
# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

## ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS


# ZPRACOVÁNÍ A VYUŽITÍ INFORMACÍ NA TRZÍCH ALTERNATIVNÍCH MĚN
COLLECTING AND INTERPRETING INFORMATION ON DIGITAL CURRENCY EXCHANGES

**DIPLOMOVÁ PRÁCE**
MASTER'S THESIS


**AUTOR PRÁCE** — **Ing. Václav Uhlíř**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Jan Budík, Ph.D.**
SUPERVISOR


**BRNO 2016**

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2015/2016
Ústav informatiky

# ZADÁNÍ DIPLOMOVÉ PRÁCE

**Uhlíř Václav, Ing.**

Informační management (6209T015)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává diplomovou práci s názvem:

**Zpracování a využití informací na trzích alternativních měn**

v anglickém jazyce:

**Collecting and Interpreting Information on Digital Currency Exchanges**

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy
Teoretická východiska práce
Analýza problému
Vlastní návrhy řešení
Závěr
Seznam použité literatury

Seznam odborné literatury:

CHUEN, D. Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data. USA: Academic Press 2015. ISBN 978-0-12-802-117-0.
SWAN, M. Blockchain: Blueprint for a New Economy. USA: O'Reilly Media, 2015. ISBN 978-14-919-2049-7.
VIGNA, P. a M. CASEY. The Age of Cryptocurrency: How Bitcoin and Digital Money are Challenging the Global EconomicOrder. USA: St. Martin's Press, 2015. ISBN: 978-12-500-6563-6.
WILLIAMS, L. Long-Term Secrets to Short-Term Trading. USA: Wiley-Interscience,1999. 255 p. ISBN 0-471-29722-4.

Vedoucí diplomové práce: Ing. Jan Budík, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2015/2016.

L.S.

| | |
|---|---|
| doc. RNDr. Bedřich Půža, CSc. | doc. Ing. et Ing. Stanislav Škapa, Ph.D. |
| Ředitel ústavu | Děkan fakulty |

V Brně, dne 29.2.2016

# Abstrakt

Tato práce se zabývá principiální analýzou zpracování a využití informací na trzích alternativních měn a následně stanovuje požadavky a implementuje modulární systém využitelný pro akademické účely s cílem vytvoření platformy pro práci s daty na stanovených trzích či na jiných obdobných datech.



# Abstract

This student paper discusses principals of data collecting and subsequent analysis of data on digital currency exchanges followed by proposition and full implementation of research oriented system capable of solving all relevant tasks and presenting a way for implementing solutions for broad spectrum of related problems.



# Klíčová slova

Bitcoin, Litecoin, digitální měny, kryptoměny, grafy, zpracování dat.

# Keywords

Bitcoin, Litecoin, digital currency, cryptocurrency, graphs, data analysis.


# Citace

UHLÍŘ, V. *Zpracování a využití informací na trzích alternativních měn.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2016. 61 s. Vedoucí diplomové práce Ing. Jan Budík, Ph.D..

# Zpracování a využití informací na trzích alternativních měn

## Prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

.....
Václav Uhlíř
26. května 2016

# Obsah

# Kapitola 1

# Úvod

Už několik let se stále více a více objevují zmínky o takzvaných „trzích alternativních měn“ zejména pak ve spojitosti s kryptoměnou Bitcoin. Jedná se zpravidla o přirozený proces hledání monetárních možností v prostředí globální sítě, kde jsou lidmi či skupinami lidí vytvářeny prostředky, jež zaplňují určitou mezeru na trhu. Prostředí internetu pak umožňuje snadné implementace takovýchto sítí a umožňuje velmi rychle a levně dostupnost nového produktu po celém světě.

Ač tyto trhy většinou mimikují fyzické či delší dobu zavedené pevné trhy, tak se fundamentálně odlišují svojí nevázaností na určité prvky fyzických struktur. Tyto odlišnosti jsou různé od produktu k produktu a často nejsou na první pohled zřetelné.

Tato práce se bude zabývat analyzováním a principem hledání využitelných dat na trzích alternativních měn a způsoby algoritmické interpretace těchto dat, ve smyslu jejich využití, pro zpracování navazujícími pokročilými algoritmy dosahujícími výsledné přidané hodnoty pro uživatele. Pojem „alternativní měna“ je tedy pro účely této práce definován jako virtuální prostředek pro výměnu bez regulace či přímé vazby na územní nebo samosprávní jednotky.

Podrobné cíle a metody práce jsou definovány v kapitole 2. Samotná realita trhů alternativních měn je poté popsaná v kapitole 3, konkrétně pozadí prostředí jednotlivých směnáren a měn je popsáno v sekci 3.1 následováno teorií o zpracování a interpretování informací vycházejících z tohoto typu zdrojů informací v sekci 3.2.

Kapitola 4 se pak zabývá analýzou příkladů virtuálních měn a vybraným vzorkem směnáren reprezentující konkrétní situace vznikající na tomto trhu.

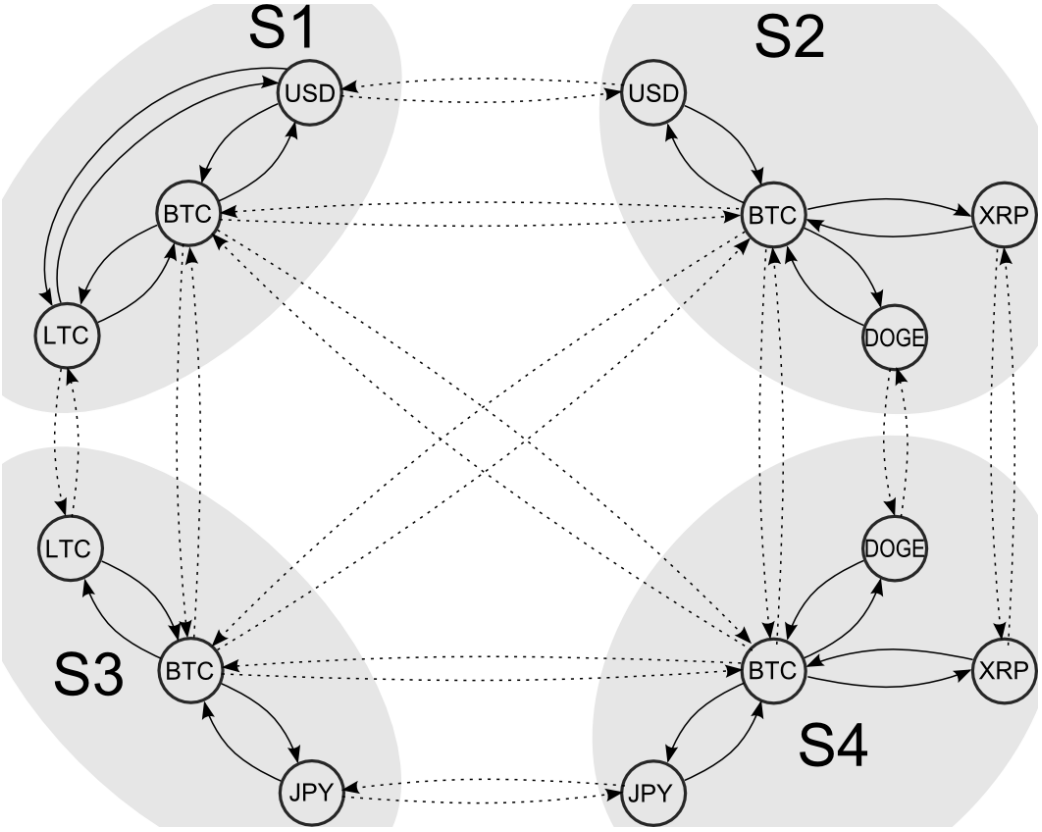
Vlastní řešení problému je poté navrženo a implementováno v kapitole 5, která zároveň obsahuje řadu příkladů a způsobů využití implementovaných algoritmů doplněných o reálné příklady aplikací postavených na implementovaném systému.

