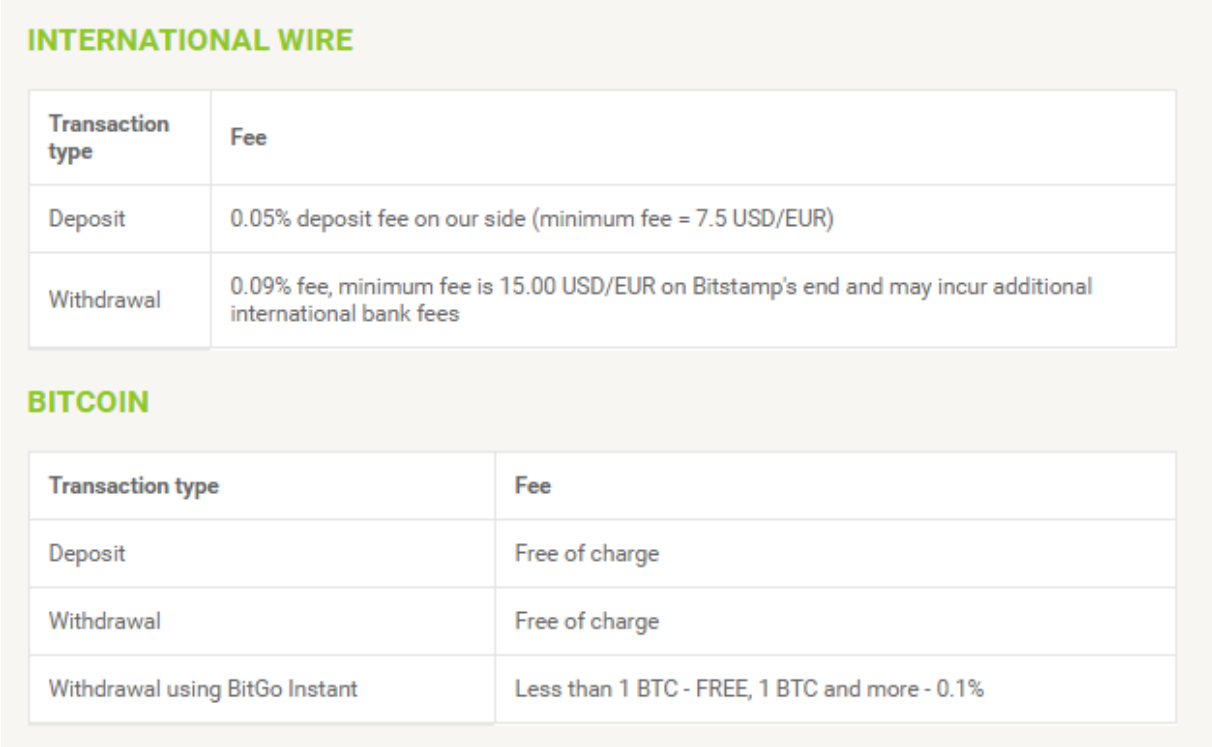
no exterior com volumes de transações diárias da ordem de dezenas de milhares de bitcoins. Usuários que desejam realizar uma compra ou venda com volume alto têm dificuldades nas bolsas brasileiras. Nestes casos, eles podem realizar a transação instantânea e serão penalizados com uma grande oscilação nos preços, ou criar e manter uma ordem no livro de ordens para ser executada aos poucos. O uso de bots também é adequado para auxiliar o usuário na compra e venda de grandes volumes de bitcoins de forma rápida, reduzindo a oscilação no preço. A Figura 3.1 mostra um histórico de ordens executadas em Mercado Bitcoin.

3.2.2 Câmbio de bitcoin por dólar

O mercado de câmbio de bitcoins por dólar é composto por diversas empresas, entre elas estão Bitstamp com sede em Londres, Reino Unido e OKCoin com sede em Pequim, China. Estas duas empresas estão entre as maiores bolsas que negociam bitcoin por dólar, e movimentam milhares de bitcoins diariamente em suas transações, a Figura 3.2 apresenta uma tabela comparativa de diversas empresas que negociam bitcoins por dólar. O maior volume de transações cria um cenário de mercado com taxas menores para as operações de compra e venda de bitcoins, reduz a diferença de preço entre as ordens de compra e venda, e a quantidade de transações por intervalo de tempo aumenta. Ambas as bolsas de bitcoins citadas oferecem APIs que possibilitam a criação de bots autônomos para atuar no mercado realizando operações de compra e venda entre outras.

Outro diferencial dessas bolsas em relação ao mercado brasileiro é que oferecem uma taxa de compra e venda variável, de acordo com o volume em dólares que cada usuário movimenta em sua conta nos últimos 30 dias. Isso beneficia usuários com grande volume de transações permitindo que paguem taxas cada vez menores podendo chegar à zero em alguns casos.

Bitstamp é uma bolsa de bitcoin cujo maior volume de transações envolve negociações de bitcoin por dólar, a plataforma também oferece a possibilidade de negociar bitcoin por euro e dólar por euro. Depósitos e retiradas de bitcoin ainda não têm cobrança de taxas, exceto ao usar BitGo, depósitos e retiradas de dólar e euro têm uma pequena taxa envolvida, mais detalhes na Figura 3.7. As taxas cobradas em operações de compra e venda de bitcoin variam entre 0,25% e 0,10%, dependendo do volume de transações do usuário, ordem executada e executora têm as mesmas taxas, a Tabela 3.1, apresentam as taxa relativas a negociações de bitcoin por dólar. Outras particularidades da empresa Bitstamp são a emissão de cartões de débito e crédito para seus usuários, e possibilidade de realizar retirada de fundos em barras de ouro, o preço do ouro é cotados em USD e atualizados a cada 5 minutos.



INTERNATIONAL WIRE

Transaction type	Fee
Deposit	0.05% deposit fee on our side (minimum fee = 7.5 USD/EUR)
Withdrawal	0.09% fee, minimum fee is 15.00 USD/EUR on Bitstamp's end and may incur additional international bank fees

BITCOIN

Transaction type	Fee
Deposit	Free of charge
Withdrawal	Free of charge
Withdrawal using BitGo Instant	Less than 1 BTC - FREE, 1 BTC and more - 0.1%

Figura 3.7: Taxas cobradas para depósitos e retiradas (data 01/12/2016).

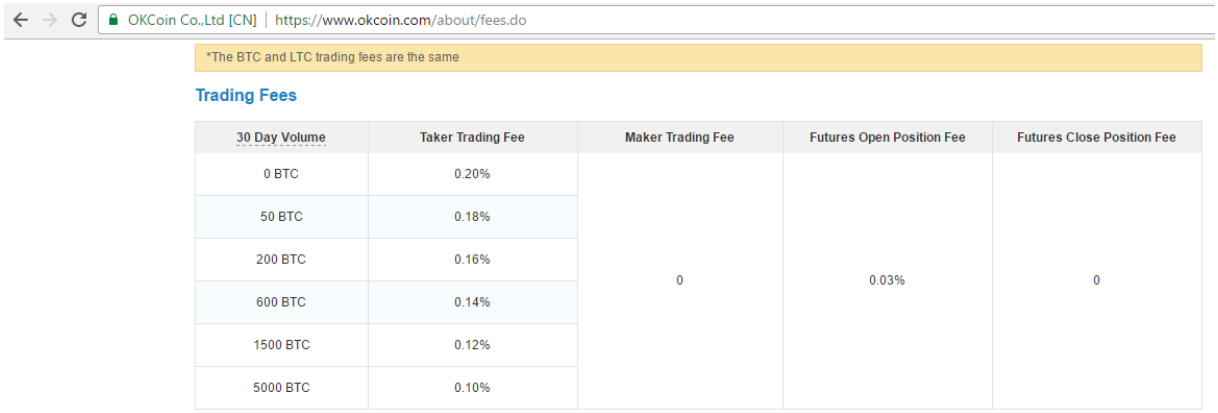
A Figura 3.7 apresenta as taxas cobradas em saques e depósitos para dolar, euro e bitcoins. Onde se destacam as operações em bitcoins que não têm cobrança de taxa o que facilita a transferência de capital em bitcoins sem custos ao investidor.

Taxa	Volume em dólar negociado nos últimos 30 dias
0,25%	< $20.000
0,24%	< $100.000
0,22%	< $200.000
0,20%	< $400.000
0,15%	< $600.000
0,14%	< $1.000.000
0,13%	< $2.000.000
0,12%	< $4.000.000
0,11%	< $20.000.000
0,10%	> $20.000.000

Tabela 3.1: Taxas das transações de compra e venda de bitcoin por dólar (data 01/12/2016)

OKCoin é uma bolsa de bitcoin que permite negociar bitcoin por dólar, através do domínio https://www.okcoin.com, alguns diferenciais são as taxas de compra e venda de bitcoins que podem chegar a zero e conta com uma ferramenta de alavancagem, que permite ao usuário pegar empréstimo de bitcoin ou dólares, também trabalha com o mercado de bitcoin futuro. As taxas envolvidas nas negociações de compra e venda de moedas digitais, são baseadas no volume de bitcoins movimentado pelo usuário e o tipo de ordem que foi executada. Ordens colocadas

no livro de ordens, são consideradas ordens passivas que podem ser canceladas ou executadas, quando executadas são taxadas pela coluna *Maker Trading Fee* da Figura 3.8, que é de 0% para qualquer operação de compra ou venda. Ordens que são executadas instantaneamente sem ir para o livro de ordens, são consideradas ordens executoras e são taxadas pela coluna *Taker Trading Fee* da Figura 3.8, que variam entre 0,2% e 0,1% dependendo do volume de bitcoins negociado pelo usuário nos últimos 30 dias.


OKCoin Co.,Ltd [CN] | https://www.okcoin.com/about/fees.do

*The BTC and LTC trading fees are the same

Trading Fees

30 Day Volume	Taker Trading Fee	Maker Trading Fee	Futures Open Position Fee	Futures Close Position Fee
0 BTC	0.20%	0	0.03%	0
50 BTC	0.18%			
200 BTC	0.16%			
600 BTC	0.14%			
1500 BTC	0.12%			
5000 BTC	0.10%			

Figura 3.8: Taxas para transações de compra e venda de moedas digitais em OKCoin (data 01/12/2016).

A Figura 3.8 apresenta a tabela de taxas, onde podemos notar que um aumento no volume de bitcoins negociados nos últimos 30 dias resulta em redução nas taxas para ordens do tipo *taker*. Além disso, a cobrança de taxa zero para ordens do tipo *maker* que são colocadas no livro de ordens permite que o investidor negocie bitcoins sem nenhum custo adicional.

O mecanismo de alavancagem em OKCoin oferece crédito baseado no valor da conta do usuário, este valor é calculado baseado no saldo em dólares e moedas digitais presentes em sua conta, e permite ao usuário realizar empréstimos de até 3x o valor de sua conta. O empréstimo pode ser em dólar, bitcoin ou litcoin, taxas de juros são cobradas diariamente pelos empréstimos abertos, e o usuário fica impedido de retirar fundos que comprometam o pagamento dos empréstimos. Um mecanismo de controle de risco fica ativo monitorando o valor da conta de usuários que têm empréstimos abertos, se o valor da conta cair a níveis que representam risco para o pagamento dos empréstimos, um protocolo automático é executado que utiliza o saldo do usuário para pagar os empréstimos.

O preço do bitcoin é similar em ambas as bolsas de bitcoin que trabalham com dólar, as variações de preço são semelhantes e podem ocorrer grandes variações em curtos períodos de tempo como mostrado em Figura 3.9 e Figura 3.10, as bolsas de câmbio Bitstamp e OKCoin, oferecem APIs distintas, que podem ser usadas para criação de bots que funcionam automaticamente realizando negociações ou coletas de dados.

Figura 3.9: Grafico de negociações na bolsa de bitcoin Bitstamp.

A Figura 3.9 apresenta um gráfico de negociações de bitcoins, onde podemos notar uma grande variação nos preços em um curto intervalo de tempo. Esse tipo de variação é comum no mercado de bitcoins devido ao baixo volume de ordens em aberto nos livros, e com isso quando um volume alto é vendido rapidamente cria uma queda nos preços instantaneamente.

Figura 3.10: Grafico de negociações na bolsa de bitcoin OKCoin.

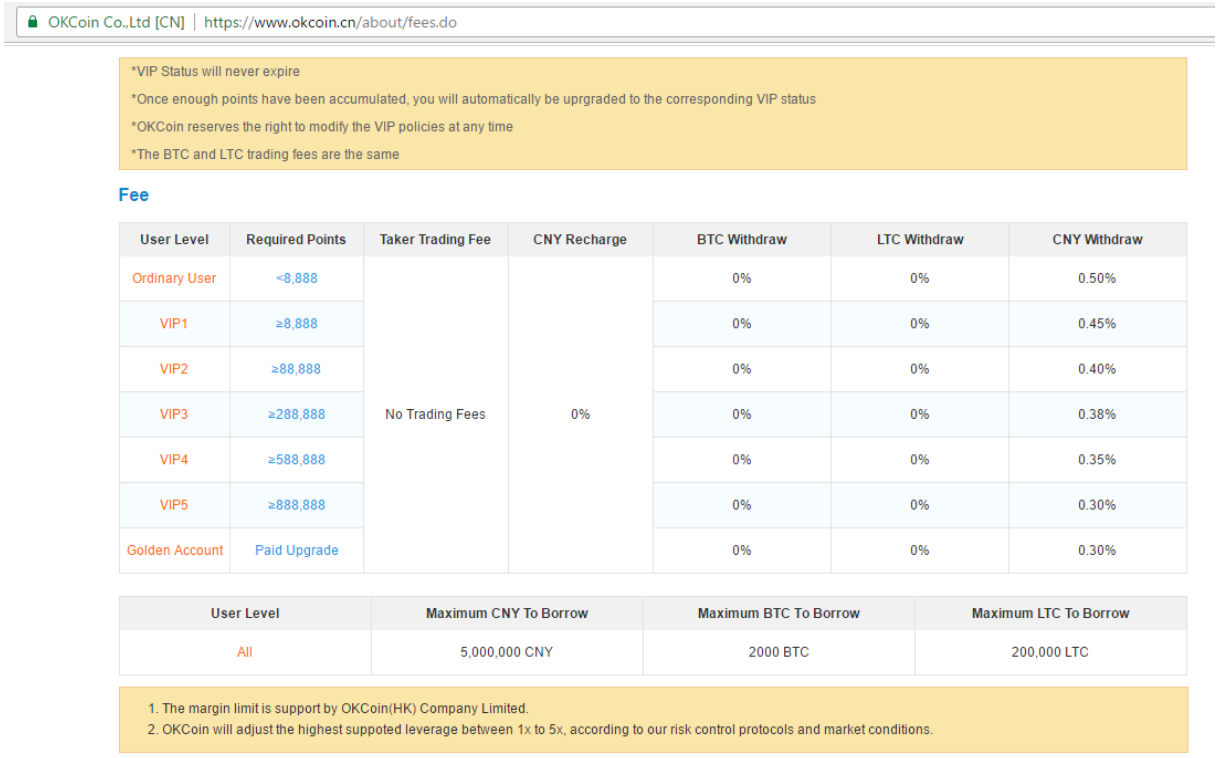
A Figura 3.10 apresenta um gráfico de negociações de bitcoins, onde podemos destacar uma grande queda no preço da moeda em um intervalo de um minuto, a parte inferior da figura contém um gráfico de volume onde podemos notar que durante o período de queda houve

um grande volume de bitcoins negociado. Concluímos que operações com volumes elevados criam variações bruscas no preço do bitcoin, devido a baixa quantidade de ordens em aberto na *exchange*.

3.2.3 Câmbio de bitcoin por Yuan chinês

Até o início de 2017, o mercado de câmbio para bitcoins em yuan chinês foi muito maior em termos de volumes de transação, algumas bolsas de bitcoin movimentavam valores da ordem de milhões de bitcoins diariamente. A Figura 3.2 mostra uma tabela com alguma bolsas de bitcoin, entre elas OKCoin que foi citada anteriormente, e também atua na bolsa de bitcoin por yuan no domínio https://www.okcoin.cn. O volume elevado de transações cria um ambiente com taxas de compra e venda ainda menores, mecanismo de alavancagem que permitem empréstimos maiores, centenas de operações são realizadas por segundo e grandes oscilações de preço ocorrem em curtos intervalos de tempo.

A bolsa de bitcoin por yuan da empresa OKCoin apresentou como diferencial as taxas de compra e venda de 0% para ambos os tipos de ordens *taker* (executora) e *maker* (executada), também permite empréstimos de até 5x o valor da conta. A conta do usuário é classificada num sistema de níveis, que reduz as taxas para saques de yuan Chinês nos níveis mais altos. A redução de taxas pode ser conquistada com o acúmulo de pontos ou na compra do status de *Golden Account*, a Figura 3.11 apresenta mais detalhes. Empréstimos são monitorados por um sistema de controle de riscos que impede o usuário de retirar fundos da conta que comprometam o pagamento, são cobrados juros diariamente pelos empréstimos abertos, um sistema de controle de risco monitora o saldo na conta, e pode executar o pagamento automaticamente caso o usuário fique com saldo abaixo de um limite estipulado que comprometam o pagamento dos empréstimos. Os empréstimos podem ser pagos parcialmente ou totalmente a qualquer momento. No início de 2017 a *exchange* aumentou a taxa de negociação para 0,2% o que resultou em uma redução no volume de bitxoins negociados diariamente para valores da ordem de milhares de bitcoins por dia.



OKCoin Co.,Ltd [CN] | https://www.okcoin.cn/about/fees.do

*VIP Status will never expire
*Once enough points have been accumulated, you will automatically be uprgraded to the corresponding VIP status
*OKCoin reserves the right to modify the VIP policies at any time
*The BTC and LTC trading fees are the same

Fee

User Level	Required Points	Taker Trading Fee	CNY Recharge	BTC Withdraw	LTC Withdraw	CNY Withdraw
Ordinary User	<8,888	No Trading Fees	0%	0%	0%	0.50%
VIP1	≥8,888			0%	0%	0.45%
VIP2	≥88,888			0%	0%	0.40%
VIP3	≥288,888			0%	0%	0.38%
VIP4	≥588,888			0%	0%	0.35%
VIP5	≥888,888			0%	0%	0.30%
Golden Account	Paid Upgrade			0%	0%	0.30%

User Level	Maximum CNY To Borrow	Maximum BTC To Borrow	Maximum LTC To Borrow
All	5,000,000 CNY	2000 BTC	200,000 LTC

1. The margin limit is support by OKCoin(HK) Company Limited.
2. OKCoin will adjust the highest suppoted leverage between 1x to 5x, according to our risk control protocols and market conditions.

Figura 3.11: Taxas de transação e limites para empréstimos (data 02/12/2016).

A Figura 3.11 apresenta a tabela de taxas cobradas na *exchange* OKCoin antes do aumento nas taxas que ocorreu no início de 2017. Com a ausência de taxas os usuários tem a liberdade de aumentar a frequência de negociações sem serem penalizados pela taxa. Os limites elevados para empréstimos, em comparação com plataforma da OKCoin que negocia por bitcoin por dólar, também são um diferencia da *exchange* que possibilita aos usuários maiores margens de lucro.

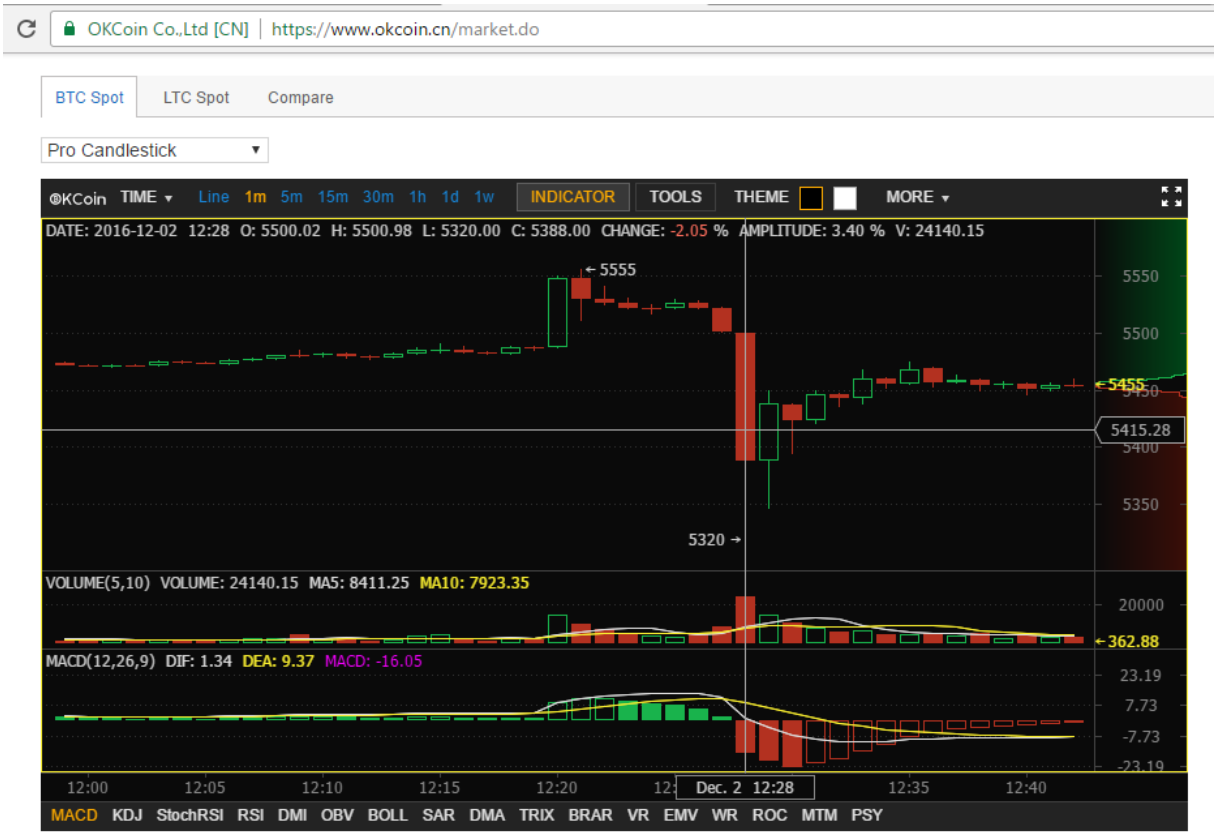


Figura 3.12: Alta volatilidade.

A Figura 3.12 apresenta um gráfico de negociações de bitcoin por yuan, onde podemos observar uma variação no preço de 3,4% no período de 1 minuto, onde foram negociados mais de 24 mil bitcoins. Concluímos que grandes volumes negociados tem ligação com maiores variações nos preços.

3.3 MERCADO DE FINANCIAMENTO

Uma das *exchanges* de bitcoin que oferecem uma plataforma de financiamento é a Bitfinex, onde usuários podem negociar bitcoins com fundos próprios depositados previamente, ou operações mistas com parte do valor financiado. "As transações financiadas em tokens digitais são permitidas através da funcionalidade de financiamento habilitada para plataforma do site. Os provedores de financiamento podem oferecer financiamento através de sua própria conta, se assim o desejarem. Os beneficiários de financiamento podem aceitar financiamento de provedores de financiamento para até 70% do valor de uma compra de token digital. Por exemplo, se um comerciante depositar US $ 30,00 no Site, ela pode obter financiamento em um valor não superior a US $ 70,00 para comprar US $ 100,00 em bitcoins em uma transação financiada"Bitfinex (2016).