# Kapitola 2

# Cíle práce, metody a postupy

## 2.1 Cíle

Výstupem této práce by měla být komplexní teoretická analýza problematiky v prostředí alternativních měn, následovaná návrhem a implementací algoritmů využívající znalosti z analyzovaného prostředí. Jmenovitě teorie fungování a možnosti využití vnitřních vazeb prostředí a aplikace poznatků z hodnot závislých na určitém časovém rozhraní.




Obrázek 2.1: Znázornění možných vazeb mezi čtyřmi různými servery $S$ a jejich měnami.

Výsledné algoritmy by tedy měly dosahovat těchto cílů za udržení co nejvyšší variability, kde bude kladen hlavní důraz na vznik akademické platformy pro případné další

algoritmické nástavby. Dalším cílem bude na vytvořené platformě demonstrovat hledání užitečných vazeb a informací, ze kterých bude moci uživatel čerpat přidanou hodnotu či přímo provádět transakce na trzích alternativních měn. Algoritmy se budou muset nejenom vypořádat s různými zdroji dat a porovnáváním směnných poměrů, ale také se spolehlivostí a časovým zpožděním jednotlivých transakcí (jak vlivem rychlostí protokolů měn, tak vlivem úmyslného zpožďování transakcí směnárnami).

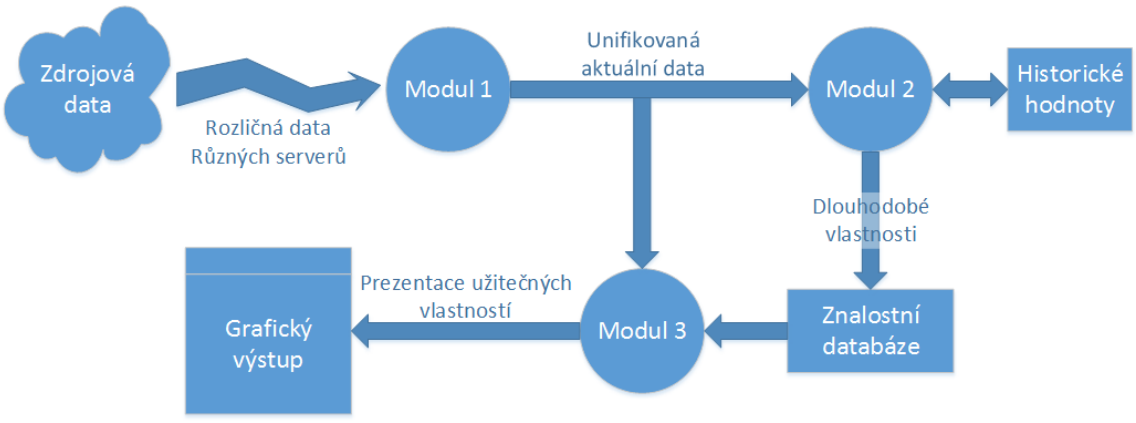
Veškeré algoritmy by měly být navrženy a implementovány tak, aby bylo možné snadné implementování na jiné podobné problémy s co nejmenší změnou kódu a tedy jako modulární části komunikující přes stanovené rozhraní s možností adaptace na jiné zdroje či cíle.

Výstupem samotných algoritmů pak musí být nejenom záznam o zpracovaných informacích, ale také přehledný grafický výstup znázorňující nejenom aktuální stav, ale i možnosti jeho využití s ohledem na minulost a predikci následného vývoje.

## 2.2 Metody a postupy

Implementační část práce bude rozdělena na minimálně tři části, kde základními prvky budou moduly:

- 1. modul bude řešit sběr informací od serverů (jednotlivých směnáren) a jeho výstupem budou data v jednotném formátu pro analýzu a zpracování dalšími částmi.
- 2. modul bude zpracovávat a vyhodnocovat data od prvního modulu za účelem získání informací o dlouhodobém stavu a vlastnostech jednotlivých protokolů měn a směnáren.
- 3. modul bude kombinovat aktuální data z prvního modulu se znalostmi dlouhodobých vlastností produkovaných druhým modulem a jeho výstupem bude jak textová tak grafická reprezentace vztahů a možností jejich využití.




Obrázek 2.2: Znázornění interakce jednotlivých modulů.

# Kapitola 3

# Teoretická východiska práce

## 3.1 Prostředí

Tato kapitola se kromě prostředí a původu trhů alternativních měn zabývá také možnostmi a nejčastějšími způsoby interakce v rámci těchto trhů. Problémem je však, že tento trh je tak diverzifikovaný, že není možné popsat všechny vlastnosti do detailů a budou zde tedy nastíněny převážně hlavní principy a poté bude demonstrováno několik konkrétních případů pro lepší přehled o celkových možnostech.

### 3.1.1 Měny

Jednotlivé měny používané na internetových směnárnách jsou vytvářeny individui či skupinami s různými cíli a jejich vytváření je limitováno pouze časovou náročností jejich implementace tvůrci. Takovýchto měn se tedy klidně denně mohou objevovat desítky či stovky a takovéto měny se navzájem liší nejenom názvy, ale většinou i principy od způsobu emitovaní přes zabezpečení po privátní politiky transakcí. I přes všechny rozdíly se však snaží většina těchto měn zachovat **nezávislost**, **otevřenost**, **nevratnost** a **bezpečnost**. Nedodržování těchto nepsaných pravidel většinou vede k nedůvěře ze strany lidí a postupném zániku měny.

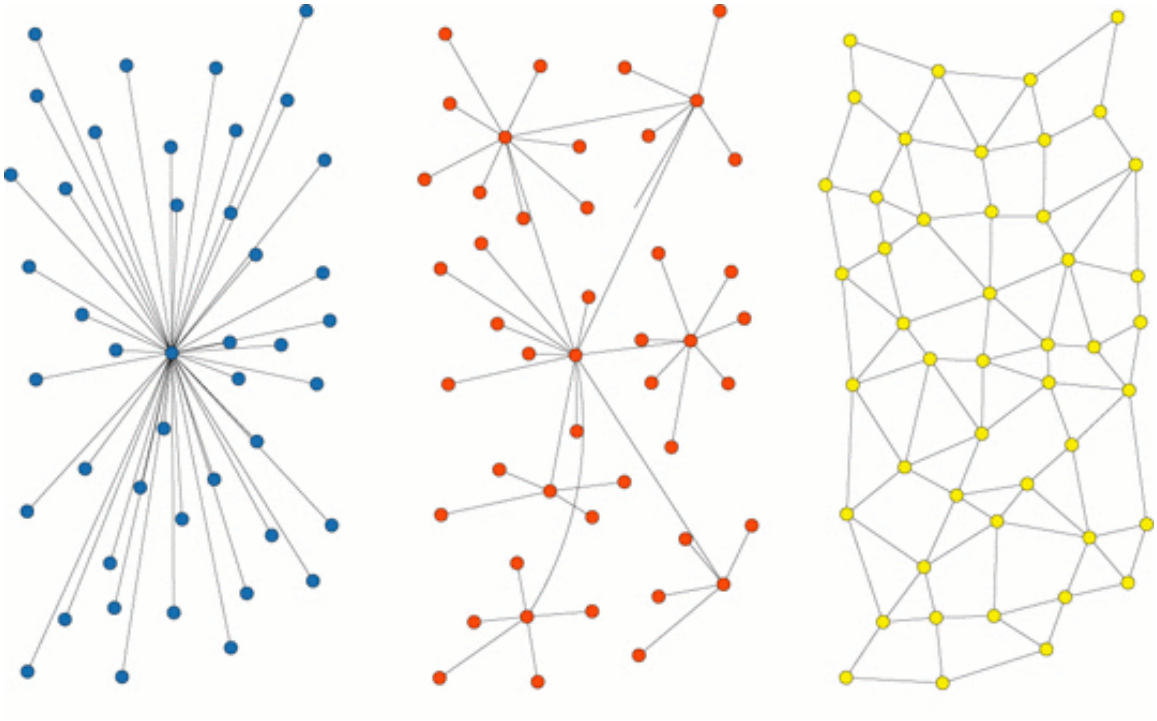
#### Nezávislost

Nezávislost měny je cílena na nezávislost její hodnoty na vládních jednotkách a hodnota je tedy určována poptávkou a nabídkou. U některých měn pak může být cena ovlivňována změnami emisních pravidel, či manipulací velkých sum na jednotlivých trzích (vytvářením neúměrně vysoké poptávky či nabídky). Takovéto měny se však tímto chováním většinou jeví jako dlouhodobě neperspektivní a lidé je používají maximálně jako tranzitivní, čímž měna ztrácí hodnotu.

#### Otevřenost

Otevřeností je myšleno hlavně zveřejnění zdrojových kódů samotného protokolu a případně i implementací klientů. Téměř standardem v oblasti těchto měn je anonymita autora či autorů a důvěra se tedy vkládá jen v samotný kód, který si uživatelé mohou prověřit sami.

Většina měn v rámci otevřenosti využívá distribuovaných sítí, tedy sítí bez centrální autority, která by mohla ovlivňovat provoz a obsah dat (obrázek 3.1 vpravo - žlutá barva).



Obrázek 3.1: Typy síťového propojení, zleva: centralizovaná (modře), decentralizovaná (červeně) a distribuovaná (žlutě) [24].

Takovéto sítě bývají implementovány na P2P protokolech, tedy Klient-Klient (Peer-to-Peer), kde se klienti propojují mezi sebou. Výhodou takovýchto sítí je nezávislost na výkonu centrálních serverů a vyšší odolnost vůči standardním útokům.

### Nevratnost

Na rozdíl od běžných transakcí, ať už bankovních či fyzických, elektronické měny zpravidla používají algoritmy, které zajišťují nemožnost stornování platby jak odesílatelem tak třetí stranou. Pokud tedy platba jednou odešla, tak jsou jednotky odečteny a pouze samotný příjemce s nimi může nakládat dále. Tento model je považován za důvěryhodnější kvůli implicitní nedůvěře v odesílatele (zpravidla anonymního), aby po nenávratném doručení smluvního zboží nemohl platbu stornovat.

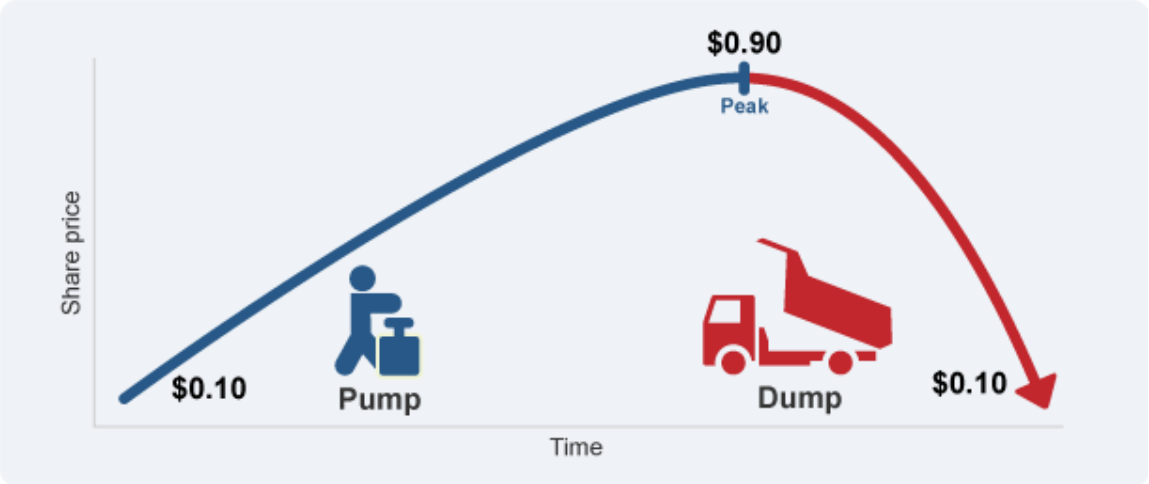
Díky nenávratnosti je tedy na plátci, aby si dopředu prověřil důvěryhodnost příjemce či využil třetí důvěryhodnou stranu jako arbitra, který nenávratně přijme platbu od plátce a teprve po úspěšné či neúspěšné transakci nasmlouvaných služeb nenávratně pošle peníze příjemci či zpět plátci.

### Bezpečnost

Bezpečnost elektronických měn je zpravidla založena na kryptografických asymetrických algoritmech a tedy na nepravděpodobnosti. Samotné algoritmy bývají řádově bezpečnější než ty používané dnes na zabezpečení ve světových bankách, avšak problémy nastávají v takzvaném „back-door“ přístupu tedy obejití zabezpečení jinou cestou. Částečně se tomuto

dá zabránit zveřejněním zdrojového kódu před využitím a vypsáním odměn za nalezení takovýchto možností.