Segundo Bitfinex (2016), "O *shorting* é um tipo de transação financiada permitida na plataforma da Bitfinex. Em uma venda longa, típica de bitcoin, o vendedor entra em um comércio

local de bitcoin e resolve a transação, entregando bitcoin que ela possui de forma definitiva. Em uma venda curta de bitcoin, o vendedor também entra em uma venda regular de bitcoin, exceto que a transação é concluída mediante entrega de bitcoin adquiridos de terceiros através de empréstimo".

Os tokens digitais podem ser emprestados para fins de vendas curtas através da plataforma de financiamento da *exchange*. O usuário pode criar ofertas de *borrow* (pedido de empréstimo), criar ofertas de *lend* (oferta de empréstimo), ou executar ofertas no livro de pedidos de financiamento. Quem recebe o empréstimo não pode pedir mais de 70% dos bitcoins vendidos em uma venda curta. O resultado de qualquer venda curta serve como garantia para o empréstimo dos bitcoins até que eles sejam reembolsados Bitfinex (2016).

Os usuários da plataforma podem obter financiamento de duas formas: podem fazer lances para financiamento no livro de pedidos de financiamento ou podem optar por ser automaticamente combinados através do mecanismo de correspondência de pedidos da *exchange*, com um ou mais provedores de financiamento no livro de pedidos de financiamento com o melhor preço prevalecente no livro de pedidos de financiamento, o mecanismo automático é ativado no momento que uma negociação de margem é realizada Bitfinex (2016).

O livro de pedidos de financiamento opera independentemente do livro de pedidos de negociação. Uma vez que o financiamento desejado é assegurado por um beneficiário de financiamento, as transações financiadas e não financiadas no livro de pedidos de negociação são indistinguíveis entre si para o mecanismo de correspondência comercial da Bitfinex.

O valor do financiamento, o prazo do financiamento e a taxa de juros são todos os termos comerciais negociados através do livro de ordens de financiamento entre provedores de financiamento e destinatários de financiamento. Por exemplo, suponha que Alice tenha US$30,00 em sua conta na *exchange*. Alice pode obter US$70,00 em financiamento à taxa de juros X para o período Y no livro de pedidos de financiamento, Alice tornando-se uma beneficiária de financiamento, e Bob um provedor de financiamento. Com esse valor agregado de US$100,00, Alice pode comprar US$100,00 em bitcoins no livro de pedidos de negociação de Cindy, ou de mais vendedores. Alice tem o direito de reembolsar o financiamento (incluindo os juros acumulados) a qualquer momento sem pré-pagamento ou outra penalidade. A obtenção de financiamento não cria qualquer obrigação de comprar bitcoins no livro de pedidos de negociação. Alice também pode substituir o financiamento de Bob a qualquer momento com outro financiamento mais favorável Bitfinex (2016).

No exemplo acima, os bitcoins comprados por Alice (US$100,00) estão sujeitos a um penhor a favor de Bob, até o montante total de financiamento fornecido por Bob (US$70,00 mais qualquer componente de juros) serem pagos. Alice pode remover qualquer quantidade de bitcoins do site que não está sujeito ao penhor. Se o patrimônio do beneficiário do financiamento cair para 15% ou abaixo disso, do valor total do empréstimo - calculado como o quociente (expresso em porcentagem) obtido dividindo (a) por (b) descritos abaixo:

- (a) o excesso de (i) sobre (ii)

(i) o valor de mercado dos bitcoins comprados

(ii) o montante total do empréstimo (acrescido de juros acumulados e não pagos) relativos a todo o financiamento pendente

- (b) o valor de mercado dos bitcoins, (i) usado acima.

A Bitfinex forçará a liquidação dos bitcoins na conta da Alice sem aviso prévio, para devolver o financiamento ao fornecedor, com juros acumulados, e devolver o saldo que restar ao destinatário do financiamento Alice. A Bitfinex não faz chamadas de margem. Por exemplo, se o valor dos bitcoins comprados cai de US$100,00 para US$82,35, a diferença entre esse valor e o financiamento obtido por Alice seria de US$82,35 - US$70,00 ou US$12,35. Tomada como uma porcentagem do valor dos bitcoins, US$12,35 / US$82,35 é inferior a 15%. Em outras palavras, se o valor dos bitcoins caiu para US$82,35 no total, os bitcoins da Alice seriam liquidados pelo Bitfinex no livro de pedidos de negociação, Bob seria reembolsado e qualquer diferença restante (US$12,35, excluindo juros) seria entregue a Alice.

De acordo com os termos de serviço da Bitfinex, o usuário concede à *exchange* permissão para implementar, cobrar, monitorar e manter todos e quaisquer privilégios em favor dos provedores de financiamento e forçar a liquidação de qualquer tokens digitais em seu nome ou controle no site se necessário, para certifique-se de que qualquer provedor de financiamento no site de quem o usuário obteve financiamento será reembolsado na íntegra.

Segundo Bitfinex (2016), "mercados de tokens digitais podem mudar rapidamente. Os movimentos de preços podem ser inesperados. Não há garantia contra perdas no site. O usuário pode perder tudo que está em suas diversas carteiras no site, caso tenha se envolvimentos em financiamento. O usuário é responsável por qualquer atividade comercial e não comercial na sua conta Bitfinex, mas a Bitfinex deve sempre manter a capacidade de proteger os provedores de financiamento, liquidando a conta do beneficiário de financiamentos, quando necessário. A Bitfinex não pode garantir que evitara perdas mesmo com a capacidade de forçar a liquidação de qualquer uma das suas posições (devido, por exemplo, à volatilidade do mercado e liquidez). A Bitfinex não será e não é responsável por nenhum provedor de financiamento que perca fundos ou tokens digitais para qualquer destinatário de financiamento no Bitfinex".

Apesar do forte controle exercido pela *exchange*, ainda é possível que o saldo total na conta do beneficiário de financiamento não seja suficiente para pagar totalmente o financiamento, e, portanto, o provedor de financiamento também está vulnerável a perdas. Outras *exchanges* que têm plataforma de financiamento tem protocolos de segurança semelhante, sempre visando proteger o provedor de financiamento da melhor forma possível.

3.3.1 Negociação de margem

Negociação de margem (*Margin Trading*) são negociações de bitcoins com fundos alavancados, empréstimos de terceiros que possibilitam o investidor realizar investimentos maiores,

pagando taxas de juros diárias pelos fundos financiados. As ordens de margem executadas não liquidam imediatamente os saldos da carteira. Em vez disso, eles abrem, fecham ou modificam uma posição alavancada. Os saldos da carteira são liquidados apenas quando o usuário fecha ou reivindica uma posição Bitfinex (2016). No modo de margem, o usuário pode abrir uma posição de compra realizando a compra de bitcoins com saldo de terceiros, uma posição aberta pode ser ampliada quando se realiza uma nova compra, ou reduzida quando se realiza uma venda, o oposto é valido para criar uma posição de venda.

Em uma negociação de margem, por exemplo, se o usuário com US$30,00, financiar US$70,00, para comprar US$100,00 em bitcoins e o valor dos bitcoins sobe para valer US$120,00, o usuário pode realizar uma retirada de bitcoins no valor de US$20,00 de sua carteira, ou realizar um pagamento parcial do empréstimo ao fornecedor do financiamento, o pagamento antecipado reduz a dívida e consequentemente o montante dos juros aplicados sobre ela.

3.4 DIFERENÇAS DE MERCADO

Entre as diversas diferenças entre bolsas de bitcoin, a liquidez do mercado, o volume de bitcoins negociados e as taxas de transação são pontos chaves.

3.4.1 Liquidez do mercado

O mercado de bitcoin por dólar, no qual Bitstamp, Bitfinex e OKCoin estão inseridas, apresenta maior liquidez comparado ao mercado de bitcoin por real, composto por Mercado Bitcoin e Foxbit entre outras. Observando as Figura 3.13 e Figura 3.14, podemos notar a diferença entre o volume de ordens em aberto entre Mercado Bitcoin e Bitstamp, um livro de ordens com maior volume permite que clientes realizem operações com grande volume instantaneamente sem grandes variações no preço, vamos explicar a diferença mostrando como seria executada uma compra de 450 bitcoins, nos dois cenários.

Na Figura 3.13, podemos observar que o volume de ordens de compra representado pela areá azul no gráfico apresenta picos de crescimento em alguns setores distantes do valor atual do bitcoin, isso representa usuários que criam ordens de compra a valores baixos esperando uma grande queda nos preços para realizar a compra. A região em vermelho que representa as ordens de venda em aberto indicam um baixo volume de ordens no livro distribuídas a preços muito acima do preço atual. O baixo volume de ordens implica em grandes variações de preço em momentos com alto volume de negociações.

Figura 3.13: Gráfico com volume do livro de ordens em Mercado Bitcoin (data 30/05/17).

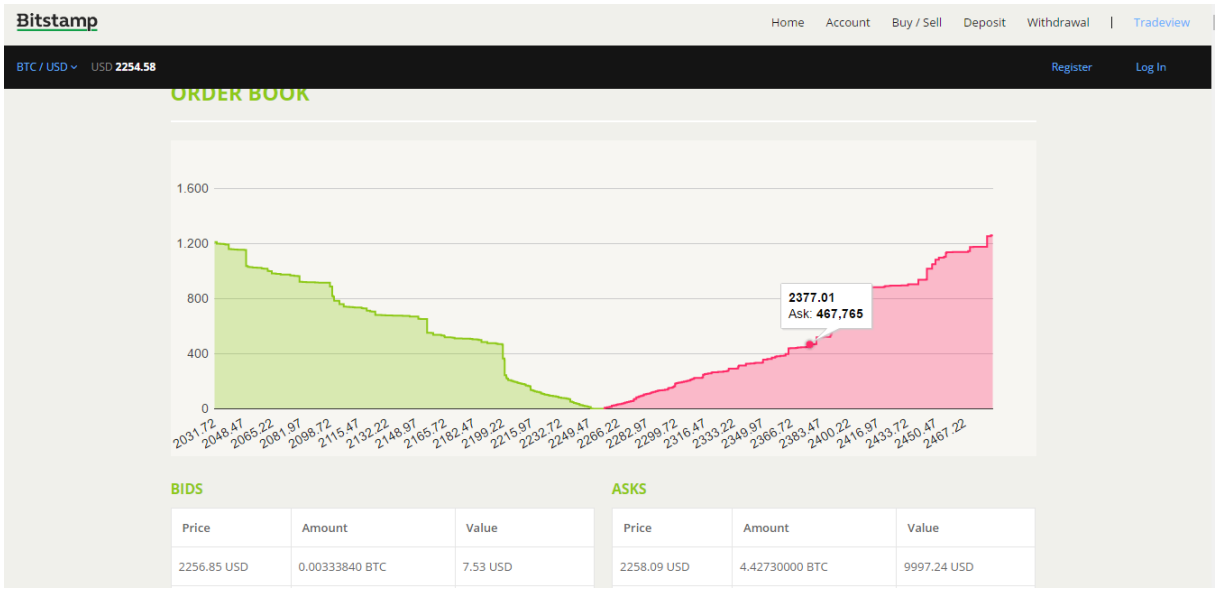


Figura 3.14: Gráfico com volume do livro de ordens em Bitstamp (data 30/05/2017).

Na Figura 3.14, o gráfico representa o volume de ordens em aberto, que quando comparado ao livro de ordens em Figura 3.13, podemos notar que é mais denso, que torna possível

negociar maiores volumes causando pouca variação nos preços.

- **Compra de 450 bitcoins em Mercado Bitcoin:** Neste caso, quando o usuário envia o comando para executar a compra de 450 bitcoins, as ordens de venda no livro de ordens serão executadas em sequência até atingir o volume de compra, portanto o preço de compra vai variar de R$ 9.810,00 até R$ 16.939,14, uma variação de 72,67% a partir do preço inicial de R$ 9.810,00.

- **Compra de 450 bitcoins em Bitstamp:** Quando o usuário envia o comando para executar a compra de 450 bitcoins, o preço de compra vai variar de USD 2.258,09 até USD 2.377,01, uma variação de 5,26% a partir do preço inicial de USD 2.258,09.

O problema da liquidez pode ser amenizado, caso o cliente realize diversas compras menores periodicamente, isso daria tempo para outros usuários criarem novas ordens de venda, reabastecendo o livro de ordens, porém esse processo pode levar horas ou até dias, porque tornasse dependente do volume de transações diárias, que em bolsas de bitcoins brasileiras ainda é baixo comparado ao mercado estrangeiro.

Com o uso de robôs podemos apresentar duas soluções para tratar o problema da liquidez de forma eficiente, com menor variação nos preços e reduzindo drasticamente o esforço do investidor. Vamos apresentar a noção de funcionamento do robô, ele deve comprar 450 bitcoins em Mercado Bitcoin ao longo do tempo.

- **Robô 1:** Realiza compras de valores fixos periodicamente com o período determinado pelo usuário, e sempre que o preço de venda estiver num intervalo de no máximo x% acima do preço inicial, valor de x determinado pelo usuário, o robô pode rodar até comprar o montante desejado de 450 bitcoins.

- **Robô 2:** Cria pequenas ordens de compra no valor de mercado, com volume determinado pelo usuário, a criação de ordens é feita periodicamente e ordens não executadas são canceladas.

Os robôs citados acima são bastante simples e têm possibilidades de diversas melhorias para aumentar os benefícios ao usuário. No estado citado apresentam os seguintes benefícios:

- **Redução de tempo e esforço do usuário:** O tempo que o usuário passaria, criando ordens e esperando que o livro de ordens fosse novamente preenchido com preços adequados é poupado, porque o robô consegue realizar essas operações de forma automática, após receber os parâmetros de execução. Devido ao funcionamento 24 horas das bolsas de bitcoins esse tipo de situação é ainda mais adequada a robôs que podem funcionar em horários que seriam incômodos ao investidor.

- **Redução no preço médio de compra:** Um robô que cria ordens de compra vai realizar as operações a um preço mais baixo e também irá pagar uma taxa de transação menor, consequentemente reduzindo o preço médio de compra.

- **Redução no tempo necessário para realizar a tarefa completa:** Algumas ordens são colocadas no livro e executadas em questão de segundos, devido ao uso de robôs por outros usuários e também porque existem outros usuários concorrendo para realizar as negociações ao melhor preço.

3.4.2 Volume negociado diariamente

O volume de bitcoin negociado diariamente por uma bolsa é um fator importante e está ligado a liquides do mercado, bolsas com volume diário mais elevado têm uma liquidez melhor. O mercado de bitcoin por real vem crescendo ao longo do tempo, mas as bolsas de bitcoin por dólar têm um volume diário de transações muito superior como mostrado nas Figuras 3.15 e 3.16. A partir disso, podemos supor que o perfil de usuário atuando em cada mercado é diferente, e que em mercados com maior volume, a velocidade de negociação é mais rápida e também são negociadas maiores quantidades de bitcoins em cada transação, podemos observar isso na Figura 3.17.

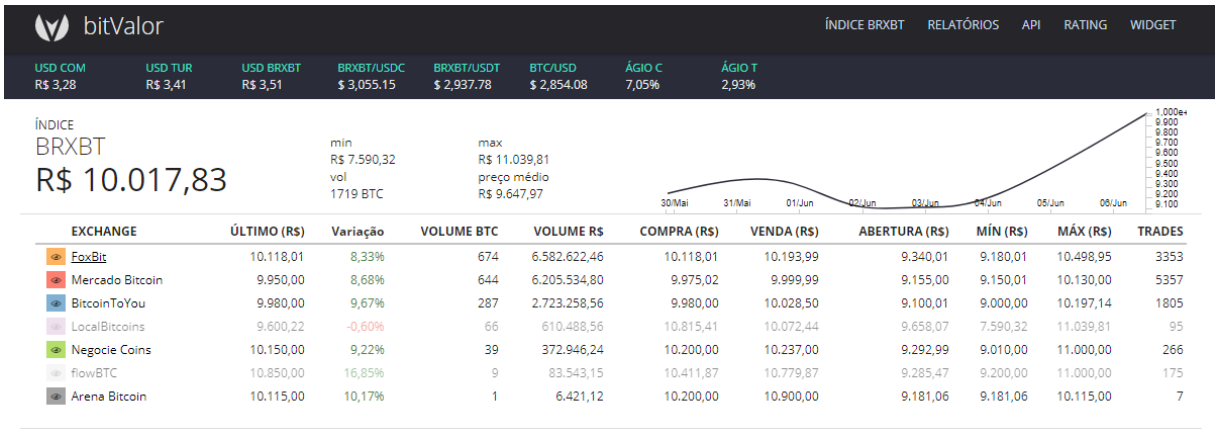


Figura 3.15: Estatísticas diárias de bolsas de bitcoin por real.

A Figura 3.15 apresenta uma tabela com varias *exchanges* que negociam bitcoin por real, a partir dos dados podemos notar uma diferença no preço do bitcoin que reflete o comportamento dos usuários negociando em ambientes diferentes, a difereça de preços em alguns casos pode ser usada para gerar lucro comprando em uma *exchange* para vender em outra.

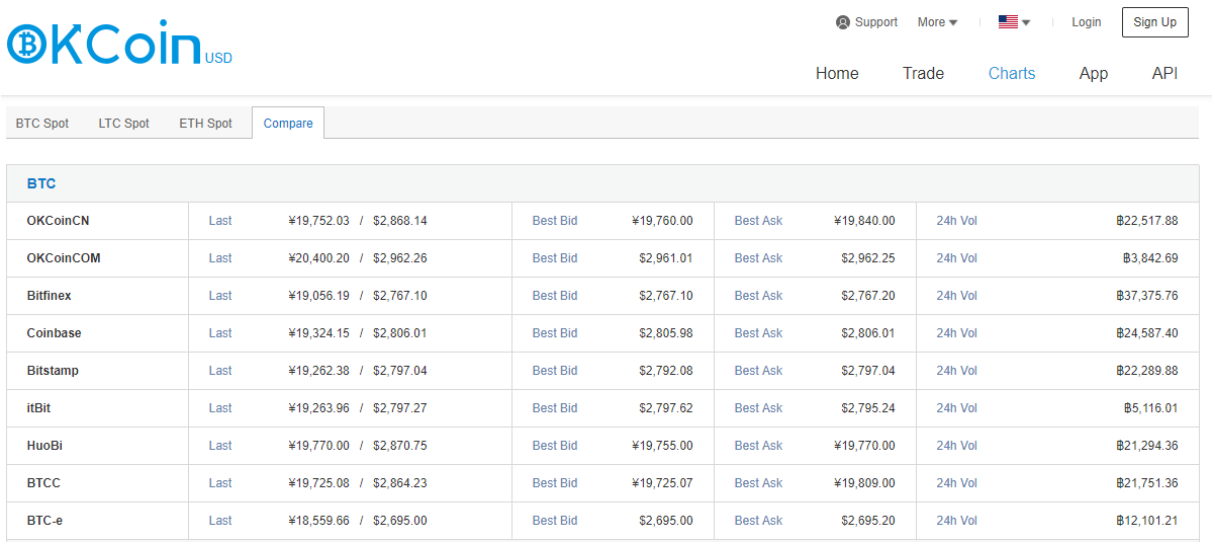
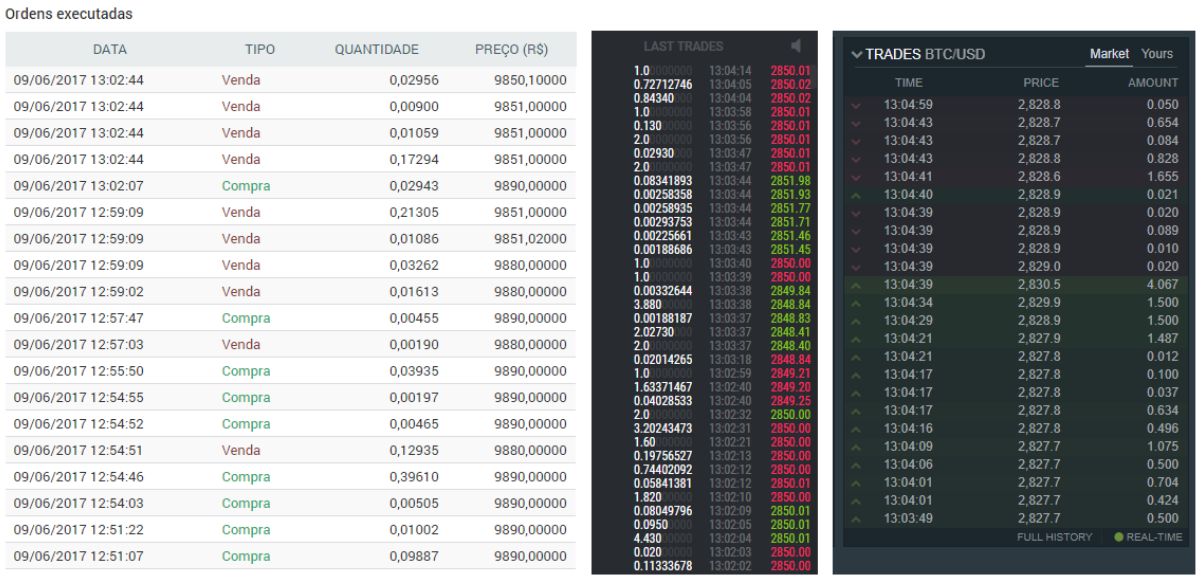


Figura 3.16: Estatísticas diárias de bolsas de bitcoin por dólar e bitcoin por yuan.

A Figura 3.16 apresenta uma tabela com varias *exchanges* que negociam bitcoin. A partir da segunda coluna que apresenta o preço estimado da ultima ordem executada em dólar e yuan podemos notar uma diferença no preço da moeda em diferentes *exchanges*, neste caso o preço do bitcoin em dólar está 7% mais caro na *exchange* OKCoinCOM em relação ao preço da *exchange* Bitfinex. Isso mostra que é possível obter lucro negociando entre *exchanges* distintas.



Ordens executadas

DATA	TIPO	QUANTIDADE	PREÇO (R$)
09/06/2017 13:02:44	Venda	0,02956	9850,10000
09/06/2017 13:02:44	Venda	0,00900	9851,00000
09/06/2017 13:02:44	Venda	0,01059	9851,00000
09/06/2017 13:02:44	Venda	0,17294	9851,00000
09/06/2017 13:02:07	Compra	0,02943	9890,00000
09/06/2017 12:59:09	Venda	0,21305	9851,00000
09/06/2017 12:59:09	Venda	0,01086	9851,02000
09/06/2017 12:59:09	Venda	0,03262	9880,00000
09/06/2017 12:59:02	Venda	0,01613	9880,00000
09/06/2017 12:57:47	Compra	0,00455	9890,00000
09/06/2017 12:57:03	Venda	0,00190	9880,00000
09/06/2017 12:55:50	Compra	0,03935	9890,00000
09/06/2017 12:54:55	Compra	0,00197	9890,00000
09/06/2017 12:54:52	Compra	0,00465	9890,00000
09/06/2017 12:54:51	Venda	0,12935	9880,00000
09/06/2017 12:54:46	Compra	0,39610	9890,00000
09/06/2017 12:54:03	Compra	0,00505	9890,00000
09/06/2017 12:51:22	Compra	0,01002	9890,00000
09/06/2017 12:51:07	Compra	0,09887	9890,00000

Figura 3.17: Histórico de ordens executadas em Mercado Bitcoin, Bitstamp e Bitfinex.

Na Figura 3.17, temos um comparativo de histórico de ordens entre três *exchanges*, podemos notar uma diferença no volume das ordens executadas assim como na frequência com que as negociações ocorrem, considerando que a Figura registra um momento aleatório. Concluímos que no geral os usuários negociando em cada mercado apresentam um comportamento diferente.

No mercado de bitcoin por real foi observado que existe uma redução no volume e frequência das negociações durante a madrugada, na Figura 3.18 temos um gráfico que mostra

o volume de bitcoins negociado por hora, destacando períodos da madrugada onde o volume negociado é baixo. Nos fins de semana também ocorre uma redução no volume que é mostrada na Figura 3.19, onde o gráfico apresenta o volume de bitcoins negociados diariamente, destacando os fins de semana. O comportamento observado durante as madrugadas e fins de semanas foi observados ao longo de mais de dois anos com algumas poucas exceções. Supomos que tem relação com o perfil dos usuários, que em determinados horários não está disponível para negociar. Em relação ao mercado de bitcoin por dólar é possível que o uso global das plataformas torne o mercado mais dinâmico ao longo das 24 horas todos os dias, também é comum em bolsas de bitcoin por dólar, um sistema para beneficiar o usuário que negocia grandes volumes mensalmente.

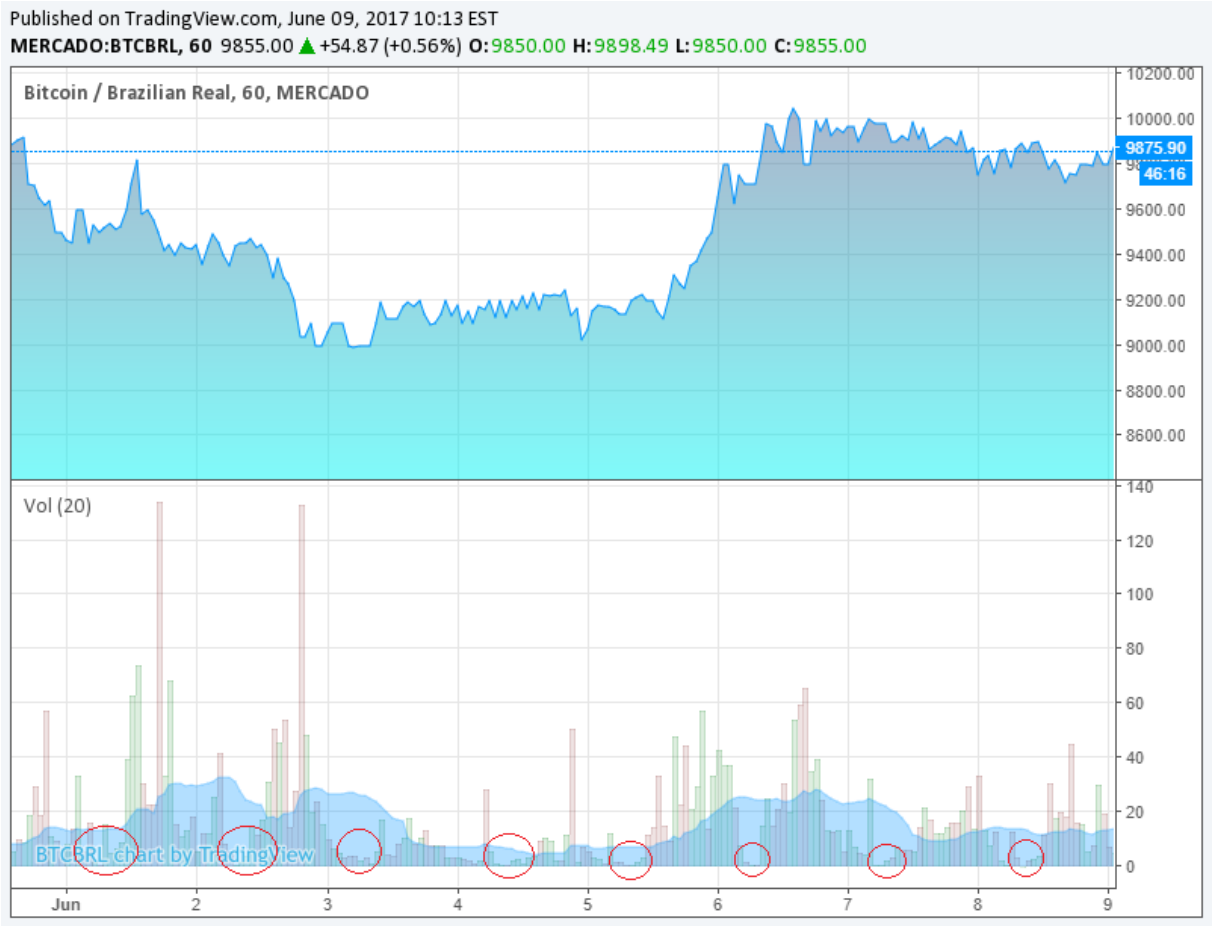


Figura 3.18: Redução no volume de transações na madrugada para o início da manhã.

A Figura 3.18 apresenta o preço e o volume negociado numa escala com períodos de uma hora, ao observar o comportamento podemos notar uma tendência de redução no volume negociado em períodos entre a madrugada e o início da manhã, a redução no volume não apresenta nenhuma relação visível com os preços. Esse comportamento reflete a dificuldade do usuário negociando manualmente ficar ativo no mercado por longos períodos consecutivos.

A Figura 3.19 apresenta o preço e o volume negociado numa escala com períodos de um dia, ao observar o comportamento podemos notar uma tendência de redução no volume

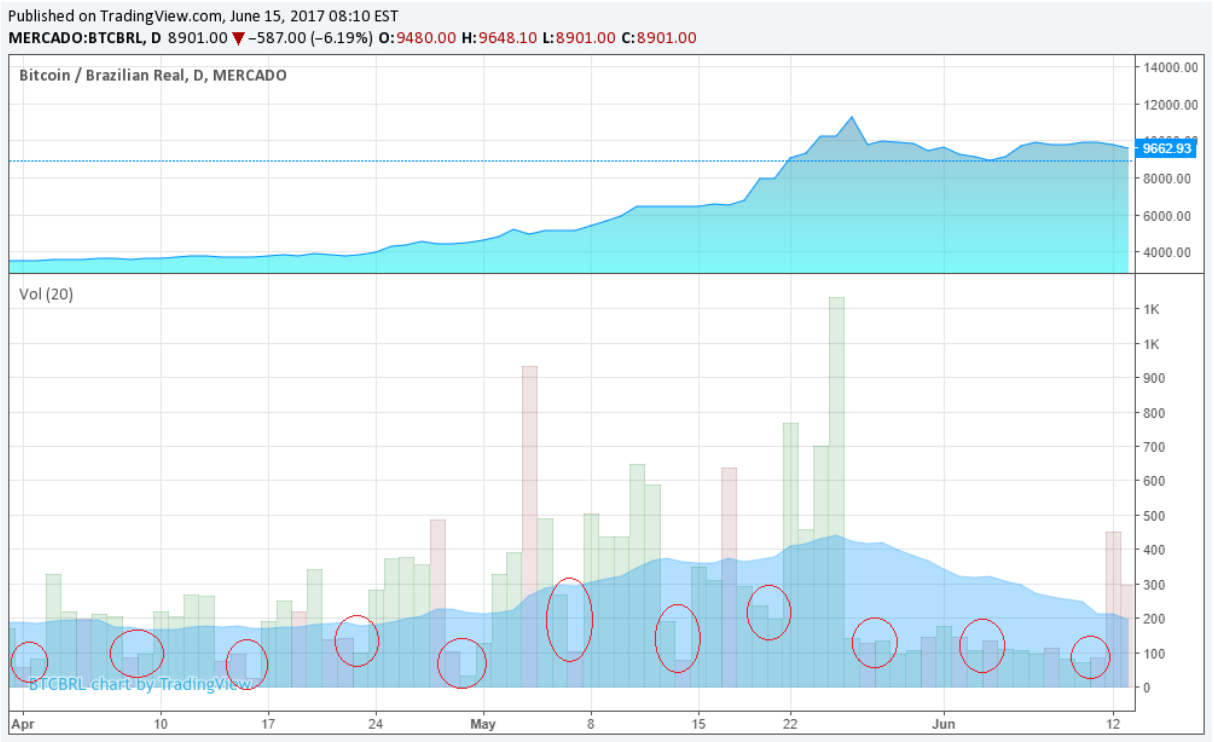


Figura 3.19: Redução no volume de transações nos fins de semana.

negociado nas datas referentes a fins de semana (sábado e domingo), a redução no volume não apresenta nenhuma relação visível com os preços. O comportamento apresenta uma possível relação com a cultura de reduzir a carga de trabalho nos fins de semana.

O mercado de bitcoin por dólar conta com algumas ferramentas de alavancagem que contribuem também para um aumento no volume de negociações, permitindo que o usuário realize compra ou venda de bitcoins com capital de investidores, o empréstimo de capital está sujeito a taxas de juros diárias e é supervisionado por protocolos de segurança na própria *exchange* de bitcoins, tais protocolos garantem o pagamento das dívidas criando um ambiente com segurança para captar investimos externos. Algumas bolsas de bitcoin que oferecem possibilidade de alavancagem são OKCoin e Bitfinex.

3.4.3 Taxas de transação

As taxas cobradas pela bolsa de bitcoins, no momento que os usuários realizam uma compra ou venda, são um dos pontos mais importantes no mercado de bitcoins. As taxas influenciam diretamente os lucros dos investidores e o volume negociado na bolsa. Algumas *exchanges* trabalham com um sistema de taxas que diferencia as ordens de compra e venda, envolvidas em cada transação, em dois tipos:

- **Ordem EXECUTADA ou MAKER:** é o tipo de ordem que antes de ser executada esteve presente no livro de ordens, esse tipo é essencial para preencher o livro de ordens, e contribui para a liquidez do mercado. Em algumas *exchanges*, esse tipo de ordem é beneficiada com a cobrança de taxa reduzida, no momento da sua execução.

- **Ordem EXECUTORA ou TAKER:** é o tipo de ordem que quando criada executa instantaneamente outra(s) ordens do livro e conclui uma transação, em algumas bolsas de bitcoins como Mercado Bitcoin, Foxbit e Bitfinex são cobradas taxas maiores para esse tipo de ordem. Em casos onde a ordem é executada parcialmente o montante restante fica no livro de ordens, e será tratado como ordem EXECUTADA caso seja realmente executada no futuro.

Uma prática diferenciada encontrada em mercados de bitcoin por dólar que ainda não foi vista em mercados de bitcoin por real é a taxa variável baseada em volume. Essa prática é vista em bolsas como Bitstamp, Bitfinex, OKCoin e outras bolsas, onde, baseado no volume mensal negociado pelo usuário, ele obtém redução nas taxas, a Tabela 3.1 apresentada anteriormente, mostra o sistema de taxas em Bitstamp, que cobra uma taxa igual para ordens *Maker* e *Taker*, a Tabela 3.2 apresenta a tabela de taxas em Bitfinex, com taxas também decrescentes baseadas no volume negociado nos últimos 30 dias, porém existe uma diferença entre ordens do tipo *Maker* e *Taker*. No caso da Bitfinex, podemos observar que é possível obter taxa 0 para ordens do tipo *Maker* que oferece ao investidor um grande diferencial em relação a outras *exchanges*.

O uso de robôs pode contribuir de forma significativa para um aumento no volume de transações mensal do usuário, resultando em melhores taxas de negociação para o investidor.

Volume em dólar negociado nos últimos 30 dias	Maker order*	Taker order**
$0 ou mais	0,10%	0,20%
$500.000 ou mais	0,08%	0,20%
$1.000.000 ou mais	0,06%	0,20%
$2.500.000 ou mais	0,04%	0,20%
$5.000.000 ou mais	0,02%	0,20%
$7.500.000 ou mais	0,00%	0,20%
$10.000.000 ou mais	0,00%	0,18%
$15.000.000 ou mais	0,00%	0,16%
$20.000.000 ou mais	0,00%	0,14%
$25.000.000 ou mais	0,00%	0,12%
$30.000.000 ou mais	0,00%	0,10%

* ordem executada

** ordem executora

Tabela 3.2: Taxas cobradas nas operações de compra e venda de moedas digitais na *exchange* Bitfinex

3.5 CONTRIBUIÇÕES DO ESTUDO DE MERCADO

Com o estudo do mercado foi possível apresentar diferenças entre o as *exchanges* citadas, identificamos alguns cenários onde o uso de robôs oferece benefícios ao investidor, também conseguimos detectar algumas tendências de queda no volume de bitcoins negociado diariamente no mercado brasileiro em finais de semana e durante as madrugadas que leva a uma dificuldade maior de negociar grandes volume nesses períodos. Também observamos uma relação entre *exchanges* que cobram taxas menores apresentarem um volume de negociação diário maior, e o volume maior indica uma maior liquidez no mercado. Sugerimos duas abordagens para criação de robôs para lidar com o problema da baixa liquidez.

No mercado de bitcoins por yuan Chinês, que passou por modificações no início de 2017, relacionadas às taxas de transações que eram de 0% foi modificada para 0,2%, levou a uma redução no volume de bitcoins negociados pelas respectivas *exchanges* que realizaram a mudança. Disso concluímos que taxas menores contribuem de forma efetiva para um maior volume de negociações na *exchange*.

4

CRIAÇÃO E USO DE ROBÔS

Neste capítulo vamos descrever o funcionamentos das APIs, e apresentar os resultados obtidos com relação ao uso das APIs e criação de robôs nas seguintes bolsas de bitcoins: Mercado Bitcoin e Bitfinex. Vamos descrever e discutir a implementação dos robôs e apresentar os resultados obtidos com sua execução que também serão discutidos.

4.1 CAPACIDADE DAS APIS

O funcionamento dos robôs é limitado pela capacidade das APIs usadas na sua criação. As APIs são fornecidas pelas bolsas de bitcoins, e têm alguns limites, um deles é a quantidade de requisições realizadas pelo cliente, cada bolsa tem limites diferentes que serão citados mais a frente, contudo os limites são bem agradáveis, na maioria dos casos é pelo menos 1 requisição por segundo, com risco de negação de serviço momentânea, ao usuário que exceder o limite. A segunda limitação das APIs está relacionada aos métodos disponíveis, como cada empresa tem sua própria API, a capacidade varia, e, no geral, empresas no exterior têm APIs mais robustas, com métodos mais sofisticados.

Exchanges de bitcoin disponibilizam APIs, com objetivo de permitir que usuários criem aplicações para operar no mercado criado por elas, a empresa Mercado Bitcoin descreve sua intenção com a seguinte citação a respeito da documentação de sua API, "Essa página é direcionada a desenvolvedores de software que desejem operar no Mercado Bitcoin de forma automatizada. A documentação abaixo descreve, com exemplos, como utilizar a interface de negociações (também chamada de TAPI, acrônimo de Trade API)"MercadoBitcoin (2016). Segundo a Bitstamp, sua API permite seus clientes acessar e controlar suas contas, usando softwares personalizados, e com respeito ao limite de requisições afirma 'Do not make more than 600 requests per 10 minutes or we will ban your IP address.'Bitstamp (2016).

Citado por Bovaird (2016), segundo Arthur Hayes, co-fundador e CEO da plataforma de negociação bitcoin alavancada Bitmex (2017) "Não existe uma API padrão para todas as trocas de câmbio digitais, e algumas trocas possuem melhores APIs do que outras", afirmou. "Isso significa que é necessário gastar muito tempo e energia, certificando-se de que a lógica de negociação pode lidar com falhas e calcular corretamente as métricas de risco do portfólio".

As APIs podem ser separadas em duas partes que são classificadas como API de dados, que contêm métodos públicos que podem ser acessados sem a necessidade de criar conta na *exchanges* e Trade API, que contêm métodos para acessar e manipular dados na conta do usuário.

4.1.1 Velocidade de acesso

A velocidade com que um robô realiza uma chamada de método, e obtém a resposta com os dados da *exchange*, é um ponto crítico. Em *exchanges*, onde o volume de bitcoins negociados diariamente é baixo, também percebemos que a velocidade com a qual são realizadas negociações também é baixa, ainda assim, é comum que observar mudanças no livro de ordens no intervalo entre uma consulta de preços e uma criação de ordem pelo robô. Em *exchanges* de bitcoin por dólar onde a velocidade das negociações é muito maior, o tempo necessário para um robô realizar duas chamadas é suficiente para ocorrer várias mudanças no livro de ordens. E uma mudança no livro de ordens pode implicar em mudança nos preços ao qual o robô irá comprar ou vender os bitcoins, na maioria dos casos a mudança no livro causa uma pequena variação nos preços e em casos raros pode causar mudanças relevantes no preço. Os riscos são maiores em casos onde o robô cria ordens do tipo "market", que são executadas instantaneamente no preço do livro de ordens seja ele qual for.

Mudanças no livro de ordens também podem afetar as taxas pagas pelo usuário, por exemplo se um robô é programado para em determinado momento criar uma ordem de compra, para o topo do livro, primeiro ele precisa consultar o valor da ordem atual no topo do livro, em seguida, criar uma ordem num preço superior, porém no intervalo entre a consulta e a criação da ordem outro usuário pode criar uma ordem de venda no valor que o nosso robô irá criar a ordem de compra e, portanto, nossa ordem de compra será considerada uma ordem executora, e algumas *exchanges* cobram uma taxa maior em ordens executoras.

4.1.2 API de dados

API de dados é constituída por métodos de acesso público que podem ser executados sem a necessidade de criação de conta na respectiva *exchange*. Tais métodos têm foco na manipulação de dados públicos como histórico de transações passadas, preços atuais e acesso a dados do livro de ordens. Vamos realizar uma apresentação das APIs em Mercado Bitcoin e Bitfinex, que representam bem o mercado de bitcoin por real e bitcoin por dólar respectivamente.

O Mercado Bitcoin até o momento negocia bitcoins por real, e também litcoin por real, até o momento não atua no mercado de bitcoin por litcoin diretamente, os métodos disponíveis na API podem consultar dados de um dos dois mercados a depender de parâmetros passados na chamada. Vamos apresentar os métodos para o mercado de bitcoin, a resposta das requisições é obtida no formato JSON. As chamadas de métodos foram implementada na linguagem de programação Python, versão 2.7.12, rodando em Windows10 de 64 bits. A documentação

completa disponibilizada pela própria *exchange* se encontra em MercadoBitcoin (2016). Os métodos implementados com exemplo de chamada estão disponíveis no APÊNDICE A:

A *exchange* Bitfinex negocia bitcoin e outras moedas digitais por dólar e também entre elas. Sua API de dados também chamada de REST PUBLIC ENDPOINTS oferece métodos para lidar com diversas criptomoedas, vamos tratar do uso desses métodos aplicados ao bitcoin. O formato de resposta também é JSON. A documentação completa disponibilizada pela própria *exchange* se encontra em Bitfinex (2016). Nossa implementação dos métodos com exemplos de chamadas está disponível no APÊNDICE B.

Os métodos públicos são muito semelhantes do ponto de vista de código, devido ao fato que não exigem muitos parâmetros, também podem ser executados digitando a URL direto do navegador, exemplo na Figura 4.1, um uso comum dos métodos públicos é para criar aplicações que usem os dados obtidos nas bolsas de bitcoins para criar relatórios, ou criar robôs que negociam em uma bolsa baseado em informações de variações de preços em outros mercados, mas também indispensável na criação de robôs que atuam em uma única *exchange*. Outras empresas como Foxbit, Bitstamp e OKCoin também oferecem métodos semelhantes. A documentação completa disponibilizada pela própria *exchange* se encontra em Bitfinex (2016).

Figura 4.1: Chamada do método candles de Bitfinex direto no navegador.

4.1.3 Trade API

A Trade API, também chamada de TAPI ou API de negociação, é constituída de métodos que manipulam informações de um usuário, para isso os métodos precisam de informações que identificam unicamente o usuário; (*chave_pública*, *chave_secreta*), o usuário precisa ter a conta na *exchange* para criar esse par, e, no momento da criação, definir as premiações de acesso que podem envolver leitura e manipulação de métodos que usem o saldo. Com as devidas chaves então o usuário pode criar aplicações que realizam requisições a TAPI. Dependendo da *exchange*, os passos para a criação das chaves podem ser mais complexos com o objetivo de proteger o usuário.

4.1.3.1 Comunicação com trade API do Mercado Bitcoin

Com as devidas chaves, podemos realizar a comunicação com a API, composta por uma requisição e uma resposta. O processo de comunicação com a TPAI no Mercado Bitcoin tem uma taxa limite de requisições, "é limitado por padrão ao máximo de 60 requisições a cada 60 segundos, por usuário e não por chave (os limites podem ser reduzidos). Em caso de exceder o limite, o usuário pode sofrer um bloqueio temporário, em casos de reincidência, bloqueios mais longos ou desativação da conta podem ser aplicados."MercadoBitcoin (2016). A documentação completa é encontrada em https://www.mercadobitcoin.com.br/trade-api/.

A estrutura da requisição tem como elementos principais: URL, parâmetros e cabeçalho.

- **URL:** É necessário fazer uma chamada HTTP, método POST, para;

  www.mercadobitcoin.net/tapi/<versão>/

- **Parâmetros:** Os dados são codificados no formato Form URL Encoded (Content-Type: application/x-www-form-urlencoded). Dois parâmetros são obrigatórios em todas as requisições. Exemplo em python, Código 4.1:

```
1 params = {
2     'tapi_method': 'list_orders',
3     'tapi_nonce': 1
4 }
5 #deve ser convertido para em:
6 tapi_method=list_orders&tapi_nonce=1
```

Código 4.1: Formato dos parâmetros para uso da Trade API em Mercado Bitcoin

- **Cabeçalho:** O cabeçalho é composto pelos itens:

  TAPI-ID: Chave pública.

  TAPI-MAC: Código MAC , utilizado para autenticar os dados da requisição. Com isso, garante a segurança da informação transmitida e assim evita ataques Man-In-The-Middle (MITM), uma referência sobre ataques MITM pode ser encontrada em Karapanos e Capkun (2014). A informação ou mensagem a ser autenticada é composta pelo path da requisição concatenado com o caractere ? (interrogação) e com a lista de parâmetros codificada no formato de código HMAC, Código 4.2. O MAC é assinado com a chave secreta do usuário, Código 4.3.

```
1 # Path:
2   /tapi/v3/
3
4 # Lista de parâmetros codificada no formato url-encoded:
5   tapi_method=list_orders&tapi_nonce=1
```

```
6
7  # Mensagem concatenada e formatada:
8    /tapi/v3/?tapi_method=list_orders&tapi_nonce=1
```

Código 4.2: Informação que será autenticada para uso da Trade API em Mercado Bitcoin

A mensagem então deve ser autenticada com algoritmo HMAC-SHA-512, usando a chave secreta do usuário.

```
1  import urllib
2  import hashlib
3  import hmac
4
5  # Constantes
6  MB_TAPI_SECRET = '<user_tapi_secret>'
7  REQUEST_PATH = '/tapi/v3/'
8
9  # Parâmetros (variam de acordo com o método)
10 params = {
11     # Do exemplo acima, 'list_orders'
12     'tapi_method': '<tapi_method>',
13     # Do exemplo acima, 1
14     'tapi_nonce': <next_nonce>
15 }
16
17 # Parâmetros formatados
18 # Utilizado no request
19 params = urllib.urlencode(params)
20 # Utilizado apenas para gerar o MAC
21 params_string = REQUEST_PATH + '?' + params
22
23 # Gerar MAC
24 H = hmac.new(MB_TAPI_SECRET, digestmod=hashlib.sha512)
25 H.update(params_string)
26 tapi_mac = H.hexdigest()
```

Código 4.3: Assinatura para uso da Trade API em Mercado Bitcoin

Vamos apresentar o código em Python para realizar a chamada de alguns métodos da Trade API no Mercado Bitcoin, com exemplos de resposta. Dados que contém informações pessoais que identifiquem o usuário não serão apresentados. A implementação dos métodos está disponível no APÊNDICE A.