Většina měn nedosáhne využití více než pár lidmi a zanikne stejně rychle jako se objevila. Toto je způsobeno především špatným či žádným marketingem a neschopností měny přinést něco nového, co by lidi zaujalo. Nemalá skupina měn se také stává obětmi „Pump and Dump“ strategie, kdy jedinec či skupina cíleně zvyšují poptávku skupováním dostupné měny často doprovázené dezinformací o budoucí prosperitě právě této měny. Toto většinou vyústí v zájem dalších lidí a růst ceny měny na několikanásobek původní hodnoty. Po dosažení určité hranice ceny za jednotku, pak iniciující jedinec či skupina začne s masivním odprodejem zásob, což vede ke strmému pádu kurzu, ztrátě důvěry v budoucnost měny od ostatních lidí a většinou i totální devalvaci měny.




Obrázek 3.2: Znázornění strategie „Pump and Dump“ [3].

### 3.1.2 Elektronické směnárny

Elektronické směnárny vznikly jako prostředí pro ulehčení vstupu či výstupu mezi tradičními a alternativními měnami, avšak se velmi rychle staly spekulativním prostředím s podobnými principy jako tradiční měnové trhy.

Elektronické směnárny slouží nejenom ke směnám tradičních měn s měnami „virtuálními“, ale i ke směnám virtuální - virtuální měna, což se mimo jiné využívá například pro rychlejší přesun hodnoty přes výkonnější protokol či pro diverzifikaci a snížení rizika.

K založení elektronické směnárny není potřeba téměř nic kromě webové adresy a hostingového serveru. Samotné implementace směnáren se dají dohledat dostupné na internetu a stačí je pak už jen nakopírovat a vhodně nastavit. Díky takto relativně jednoduché možnosti zřízení směnáren jich je na internetu, dle očekávání, velmi mnoho. Najít mezi nimi ty, které jsou důvěryhodné je pak samozřejmě mnohem náročnější.

#### Důvěryhodnost

Důvěryhodnost jednotlivých směnáren se stanovuje velmi obtížně, jelikož anonymita je brána téměř jako standard a dopátrat se skutečného majitele či provozovatele je většinou téměř nemožné. Zatímco člověk zkušený v tomto oboru má schopnost rozpoznat některé směnárny založené s podvodným úmyslem, pro nově příchozího člověka do tohoto sektoru to bývá úkol většinou nadlidský.

| Name | Trading Pairs | Total Volume | Logarithmic |
|---|---|---|---|
| BTCChina | 2 | 56,071.60 BTC | |
| OKCoin | 4 | 35,196.75 BTC | |
| Bitfinex | 6 | 34,208.15 BTC | |
| BTC-e | 23 | 9,534.12 BTC | |
| CEX.IO | 7 | 5,990.23 BTC | |
| Kraken | 29 | 3,371.05 BTC | |
| BTC38 | 53 | 2,428.39 BTC | |
| hitbtc | 20 | 1,968.13 BTC | |
| Bter | 179 | 1,635.38 BTC | |
| Cryptsy | 469 | 1,063.10 BTC | |

| Name | Trading Pairs | Total Volume | Logarithmic |
|---|---|---|---|
| OKCoin | 4 | 170,341.44 BTC | |
| CEX.IO | 31 | 48,092.87 BTC | |
| Poloniex | 335 | 15,322.35 BTC | |
| Bitfinex | 8 | 13,821.63 BTC | |
| Gatecoin | 6 | 10,959.16 BTC | |
| Bittrex | 773 | 2,058.18 BTC | |
| BTC38 | 62 | 1,696.59 BTC | |
| The Rock Trading | 15 | 582.58 BTC | |
| Vaultoro | 1 | 407.51 BTC | |
| hitbtc | 23 | 374.11 BTC | |

Obrázek 3.3: Deset nejobsáhlejších směnáren dle serveru [9] k 6.1. 2015 (vlevo) a stejný server k 25.3.2016 (vpravo).

Jednou z možností jak získat základní přehled o směnárnách používaných dalšími uživateli (a tedy směnárny, kterým důvěřuje větší množství lidí), je sledovat seznamy zveřejňující tržní kapitalizaci jednotlivých směnáren, tedy hodnoty vyjadřující množství obchodovaných měn a jejich hodnotu. Příkladem takových seznamů mohou být například [10], [8] nebo [9]. Samotné seznamy jsou zpravidla řazeny dle celkové USD či BTC hodnoty (jak lze vidět na obrázku 3.3).

Samotná velikost směnárny však nezaručuje její bezpečnost (viz. směnárna Mt. Gox - kapitola 4.2.1) a je proto vždy vhodné diverzifikovat, aby v případě pádu jedné směnárny nepřišel uživatel o veškeré úspory. Z dlouhodobého hlediska je pak samozřejmě výhodnější v době, kdy nejsou směny prováděny, mít měny uloženy na vlastním zabezpečeném úložišti.

### 3.1.3 Poplatky

Prostředí alternativních měn se zpravidla vyznačuje velmi nízkými poplatky za operace. Tyto poplatky v rámci jednotlivých měn většinou bývají stanoveny hlavně kvůli zabránění zbytečného zatěžování sítě velkým počtem bezvýznamně malých transakcí. V případě jednotlivých směnáren jsou pak poplatky za směnu tlačeny konkurenčním bojem na částky pohybující se většinou v rozmezí 0 až $0,25\%$.

### 3.1.4 Rizika

Nízké poplatky v tomto prostředí jsou však kompenzovány vysokým rizikem, kdy kurzy jednotlivých měn se mohou měnit o desítky procent denně a směnárny se většinou objevují

a zanikají bez předchozího varování.

## 3.2 Predikce vývoje kurzu

Předpokladem úspěšné dlouhodobé strategie je schopnost levně nakupovat a draze prodávat. Tedy přesněji prodávat za vyšší cenu než bylo nakoupeno, kde aktuální cena není tak důležitá jako informace zdali cena poroste či klesne v budoucnu.

Návodem k úspěchu se tedy stala potřeba predikce vývoje kurzu, kde kurz je tvořen nabídkou a poptávkou (regulovanými kurzy se zabývat nebudeme). Takováto nabídka a poptávka je však ovlivňována řadou faktorů od historie vývoje, přes ekonomické situace jednotlivých subjektů až po faktory jako politická situace a počasí. Kvantifikovat tedy všechny faktory je prakticky nemožné a praxe ukázala, že je možné činit kvalifikované odhady na základě předchozího vývoje ceny. Jedním z prvních větších zdokumentovaných obchodníků rozpoznávajících určité opakující vzory byl Homma Munehisa [21].

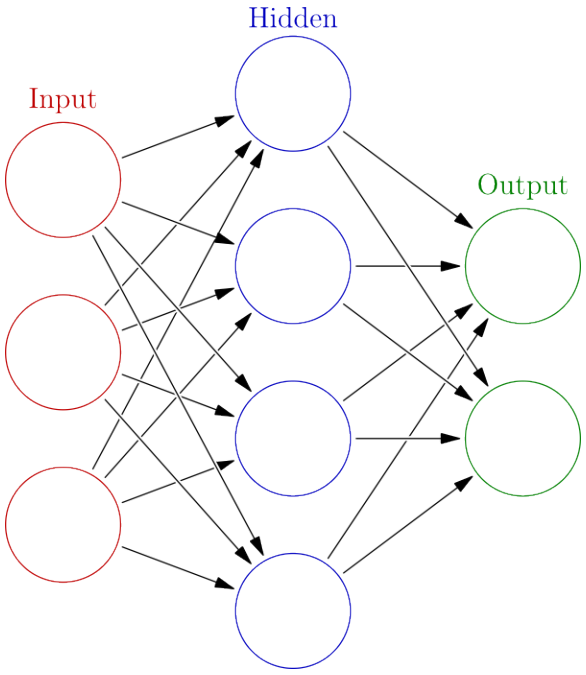
Obrázek 3.4: Homma Munehisa - „otec“ účelového obchodování a analyzování trhu [12].

Munehisa napsal rozsáhlé dílo odhalující způsob jeho obchodování na základě vývoje ceny a předpokladu, jak se zachovají ostatní lidé na trhu, tedy byl schopen předvídat změny poptávky a nabídky, které ovlivňovaly ceny. Podobně v průběhu vývoje a rozmachu burzovních trhů řada dalších lidí byla schopna identifikovat určité znaky v průběhu vývoje kurzu, díky kterým byli či stále jsou schopni předpovědět vývoj ceny komodity. Tato schopnost dnes již většinou přechází na výpočetní techniku, která se snaží emulovat různé části procesu rozpoznávání vzorů lidí a případně i hledat nové způsoby rozpoznávání, které jsou pro lidi nemožné.

Jednou z technik umožnující hledání a rozpoznávání vzorů jsou umělé neuronové sítě.

### 3.2.1 Umělé neuronové sítě

Umělé neuronové sítě se snaží napodobit proces „přemýšlení“ biologických organizmů. Jejich standardní topologií je rozdělení do tří vrstev (vyobrazeno na obrázku 3.5) na vstupní (input - červeně), skrytou (hidden - modře) a výstupní (output - zeleně).




Obrázek 3.5: Příklad topologie umělé neuronové sítě [33].

Vstupní vrstva slouží jako zdroj externích informací, kde na každý neuron je napojená jiná informace. Za vstupní vrstvou následuje skrytá vrstva, která může obsahovat i více řad dopředných neuronů, či jiné propojení. skrytá vrstva nemá vazby na externí okolí sítě a využívá pouze informace ze vstupní vrstvy, případně od sama sebe. Poslední vrstvou je výstupní vrstva, která stanovuje výsledky neuronové sítě na základě předchozích vstupů.

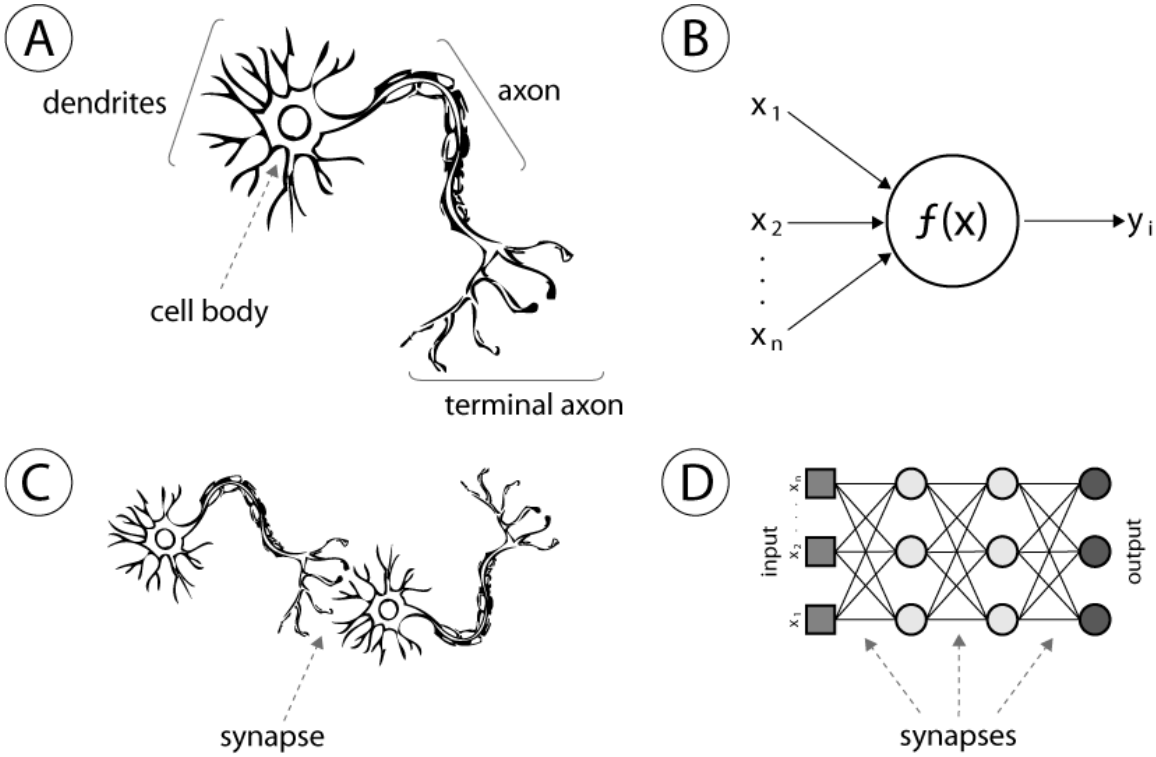
#### Umělý neuron

Jednotlivé prvky neuronové sítě se nazývají neurony. Umělé neurony se snaží emulovat biologické neurony, tedy na základě vstupů (u biologického neuronu - „dendritů“ ) vyhodnotit vstupní signály - většinou váženou sumou (část „soma“ u biologického neuronu) a stanovit výstup (u biologického neuronu část „axon“).

Výstup takové simulace neuronu pak můžeme zapsat jako [34]:

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right), \tag{3.1}$$

kde $m + 1$ je celkový počet vstupů, $x_0$ až $x_m$ značí jednotlivé vstupy a $w_0$ až $w_m$ příslušné váhy k jednotlivým vstupům. $\varphi$ následně značí aktivační funkci, která zpravidla má výstup v rozsahu 0 až 1. Vstup $x_0$ zpravidla bývá takzvaný *bias*, tedy speciální vstup na vyvažování, který většinou má hodnotu na konstantě 1 a celkově je tedy počet proměnných vstupů $m$.