Além dos métodos apresentados no APÊNDICE A existem outros, para lidar com saques(*withdraw*), que permite ao usuário realizar saques em reais para sua conta bancária e saques em bitcoins para outra carteira, tais métodos exigem mais cuidado com a segurança e

devem cumprir algumas exigências específicas que são claramente explicadas em MercadoBitcoin (2016), onde é encontrada a documentação completa da API. Os métodos acima são suficientes para criar um robô que lê o livro de ordens, histórico de transações passadas e realiza operações de compra e venda de forma automática. Contudo o uso de robôs é um método arriscado, que deve ser usado com certa supervisão para monitorar se o algoritmo implementado está sendo eficiente, pois o mercado de bitcoins muda constantemente.

4.1.3.2 Comunicação com trade API do Bitfinex

Para utilizar os métodos da Trade API o usuário precisa criar o par de chaves (*public_key*, *ecret_key*), o usuário pode ter várias chaves com permissões distintas, configuradas por ele, de acordo com a Tabela 4.1. O limite de requisição é contado a partir do IP que realiza a chamada, e é de 90 por 1 minuto, para cada método, os limites podem ser alterados a qualquer momento pela *exchange*, o usuário que exceder o limite é penalizado com negação de serviço momentânea.

PERMISSIONS	Leitura	Escrita
ACCOUNT INFO	Sim	Não
ACCOUNT HISTORY	Sim	Sim
ORDERS	Sim	Sim
MARGIN TRADING	Sim	Sim
MARGIN FUNDING	Sim	Sim
WALLETS	Sim	Sim
WITHDRAW	Não	Sim

Tabela 4.1: Configurações de permissão para acesso a Trade API da Bitfinex.

Vamos apresentar no APÊNDICE B, a implementação de diversos métodos em Python 2.7.12. Os exemplos de implementação serão realizados em uma conta privada, portanto informações de chaves e outras constantes que identificam o usuário não serão apresentadas. A documentação completa disponível para *exchange* está disponível em https://docs.bitfinex.com/v1/reference, a *exchange* Bitfinex está desenvolvendo a API versão 2 que está em fase beta no momento.

A estrutura da requisição tem como elementos principais: URL, parâmetros e cabeçalho. Os parâmetros especificam o método a ser chamado e os dados de entrada para execução do mesmo. O cabeçalho inclui uma assinatura digital com a chave secreta do usuário. Ambos podem ser escritos no seguinte formato:

- **URL:** É necessário fazer uma chamada HTTP, POST, para a URL;

  https://api.bitfinex.com/<versão>/<método>

- **Parâmetros:** Dois parâmetros são obrigatórios em todas as requisições, request e nonce, outros parâmetros serão adicionados de acordo com o método que deve ser executado. Exemplo em Código 4.4:

```
payloadObject = {
    'request':'/<versão>/<método>',
    'nonce':str(time.time())
}
payload_json = json.dumps(payloadObject)
```

Código 4.4: Parâmetros para uso da Trade API em Bitfinex

- **Cabeçalho:** O cabeçalho é composto pelos itens abaixo: Implementação em Código 4.5.

  X-BFX-APIKEY: Chave pública do usuário.

  X-BFX-PAYLOAD: Parâmetros no formato adequado (b64encode).

  X-BFX-SIGNATURE: Assinatura digital assinada com a chave secreta do usuário.

```
payload = base64.b64encode(bytes(payload_json))
m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
m = m.hexdigest()
headers = {
  'X-BFX-APIKEY' : bitfinexKey,
  'X-BFX-PAYLOAD' : payload,
  'X-BFX-SIGNATURE' : m
}
```

Código 4.5: Assinatura para uso da Trade API em Bitfinex

Os métodos disponíveis no APÊNDICE B, são métodos comuns para serem usados em um robô, outros métodos são oferecidos pela API da Bitfinex para lidar com informações do usuário, ordens, informações históricas e financiamentos. A Bitfinex tem alguns projetos de Open Source em andamento em diversas linguagens; Ruby, Go, Node.js, Python e C++, que são encontrados em https://bitfinex.readme.io/v1/docs/open-source-libraries.

4.2 CRIAÇÃO DE ROBÔS

Como destacado no artigo da CoinDesk Bovaird (2016), o uso de robô em mercados financeiros é algo consolidado, e no mercado de criptomoedas que foi criado a menos de uma década ainda está em desenvolvimento, e com o uso de robôs os investidores podem desenvolver algoritmos para operar nesse mercado que ainda está em fase inicial.

Citado por Bovaird (2016), qualquer pessoa pode participar de negociações com bot. Os comerciantes podem usar as soluções de prateleira, embora confiar em programas de software

pré-fabricados pode se tornar perigoso. Bovaird também aponta duas estratégias que são usadas por usuários de bots;

- **Arbitragem:** Compra de ativos em um mercado e, em seguida, vendê-los em outro por um preço mais alto, ganhando assim lucros com a diferença.
- **Fabricação de mercado:** Fornecer preços contínuos de compra e venda em uma variedade de moedas digitais spot e contratos de derivativos de moeda digital, em um esforço para capturar a amplitude entre o preço de compra e venda, segundo Arthur Hayes, citado por Bovaird (2016).

Segundo Redman (2017), "Bots de negociação ou negociação algorítmica é uma técnica que usa software pré-programado que analisa ações de mercado, como tempo, preço, pedidos e volume. Os bots ou a negociação com programas são usados em muitas bolsas de valores globais e é uma prática legal na maior parte. Os bots de negociação bitcoin dizem estabelecer uma negociação mais eficiente e podem ser utilizados em muitas e bem conhecidas *exchanges* de criptomoedas hoje. Existem bots que são gratuitos e podem ser baixados on-line. Algumas pessoas também projetaram seu próprio software de bot de negociação bitcoin. Também, há serviços de bot de negociação que são vendidos, por várias empresas de mecanismos de negociação e de programação".

- **Haasonline Software:** Empresa criada em janeiro de 2014. Dedicada a fornecer o software de negociação avançado e para negociar criptomoedas para clientes. Segundo a Haasonline, o software é o produto das sugestões de usuários, bem como a necessidade de recursos comerciais específicos, como arbitragem e bots de negociação. Eles afirmam que o software é bastante diversificado, que até mesmo comerciantes novatos podem utilizar e é poderoso o suficiente para comerciantes profissionais colherem os benefícios do Haasbot, citado por HaasOnline (2017).
- **BTC Robot:** O BTC Robot se considera o primeiro robô eletrônico de criptomoedas do mundo. O robô comercial oferece vários planos para diferentes tipos de software e associação. A empresa observa que sua negociação algorítmica não prevê mercados perfeitamente, e sempre haverá negociações com perdas e ganhos Redman (2017).
- **Cryptotrader:** É uma plataforma de negociação algorítmica para bitcoin e outras criptomoedas. "Nosso objetivo é fornecer aos comerciantes soluções de agentes de negociação automatizados, baseadas em nuvem, alimentadas por tecnologia de ponta"Cryptotrader (2017).

Segundo News (2017), OKCoin, uma das três maiores *exchanges* de bitcoins da China, estima que 60% de suas transações são executadas por comerciantes automatizados, enquanto as *exchanges* Huobi e a BTC China apresentam estimativa em torno de 80%.

4.2.1 Robôs no mercado de bitcoin por real

Vamos apresentar algumas sugestões para criação de robôs que possam atuar no mercado de bitcoins por real. O tipo mais simples de robô que não exige a criação de conta na *exchange* pode usar métodos da API de dados, para coletar dados das operações passadas executadas na *exchange* por outros usuários e também coletar dados do livro de ordens.

- **mb-r1:** Nosso primeiro robô apresentado para o Mercado Bitcoin vai realizar uma coleta de dados, usando os três métodos disponível pela API de dados, ticker, trades e orderbook. Esses métodos dão acesso ao resumo das últimas 24 horas, o histórico das últimas negociações realizadas e o estado do livro de ordens. Tais dados podem ser tratados para ajudar na tomada de decisões de compra e venda. Nosso robô vai apresentar o resumo das últimas 24 horas. Em seguida, baseado nas últimas 1000 negociações apresentar o menor preço, o maior preço, a diferença entre eles, o volume de bitcoins comprado, o volume de bitcoins vendido e o tempo, em horas, no qual ocorreram as negociações. E, por fim, baseado no livro de ordens apresentar o volume de ordens em aberto para compra de bitcoins, o volume de ordens em aberto para venda de bitcoins, o maior preço de ordem de compras aberta, o menor preço de ordem de venda aberta e por último o volume de ordens de compra em aberto nos preços abaixo de 20% do maior preço de compras, em Código 4.6.

```
1  def mb_r1():
2      while True:
3          try:
4              #obter resumo das últimas 24h
5              tick = ticker()
6              #obter dados das últimas ordens executadas
7              lista = trades()
8              t1 = lista[0].get('date')
9              t2 = lista[999].get('date')
10             dt = (t1-t2)/3600.0
11             buyvolume = sellvolume = 0
12             menorPreco = float("inf")
13             maiorPreco = 0.00
14             for x in lista:
15                 amount = x.get('amount')
16                 side = x.get('type')
17                 price = x.get('price')
18                 if(menorPreco > price): menorPreco = price
19                 if(maiorPreco < price): maiorPreco = price
20                 if(side == 'buy'):
21                     buyvolume += amount
22                 else:
```

```
23                      sellvolume += amount
24              #obter dados do livro de ordens
25              book = orderbook()
26              bids = book.get('bids')
27              asks = book.get('asks')
28              volumeBids = volumeAsks = 0
29              for x in bids:
30                  volumeBids += x[1]
31              for x in asks:
32                  volumeAsks += x[1]
33              #calcular volume de bitcoins no livro de compras no valor
    ↪  abaixo de 20% do valor de mercado
34              firstBidPrice = bids[0][0]
35              firstAskPrice = asks[0][0]
36              volumeBids20 = 0
37              for x in bids:
38                  if(x[0] < firstBidPrice*0.2):
39                      volumeBids20 += x[1]
40              print tick
41              print 'menor preco', menorPreco, 'maior preco', maiorPreco,
    ↪ 'diferenca', maiorPreco-menorPreco
42              print 'volume comprado', buyvolume, 'volume vendido',
    ↪  sellvolume, 'em :', round(dt, 2), 'horas'
43              print 'Volume de ordens em aberto no livro: bids', volumeBids,
    ↪ 'asks', volumeAsks
44              print 'maior preco de compra', firstBidPrice, 'menor preco de
    ↪  venda', firstAskPrice
45              print 'volume de compra em aberto abaixo de', firstBidPrice*0.2,
    ↪ '=', volumeBids20
46              time.sleep(30)
47          except Exception as erro:
48              print erro
49              pass
50              time.sleep(5)
```

Código 4.6: Robô para coleta de dados no Mercado Bitcoin

O programa Código 4.6, funciona em um *loop* infinito, com um *try-except* que engloba todo o código para evitar que erros inesperados interrompam a execução. Nosso código está apenas apresentando a informação através da saída padrão para imprimir na tela, porém os dados processados após a coleta podem ser tratados de acordo com a necessidade do usuário: salvos em arquivo ou armazenado em variáveis para serem lidos por outros métodos que executem outras funções. Na linha 5, obtemos o *tick* que contêm informações da *exchange* relativas as últimas 24 horas. Na linha 7, obtemos a lista com histórico das últimas 1000 negociações realizadas, o método *trades()* pode ser modificado para retornar a lista de ordens baseada em outros parâmetros como: intervalo de tempo ou intervalo de IDs. As linhas de 8 a 23 fazem o tratamento das

informações da *lista* para obter: o intervalo de tempo entre a primeira e última ordem da *lista*, os preços mínimo e máximo ao qual o bitcoin foi negociado, e os volumes de bitcoins comprados e vendidos, e a diferença entre o preço máximo e o mínimo. Na linha 25, obtemos o livro de ordens, e nas linhas de 26 a 39 fazemos a manipulação das informações em *book*. Os preços 'sell' e 'buy' na variavel *tick* representam os preços no topo das listas 'bids' e 'asks' do livro de ordens, porém é importante observar que o intervalo de tempo entre a chamada do método *ticker()* e *orderbook()* é suficiente para mudar o livro de ordem, essa diferença pode ser vista em Código 4.7, comparando as linhas 1 e 5, onde o maior preço de compra está diferente do valor de 'buy' da variável *tick*. Na linha 46, o programa fica em espera por 30 segundos, esse tempo de espera deve ser definido de acordo com a necessidade do usuário, para que o programa execute o mais rápido possível sem ultrapassar o limite máximo de requisições permitido pela API, no caso do Mercado Bitcoin não encontramos relatos sobre número máximo de requisições dos métodos da API de dados.

```
1    {u'ticker': {u'sell': 8294.99, u'buy': 8121.39883, u'last': 8294.99,
   ↪  u'vol': 321.64039929, u'high': 8700.00004, u'low': 8000.0, u'date':
   ↪  1499730073}}
2    menor preco 8000.0 maior preco 8579.99999 diferenca 579.99999
3    volume comprado 44.0800398 volume vendido 82.24785116 em : 3.7 horas
4    Volume de ordens em aberto no livro: bids 2584.77294079 asks
   ↪  477.75353292
5    maior preco de compra 8124.66515 menor preco de venda 8294.99
6    volume de compra em aberto abaixo de 1624.93303 = 2078.76705152
```

Código 4.7: Resposta obtida em uma execução do código mb_r1

Com respeito às informações apresentadas em Código 4.7, a partir das linhas 1 e 2, podemos concluir que o preço está próximo do menor preço registrado nas últimas 24 horas, e que esse valor foi registrado nas últimas 3,7 horas. Da linha 3, temos que o volume de bitcoins vendidos é maior que o volume de bitcoins comprados nas últimas 1000 negociações. Da linha 4, podemos notar que o volume de ordens de compra é maior que o volume de ordens de venda, porém na linha 6, vemos que grande parte do volume das ordens de compra são de ordens com valor abaixo de 20% do valor de mercado.

- **mb_r2:** Nossa segunda proposta é um robô com capacidade de coletar dados do mercado e criar ordens de compra e venda, sendo capaz de negociar automaticamente por períodos significativos na *exchange* Mercado Bitcoin. O robô funciona associado a uma conta de usuário, que precisa ter saldo disponível para executar operações de compra e venda. Ele coleta informações do livro de ordens, lista de ordens em aberto criadas pelo usuário, cancela ordens em aberto, coleta informações de saldos disponíveis e total do usuário e cria ordens de compra e venda para o livro. A

execução dos métodos é controlada por um algoritmo que determina o momento para criar ou cancelar ordens baseado nas informações coletadas do livro de ordens. A ideia para o algoritmo do robô **mb_r2**, é observar o livro de ordens para determinar o momento onde o preço de compra está distante do preço de venda, se a distância for suficiente para gerar lucro na operação de compra e venda, então criamos ordens de compra ou venda de acordo com o saldo disponível do usuário. Baseado no volume de ordens no livro, colocamos as ordens criadas próximas do topo da lista de ordens na *exchange*, quando ordens de outros usuários com volume significativo são criadas acima da nossa cancelamos nossa ordem para recriar outra ordem, em uma posição adequada que será determinada pelo próprio algoritmo. O robô funciona em *loop* infinito seguindo o algoritmo programado, e imprime algumas informações na tela relativas a criação, cancelamento de ordens e o valor estimado na conta do usuário, tais informações são uteis para o investidor realizar um acompanhamento das operações realizadas pelo robô Código 4.8.

```
1  def mb_r2():
2      volumeBase = 0.01
3      maxEstimativa = 0
4      while(True):
5          try:
6              book = list_orderbook()#1
7              bids = book.get('bids')
8              asks = book.get('asks')
9              precoCompra = float(bids[0].get('limit_price'))
10             precoVenda = float(asks[0].get('limit_price'))
11
12             volBids = 0
13             for x in bids:
14                 if(x.get('is_owner')): continue
15                 volBids += float(x.get('quantity'))
16                 if(volBids >= volumeBase):
17                     precoCompra = float(x.get('limit_price'))
18                     break
19             volAsks = 0
20             for x in asks:
21                 if(x.get('is_owner')): continue
22                 volAsks += float(x.get('quantity'))
23                 if(volAsks >= volumeBase):
24                     precoVenda = float(x.get('limit_price'))
25                     break
26
27             opemOrders = list_orders()#2
28             for x in opemOrders:
```

```
29                quanti = float(x.get('quantity'))
30                if(quanti >= 0.2): continue
31                side = x.get('order_type')
32                price = float(x.get('limit_price'))
33                orderId = x.get('order_id')
34                if(side == 1):
35                    if(price != precoCompra + 0.00001):
36                        order = cancelar(orderId)#3
37                        print 'Cancelar Compra preco:',
    ↪ order.get('limit_price')
38                else:
39                    if(price != (precoVenda - 0.00001)):
40                        order = cancelar(orderId)#4
41                        print '.....Cancelar Venda preco:',
    ↪ order.get('limit_price')
42
43            if(precoVenda/precoCompra < 1.006):
44                print 'ask e bid proximos'
45                continue
46
47            bal = get_account_info().get('balance')#5
48            balBrl = bal.get('brl')
49            balBtc = bal.get('btc')
50            freeReal = float(balBrl.get('available'))
51            freeBtc = float(balBtc.get('available'))
52
53            if(freeReal > 20):
54                precoCompra += 0.00001
55                buyamount = round(freeReal/precoCompra, 5)-0.00001
56                order = place_buy_order(buyamount, precoCompra)#6
57                print 'Criar Compra volume:', order.get('quantity'),
    ↪ 'preco', order.get('limit_price')
58            if(freeBtc > 0.002):
59                precoVenda -= 0.00001
60                order = place_sell_order(freeBtc, precoVenda)#7
61                print '.....Criar Venda volume:', order.get('quantity'),
    ↪ 'preco:', order.get('limit_price')
62
63            totalReal = float(balBrl.get('total'))
64            totalBtc = float(balBtc.get('total'))
65            estimativa = totalReal + totalBtc*precoVenda
66            if(maxEstimativa < estimativa):
67                maxEstimativa = estimativa
68            printData()
69            print 'BRL disponivel', freeReal, 'BTC disponivel', freeBtc,
    ↪ 'valor estimado da conta', round(estimativa,2), 'valor maximo
    ↪ registrado', round(maxEstimativa,2)
```

```
70
71          except Exception as erro:
72              print erro
73              pass
74              time.sleep(5)
```

Código 4.8: Robô que compra e vende bitcoins em Mercado Bitcoin

O programa Código 4.8, inicia na linha 2, criando a variável *volumeBase* que será usada no momento de definir o preço de compra e venda. Na linha 3, cria a variável *maxEstimativa* que é usada para armazenar uma estimativa do valor máximo da conta do usuário alcançado durante a execução do robô. As linhas 4 e 5 definem um *loob* infinito com o *try-except*, que contêm todo o código do programa, para evitar que o robô interrompa sua execução em casos de exceções inesperadas. Na linha 6, obtemos o livro de ordens através do método *list_orderbook()*. Nas linhas 7 e 8, extraímos as listas *bids* e *asks* do livro de ordens. Nas linhas 9 e 10, obtemos os preços de compra e venda do topo das listas. Nas linhas de 12 a 18, vamos identificar o preço de compra, percorrendo a lista *bids*, descartando ordens do próprio usuário e ordens cujo volume acumulado em bitcoins são inferiores ao valor de *volumeBase* definido na linha 2. Na linha 17, atualizamos *precoCompra* com um novo preço de compra, que pode ser um pouco abaixo do topo da lista de *bids*. Nas linhas 19 a 25, realizamos um processo similar para encontrar o novo valor de *precoVenda*. Na linha 27, obtemos a lista de ordens em aberto através do método *list_orders()*. Nas linhas de 28 a 41, comparamos os preços das ordens abertas com os preços de compra e venda sugeridos pelo nosso algoritmo. Nas linhas 36 e 40, cancelamos as ordens em aberto se houver alterações nos valores de *precoCompra* ou *precoVenda*. Nas linhas 29 e 30, cria-se uma condição para evitar que ordens criadas pelo usuário para outros fins sejam canceladas com a excussão desse robô. Nas linhas 43 a 45, verificamos se a diferença entre o preço de compra e o de venda é suficiente para gerar lucro, considerando que taxa paga ao Mercado Bitcoin para realizar uma compra seguida da venda varia de 0,6% a 1,4%, se a diferença de preço é baixa, reiniciamos a execução do *loop*. Na linha 47, obtemos os saldos na conta do usuário com o método *get_account_info().get('balance')*. Nas linhas 50 e 51, obtemos o saldo disponível em reais e bitcoins para as variáveis *freeReal* e *freeBtc* respectivamente. Nas linhas 53 a 57, se houver saldo disponível em reais suficiente para criar uma ordem compra, então incrementamos *precoCompra* com a menor fração possível para que nossa ordem de compra fique melhor posicionada no livro de ordens, e criamos a ordem no novo valor *precoCompra*, com o volume baseado no saldo disponível na conta do usuário. A linha 57 imprime dados sobre a ordem criada para fins de monitoramento do funcionamento do programa. Nas linhas 58 a 61, se houver saldo disponível em bitcoins suficiente para criar uma ordem de venda, realizamos o processo semelhante ao de criação da ordem de compra. Nas linhas 63 a 64, obtemos o saldo total em reais e bitcoins na conta do usuário para criar uma estimativa 'valor estimado da conta' e 'valor máximo registrado'. Nas linhas 68 e 69, imprimimos na tela o horário e os valores de *freeReal*, *freeBtc*, *estimativa* e *maxEstimativa* que foram obtidos ao longo da execução do *loop*.

As linhas 71 a 74 realizam a captura das exações geradas no *loop*, apresentam as informações na tela e reiniciam o *loop*.

```
1  horario 6 46 28
2  .....Cancelar Venda preco: 7944.90290
3  Cancelar Compra preco: 7826.60208
4  BRL disponivel 124.38566 BTC disponivel 0.02147888 valor estimado da conta
     ↪  295.03 valor maximo registrado 295.03
5  Criar Compra volume: 0.01588000  preco: 7826.60210
6  .....Criar Venda volume: 0.02147888 preco: 7944.89289
7  horario 6 46 40
8  .
9  .
10 .
11 horario 16 7 10
12 BRL disponivel 0.08823 BTC disponivel 0.0399797 valor estimado da conta
     ↪  327.52 valor maximo registrado 331.32
13 .....Cancelar Venda preco: 8189.95996
14 .....Criar Venda volume: 0.03997970 preco: 8189.95995
15 horario 16 7 18
```

Código 4.9: Resultado obtido com a execução do robô mb_r2

Código 4.9 apresenta dois trechos do arquivo de saída criado pela execução do robô **mb_-r2**, com um intervalo de aproximadamente 10 horas, onde foi registrado um aumento no valor da conta do usuário de 11% obtidos pela razão (327,52/295,03). Tais resultados foram possíveis devido a diversas circunstâncias entre elas: o estado de alta nos preços do bitcoin registrado no período, que foi de aproximadamente 4,7% obtidos pela razão (8189,95995/7826,60210), a baixa liquidez no mercado que permitia grandes variações nos preços durante o período, e o baixo volume de bitcoins manipulados pelo robô. Um aumento no volume de recursos para serem usados pelo robô implica em redução na porcentagem de ganho devido à baixa liquidez do mercado, porém a baixa liquidez é uma das condições que favorece esse tipo de estratégia usada pelo robô **mb_r2**.

- **mb_r3:** A proposta desse robô é coletar dados das transações passadas do usuário, que foram executadas totalmente ou parcialmente, para elaborar relatório sobre o desempenho da estratégia que foi usada durante o período de negociações. Iremos coletar e analisar dados das de transações executadas pelo robô **mb_r2**. O código para **mb_r3** é apresentado em Código 4.10, e é descrito na sequência.

```
1  def mb_r3():
2      lista = list_orders_with_fills()
3      qtd = len(lista)
4      t1 = float(lista[0].get('created_timestamp'))
5      t2 = float(lista[qtd-1].get('created_timestamp'))
6      lastPrice = float(lista[0].get('executed_price_avg'))
7      firstPrice = float(lista[qtd-1].get('executed_price_avg'))
8      qtdHoras = (t1-t2)/3600.0
9      maiorOrdem = 0
10     buyVol = buyPrice = buyMedia = 0
11     sellVol = sellPrice = sellMedia = 0
12     for x in lista:
13         vol = float(x.get('executed_quantity'))
14         precoMedio = float(x.get('executed_price_avg'))
15         if(vol*precoMedio > maiorOrdem): maiorOrdem = vol*precoMedio
16         tipo = x.get('order_type')
17         if(tipo == 1):
18             buyMedia += precoMedio*vol
19             buyVol += vol
20         if(tipo == 2):
21             sellMedia += precoMedio*vol
22             sellVol += vol
23     buyPrice = buyMedia/buyVol
24     sellPrice = sellMedia/sellVol
25     data1 = datetime.fromtimestamp(t1).isoformat()
26     data2 = datetime.fromtimestamp(t2).isoformat()
27     print 'Intervalo dos dados de', data2, 'ate', data1, qtdHoras, 'horas'
28     print 'Quantidade de ordens', qtd
29     print 'Volume de bitcoins comprados', buyVol
30     print 'Volume de bitcoins vendidos', sellVol
31     print 'Preco da primeira ordem no intervalo', firstPrice
32     print 'Preco da última ordem no intervalo', lastPrice
33     print 'Preco medio de compra', buyPrice
34     print 'Preco medio de venda', sellPrice
35     print 'Valor da maior ordem executada', maiorOrdem
36     razao = sellPrice/buyPrice
37     razao -= 1.006
38     print 'Lucro estimado', min(buyVol, sellVol)*(buyPrice)*razao
```

Código 4.10: Robô para criar relatório de transações executadas pelo usuário

Na linha 2, obtemos a lista de negociações passadas executadas que iremos analisar. O método *list_orders_with_fills()* é uma variação do Código 6, com a seguinte modificação nos parâmetros Código 4.11, que permite retornar uma lista de ordens com execução parcial ou total,

no intervalo de *from_id* até *to_id*. Nas linhas 4 e 5, obtemos as datas da primeira negociação e da última no intervalo de dados em *lista*. Nas linhas 6 e 7, obtemos os preços relativos à primeira e à última ordem em *lista*. Nas linhas de 10 a 24, realizamos o cálculo do volume de bitcoins comprado, o volume de bitcoins vendido, o valor da maior ordem negociada, a média de preços de compra e média de preços de venda. O preço médio é calculado através de média ponderada. Na linha 13, obtemos apenas o volume de bitcoins executado, onde o volume executado pode ser diferente do volume de bitcoins na ordem, em casos de execução parcial. Na linha 14, obtemos o preço médio de execução da ordem, pois uma ordem pode ser executada em mais de uma parte com preços distintos. Na linha 15, calculamos o valor da maior ordem executada. Na linha 16, obtemos o tipo da ordem que pode ser '1' equivalente a 'buy' ou '2' equivalente a 'sell'. Nas linhas 23 e 24, as variáveis *buyPrice* e *sellPrice* recebem o preço médio de compra e venda respectivamente. Nas linhas 25 e 26, convertemos o tempo para um formato legível. Na linha 27, apresentamos as informações relativas ao período de tempo dos dados. Na linha 28, apresenta a quantidade de ordens em *lista* executadas total ou parcialmente. As linhas 29 e 30 apresentam o volume de bitcoins comprados e vendidos. As linhas 31 e 32 apresentam o preço da primeira e última ordem executada. As linhas 33 e 34, mostram o preço médio de compra e venda. Na linha 35, apresentamos o valor da maior ordem executada no período. As linhas de 36 a 38 calculam e apresentam o valor do lucro aproximado no período de negociações, com o desconto das comissões cobradas pela *exchange* Mercado Bitcoin nas negociações de compra e venda, que é de 0,3% para cada operação.

```
1 params = {
2         'tapi_method': 'list_orders',
3         'tapi_nonce': str(int(time.time())),
4         'coin_pair': 'BRLBTC',
5         'status_list': '[3,4]', #open2, cancelled3, filed4
6         'has_fills': True,
7         'from_id': 37962000,
8         'to_id': 37993311
9     }
```

Código 4.11: Parâmetros para o método *list_orders_with_fills()* usado no robô **mb_r3**

Uma consideração a respeito do código em Código 4.10 é que a taxa cobrada por negociação pode ser de 0,7%, para o caso de ordem executora, apesar de nosso algoritmo em **mb_r2** tentar evitar a criação de ordem considera executora, porém é possível que mudanças rápidas no livro de ordens levem a criação indesejada de ordens que serão executoras. A variável *lista* contêm dados referentes às comissões pagas nas negociações e podem ser usados para calcular o lucro com maior exatidão. Vamos discutir o resultado obtido pela execução do robô **mb_r2**.

A partir dos dados em Código 4.12 podemos notar que em um intervalo de aproximadamente 21 horas, entre 2017-07-15T14:03:28 e 2017-07-16T10:59:16, foram registradas 200

ordens executadas. Nas linhas 7 e 8, podemos observar que o preço médio de compra e venda tem uma distância considerável o que vai implicar no lucro estimado apresentado na linha 10. A linha 9 apresenta o valor da maior ordem negociada pelo robô no intervalo. Temos que o lucro estimado representa aproximadamente 21% do valor da maior ordem criada pelo robô, onde o valor da maior ordem representa o saldo disponível na conta do usuário para negociar. Assim temos que o robô apresentou um lucro aproximado de 21% do capital investido em 21 horas. Além disso, temos que o preço da primeira ordem e superior ao preço da última ordem executada no intervalo, o que demostra que os preços estavam caindo, e uma estratégia de comprar e segurar os bitcoins teria levado ao prejuízo de aproximadamente 6,9%, nesse período. A seguir vamos apresentar resultados com intervalos de tempo maiores.

```
1  Intervalo dos dados de 2017-07-15T14:03:28 ate 2017-07-16T10:59:16 20.93
     ↪ horas
2  Quantidade de ordens 200
3  Volume de bitcoins comprados 2.91282624
4  Volume de bitcoins vendidos 2.90408776
5  Preco da primeira ordem no intervalo 6551.00001
6  Preco da ultima ordem no intervalo 6099.99899
7  Preco medio de compra 6352.8234341
8  Preco medio de venda 6421.60750089
9  Valor da maior ordem executada 423.067412496
10 Lucro estimado 89.0600257941
```

Código 4.12: Relatório obtido pela execução do robô **mb_r3**

A seguir apresentamos outro relatório da execução do robô **mb_r2** durante o período de aproximadamente 18 dias. Os dados em Código 4.13, mostram que durante o período entre a primeira e última ordem executada pelo robô, os preços aumentaram em aproximadamente 48,9%, esse seria o lucro obtido pelo investidor ao usar a estratégia de comprar e manter os bitcoins. Na linha 1, temos que o valor da maior ordem executada pelo robô nas primeiras 200 ordens a partir da data 2017-07-28T17:03:10, foi de 1225,13 reais, esse valor pode ser visto como uma aproximação do capital inicial disponível para o robô. Na linha 11, temos o lucro estimado de 1328,75 que representa um ganho de aproximadamente 108,4% no período. As linhas 4 e 5 apresentam o volume negociado pelo robô de aproximadamente 194,76 bitcoins. Nas linha 8 e 9, temos o preço médio de compra e de venda. E na linha 10, o valor da maior ordem executada pelo robô no período. Considerando a taxa de transação cobrada pelo Mercado Bitcoin de 0,3%, e o volume negociado pelo robô, estimamos que no período foram pagos mais de 6358,00 reais de taxas.

```
1  valor da maior ordem nas primeiras 200 ordens executadas 1225.13258318
2  Intervalo dos dados de 2017-07-28T17:03:10 ate 2017-08-15T17:55:36
     ↪ 432.873888889 horas
```

```
Quantidade de ordens 4000
Volume de bitcoins comprados 97.61347365
Volume de bitcoins vendidos 97.1542049
Preco da primeira ordem no intervalo 8899.99999
Preco da ultima ordem no intervalo 13256.00001
Preco medio de compra 10843.8316496
Preco medio de venda 10922.5714147
Valor da maior ordem executada 2079.11985413
Lucro estimado 1328.75622802
```

Código 4.13: Segundo relatório de negociações

No Código 4.14, temos outro modelo de relatório. Usamos uma versão modificada do Código 4.10 que executa separadamente para listas distintas, obtidas dos dados de histórico de ordens executadas pelo robô **mb_r2**. Ao longo do tempo, podemos notar que a estratégia apresenta lucro, porém em determinados intervalos encontramos resultados negativos, quanto ao lucro estimado, indicado que existe um risco de perdas em alguns cenários não favoráveis a execução desse robô. Apesar disso, ao longo do tempo, as perdas são compensadas com lucros maiores. Também notamos que, à medida que o saldo da conta aumenta, o desempenho do robô é prejudicado, devido a baixa liquidez no mercado. Em certo ponto, reduzir o capital disponível para o robô possibilita maiores lucros.

```
quantidade 200 ultimo order_id da lista 39527151
Intervalo dos dados de 2017-08-15T11:39:19 ate 2017-08-15T17:55:36
  ↪ 6.27138888889 horas
volume | preco de compra | preco de venda | maior ordem | lucro estimado
7.21239227 | 12953.0895473 | 13042.1678537 | 913.019366773 | 40.5136724053

quantidade 200 ultimo order_id da lista 39513371
Intervalo dos dados de 2017-08-15T06:52:13 ate 2017-08-15T11:38:36
  ↪ 4.77305555556 horas
volume | preco de compra | preco de venda | maior ordem | lucro estimado
6.57956875 | 12814.8228915 | 12935.7657629 | 862.109784255 | 143.901549834

quantidade 200 ultimo order_id da lista 39488571
Intervalo dos dados de 2017-08-14T19:10:42 ate 2017-08-15T06:50:27 11.6625
  ↪ horas
volume | preco de compra | preco de venda | maior ordem | lucro estimado
5.81955503 | 14349.3076563 | 14464.3639469 | 814.954501708 | 84.0614712319

quantidade 200 ultimo order_id da lista 39472544
Intervalo dos dados de 2017-08-14T12:43:16 ate 2017-08-14T19:10:20
  ↪ 6.45111111111 horas
volume | preco de compra | preco de venda | maior ordem | lucro estimado
5.32083899 | 14307.2160241 | 14399.2553049 | 739.329586739 | 16.4834814857
```

```
20
21 quantidade 200 ultimo order_id da lista 39430493
22 Intervalo dos dados de 2017-08-13T17:07:46 ate 2017-08-14T12:42:23
     ↪ 19.5769444444 horas
23 volume | preco de compra | preco de venda | maior ordem | lucro estimado
24 5.20227193 | 13546.9271085 | 13624.4357211 | 728.03930849 | -9.80041162873
25
26 quantidade 200 ultimo order_id da lista 39352876
27 Intervalo dos dados de 2017-08-12T11:02:10 ate 2017-08-13T17:06:06
     ↪ 30.0655555556 horas
28 volume | preco de compra | preco de venda | maior ordem | lucro estimado
29 9.58768568 | 12641.7872285 | 12729.9999522 | 2063.93082106 | 58.9128695441
30
31 quantidade 200 ultimo order_id da lista 39298664
32 Intervalo dos dados de 2017-08-11T16:18:28 ate 2017-08-12T11:01:44
     ↪ 18.7211111111 horas
33 volume | preco de compra | preco de venda | maior ordem | lucro estimado
34 12.40774876 | 12493.5279335 | 12546.1581366 | 2079.11985413 |
     ↪ -138.242359301
35
36 quantidade 200 ultimo order_id da lista 39250667
37 Intervalo dos dados de 2017-08-10T21:42:26 ate 2017-08-11T16:14:37
     ↪ 18.5363888889 horas
38 volume | preco de compra | preco de venda | maior ordem | lucro estimado
39 11.48962441 | 11812.0407025 | 11877.3168863 | 2042.58514869 |
     ↪ -31.9788806643
40
41 quantidade 200 ultimo order_id da lista 39224667
42 Intervalo dos dados de 2017-08-10T11:27:26 ate 2017-08-10T21:42:07
     ↪ 10.2447222222 horas
43 volume | preco de compra | preco de venda | maior ordem | lucro estimado
44 12.11336633 | 11270.3901732 | 11344.6449609 | 2012.53543843 | 39.7084289989
45
46 quantidade 200 ultimo order_id da lista 39173222
47 Intervalo dos dados de 2017-08-09T13:43:11 ate 2017-08-10T11:26:13
     ↪ 21.7172222222 horas
48 volume | preco de compra | preco de venda | maior ordem | lucro estimado
49 9.41371437 | 11112.6794555 | 11192.0964773 | 1969.98972814 | 59.1661620394
```

Código 4.14: Segundo relatório de negociações

4.2.2 Robôs no mercado de bitcoin por dólar

Para descrever a criação de robôs no mercado de bitcoin por dólar vamos apresentar um robô desenvolvido para a *exchange* Bitfinex. Usaremos uma técnica de análise de dados chamada Convergência e Divergência de Médias Móveis (MACD). Segundo Markets (2015), o MACD é

um estudo de análise técnica amplamente utilizado. MACD é efetivamente um indicador híbrido, que compreende elementos de impulso e análise de média móvel. O indicador usa duas médias móveis exponenciais, que ajudam a medir o impulso nos movimentos de preço. A linha MACD é a diferença entre essas duas médias móveis planejadas em relação a uma linha central. A linha central é o ponto em que as duas médias móveis são iguais. Junto com o MACD e a linha central, uma média móvel exponencial do próprio MACD pode ser plotada em gráfico para produzir a linha de sinal. A ideia por trás deste indicador é medir o impulso a curto prazo em relação ao impulso de longo prazo para ajudar a sinalizar a direção do impulso nos preços Markets (2015).

Segundo ADVFN (2017), "Um sinal de compra é gerado sempre que a linha MACD cruza para cima (no sentido ascendente) sua linha de sinalização. Quanto mais baixo, no campo negativo, ocorrer este cruzamento, ou seja, quanto mais afastadas abaixo da linha zero (eixo horizontal) estiverem as linhas, maior a expectativa de alta. A confirmação deste sinal de compra ocorre quando a linha MACD cruzar o eixo horizontal (zero), saindo da região negativa". O sinal de venda é gerado quando a linha MACD cruza no sentido descendente a linha de sinal, e o sinal de venda é confirmado quando a linha MACD cruza o eixo horizontal da região positiva para negativa.

Para criação do nosso robô vamos usar a API fornecida pela Bitfinex para coletar informações do gráfico de candles que contêm os dados para aplicação da análise MACD. A implementação do MACD consiste em calcular as listas: *ema12*, *ema26*, *macd*, *sinal* e *histograma*, onde *ema12*, *ema26* e *sinal* são MME's (Médias móveis exponenciais) com períodos de 12, 26 e 9 dias respectivamente, calculadas através do código Código 4.15. A lista *macd* é a diferença *ema12* - *ema26*. A lista *histograma* é obtida da diferença *macd* - *sinal*. Todos os dados relacionados ao indicador MACD são calculados com a chamada do método Código 4.16, que recebe como parâmetro a lista de preços de fechamento obtidas da *exchange* Bitfinex e o tamanho da lista.

```
1 def EMAn(precoFechamento, pastEMA, periodo):
2     return (precoFechamento*(2.0/(periodo+1)) +
  ↪  pastEMA*(1-(2.0/(periodo+1))))
```

Código 4.15: Média móvel exponencial

```
1 def buildHistograma(dados, tamanho):
2     global ema12
3     global ema26
4     global macd
5     global sinal
6     global histograma
7     ema12 = range(tamanho)
8     ema26 = range(tamanho)
```

```
9       macd = range(tamanho)
10      sinal = range(tamanho)
11      histograma = range(tamanho)
12
13      pos = 0
14      ema12[12] = mediaaritimetica(dados[1:13])
15      ema26[26] = mediaaritimetica(dados[1:27])
16      for x in dados:
17          if(pos > 12):
18              ema12[pos] = EMAn(float(dados[pos]), float(ema12[pos-1]), 12.0)
19          if(pos > 26):
20              ema26[pos] = EMAn(float(dados[pos]), float(ema26[pos-1]), 26.0)
21          if(pos >= 26):
22              macd[pos] = ema12[pos] - ema26[pos]
23          if(pos == 34):
24              sinal[pos] = mediaaritimetica(macd[26:35])
25              histograma[pos] = macd[pos]-sinal[pos]
26          if(pos > 34):
27              sinal[pos] = EMAn(float(macd[pos]), float(sinal[pos-1]), 9.0)
28              histograma[pos] = macd[pos]-sinal[pos]
29          pos += 1
```

Código 4.16: Constrói uma lista que representa o histograma da análise MACD

O método em Código 4.16 calcula as listas que são usadas na construção do indicador MACD. As linhas 2 a 6 referência listas globais para serem usadas por outros métodos do programa, e as linhas de 7 a 11 iniciam as listas. A lista *ema12* tem seu primeiro elemento válido na posição 12, obtido da média aritmética de 12 períodos de *dados* na linha 14, os elementos seguintes são calculados na linha 18, pela função **EMAn**. A lista *ema26* tem seu primeiro elemento válido na posição 26, obtido da média aritmética de 26 períodos de *dados* na linha 15, os elementos seguintes são calculados na linha 20, pela função **EMAn**. A lista *macd* tem seus elementos válidos a partir da posição 26, calculados na linha 22, obtido da diferença dos elementos em *ema12* e *ema26*. A lista *sinal* tem seu primeiro elemento válido na posição 34, calculado pela média aritmética de 9 períodos de *macd*. Na linha 24, os elementos seguintes são calculados na linha 27, pela função **EMAn**. A lista *histograma* tem seu primeiro elemento válido na posição 34, calculado pela diferença entre os elementos de *macd* e *sinal* na linha 25, os elementos seguintes são calculados na linha 28. A função *mediaaritimetica*, usada nas linhas 14 e 15, retorna a média aritmética dos valores na lista passada como parâmetro.

- **bf_r1:** A ideia para nosso robô é realizar uma coleta de dados da *exchange* Bitfinex, em seguida, realizar uma simulação baseada no indicador MACD para decidir o momento de compra e venda. Para o funcionamento do robô não é necessária a criação de conta na Bitfinex, porque usaremos apenas o método público *candles* para coletar informações dos preços de fechamento no período desejado. A ideia

do algoritmo executado pelo robô é realizar a coleta de dados, criar as listas que representam o histograma e o macd, e realizar simulação de compra e venda nos momentos onde os indicadores atingem o estado que confirmam o sinal de compra ou sinal de venda descrito em ADVFN (2017). E, por fim, apresentar alguns dados obtidos na simulação em Código 4.17.

```
def bf_r1():
    dolar = 1000
    bit = 0
    price = 0
    qtdoperacoes = 0
    qtd = 250*4
    candle = candles('6h', qtd, 'BTCUSD') #time, open, close, high, low,
  ↪  vol
    candle.reverse()

    dados = []
    for x in candle:
        dados.append(float(x[2]))
    buildHistograma(dados, qtd)

    pos = 0
    for x in candle:
        if(pos < 36):
            pos += 1
            continue
        datahora = datetime.fromtimestamp(float(x[0]/1000))
        price = float(x[2])
        histo = histograma[pos]
        vmacd = macd[pos]


        if(histo > 0 and vmacd > 0):
            if(dolar > 0):
                gasto = dolar
                dolar -= gasto
                bit += gasto/price
                bit*= 0.998
                print 'Comprar', datahora, 'BTC', bit, 'Preco', price
                qtdoperacoes += 1
        if(histo < 0 and vmacd < 0):
            if(bit > 0 ):
                gasto = bit
                bit -= gasto
                dolar += gasto*price
```

```
39                dolar*= 0.998
40                print '......vender', datahora, 'USD', dolar, 'Preco',
  ↪  price
41                qtdoperacoes += 1
42
43          pos += 1
44      datainicial = datetime.fromtimestamp(float(candle[1][0]/1000))
45      datafinal = datetime.fromtimestamp(float(candle[qtd-1][0]/1000))
46      print 'Periodo dos dados: ', datainicial, datafinal
47      print 'valor estimado da conta' , dolar + bit*price, 'quantidade de
  ↪  trocas', qtdoperacoes, 'preco', price
```

Código 4.17: Robô para realizar simulação de compra e venda utilizando o indicador MACD, na *exchange* Bitfinex