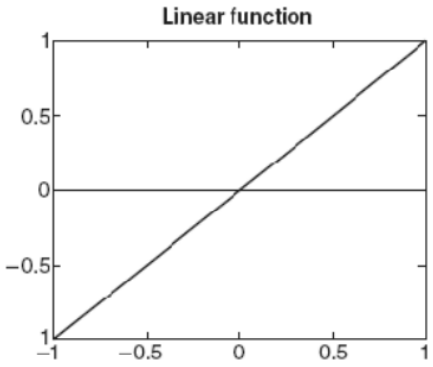

Obrázek 3.6: Porovnání neuronů: A) Biologický neuron, B) reprezentace umělého neuronu, C) propojení biologických neuronů a D)propojení umělých neuronů [20].

### Aktivační funkce

Aktivační funkce $\varphi(x)$ umělých neuronů pak nejčastěji jsou:

- Lineární funkce, která převádí poměrově vstup na výstup.

$$\varphi(x) = x \tag{3.2}$$



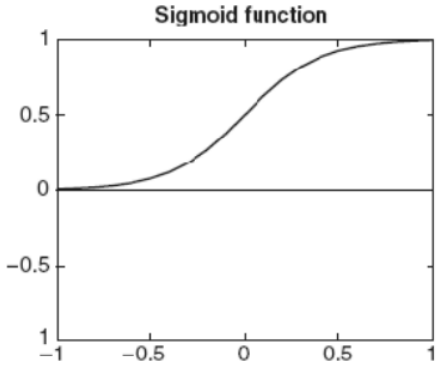

Obrázek 3.7: Lineární aktivační funkce [16].

Lineární aktivační funkce se využívají například často u vstupních neuronů, kde stačí suma vážených vstupů. Samotná suma pak již může být transponována pomocí bias parametru.

- Sigmoidní funkce - pojmenovaná podle tvaru křivky do písmene S

$$\varphi(x) = \frac{1}{1 + e^{-x}}. \tag{3.3}$$

Sigmoidní aktivační funkce je jedna z relativně jednoduchých nelineárních funkcí.




Obrázek 3.8: Sigmoidní aktivační funkce [16].

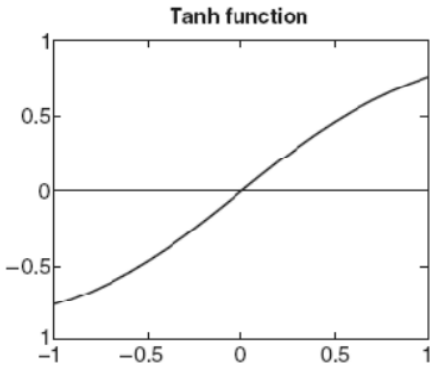
Tato aktivační funkce má výhodu jednoduché derivace, která se využívá pro výpočet vah, například velmi často využívaná u backpropagation sítí.

$$\frac{d\varphi(x)}{dx} = \varphi(x)(1 - \varphi(x)) \tag{3.4}$$

- Funkce hyperbolického tangentu

$$\varphi(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.5}$$

Funkce hyperbolického tangentu je další variantou sigmoidní funkce - její výhodou

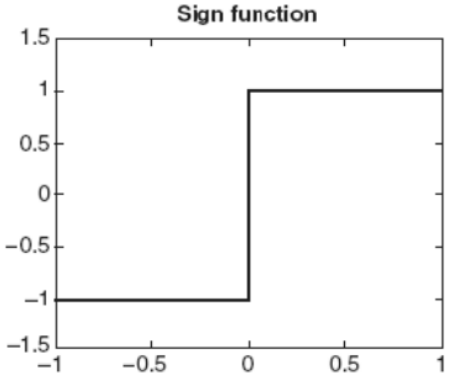



Obrázek 3.9: Aktivační funkce hyperbolického tangentu[16].

je nativní podpora na některých procesorech a tím i výrazné zrychlení některých algoritmů, využívajících tuto aktivační funkci.

- Skoková funkce určuje binární výstup na základě určité hranice $\theta$ - nejčastěji 0.

$$\varphi(x) = \begin{cases} 1 & \text{pokud } x \geq \theta \\ 0 & \text{pokud } x < \theta \end{cases} \tag{3.6}$$




Obrázek 3.10: Skoková aktivační funkce [16].

Tato aktivační funkce se velmi často používá u výstupní vrstvy, tedy u neuronů, které rozhodují například o příslušnosti vstupů do určité skupiny, čímž efektivně fungují jako klasifikátory.

### Učení neuronových sítí

Umělé neuronové sítě se nejprve musí „naučit“ řešit daný problém. Standardní přístupy k učení neuronových sítí jsou učení s učitelem, učení bez učitele a zpětnovazební učení. Všechny tyto metody mají svoje výhody a nevýhody - zvláště pak v řešení určitých problémů.

- Učení s učitelem

Základním postupem učení neuronových sítí je metoda učení s učitelem. Požadavkem pro takovéto učení tedy je mít k učící sadě dat správné výsledky, které si může algoritmus kontrolovat (zeptat se učitele, jestli to má správně).

Tento typ učení je vhodný například na rozpoznávání vzorů - klasifikování objektů do příslušných skupin či aproximování dat na určitou hodnotu (regresní analýza). Tato metoda je taky často využívaná na sekvenční data jako například rozpoznávání řeči či gest.

Nevýhodou je hlavně nebezpečí takzvaného přeučení, které nastává hlavně u nedostatečných trénovacích dat či příliš komplexních sítích. Přeučení se pak projevuje neschopností správně zpracovat obecnější data a mělo by být detekováno během fáze validování sítě. Řešením problému přeučení zpravidla bývá snížení komplexnosti sítě,

či rozšíření testovací množiny (například přidáním šumu). Dalšími možnostmi pak jsou předčasné ukončení procesu učení (většinou na základě dosáhnutí spodní hranice dostatečné přesnosti) nebo průběžného testování oproti validačním datům.

- Učení bez učitele

  Pro metodu učení bez učitele se používají testovací data, která nemají jednoznačně určené, jaký má být správný výstup sítě. Požadovaný výstup je tedy hodnocen pouze funkcí chybovosti, kdy si algoritmus může ověřit jestli jde správným směrem.

  Učení bez učitele se často používá na problémy klasifikace, kde je znám například jen počet skupin. Úspěch takové klasifikace se pak většinou hodnotí podle průměrné vzdálenosti bodů od středu shluku či velikosti vzájemného překryvu. Dalšími problémy, kde se často učení bez učitele využívá, jsou odhady statistických distribucí a filtrování dat.

  Nevýhody algoritmu učení bez učitele jsou, že předem nemusí být zřejmé, jaké mají být správné výsledky. Toto je však zároveň i výhoda ve smyslu, že algoritmus může najít správné řešení, které zadavatel nečekal či nevěděl, že je možné.

- Zpětnovazební učení (Reinforcement learning)

  Zpětnovazební učení se využívá u problémů, kde nejsou či nemohou být trénovací data. Modely využívající zpětnovazební učení tak pracují přímo s datovými vstupy a snaží se minimalizovat celkové ztráty či maximalizovat celkový zisk.