O programa Código 4.17, define os valores iniciais de USD e BTC para nossa simulação nas linhas 2 e 3. Na linha 6, criamos a variável *qtd* que representa a quantidade de candles que serão coletados, os dados coletados são referentes a 250 dias, com intervalos de 6h totalizando 1000 candles. Na linha 7, realizamos a coleta dos dados para a variável *candle*, que na linha 8 tem a ordem dos elementos invertida. Na linha 10, criamos a variável *dados*. Nas linhas 11 e 12, extraímos os preços de fechamentos de *candle* para *dados*. O valor na posição zero da lista de *dados* não será usado. Na linha 13, realizamos a chamada da função **buildHistograma**, passando como parâmetro a lista de dados com seu respetivo tamanho, essa função calcula os valores das listas *macd* e *histograma* que serão usados para tomada de decisão de compra e venda. Na linha 16, iniciamos a interação na lista *candle*. As linhas 17 a 19 interrompem a simulação antes da posição 36, porque não existem dados relativos ao histograma nesse período, devido ao algoritmo de criação do indicador MACD. Na linha 20, a variável *datahora* obtém a data do início do período relativo ao 'candle' daquela interação. As linhas 21 a 23 obtêm respectivamente o preço de fechamento, o valor do *histograma* e o valor do *macd*. As linhas 26 a 33 realizam a simulação de compra se os valores em *histo* e *vmacd* são positivos e existe saldo em *dolar*, então simulamos a compra ao preço de fechamento *price*. Na linha 31 a simulação do pagamento da taxa de transação de 0,2%. A linha 32 imprime as informações relativas à compra, na tela para relatório. As linhas de 34 a 41 realizam a simulação da venda, que é um processo semelhante ao de compra. Nas linhas 44 e 45, obtemos o valor que representa as datas do intervalo de dados usados no experimento, a data representa o início do período de um 'candle' e difere em 6 horas da data que registra o preço de fechamento. As linhas 46 e 47, imprimem os dados relativos ao período no qual os dados foram coletados, valor estimado da conta, quantidade de negociações e o preço do bitcoin.

```
1 Comprar 2016-11-17 21:00:00 BTC 1.34208332213 Preco 743.62
2 ......vender 2016-11-24 21:00:00 USD 982.301946639 Preco 733.39
3 Comprar 2016-11-30 09:00:00 BTC 1.32417178965 Preco 740.34
4 ......vender 2017-01-06 09:00:00 USD 1205.89014454 Preco 912.5
```

```
5  Comprar 2017-01-17 09:00:00 BTC 1.33868561095 Preco 899.0
6  ......vender 2017-02-10 03:00:00 USD 1295.91463246 Preco 969.99
7  Comprar 2017-02-15 09:00:00 BTC 1.27811325545 Preco 1011.9
8  ......vender 2017-03-08 03:00:00 USD 1539.59733393 Preco 1207.0
9  Comprar 2017-03-13 09:00:00 BTC 1.24444653702 Preco 1234.7
10 ......vender 2017-03-16 21:00:00 USD 1414.58975646 Preco 1139.0
11 Comprar 2017-03-28 15:00:00 BTC 1.35135500808 Preco 1044.7
12 ......vender 2017-05-27 09:00:00 USD 2714.56734555 Preco 2012.8
13 Comprar 2017-05-30 03:00:00 BTC 1.20905887038 Preco 2240.7
14 ......vender 2017-06-12 21:00:00 USD 3146.67775473 Preco 2607.8
15 Comprar 2017-06-19 21:00:00 BTC 1.2102606749 Preco 2594.8
16 ......vender 2017-06-24 21:00:00 USD 3050.88344385 Preco 2525.9
17 Comprar 2017-07-03 21:00:00 BTC 1.16711962472 Preco 2608.8
18 ......vender 2017-07-08 03:00:00 USD 2867.46866194 Preco 2461.8
19 Periodo dos dados:  2016-11-09 03:00:00 2017-07-16 15:00:00
20 valor estimado da conta 2867.46866194 quantidade de trocas 18 preco 1865.7
```

Código 4.18: Relatório de execução do robô bf_r1, na *exchange* Bitfinex

A respeito da execução do robô **bf_r1** e seu resultado apresentado em Código 4.18, usamos 1000 intervalos de 6 horas, porém a simulação das negociações de compra e venda só foram iniciadas a partir do 37º período, devido à necessidade de dados para calcular a lista do histograma. Realizamos tentativas de simulação para intervalos de 1 dia, 12 horas, 6 horas, 3 horas, 1 hora, 30 minutos, 15 minuto e 5 minutos. O melhor resultado foi obtido com o intervalo de 6 horas. O último 'candle' dos 1000 não tem o intervalo de 6 horas, devido ao horário que o método *candle* foi executado, não foi necessário remover esse 'candle', porque ele não teve participação nas negociações. A variação no preço do bitcoin entre a primeira compra e a última venda, linhas 1 e 18, foi de 1718,18 (2461,80 - 743,62), isso representa 231% de aumento no preço do bitcoin, enquanto que o valor da na conta do usuário passou de 1000 para 2867,47, um aumento de 186,75%, isso significa que se o usuário tivesse mantido os bitcoins da primeira compra até a data da última venda teria um lucro maior. No entanto, considerando o período da primeira compra até final do intervalo de dados a diferença nos preços, linhas 1 e 20, foi de 1122,08 (1865,70 - 743,62), que representa 151,9% de aumento no preço, nesse caso o aumento no valor da conta continua de 186,75%, e nesse caso, o lucro usando o indicador MACD é maior.

Vamos destacar 3 operações que têm o maior impacto nos resultados da simulação Código 4.18. Nas linhas 11 e 12, foi registrado o maior lucro da simulação, justamente porque o bitcoin quase dobrou de valor sem apresentar quedas significativas o suficiente para o algoritmo atingir a condição de venda. A terceira operação que queremos destacar é na linha 18, que foi a venda responsável por evitar a perda, em uma das maiores quedas no preço registradas no período da simulação. Observando os preços de venda em relação ao preço da próxima compra, também podemos notar que diversas vezes o preço aumentou sinalizando que a estratégia falha em detectar o ponto de venda adequado, as mesmas falhas também ocorrem nos pontos de compra algumas vezes.

Também é importante considerar que o preço simulado é diferente do preço ao qual o usuário iria comprar ou vender uma ordem real. Por exemplo, no momento do fechamento de um 'candle', o preço de fechamento é o preço registrado da última negociação realizada, se o usuário vai fazer uma compra em seguida, o preço de venda, na lista de 'asks' do livro de ordens, pode ser maior, igual ou até menor, porém a possibilidade de ser menor ou igual é muito baixa. Concluímos que o uso dessa estratégia apresenta possibilidade de lucro, porém com grandes riscos.

4.3 OUTROS CENÁRIOS PARA USO DE ROBÔ

4.3.1 Flash Crash

Como consequência de uma das maiores e mais rápidas variações de preços de uma criptomoeda, houve uma paralisação nas negociações de ether por dólar (EHT-USD) na *exchange* Gdax. Segundo relatos de White (2017), em 21 de junho de 2017, uma ordem de venda de milhões de dólares em ether foi colocada para ser realizada a preço de mercado (tipo de ordem que vende no melhor preço possível instantaneamente), no mercado de ETH-USD. Esse evento resultou em uma queda no valor do ETH de USD 317,81 para USD 224,48, devido a execução de todas as ordens do livro de compras em valores acima de USD 224,48, uma queda de 29,4% no preço de compra. Tal redução no preço deu início a execução de aproximadamente 800 ordens de *stop loss* (ordens de venda programadas para executar automaticamente se o preço cair abaixo de um valor definido) e *margin funding liquidations* (ordens são executadas automaticamente a fim de adquirir fundos para o usuário liquidar financiamentos existentes). Como consequência da execução automática desse grande volume de vendas, o preço do ETH caiu para USD 0,10 Tech (2017).

No caso da Gdax, fica visível que ethers foram vendidos num valor abaixo de 1% do valor real da moeda, e quem realizou venda nesses valores sofreu uma perda de mais de 99% do capital mantido naquela moeda. Vamos assumir que algumas ordens de *stop loss*, que deveriam ser executadas com valores acima de 50%, foram executadas por valores menores devido a falta de liquidez do mercado, porém se essas ordens não fossem todas executadas em sequência seria possível uma recuperação do mercado para a execução das ordens em valores adequados. Se considerarmos que o preço do ether em outras bolsas que negociam ether por dólar, não apresentou queda significativa, podemos concluir que as ordens de *stop loss* embora tenham executado normalmente não cumpriram o papel de proteger o investidor de perdas maiores, mas foram, de fato, a causa de perdas extremamente altas.

Com base no cenário ocorrido na exchage Gdax, temos uma proposta de robô para atuar em cenários similares. Vamos apresentar à implementação do robô para atuar na echange Bitfinex, que negocia bitcoin, ether e outras criptomoedas e também tem um forte mercado com opções de financiamento.

- **stopLossLimited:** Simular um *stop loss*, limitando o valor de venda com um valor mínimo, o qual seria desnecessário vender abaixo daquele valor. Por exemplo, uma moeda comprada a 100, se cair abaixo de 90 pode ser vendida para evitar perdas maiores, porém se a queda for tão rápida que o valor chegou abaixo de 20, não vale a pena realizar a venda. Implementação disponível em Código 4.19.

```
1  def stopLossLimited(symbol, currency, wallet, sellprice, sellpricelimit):
2      orderId = 0
3      while True:
4          try:
5              status = orderStatus(orderId)
6              if(status.has_key('is_live') and status.get('is_live')):
7                  cancel(orderId)
8                  continue
9              balance = balances() #confirmar saldo
10             amount = 0
11             for x in balance:
12                 if (x.get('currency') == currency and x.get('type') ==
   ↪  wallet):
13                     amount = x.get('available')
14                     break
15             if(amount == 0): return 0
16             book = orderbook(symbol)
17             bids = book.get('bids')
18             buyprice = float(bids[0].get('price'))
19             if(buyprice <= sellprice):
20                 order = newOrder(amount, sellpricelimit, 'sell')
21                 orderId = order.get('order_id')
22         except Exception as erro:
23             print erro
24             time.sleep(5)
25             pass
```

Código 4.19: Robô para stop loss limited

O programa Código 4.19 na linha 1, recebe como parâmetro o simbolo do mercado que vai atuar *symbol*, a moeda que vai monitorar *currency*, o nome da carteira onde a criptomoeda está armazenada *wallet*, o preço base que o robô ira monitorar *sellprice*, para iniciar o processo de venda, caso o valor do bitcoin no livro de ordens de compra se torne menor ou igual e, por fim, o preço mínimo *sellpricelimit* ao qual a criptomoeda pode ser vendida. Na linha 2, inicia a variável *orderId* que será usada para armazenar um identificador da ordem de venda caso ela seja criada. As linhas 3 e 4 definem um *loop* infinito com um *try-except* que englobam todo o código que manipula a conta do usuário. As linha de 5 a 8, verificam se uma ordem criada anteriormente

pelo robô ainda esta ativa e a cancela caso exista. As linhas de 9 a 14, obtém o saldo da moeda monitorada e na linha 15, se o saldo é igual a zero encera a execução do robô. As linhas de 16 a 18, obtém o preço de compra mais alto no livro de ordens. A linha 19 verifica se o preço de compra no livro de ordens é inferior ou igual ao preço *sellprice*. A linha 20 cria uma ordem de venda no valor *sellpricelimit*, que sera executada caso existam ordens de compra no livro com valor igual ou superior a *sellpricelimit*. A linha 21 armazena o identificador da ordem criada na variável *orderId*, para que a ordem seja cancelada em uma próxima execução do *loop* caso não tenha sido totalmente executada. As linhas 22 a 25, tratam exceções geradas na execução das chamadas a API, o tratamento consiste em imprimir a mensagem de erro e manter o programa em estado de espera por 5 segundos antes de reiniciar o *loop*.

Para a sugestão do robô **stopLossLimited** (Código 4.19), queremos deixar claro que existem diversas formas de melhorar o código, para tornar o robô mais eficiente ou até monitorar diversas moedas em paralelo. Para replicar a ideia em outras *exchanges*, é preciso observar as diferenças nas funções oferecidas pela API, o que pode levar a implementações distintas, o principal objetivo aqui é mostrar a viabilidade de se construir um robô para executar automaticamente. Segundo Metz (2017), Martin Froehler diz que "Independente do método que você usa nas finanças quantitativas - seja ele aprendizado de máquina ou métodos tradicionais quantitativos - há um número infinito de maneiras de falhar". Assim, como o caso do *flash crash* do ether na *exchange* Gdax era inesperado para muitos, também devem acontecer situações inesperadas para usuários de robôs que podem levar a falhas. Segundo Rowlands (2014), que cita outro caso de *flash crash* ocorrido em 2010, e afirma que, robôs atuando no mercado podem contribuir para quedas maiores, em casos onde a queda inicial nos preços leva o robô a realizar vendas, que vai derrubar ainda mais os preços levando possivelmente ouros robôs a realizarem vendas em instantes.

4.4 RESULTADOS

Como resultado do estudo das APIs foi possível implementar e testar diversos métodos que estão disponíveis nos Apêndices A e B, vários deles foram usados na elaboração dos robôs.

Apresentamos três robôs para o mercado de bitcoin por real, e um para o mercado de bitcoins por dólar como resultados de nossos experimentos de criação de robôs.

A partir relatório gerado pelo robô **mb_r1** concluímos que existe um grande volume de ordens de compra a preços muito baixos, esse tipo de ordem tem a possibilidade de ser executada em momentos de grande queda nos preços.

Conseguimos obter resultados positivos com o uso do robô **mb_r2**, na *exchange* Mercado Bitcoin, onde um de nossos objetivos era demostrar a viabilidade de criar robôs para executar de forma autônoma em *exchanges* de bitcoin. Nosso robô também contribui de forma significativa para aumentar a liquidez na *exchange* devido ao volume negociado criando ordens para o livro. Concluímos que o uso de robôs com capacidade de negociar contribui de forma positiva para

exchanges de bitcoin. Nosso robô **mb_r2**, foi capaz de explorar uma característica momentânea do mercado de bitcoins, para realizar negociações de forma autônoma, e obteve resultados com lucros a curto prazo.

O robô **mb_r3** foi criado como uma ferramenta para medir o desempenho do robô **mb_r2**, através dos relatórios criados por ele foi possível identificar os lucros superiores a 100% em 18 dias, e também que as operações executadas no período resultaram em pagamento de taxas 4,78 vezes maior que o lucro. Concluindo que além dos lucros ao investidor o uso do robô proporcionou lucros ainda maiores para a *exchange*.

Para o mercado de bitcoin por dólar criamos o robô **bf_r1** que realizou simulações de compra e venda na *exchange* bitfinex, as simulações não apresentaram resultados satisfatórios para justificar a criação de um robô com capacidade de negociar em ambiente real. Também apresentamos uma sugestão de robô para atuar monitorando a conta do usuário na tentativa de reduzir perdas em uma situação de *flash crash*.

5

CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

Ao longo de 2015 a 2017, alguns robôs com estratégias distintas em seus algoritmos foram implementados, e usados em *exchanges* como Foxbit, Mercado Bitcoin, Bitstamp e Bitfinex, em todos os casos o robô funcionou da forma esperada. Os robôs podem ser usados para gerar ganho de capital, porém devido à natureza das variações nos preços, dificilmente previsíveis, o uso para esse fim apresenta riscos elevados assim como o potencial de ganho.

Ao longo de nossa pesquisa em *exchanges* de bitcoins, identificamos que o ambiente de negociação apresenta comportamento diferenciado em *exchanges* distintas. Com isso, concluímos que o uso de robôs para realizar comércio, deve ser personalizado para cada ambiente, e normalmente as estratégias usadas necessitam de atualização com frequência, para se adaptar às mudanças de comportamento na própria *exchange*, também são frequente, casos onde o mesmo algoritmo pode gerar lucro num período e prejuízo em outro período muito próximo, devido a mudanças bruscas no comportamento do mercado. O uso da mesma estratégia por muitos usuários no mesmo mercado pode inviabilizar os lucros, devido a concorrência criada.

O uso de robôs em bolsas de bitcoins é viável, e de acordo com o objetivo do robô e a estratégia usada na criação do algoritmo usado pelo robô, é possível aumentar os lucros do investidor usuário de robôs.

Robôs que realizam operações de compra ou venda, quando executando podem sofrer com variação nos preços no intervalos entre uma consulta de preços e a criação da ordem, o que leva a pagamentos de taxas maiores em alguns casos de *exchanges* que cobram valores diferentes para ordens executora ou executada. A depender da implementação do robô, e os métodos usados, as ordens criadas pelos robôs podem executar a preços diferentes do preço desejado, devido a mudanças extremamente rápidas no livro de ordens. Segundo Rowlands (2014), à medida que a negociação automatizada se tornou mais predominante e competitiva, a latência e a proximidade com a *exchange* se tornaram cada vez mais importantes, com os negócios que ocorrem tão rapidamente quanto em milissegundos. A velocidade com que os dados são recebidos da *exchange* precisa ser tão baixa quanto possível.

O bitcoin tem um auto potencial para continuar crescendo, e deve continuar aumentando

seu valor devido a oferta limitada pela mineração e a demanda crescente que tem sido vista. E com o passar do tempo as atividades nas *exchanges* devem se tornar cada vez mais intensas. Já existe uma tendência de implantação de taxas nas transações de bitcoins que envolvem o *blockchain* devido a limitação da criação de moedas na mineração, isso deve afetar as políticas de taxas nas *exchanges*.

5.2 TRABALHOS FUTUROS

Como não foi possível desenvolver ao longo de nossa pesquisa robôs para realizar arbitragens entre *exchanges* de bitcoins esse é um objetivo para trabalhos futuros, pesquisar métodos de aprendizagem de máquina para criar robôs com capacidade de negociar volume maiores de capital. Desenvolver nosso experimento de criação de robôs para o mercado de bitcoins por altcoins. Buscar desenvolver aplicações que usem robôs para incentivar o uso de bitcoins na sociedade. Pesquisar a viabilidade de criar uma *exchange* de bitcoins.

REFERÊNCIAS

ADVFN. Análise técnica: Macd. 2017. Disponível em: <https://br.advfn.com/educacional/analise-tecnica/macd>. Acesso em: 22 outubro 2017.

BADEV, A.; CHEN, M. Bitcoin: Technical background and data analysis. 2014. Disponível em: <https://www.federalreserve.gov/econresdata/feds/2014/files/2014104pap.pdf>. Acesso em: 22 outubro 2017.

BELL, T. Bitcoin trading agents. 2015. Disponível em: <http://www.tomjbell.co.uk//wp-content/uploads/2015/05/BitcoinTradingAgents.pdf>. Acesso em: 22 outubro 2017.

BITFINEX. Bitfinex: Bolsa de bitcoin online. 2016. Disponível em: <https://www.bitfinex.com>. Acesso em: 22 outubro 2017.

BITMEX. Bitmex: Bolsa de bitcoin online. 2017. Disponível em: <https://www.bitmex.com>. Acesso em: 22 outubro 2017.

BITSTAMP. Bitstamp: Bolsa de bitcoin online. 2016. Disponível em: <https://www.bitstamp.net>. Acesso em: 22 outubro 2017.

BITVALOR. Bitvalor: Cotação do bitcoin em mercados brasileiros. 2016. Disponível em: <https://bitvalor.com/>. Acesso em: 22 outubro 2017.

BOVAIRD, C. How bots are fueling high-speed bitcoin trading. 2016. Disponível em: <http://www.coindesk.com/how-bots-are-fueling-high-speed-bitcoin-trading/>. Acesso em: 22 outubro 2017.

COINMARKETCAP. Cryptocurrency market capitalizations. 2016. Disponível em: <https://coinmarketcap.com/>. Acesso em: 22 outubro 2017.

CRYPTOTRADER. Cryptotrader: Fornecedora de software para negociação de cripto-moedas. 2017. Disponível em: <https://cryptotrader.org/>. Acesso em: 22 outubro 2017.

FOXBIT. Foxbit: Bolsa de bitcoin online. 2016. Disponível em: <https://foxbit.com.br>. Acesso em: 22 outubro 2017.

GARCIA, D. et al. The digital traces of bubbles: feedback cycles between socio-economic signals in the bitcoin economy. 2014. Disponível em: <https://arxiv.org/abs/1408.1494>. Acesso em: 22 outubro 2017.

HAASONLINE. Haasonline software: Fornecedora de software para negociação de cripto-moedas. 2017. Disponível em: <https://www.haasonline.com/>. Acesso em: 22 outubro 2017.

KAMINSKI, J. C. Nowcasting the bitcoin market with twitter signals. 2016. Disponível em: <https://arxiv.org/pdf/1406.7577.pdf>. Acesso em: 22 outubro 2017.

KARAPANOS, N.; CAPKUN, S. On the effective prevention of tls man-in-themiddle attacks in web applications. USENIX Security Symposium, 2014. Disponível em: <https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/karapanos>. Acesso em: 23 outubro 2017.

KRISTOUFEK, L. What are the main drivers of the bitcoin price? evidence from wavelet coherence analysis. 2014. Disponível em: <http://www2.humusoft.cz/www/papers/finsem14/kristoufek.pdf>. Acesso em: 22 outubro 2017.

MACDONELL, A. Popping the bitcoin bubble: An application of log-periodic power law modeling to digital currency. 2014. Disponível em: <https://economics.nd.edu/assets/134206/mac_donell_popping_the_biticoin_bubble_an_application_of_log_periodic_power_law_modeling_to_digital_currency.pdf>. Acesso em: 22 outubro 2017.

MADAN, I.; SALUJA, S.; ZHAO, A. Automated bitcoin trading via machine learning algorithms. 2015. Disponível em: <http://cs229.stanford.edu/proj2014/Isaac%20Madan,%20Shaurya%20Saluja,%20Aojia%20Zhao,Automated%20Bitcoin%20Trading%20via%20Machine%20Learning%20Algorithms.pdf>. Acesso em: 22 outubro 2017.

MARKETS, H. Mastering the macd. 2015. Disponível em: <https://www.hantecfx.com/wp-content/uploads/2015/12/mastering_the_macd.pdf>. Acesso em: 22 outubro 2017.

MCNALLY, S. Predicting the price of bitcoin using machine learning. 2016. Disponível em: <http://trap.ncirl.ie/2496/1/seanmcnally.pdf>. Acesso em: 22 outubro 2017.

MERCADOBITCOIN. Mercadobitcoin: Bolsa de bitcoin online. 2016. Disponível em: <https://www.mercadobitcoin.com.br>. Acesso em: 22 outubro 2017.

METZ, C. Ai and bitcoin are driving the next big hedge fund wave. 2017. Disponível em: <https://www.wired.com/2017/02/bots-crowds-bitcoin-driving-next-hedge-fund-wave/>. Acesso em: 22 outubro 2017.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Acesso em: 22 outubro 2017.

NEWS, B. Automated traders take over bitcoin as easy money beckons. 2017. Disponível em: <https://www.bloomberg.com/professional/blog/automated-traders-take-bitcoin-easy-money-beckons/>. Acesso em: 22 outubro 2017.

OKCOIN. Okcoin: Bolsa de bitcoin online. 2016. Disponível em: <https://www.okcoin.com>. Acesso em: 22 outubro 2017.

REDMAN, J. You might be exchanging bitcoin with trading bots and not even know it. 2017. Disponível em: <https://news.bitcoin.com/might-exchanging-bitcoin-trading-bots-not-even-know/>. Acesso em: 22 outubro 2017.

ROWLANDS, J. Bitcoin algorithmic trading bot. 2014. Disponível em: <http://studentnet.cs.manchester.ac.uk/resources/library/3rd-year-projects/2014/james.rowlands.pdf>. Acesso em: 22 outubro 2017.

TECH, I. on. Ethereum flash crash - $317 to $0.1 on gdax - programmer explains. 2017. Disponível em: <https://www.youtube.com/watch?v=CcPqkIz0CR0>. Acesso em: 22 outubro 2017.

WHITE, A. Eth-usd trading update. 2017. Disponível em: <https://blog.gdax.com/eth-usd-trading-update-5d8142b5bdc1>. Acesso em: 22 outubro 2017.

WIJK, D. van. What can be expected from the bitcoin? 2013. Disponível em: <https://thesis.eur.nl/pub/14100/Final-version-Thesis-Dennis-van-Wijk.pdf>.

ŻBIKOWSKI, K. Application of machine learning algorithms for bitcoin automated trading. 2015. Disponível em: <https://www.researchgate.net/publication/314933260_Application_of_Machine_Learning_Algorithms_for_Bitcoin_Automated_Trading>. Acesso em: 22 outubro 2017.