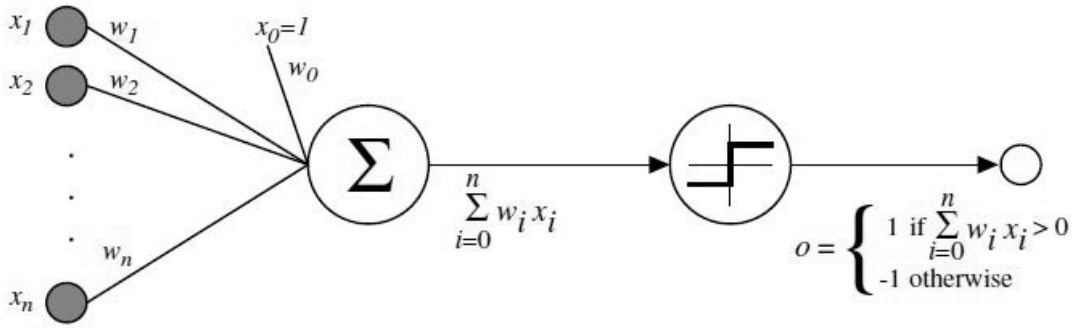
  Tento typ učení je vhodný na problémy typu „obchodní cestující“, případně na modely spadající do teorie her či problémy managementu zdrojů.

### 3.2.2 Dopředné neuronové sítě

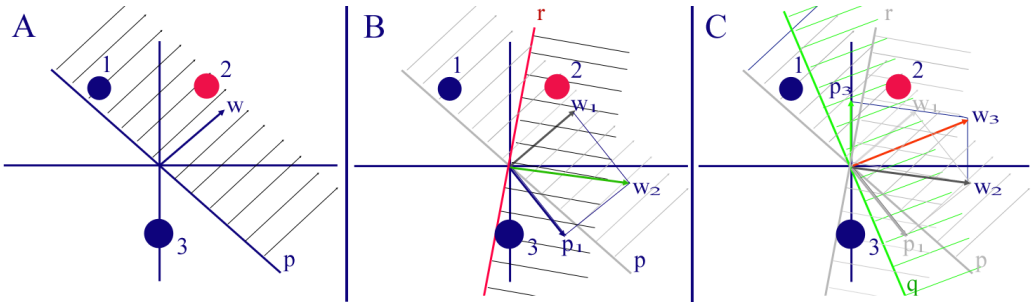
Základním typem umělých neuronových sítí jsou dopředné neuronové sítě. Tyto sítě se vyznačují topologií, kde každá vrstva může získávat informace pouze z vrstev předcházejících.

Základním členem této kategorie je jednovrstvý perceptron (Obrázek 3.11), který obsahuje jeden neuron s $n$ vstupy a jedním výstupem tvořeným skokovou funkcí na základě vážené sumy vstupů a bias prvku.

Učení perceptronu probíhá tak, že nejprve jsou náhodně iniciovány váhy vstupů a poté na základě správnosti výstupu v jednotlivých krocích aplikace testovacích vah jsou k vahám vstupů přičteny či odečteny vektory vstupů. Výsledkem je pak lineární rozdělení vektorů vstupů do binární klasifikace.




Obrázek 3.11: Jednovrstvý perceptron [29].




Obrázek 3.12: Znázornění učení perceptronu na třech testovacích bodech v dvourozměrném prostoru [38].

### 3.2.3 Systém Backpropagation

Pro složitější problémy je však důležité použít vícevrstvé sítě. Nejčastěji využívaný systém je takzvaný Backpropagation. Jedná se o vícevrstvou umělou neuronovou síť určenou pro rychlé adaptování na problémy učení s učitelem. Tyto sítě většinou používají sigmoidní aktivační funkci a nejsou tedy limitovány binární klasifikací jednotlivých neuronu.

Algoritmus učení sítě se pak dá popsat následovně:

- Výpočet výstupu:

  Nejprve se vezmou vstupy vstupní vrstvy a spočítají se jejich výstupy

  Následně se postupně počítají výsledky každé další vrstvy, kde jako vstupy jsou výsledky vrstvy předcházející.

- Zpětná propagace chyby:

  Výsledek reakce na vstup se porovná s požadovaným výstupem dle učitele.

  Rozdíl výsledku se použije na výpočet chyby v neuronech výstupní vrstvy.

  Pro každou další vrstvu od konce k začátku se porovná vypočítaný výstup a poměrový požadovaný vstup další vrstvy.

  Na základě chyb v jednotlivých neuronech se spočítají nové váhy.

- Nový test na hodnotu z testovací množiny, dokud všechny testovací vstupy nejsou klasifikovány správně nebo není dosaženo některého z dalších koncových kriterií.

Za předpokladu standardní sigmoidní aktivační funkce (vzorec 3.3) a tedy derivace dle vzorce 3.4 spočítáme chybu pro neurony výstupní vrstvy neuronu $j$ jako:

$$\delta_j = (o_j - t_j)o_j(1 - o_j), \tag{3.7}$$

kde $o_j$ je vypočítaný výstup neuronu a $t_j$ je požadovaný výstup neuronu.

V případě vnitřního neuronu se pak vzorec změní na:

$$\delta_j = (\sum_{k \in L} \delta_k w_{jk})o_j(1 - o_j), \tag{3.8}$$

kde $\delta_k$ je chyba neuronu v další vrstvě $K$, který je propojen přes váhu $w_{jk}$.

Jednotlivé váhy jsou pak nastaveny jako:

$$w_{ij} = w_{ij} - \alpha\delta_j o_i, \tag{3.9}$$

tedy nová váha vstupu z neuronu $i$ do neuronu $j$ bude váha $w_{ij}$ v minulém kroku mínus chyba $\delta_j$ neuronu $j$ vynásobena výstupem $o_i$ předchozího neuronu $i$ a koeficientem učení $\alpha$.

Stanovení koeficientu učení $\alpha$ určuje poměr aplikace chyby na změnu sítě v jednom kroku. Stanovení nízkého koeficientu má za následek velmi pomalé učení a neschopnost rozpoznat určité špatné příznaky sítě. Stanovení příliš vysokého koeficientu může mít za následek oscilaci protichůdných hodnot a celý algoritmus se stane nestabilním.

Pro stanovení učícího koeficientu existuje řada různých metod – převážně založených na okolnostech použití sítě a požadovaných výsledků. Některé metody dokonce využívají adaptivního koeficientu, který se mění mezi kroky, avšak popisování těchto metod by již bylo mimo rozsah této práce.

Problémem systému Backpropagation je, že využíváním postupné minimalizace chyby se může ustálit na lokálních minimech chyby, přičemž správným (nebo alespoň lepším) řešením sítě může být jiná varianta. Tento problém se většinou řeší více pokusy naučení sítě s různými počátečními váhami, kde je nakonec vybrána síť s celkově nejlepšími výsledky.