APÊNDICE A - MÉTODOS DA API EM MERCADO BITCOIN

Vamos apresentar a seguir a descrição seguida da implementação de métodos da API de dados na *exchange* Mercado Bitcoin.

- **Ticker:** Retorna um resumo das informações do mercado nas últimas 24h, uma versão antiga da API retorna valores a partir da meia noite. Implementação em Código 1, a resposta que contém os seguintes elementos:

  High. Decimal: Maior valor, em reais, de negociação nas últimas 24 horas.

  Low. Decimal: Menor valor, em reais, de negociação nas últimas 24 horas.

  Vol. Decimal: Volume de bitcoins/litecoins negociados nas últimas 24 horas.

  Last. Decimal: Preço unitário do último negócio, em reais.

  Buy. Decimal: Maior valor, em reais, de oferta de compra.

  Sell. Decimal: Menor valor, em reais, de oferta de venda.

  Date. Inteiro: Data e hora em Era Unix da última atualização do ticker.

```
>>> import urllib
>>> import json
>>> def ticker():
    uri = 'https://www.mercadobitcoin.net/api/v2/ticker'
    params = {}
    r = urllib.urlopen(uri)
    return json.loads(r.read())

>>> ticker()
{u'ticker': {u'sell': 8948.99999, u'buy': 8891.0, u'last': 8891.0,
  ↪  u'vol': 173.46753991, u'high': 9000.0, u'low': 8727.0,
  ↪  u'date': 1500765035}}
```

Código 1: Implementação do método ticker para Mercado Bitcoin, com exemplo de execução.

- **Orderbook:** Retorna o livro de ordens com todas as ordens de compra e venda em aberto no momento da consulta. Implementação em Código 2, a resposta contém os seguintes elementos:

  Asks: Lista de ordens de venda ordenadas pela ordem de menor valor no topo. Os elementos da lista são conjuntos com dois elementos, no formato [Decimal, Decimal]: Par, cujo primeiro elemento é o preço e o segundo é o volume disponível.

  Bids: Lista de ordens de compra ordenadas pela ordem de maior valor no topo. Os elementos da lista são conjuntos com dois elementos, no formato [Decimal, Decimal]: Par, cujo primeiro elemento é o preço e o segundo é o volume disponível.

```
>>> import urllib
>>> import json
>>> def orderbook():
    uri = 'https://www.mercadobitcoin.net/api/orderbook/'
    params = {}
    r = urllib.urlopen(uri);
    return json.loads(r.read())

>>> book = orderbook()
>>> bids = book.get('bids')
>>> asks = book.get('asks')
>>> bids[0]; bids[1]; bids[2]
[8840.27505, 0.11877]
[8840.27405, 1.41483708]
[8840.26406, 0.08986]
>>> asks[0]; asks[1]; asks[2]
[8934.0, 0.0297]
[8934.999, 0.11354833]
[8935.0, 0.1]
```

Código 2: Implementação do método orderbook para Mercado Bitcoin, com exemplo de execução.

- **Trades:** Retorna a lista com as últimas operações ocorridas em Mercado Bitcoin até o momento da consulta. Implementação em Código 3, a resposta contém os seguintes elementos:

  Date. Inteiro: Data e hora em Era Unix da operação.

  Price. Decimal: Preço unitário da operação.

  Amount. Decimal: Volume da moeda digital da operação.

  Tid. Inteiro: ID da operação.

  Type. String: Tipo da operação: sell para venda e buy para compra.

```
>>> import urllib
>>> import json
>>> def trades():
    uri = 'https://www.mercadobitcoin.net/api/trades/'
    params = {}
    r = urllib.urlopen(uri);
    return json.loads(r.read())

>>> lista = trades()
>>> len(lista)
1000
>>> lista[0]; lista[1]; lista[2]
```

```
{u'date': 1500834664, u'tid': 711218, u'price': 8825.099, u'type':
  ↪  u'sell', u'amount': 0.00797}
{u'date': 1500834557, u'tid': 711217, u'price': 8825.0, u'type':
  ↪  u'sell', u'amount': 0.02294}
{u'date': 1500834557, u'tid': 711216, u'price': 8890.0, u'type':
  ↪  u'sell', u'amount': 0.01706}
```

Código 3: Implementação do método trades para Mercado Bitcoin, com exemplo de execução.

Abaixo vamos apresentar alguns dos métodos da API de negociações.

- **Get_account_info:** "Retorna dados da conta, como saldos das moedas (real, bitcoin e litecoin), saldos considerando retenção em ordens abertas, quantidades de ordens abertas por moeda digital, limites de saque/retirada das moedas."MercadoBitcoin (2016). Implementação em Código 4.

```
>>> def get_account_info():
    params = {
        'tapi_method': 'get_account_info',
        'tapi_nonce': str(int(time.time()))
    }
    params = urllib.urlencode(params)
    params_string = REQUEST_PATH + '?' + params

    H = hmac.new(MB_TAPI_SECRET, digestmod=hashlib.sha512)
    H.update(params_string)
    sign = H.hexdigest()
    headers = {"Content-type":
  ↪  "application/x-www-form-urlencoded",
        "TAPI-ID":MB_TAPI_ID,
        "TAPI-MAC":sign}
    conn = httplib.HTTPSConnection('www.mercadobitcoin.net',
  ↪  timeout=10);
    conn.request("POST", REQUEST_PATH, params, headers);
    response = conn.getresponse();
    saida = json.load(response);
    time.sleep(1)
    conn.close()
    return saida.get('response_data')

>>> bal = get_account_info()
>>> bal.keys()
[u'withdrawal_limits', u'balance']
>>> bal.get('balance').keys()
[u'ltc', u'brl', u'btc']
```

```
>>> bal.get('balance').get('brl')
{u'available': u'800.30696', u'total': u'800.30696'}
```

Código 4: Implementação de get_account_info para Mercado Bitcoin, com exemplo de execução.

- **Get_order:** "Retorna os dados da ordem de acordo com o ID informado. Dentre os dados estão as informações das operações executadas dessa ordem. Apenas ordens que pertencem ao proprietário da chave da TAPI podem ser consultadas. Erros específicos são retornados para os casos onde o order_id informado não seja de uma ordem válida ou pertença a outro usuário."MercadoBitcoin (2016). Implementação em Código 5.

```
def get_order(pid):
    params = {
        'tapi_method': 'get_order',
        'tapi_nonce': str(int(time.time())),
        'coin_pair': 'BRLBTC',
        'order_id': pid
    }
    params = urllib.urlencode(params)
    params_string = REQUEST_PATH + '?' + params

    H = hmac.new(MB_TAPI_SECRET, digestmod=hashlib.sha512)
    H.update(params_string)
    sign = H.hexdigest()
    headers = {"Content-type":
    ↪ "application/x-www-form-urlencoded",
        "TAPI-ID":MB_TAPI_ID,
        "TAPI-MAC":sign}
    conn = httplib.HTTPSConnection('www.mercadobitcoin.net',
    ↪ timeout=10);
    conn.request("POST", REQUEST_PATH, params, headers);
    response = conn.getresponse();
    saida = json.load(response);
    conn.close();
    return saida.get('response_data').get('order')

>>> resp = get_order(buy_order.get('order_id')) #usa o id de uma
    ↪ ordem criada anteriormente
>>> resp
```

```
26 {u'status': 3, u'executed_quantity': u'0.00000000', u'has_fills':
     ↪ False, u'fee': u'0.00000000', u'order_type': 1,
     ↪ u'executed_price_avg': u'0.00000', u'updated_timestamp':
     ↪ u'1500892953', u'order_id': 38450457, u'limit_price':
     ↪ u'7890.00000', u'operations': [], u'created_timestamp':
     ↪ u'1500892943', u'coin_pair': u'BRLBTC', u'quantity':
     ↪ u'0.01200000'}
```

Código 5: Implementação de get_order para Mercado Bitcoin, com exemplo de execução.

- **List_orders:** "Retorna uma lista de até 200 ordens, de acordo com os filtros informados, ordenadas pela data de última atualização. As operações executadas de cada ordem também são retornadas. Apenas ordens que pertencem ao proprietário da chave da TAPI são retornadas. Caso nenhuma ordem seja encontrada, é retornada uma lista vazia."MercadoBitcoin (2016). Implementação em Código 6.

```
1  >>> def list_orders():
2      params = {
3          'tapi_method': 'list_orders',
4          'tapi_nonce': str(int(time.time())),
5          'coin_pair': 'BRLBTC',
6          'status_list': "[2]" #open2, cancelled3, filed4
7      }
8      params = urllib.urlencode(params)
9      params_string = REQUEST_PATH + '?' + params
10
11     H = hmac.new(MB_TAPI_SECRET, digestmod=hashlib.sha512)
12     H.update(params_string)
13     sign = H.hexdigest()
14     headers = {"Content-type":
     ↪  "application/x-www-form-urlencoded",
15         "TAPI-ID":MB_TAPI_ID,
16         "TAPI-MAC":sign}
17     conn = httplib.HTTPSConnection('www.mercadobitcoin.net',
     ↪  timeout=100);
18     conn.request("POST", REQUEST_PATH, params, headers);
19     response = conn.getresponse();
20     saida = json.load(response);
21     time.sleep(1)
22     conn.close();
23     return saida.get('response_data').get('orders')
24
25 >>> orders = list_orders()
26 >>> orders[0]
```

```
{u'status': 2, u'executed_quantity': u'0.00000000', u'has_fills':
  ↪ False, u'fee': u'0.00000000', u'order_type': 2,
  ↪ u'executed_price_avg': u'0.00000', u'updated_timestamp':
  ↪ u'1500897791', u'order_id': 38452947, u'limit_price':
  ↪ u'8879.99999', u'operations': [], u'created_timestamp':
  ↪ u'1500897791', u'coin_pair': u'BRLBTC', u'quantity':
  ↪ u'0.09299727'}
>>> orders[1].get('limit_price')
u'10000.00000'
```

Código 6: Implementação de list_orders para Mercado Bitcoin, com exemplo de execução.

- **List_orderbook:** "Retorna informações do livro de negociações (orderbook) do Mercado Bitcoin para o par de moedas (coin_pair) informado. Diferente do método orderbook público descrito em www.mercadobitcoin.com.br/api-doc/#2.2, aqui são fornecidas informações importantes para facilitar a tomada de ação de clientes TAPI e sincronia das chamadas. Dentre elas, o número da última ordem contemplada (latest_order_id) e número das ordens do livro (order_id)."MercadoBitcoin (2016). Implementação em Código 7.

```
>>> def list_orderbook():
    params = {
        'tapi_method': 'list_orderbook',
        'tapi_nonce': str(int(time.time())),
        'coin_pair': 'BRLBTC',
        'full': True #False = 20, True = 500
    }
    params = urllib.urlencode(params)
    params_string = REQUEST_PATH + '?' + params

    H = hmac.new(MB_TAPI_SECRET, digestmod=hashlib.sha512)
    H.update(params_string)
    sign = H.hexdigest()
    headers = {"Content-type":
  ↪  "application/x-www-form-urlencoded",
        "TAPI-ID":MB_TAPI_ID,
        "TAPI-MAC":sign}
    conn = httplib.HTTPSConnection('www.mercadobitcoin.net',
  ↪  timeout=100);
    conn.request("POST", REQUEST_PATH, params, headers);
    response = conn.getresponse();
    saida = json.load(response);
    time.sleep(1)
    conn.close();
    return saida.get('response_data').get('orderbook')

```

```
>>> book = list_orderbook()
>>> bids = book.get('bids')
>>> bids[0]; bids[1]
{u'order_id': 39481306, u'limit_price': u'14300.00000',
  ↪  u'is_owner': False, u'quantity': u'0.68160047'}
{u'order_id': 39480411, u'limit_price': u'14292.00000',
  ↪  u'is_owner': False, u'quantity': u'0.00100000'}
```

Código 7: Implementação de list_orderbook para Mercado Bitcoin, com exemplo de execução.

- **Place_buy_order:** "Abre uma ordem de compra (buy ou bid) do par de moedas, quantidade de moeda digital e preço unitário limite informados. A criação contempla o processo de confrontamento da ordem com o livro de negociações. Assim, a resposta pode informar se a ordem foi executada (parcialmente ou não) imediatamente após sua criação e, assim, se segue ou não aberta e ativa no livro."MercadoBitcoin (2016). Implementação em Código 8.

```
>>> def place_buy_order(bit, preco):
    params = {
        'tapi_method': 'place_buy_order',
        'tapi_nonce': str(int(time.time())),
        'coin_pair': 'BRLBTC',
        'quantity': bit,
        'limit_price': preco
    }
    params = urllib.urlencode(params)
    params_string = REQUEST_PATH + '?' + params

    H = hmac.new(MB_TAPI_SECRET, digestmod=hashlib.sha512)
    H.update(params_string)
    sign = H.hexdigest()
    headers = {"Content-type":
  ↪  "application/x-www-form-urlencoded",
        "TAPI-ID":MB_TAPI_ID,
        "TAPI-MAC":sign}
    conn = httplib.HTTPSConnection('www.mercadobitcoin.net',
  ↪  timeout=10);
    conn.request("POST", REQUEST_PATH, params, headers);
    response = conn.getresponse();
    saida = json.load(response);
    time.sleep(1)
    conn.close();
    if(saida.get('status_code') == 100):
        return saida.get('response_data').get('order')
    else:
        print saida
```

```
28          return None
29
30  >>> order = place_buy_order(0.05, 7000)
31  >>> order
32  {u'status': 2, u'executed_quantity': u'0.00000000', u'has_fills':
      ↪  False, u'fee': u'0.00000000', u'order_type': 1,
      ↪  u'executed_price_avg': u'0.00000', u'updated_timestamp':
      ↪  u'1502739595', u'order_id': 39482118, u'limit_price':
      ↪  u'7000.00000', u'operations': [], u'created_timestamp':
      ↪  u'1502739595', u'coin_pair': u'BRLBTC', u'quantity':
      ↪  u'0.05000000'}
```

Código 8: Implementação de place_buy_order para Mercado Bitcoin, com exemplo de execução.

- **Place_sell_order:** "Abre uma ordem de venda (sell ou ask) do par de moedas, quantidade de moeda digital e preço unitário limite informados. A criação contempla o processo de confrontamento da ordem com o livro de negociações. Assim, a resposta pode informar se a ordem foi executada (parcialmente ou não) imediatamente após sua criação e, assim, se segue ou não aberta e ativa no livro."MercadoBitcoin (2016). Implementação em Código 9.

```
1   >>> def place_sell_order(bit, preco):
2       params = {
3           'tapi_method': 'place_sell_order',
4           'tapi_nonce': str(int(time.time())),
5           'coin_pair': 'BRLBTC',
6           'quantity': bit,
7           'limit_price': preco
8       }
9       params = urllib.urlencode(params)
10      params_string = REQUEST_PATH + '?' + params
11
12      H = hmac.new(MB_TAPI_SECRET, digestmod=hashlib.sha512)
13      H.update(params_string)
14      sign = H.hexdigest()
15      headers = {"Content-type":
      ↪  "application/x-www-form-urlencoded",
16          "TAPI-ID":MB_TAPI_ID,
17          "TAPI-MAC":sign}
18      conn = httplib.HTTPSConnection('www.mercadobitcoin.net',
      ↪  timeout=10);
19      conn.request("POST", REQUEST_PATH, params, headers);
20      response = conn.getresponse();
21      saida = json.load(response);
22      time.sleep(1)
23      conn.close();
```

```
24      if(saida.get('status_code') == 100):
25          return saida.get('response_data').get('order')
26      else:
27          print saida
28          return None
29
30  >>> order = place_sell_order(0.05, 15000)
31  >>> order
32  {u'status': 2, u'executed_quantity': u'0.00000000', u'has_fills':
      ↪  False, u'fee': u'0.00000000', u'order_type': 2,
      ↪  u'executed_price_avg': u'0.00000', u'updated_timestamp':
      ↪  u'1502741056', u'order_id': 39483214, u'limit_price':
      ↪  u'15000.00000', u'operations': [], u'created_timestamp':
      ↪  u'1502741056', u'coin_pair': u'BRLBTC', u'quantity':
      ↪  u'0.05000000'}
```

Código 9: Implementação de place_sell_order para Mercado Bitcoin, com exemplo de execução.

- **Cancel_order:** "Cancela uma ordem, de venda ou compra, de acordo com o ID e par de moedas informado. O retorno contempla o sucesso ou não do cancelamento, bem como os dados e status atuais da ordem. Somente ordens pertencentes ao próprio usuário podem ser canceladas."MercadoBitcoin (2016). Implementação em Código 10.

```
1   >>> def cancelar(order_id):
2       params = {
3           'tapi_method': 'cancel_order',
4           'tapi_nonce': str(int(time.time())),
5           'coin_pair': 'BRLBTC',
6           'order_id': order_id
7       }
8       params = urllib.urlencode(params)
9       params_string = REQUEST_PATH + '?' + params
10
11      H = hmac.new(MB_TAPI_SECRET, digestmod=hashlib.sha512)
12      H.update(params_string)
13      sign = H.hexdigest()
14      headers = {"Content-type":
      ↪  "application/x-www-form-urlencoded",
15          "TAPI-ID":MB_TAPI_ID,
16          "TAPI-MAC":sign}
17      conn = httplib.HTTPSConnection('www.mercadobitcoin.net',
      ↪  timeout=10);
18      conn.request("POST", REQUEST_PATH, params, headers);
19      response = conn.getresponse();
20      saida = json.load(response);
```

```
21      time.sleep(1)
22      conn.close();
23      if(saida.get('status_code') == 100):
24          return saida.get('response_data').get('order')
25      else:
26          print saida
27          return None
28
29  >>> resp = cancelar(39483214)
30  >>> resp
31  {u'status': 3, u'executed_quantity': u'0.00000000', u'has_fills':
      ↪  False, u'fee': u'0.00000000', u'order_type': 2,
      ↪  u'executed_price_avg': u'0.00000', u'updated_timestamp':
      ↪  u'1502741175', u'order_id': 39483214, u'limit_price':
      ↪  u'15000.00000', u'operations': [], u'created_timestamp':
      ↪  u'1502741056', u'coin_pair': u'BRLBTC', u'quantity':
      ↪  u'0.05000000'}
```

Código 10: Implementação de cancel_order para Mercado Bitcoin, com exemplo de execução.

APÊNDICE B - MÉTODOS DA API EM BITFINEX

Vamos apresentar a seguir a descrição seguida da implementação de métodos da API de dados na *exchange* Bitfinex.

- **Ticker:** É um resumo de alto nível do estado do mercado. Mostra o preço de compra e venda, bem como o preço da última negociação. Também inclui informações como volume e o quanto o preço variou ao longo das últimas 24h. Implementação em Código 11, a resposta contém os seguintes elementos:

  Mid. Decimal: preço médio (bid+ask)/2.

  Bid. Decimal: Maior valor, em dólar, de oferta de compra.

  Ask. Decimal: Menor valor, em dólar, de oferta de venda.

  Last_price. Decimal: Preço unitário do último negócio, em dólar.

  Low. Decimal: Menor valor, em dólar, de negociação nas últimas 24 horas.

  High. Decimal: Maior valor, em dólar, de negociação nas últimas 24 horas.

  Volume. Decimal: Volume negociado nas últimas 24 horas.

  Timestamp. Time: O timestamp para o qual essa informação foi válida.

```
>>> import urllib
>>> import json
>>> def getTicker():
    bitfinexURL = 'https://api.bitfinex.com/v1/pubticker/btcusd'
    params = {}
    r = urllib.urlopen(bitfinexURL);
    return json.loads(r.read())

>>> ticker = getTicker()
>>> ticker.get('low'), ticker.get('high'), ticker.get('volume')
(u'2655.0', u'2877.5', u'35611.01355621')
>>> ticker
{u'volume': u'35611.01355621', u'timestamp':
  ↪  u'1500838530.601141957', u'bid': u'2779.1', u'last_price':
  ↪  u'2778.1', u'mid': u'2779.25', u'high': u'2877.5', u'low':
  ↪  u'2655.0', u'ask': u'2779.4'}
```

Código 11: Implementação do método ticker para a *exchange* Bitfinex, com exemplo de execução.

- **Fundingbook:** Retorna o livro de financiamentos chamado de Funding book com pedidos de financiamentos (bids) e ofertas de financiamentos (asks). A resposta é limitada a dois arrays com no máximo 50 elementos em cada. Implementação em Código 12, a resposta contém os seguintes elementos:

Bids. Array: Pedidos de financiamentos (funding bids).

Asks. Array: Ofertas de financiamentos (funding asks). Os elementos de cada array têm o seguinte formato.

Rate. Decimal: Taxa de juros em % por 365 dias.

Amount. Decimal: Valor do financiamento.

Period. Inteiro: Período máximo de dias do contrato de financiamento.

Timestamp. Time: O timestamp da criação da ordem.

Frr. [yes/no]: Se a oferta é ou não Flash Return Rate, uma média calculada pela *exchange* baseada nas últimas ofertas executadas.

```
>>> import urllib
>>> import json
>>> def lendbook():
    bitfinexURL = 'https://api.bitfinex.com/v1/lendbook/USD'
    params = {}
    r = urllib.urlopen(bitfinexURL);
    return json.loads(r.read())

>>> book = lendbook()
>>> asks = book.get('asks')
>>> asks[0]
{u'timestamp': u'1500839022.0', u'rate': u'16.79', u'frr': u'No',
  ↪  u'amount': u'0.00026848', u'period': 2}
```

Código 12: Implementação do método fundingbook para a *exchange* Bitfinex, com exemplo de execução.

- **Orderbook:** Retorna o livro de ordens com todas as ordens de compra e venda em aberto no momento da consulta. Pode ser usado para consultar o orderbook de outras moedas mudando um parâmetro. Implementação em Código 13, a resposta contém os seguintes elementos:

Bids. Lista de ordens de compra ordenadas pelo preço, com o maior preço no topo.

Asks. Lista de ordens de venda ordenadas, com o menor preço no top. Os elementos de ambas as listas são no seguinte formato.

Price. Decimal: Preço da unidade de bitcoin.

Amount. Decimal: Volume de bitcoins disponível para compra ou venda.

Timestamp. Time: Timestamp da criação da ordem.

```
>>> import urllib
>>> import json
>>> def orderbook(symbol):
    bitfinexURL = 'https://api.bitfinex.com/v1/book/'+symbol
    params = {}
    r = urllib.urlopen(bitfinexURL);
    return json.loads(r.read())

>>> book = orderbook('btcusd')
>>> bids = book.get('bids')
>>> asks = book.get('asks')
>>> bids[0]; bids[1];
{u'timestamp': u'1500854156.0', u'price': u'2754.5', u'amount':
  ↪  u'0.27746'}
{u'timestamp': u'1500854146.0', u'price': u'2754.4', u'amount':
  ↪  u'0.02048'}
>>> asks[0].get('price'), asks[0].get('amount')
(u'2755.8', u'0.01175')
```

Código 13: Implementação do método orderbook para a *exchange* Bitfinex, com exemplo de execução.

- **Trades:** Retorna a lista das ordens executadas mais recentes em um mercado definido por parâmetro. Implementação em Código 14, a resposta é uma lista com elementos no seguinte formato:

  Tid. Inteiro: O id da ordem.

  Timestamp. Time: Timestamp da criação da ordem.

  Price. Decimal: Preço da unidade de bitcoin.

  Amount. Decimal: Volume de bitcoins negociado na operação.

  Exchange. String: Nome da *exchange*; "bitfinex".

  Type. String: tipo de ordem que foi executada; "sell" ou "buy" (pode ser "").

```
>>> import urllib
>>> import json
>>> def trades(symbol):
    bitfinexURL = 'https://api.bitfinex.com/v1/trades/'+symbol
    params = {}
    r = urllib.urlopen(bitfinexURL);
    return json.loads(r.read())

>>> lista = trades('btcusd')
```

```
>>> lista[:4]
[{u'exchange': u'bitfinex', u'timestamp': 1500856652, u'price':
  ↪ u'2770.0', u'amount': u'0.04306493', u'tid': 46272930,
  ↪ u'type': u'sell'}, {u'exchange': u'bitfinex', u'timestamp':
  ↪ 1500856652, u'price': u'2770.1', u'amount': u'0.02023248',
  ↪ u'tid': 46272929, u'type': u'sell'}, {u'exchange':
  ↪ u'bitfinex', u'timestamp': 1500856652, u'price': u'2770.1',
  ↪ u'amount': u'0.01145697', u'tid': 46272922, u'type': u'sell'},
  ↪ {u'exchange': u'bitfinex', u'timestamp': 1500856652, u'price':
  ↪ u'2770.1', u'amount': u'0.07831055', u'tid': 46272916,
  ↪ u'type': u'sell'}]
```

Código 14: Implementação do método trades para a *exchange* Bitfinex, com exemplo de execução.

- **Lends:** Semelhante ao método trades, retorna a lista dos dados de financiamento mais recentes para a moeda dada, com taxa de juros para 365 dias. Implementação em Código 15, a resposta contém os seguintes elementos:

  Rate. Decimal: Taxa média de financiamento total, ou seja, a taxa de retorno instantânea anual.

  Amount_lent. Decimal: Montante total de financiamento de margem aberta.

  Amount_used. Decimal: Valor total de margem aberta utilizado em uma posição de margem.

  Timestamp. Time: timestamp.

```
>>> import urllib
>>> import json
>>> def lends(currency):
    bitfinexURL = 'https://api.bitfinex.com/v1/lends/'+currency
    params = {}
    r = urllib.urlopen(bitfinexURL);
    return json.loads(r.read())

>>> lends = lends('USD')
>>> lends[0]; lends[1]
{u'timestamp': 1500859800, u'rate': u'48.1194', u'amount_used':
  ↪ u'88014868.33970094', u'amount_lent': u'89325400.19788549'}
{u'timestamp': 1500856201, u'rate': u'48.1306', u'amount_used':
  ↪ u'88176045.46939107', u'amount_lent': u'89526571.71230403'}
```

Código 15: Implementação do método lends para a *exchange* Bitfinex, com exemplo de execução.

- **Candles:** É um método da API versão 2 que prove uma forma para acessar informações dos gráficas de candles. Retorna uma lista que representa o gráfico de candles,

conforme os parâmetros; intervalo de tempo, símbolo do mercado e quantidade de candles. Implementação em Código 16, a resposta é uma lista com elementos no seguinte formato:

MTS. Inteiro: Timestamp em milisegundos.

Opem. Float: Preço da primeira ordem executada nessa janela de tempo.

Close. Float: Preço da última ordem executada nessa janela de tempo.

High. Float: Preço mais alto das ordens executadas nessa janela de tempo.

Low. Float: Preço mais baixo das ordens executadas nessa janela de tempo.

Volume. Float: Volume de moedas negociadas nessa janela de tempo.

```
>>> import urllib
>>> import json
>>> def candles(timeframe, limit, symbol):
    bitfinexURL = 'https://api.bitfinex.com/v2/candles/trade:'
  ↪  +timeframe+':t'+symbol+'/hist/?'
    payloadObject = {
        'limit': limit
        }
    bitfinexURL = bitfinexURL + urllib.urlencode(payloadObject)
    r = urllib.urlopen(bitfinexURL);
    return json.loads(r.read())

>>> lista = candles('30m', 300, 'BTCUSD')
>>> lista[0]; lista[1]; lista[2];
[1500859800000L, 2770.6, 2776.5, 2778.1, 2761.8, 97.41923963]
[1500858000000L, 2767.3, 2772.3, 2779.9, 2760.5, 176.28715026]
[1500856200000L, 2754, 2767.2, 2776, 2744.7, 430.50323055]
```

Código 16: Implementação do método candles para a *exchange* Bitfinex, com exemplo de execução.

Abaixo vamos apresentar alguns dos métodos da API de negociações.

- **Account info:** Retorna informações a respeito das taxas de transação. Devido ao sistema de taxa decrescente baseada em volume mensal a variação de taxas pode afetar a rentabilidade do usuário. A resposta é uma lista com um elemento isso pode ser tratado para retornar diretamente o elemento na chamada do método. Exemplo de implementação em Código 17.

```
>>> def account_infos():
    bitfinexURL = 'https://api.bitfinex.com/v1/account_infos'
    payloadObject = {
            'request':'/v1/account_infos',
```

```
            'nonce':str(time.time())
    }
    payload_json = json.dumps(payloadObject)
    payload = base64.b64encode(bytes(payload_json))
    m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
    m = m.hexdigest()
    headers = {
          'X-BFX-APIKEY' : bitfinexKey,
          'X-BFX-PAYLOAD' : base64.b64encode(bytes(payload_json)),
          'X-BFX-SIGNATURE' : m
    }

    r = requests.get(bitfinexURL, data={}, headers=headers)
    return json.loads(r.content)

>>> info = account_infos()
>>> info[0].get('maker_fees'); info[0].get('taker_fees')
u'0.1'
u'0.2'
```

Código 17: Implementação de account info para *exchange* Bitfinex, com exemplo de execução.

- **Summary:** Retorna informações relacionadas ao volume negociado nos últimos 30 dias, relatório dos lucros com empréstimos e informações das taxas de negociação. A resposta também contêm informações sobre cada moeda individualmente e sobre o total acumulado. Exemplo de implementação em Código 18.

```
def summary():
    bitfinexURL = 'https://api.bitfinex.com/v1/summary'
    payloadObject = {
            'request':'/v1/summary',
            'nonce':str(time.time()),
            'options':{}
    }
    payload_json = json.dumps(payloadObject)
    payload = base64.b64encode(bytes(payload_json))
    m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
    m = m.hexdigest()
    headers = {
          'X-BFX-APIKEY' : bitfinexKey,
          'X-BFX-PAYLOAD' : base64.b64encode(bytes(payload_json)),
          'X-BFX-SIGNATURE' : m
    }

    r = requests.get(bitfinexURL, data={}, headers=headers)
    return json.loads(r.content)
```

```
20
21 >>> suma = summary()
22 >>> suma.keys()
23 [u'funding_profit_30d', u'maker_fee', u'trade_vol_30d',
     ↪ u'taker_fee', u'time']
24 >>> suma.get('funding_profit_30d')[1] #ganhos com emprestimo de
     ↪ BTC
25 {u'amount': u'0.00246677', u'curr': u'BTC'}
26 >>> for x in suma.get('trade_vol_30d'):
27         #percorrer a lista para retornar o volume em dolar
     ↪ negociado em 30 dias
28         if(x.get('curr') == 'Total (USD)'):
29                 print x
30
31 {u'vol': u'1398.98', u'curr': u'Total (USD)'}
```

Código 18: Implementação de summary para *exchange* Bitfinex, com exemplo de execução.

- **Wallet balances:** Retorna uma lista com elementos que informam o saldo em cada moeda negociada pela Bitfinex. Exemplo em Código 19.

```
1  def balances():
2      bitfinexURL = 'https://api.bitfinex.com/v1/balances'
3      payloadObject = {
4              'request':'/v1/balances',
5              'nonce':str(time.time()),
6              'options':{}
7      }
8      payload_json = json.dumps(payloadObject)
9      payload = base64.b64encode(bytes(payload_json))
10     m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
11     m = m.hexdigest()
12     headers = {
13           'X-BFX-APIKEY' : bitfinexKey,
14           'X-BFX-PAYLOAD' : base64.b64encode(bytes(payload_json)),
15           'X-BFX-SIGNATURE' : m
16     }
17
18     r = requests.get(bitfinexURL, data={}, headers=headers)
19     return json.loads(r.content)
20
21 >>> bal = balances()
22 >>> bal[5]; bal[8]
23 {u'available': u'0.0', u'currency': u'btc', u'amount': u'0.0',
     ↪ u'type': u'exchange'}
24 {u'available': u'0.20083946', u'currency': u'usd', u'amount':
     ↪ u'0.20083946', u'type': u'exchange'}
```

Código 19: Implementação de wallet balances para *exchange* Bitfinex, com exemplo de execução.

- **New order:** Cria uma nova ordem. Exemplo de implementação em Código 20, os seguintes parâmetros são necessários:

  Request. String: Parte da URL que identifica o método.

  Nonce. String: Número no formato de string que deve ser crescente a cada chamada.

  Symbol. String: Símbolo que representa o mercado, ex. 'btcusd'.

  Amount. Float: Quantidade de moeda que vai ser comprada ou vendida.

  Price. Float: Preço para comprar ou vender. Deve ser positivo. E para ordens com type=market pode ser colocado um número random.

  Side. String: 'buy' or 'sell', que identifica se é ordem de compra ou venda.

  Type. String: "market" ou "limit" ou "stop" ou "trailing-stop" ou "fill-or-kill" ou "exchange market" ou "exchange limit" ou "exchange stop" ou "exchange trailing-stop" ou "exchange fill-or-kill". (tipos começando por "exchange " são exchange orders, outros são margin trading orders). Identifica de qual carteira vai usar o saldo e o tipo de ordem que será criada. Alguns tipos vão requerer novos parâmetros adicionais.

  Exchange. String: "bitfinex", é uma constante.

```
1  def newOrder(pamount, pprice, pside):
2      bitfinexURL = 'https://api.bitfinex.com/v1/order/new'
3      payloadObject = {
4              'request':'/v1/order/new',
5              'nonce':str(time.time()),
6              'symbol':'btcusd',
7              'amount':pamount,
8              'price':pprice,
9              'side':pside,
10             'type':'exchange limit',
11             'exchange':'bitfinex'
12             #'options':{}
13     }
14     payload_json = json.dumps(payloadObject)
15     payload = base64.b64encode(bytes(payload_json))
16     m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
17     m = m.hexdigest()
18     headers = {
19           'X-BFX-APIKEY' : bitfinexKey,
```

```
            'X-BFX-PAYLOAD' : base64.b64encode(bytes(payload_json)),
            'X-BFX-SIGNATURE' : m
    }
    r = requests.post(bitfinexURL, data={}, headers=headers)
    return json.loads(r.content)

>>> order = newOrder('0.01', '2601.12', 'buy')
>>> order.keys()
[u'oco_order', u'was_forced', u'src', u'avg_execution_price',
 ↪ u'exchange', u'order_id', u'timestamp', u'symbol', u'cid',
 ↪ u'cid_date', u'price', u'is_live', u'gid', u'executed_amount',
 ↪ u'is_cancelled', u'remaining_amount', u'is_hidden',
 ↪ u'original_amount', u'type', u'id', u'side']
>>> order.get('side'), order.get('symbol'), order.get('price')
(u'buy', u'btcusd', u'2601.1')
```

Código 20: Implementação de new order para *exchange* Bitfinex, com exemplo de execução.

- **Cancel order:** Cancela uma ordem identificada pelo Id. Esse método tem duas variações: uma para cancelar uma lista de ordens e outra para cancelar todas as ordens em aberto, em uma única requisição, a escolha deve ser feita de acordo com a necessidade do usuário. Exemplo em Código 21.

  Request. String: Parte da URL que identifica o método.

  Nonce. String: Número no formato de string que deve ser crescente a cada chamada.

  Order_id. Número: Id da ordem que será cancelada.

```
def cancel(pid):
    bitfinexURL = 'https://api.bitfinex.com/v1/order/cancel'
    payloadObject = {
            'request':'/v1/order/cancel',
            'nonce':str(time.time()),
            'order_id':pid,
            'options':{}
    }
    payload_json = json.dumps(payloadObject)
    payload = base64.b64encode(bytes(payload_json))
    m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
    m = m.hexdigest()
    headers = {
          'X-BFX-APIKEY' : bitfinexKey,
          'X-BFX-PAYLOAD' : base64.b64encode(bytes(payload_json)),
          'X-BFX-SIGNATURE' : m
    }
```

```
18      r = requests.post(bitfinexURL, data={}, headers=headers)
19      return json.loads(r.content)
20
21  >>> cancel(3365652371L)
22  {u'oco_order': None, u'was_forced': False, u'src': u'api',
      ↪ u'avg_execution_price': u'0.0', u'exchange': u'bitfinex',
      ↪ u'timestamp': u'1502764020.0', u'symbol': u'btcusd', u'cid':
      ↪ 8820284404L, u'cid_date': u'2017-08-15', u'price': u'2600.1',
      ↪ u'is_live': True, u'gid': None, u'executed_amount': u'0.0',
      ↪ u'is_cancelled': False, u'remaining_amount': u'0.01',
      ↪ u'is_hidden': False, u'original_amount': u'0.01', u'type':
      ↪ u'exchange limit', u'id': 3365652371L, u'side': u'buy'}
23  >>> cancel(3365652371L)
24  {u'message': u'Order could not be cancelled.'}
```

Código 21: Implementação de cancel order para *exchange* Bitfinex, com exemplo de execução.

- **Replace order:** Substitui uma ordem por outra que pode ter preço, volume e tipo diferentes. É um método eficiente para reduzir a quantidade de requisições em casos onde o usuário precisaria cancelar a ordem para depois criar outra. Exemplo em Código 22, Os seguintes parâmetros são usados:

  Request. String: Parte da URL que identifica o método.

  Nonce. String: Número no formato de string que deve ser crescente a cada chamada.

  Order_id Número: Id da ordem que será substituída.

  Symbol. String: Símbolo que representa o mercado, ex. 'btcusd'.

  Amount. Float: Quantidade de moeda que vai ser comprada ou vendida na nova ordem.

  Price. Float: Preço para comprar ou vender. Deve ser positivo. E para ordens com type=market pode ser colocado um número random.

  Side. String: 'buy' or 'sell', que identifica se é ordem de compra ou venda.

  Type. String: "market" ou "limit" ou "stop" ou "trailing-stop" ou "fill-or-kill" ou "exchange market" ou "exchange limit" ou "exchange stop" ou "exchange trailing-stop" ou "exchange fill-or-kill". (tipos começando por "exchange " são exchange orders, outros são margin trading orders). Identifica de qual carteira vai usar o saldo e o tipo de ordem que será criada. Alguns tipos vão requerer novos parâmetros adicionais.

  Exchange. String: "bitfinex", é uma constante.

```
def replace(pid, pside, pprice, pamount):
    bitfinexURL =
  ↪ 'https://api.bitfinex.com/v1/order/cancel/replace'
    payloadObject = {
            'request':'/v1/order/cancel/replace',
            'nonce':str(time.time()),
            'order_id':pid,
            'symbol':'btcusd',
            'amount':pamount,
            'price':pprice,
            'exchange':'bitfinex',
            'side':pside,
            'type':'exchange limit'
    }
    payload_json = json.dumps(payloadObject)
    payload = base64.b64encode(bytes(payload_json))
    m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
    m = m.hexdigest()
    headers = {
          'X-BFX-APIKEY' : bitfinexKey,
          'X-BFX-PAYLOAD' : payload,
          'X-BFX-SIGNATURE' : m
    }
    r = requests.post(bitfinexURL, data={}, headers=headers)
    return json.loads(r.content)

>>> neworder = replace(3365635959L,'buy', '2600.11', '0.01')
>>> neworder
{u'oco_order': None, u'was_forced': False, u'src': u'api',
  ↪ u'avg_execution_price': u'0.0', u'exchange': u'bitfinex',
  ↪ u'order_id': 3365652371L, u'timestamp':
  ↪ u'1502764020.30986355', u'symbol': u'btcusd', u'cid':
  ↪ 8820284404L, u'cid_date': u'2017-08-15', u'price': u'2600.1',
  ↪ u'is_live': True, u'gid': None, u'executed_amount': u'0.0',
  ↪ u'is_cancelled': False, u'remaining_amount': u'0.01',
  ↪ u'is_hidden': False, u'original_amount': u'0.01', u'type':
  ↪ u'exchange limit', u'id': 3365652371L, u'side': u'buy'}
```

Código 22: Implementação de replace order para *exchange* Bitfinex, com exemplo de execução.

- **Order status:** Consulta uma ordem passando como parâmetro o ID. Exemplo em Código 23.

  Request. String: Parte da URL que identifica o método.

  Nonce. String: Número no formato de string que deve ser crescente a cada chamada.

Order_id. Número: Id da ordem a ser consultada.

```
def orderStatus(pid):
    bitfinexURL = 'https://api.bitfinex.com/v1/order/status'
    payloadObject = {
            'request':'/v1/order/status',
            'nonce':str(time.time()),
            'order_id':pid
    }
    payload_json = json.dumps(payloadObject)
    payload = base64.b64encode(bytes(payload_json))
    m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
    m = m.hexdigest()
    headers = {
          'X-BFX-APIKEY' : bitfinexKey,
          'X-BFX-PAYLOAD' : payload,
          'X-BFX-SIGNATURE' : m
    }
    r = requests.post(bitfinexURL, data={}, headers=headers)
    return json.loads(r.content)

>>> orderStatus(3365652371L)
{u'oco_order': None, u'was_forced': False, u'src': u'api',
  ↪  u'avg_execution_price': u'0.0', u'exchange': u'bitfinex',
  ↪  u'timestamp': u'1502764020.0', u'symbol': u'btcusd', u'cid':
  ↪  8820284404L, u'cid_date': u'2017-08-15', u'price': u'2600.1',
  ↪  u'is_live': True, u'gid': None, u'executed_amount': u'0.0',
  ↪  u'is_cancelled': False, u'remaining_amount': u'0.01',
  ↪  u'is_hidden': False, u'original_amount': u'0.01', u'type':
  ↪  u'exchange limit', u'id': 3365652371L, u'side': u'buy'}
```

Código 23: Implementação de order status para *exchange* Bitfinex, com exemplo de execução.

- **Active orders:** Retorna a lista de ordens abertas. Exemplo em Código 24.

  Request. String: Parte da URL que identifica o método.

  Nonce. String: Número no formato de string que deve ser crescente a cada chamada.

```
def myOrders():
    bitfinexURL = 'https://api.bitfinex.com/v1/orders'
    payloadObject = {
            'request':'/v1/orders',
            'nonce':str(time.time())
    }
    payload_json = json.dumps(payloadObject)
```

```
8       payload = base64.b64encode(bytes(payload_json))
9       m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
10      m = m.hexdigest()
11      headers = {
12            'X-BFX-APIKEY' : bitfinexKey,
13            'X-BFX-PAYLOAD' : payload,
14            'X-BFX-SIGNATURE' : m
15      }
16      r = requests.get(bitfinexURL, data={}, headers=headers)
17      return json.loads(r.content)
18
19  >>> myOrders()
20  [{u'oco_order': None, u'was_forced': False, u'src': u'api',
    ↪  u'avg_execution_price': u'0.0', u'exchange': u'bitfinex',
    ↪  u'timestamp': u'1502764020.0', u'symbol': u'btcusd', u'cid':
    ↪  8820284404L, u'cid_date': u'2017-08-15', u'price': u'2600.1',
    ↪  u'is_live': True, u'gid': None, u'executed_amount': u'0.0',
    ↪  u'is_cancelled': False, u'remaining_amount': u'0.01',
    ↪  u'is_hidden': False, u'original_amount': u'0.01', u'type':
    ↪  u'exchange limit', u'id': 3365652371L, u'side': u'buy'}]
```

Código 24: Implementação de active order para *exchange* Bitfinex, com exemplo de execução.

- **Past trades:** Retorna uma lista de ordens executadas, por padrão o tamanho da lista é de até 50 ordens, mas pode ser modificada com um parâmetro opcional. Exemplo em Código 25.

  Request. String: Parte da URL que identifica o método.

  Nonce. String: Número no formato de string que deve ser crescente a cada chamada.

  Symbol. String: Símbolo do mercado ao qual será requisitado o histórico de transações passadas.

  Limit_trades. Inteiro: Número máximo de elementos na resposta.

```
1   def myTrades():
2       bitfinexURL = 'https://api.bitfinex.com/v1/mytrades'
3       payloadObject = {
4           'request':'/v1/mytrades',
5           'nonce':str(time.time()),
6           'symbol': 'btcusd',
7           'limit_trades': 500
8       }
9       payload_json = json.dumps(payloadObject)
10      payload = base64.b64encode(bytes(payload_json))
11      m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
```

```
12      m = m.hexdigest()
13      headers = {
14            'X-BFX-APIKEY' : bitfinexKey,
15            'X-BFX-PAYLOAD' : payload,
16            'X-BFX-SIGNATURE' : m
17      }
18      r = requests.get(bitfinexURL, data={}, headers=headers)
19      return json.loads(r.content)
20
21  >>> trades = myTrades()
22  >>> len(trades)
23  500
24  >>> trades[499]
25  {u'order_id': 2139151945, u'timestamp': u'1490176635.0', u'price':
      ↪  u'1065.0', u'fee_amount': u'-0.01065', u'amount': u'0.01',
      ↪  u'fee_currency': u'USD', u'tid': 27717549, u'type': u'Sell'}
```

Código 25: Implementação de past trades para *exchange* Bitfinex, com exemplo de execução.

- **Balance history:** Retorna uma lista com histórico de balance para uma moeda e a carteira definidas como parâmetro, o tamanho da lista pode ser definida por parâmetro opcional. Exemplo em Código 26.

  Request. String: Parte da URL que identifica o método.

  Nonce. String: Número no formato de string único e crescente a cada chamada.

  Currency. String: Nome da moeda que será consultada.

  Limit. Inteiro: Número máximo de elementos na lista de resposta.

  Wallet. String: Retorna apenas o histórico da carteira escolhida; "trading", "exchange" ou "deposit".

```
1   def history():
2       bitfinexURL = 'https://api.bitfinex.com/v1/history'
3       payloadObject = {
4               'request':'/v1/history',
5               'nonce':str(time.time()),
6               'currency': 'USD',
7               'limit': 10,
8               'wallet': 'deposit', #trading, exchange, deposit.
9               'options':{}
10      }
11      payload_json = json.dumps(payloadObject)
12      payload = base64.b64encode(bytes(payload_json))
13      m = hmac.new(bitfinexSecret, payload, hashlib.sha384)
14      m = m.hexdigest()
```

```
15      headers = {
16            'X-BFX-APIKEY' : bitfinexKey,
17            'X-BFX-PAYLOAD' : payload,
18            'X-BFX-SIGNATURE' : m
19      }
20      r = requests.get(bitfinexURL, data={}, headers=headers)
21      return json.loads(r.content)
22
23  >>> hist =  history()
24  >>> len(hist)
25  10
26  >>> hist[0]
27  {u'currency': u'USD', u'amount': u'-50.0', u'balance': u'---',
      ↪  u'description': u'Transfer of 50.0 USD from wallet Deposit to
      ↪  Exchange on wallet Deposit', u'timestamp': u'-----'}
```

Código 26: Implementação de balance history para *exchange* Bitfinex, com exemplo de execução.