### 3.2.4 Zpětnovazební neuronové sítě

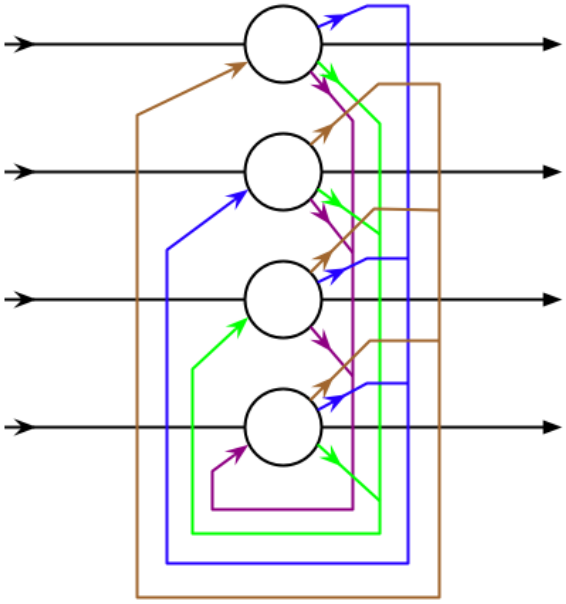
Zpětnovazební neuronové sítě (recurrent neural network – RNN) jsou typem umělých neuronových sítí, jež obsahují neurony mající vstupy od neuronů ve stejných či vyšších vrstvách též sítě čímž efektivně tvoří vnitřní cykly.

Zpětnovazební neuronové sítě mají díky vnitřním cyklům funkci obdobnou paměti, díky čemuž jsou tyto sítě velmi vhodné pro zpracovávání dat, jejichž informace jsou uvedeny v sekvencích. Příklady problémů řešící zpracování takových dat jsou mimo jiné například rozpoznávání rukopisů či řeči.

Obrázek 3.13: Příklad topologie zpětnovazební sítě: vlevo základní schéma, vpravo - vývoj sítě v čase [25].

### Hopfieldova síť

Jeden z původních konceptů zpětnovazební sítě byl popsán Johnem Hopfieldem v roce 1982. Hopfieldova síť není plně určena pro zpracování sekvenčních dat, ale ukazuje možnost použití vnitřních cyklů k hledání lokálních minim či maxim. [13]

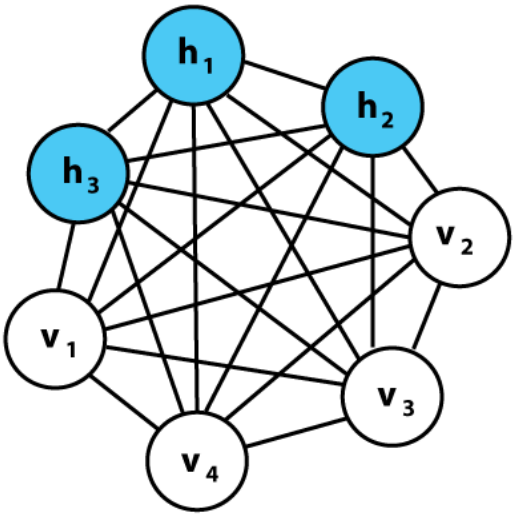


Obrázek 3.14: Znázornění Hopfieldovy sítě [36].

### Boltzmannovy stroje

Dalším příkladem sítí s vnitřním propojením jsou Boltzmannovy stroje, jejichž autory byli v roce 1985 Geoffrey Hinton a Terry Sejnowski [26]. Boltzmannovy stroje na rozdíl od Hopfieldových sítí pracují s pravděpodobnostmi a po ustálení vnitřní energie na minimu representují výsledek.

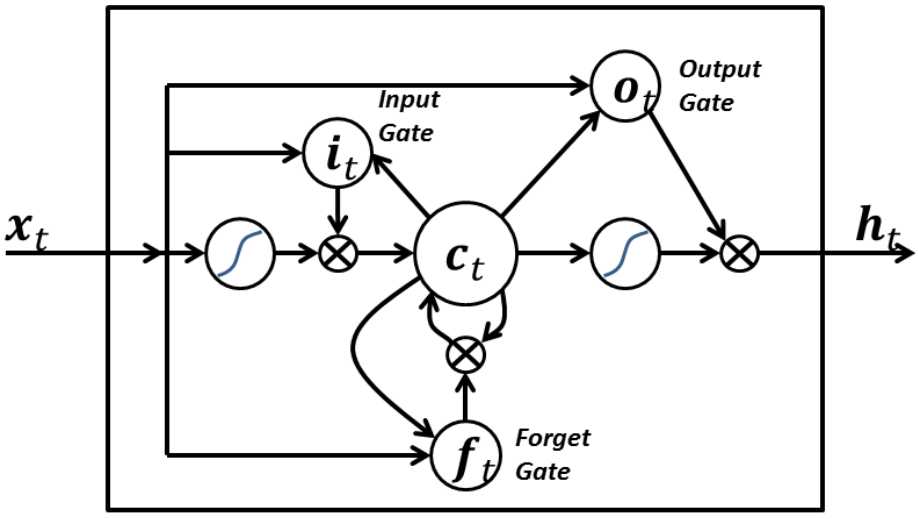
Boltzmannovy stroje byly navrhnuty s teoreticky velkými výkony a možnostmi, avšak praxe ukázala, že jsou velmi nestabilní pokud jsou zvětšeny na cokoli většího než triviální model. Vylepšení přinesly až Omezené Boltzmannovy stroje.




Obrázek 3.15: Příklad topologie Boltzmannovy stroje [35].

### LSTM sítě

LSTM (Long short-term memory) sítě navrhli Sepp Hochreiter a Jürgen Schmidhuber v roce 1997 [15]. Největší rozmach však tyto sítě zažívají od roku 2006, kdy se začaly hojně používat na rozpoznání řeči a zpracování zvuku a obrazu. Příkladem jejich využití jsou například nastavení algoritmů na rozpoznávání řeči čínským vyhledávačem Baidu [14] či vysoký výkon rozpoznání v aplikaci Google Voice [27].




Obrázek 3.16: Jednoduchý příklad LSTM [37].

LSTM sítě díky své schopnosti udržení informací regulací vstupů a výstupů jsou schopny provádět libovolné výpočty ekvivalentní možnostem konvenčních počítačů a jejich schopnost zpracovávat datové řady umožnila celou řadu přelomových algoritmů.

Samotné sítě jsou pak tvořeny jednotlivými bloky, které jsou schopny si zapamatovat konkrétní informaci po potřebnou délku času. Schopnosti zapamatování jsou pak řízeny několika branami a to nejčastěji:

- Vstupní bránou (Input gate) - určuje, kdy se informace na vstupu má zaznamenat do vnitřní paměti.
- Výstupní bránou (Output gate) - určuje, kdy se informace zaznamenaná uvnitř bloku má dostat na výstup.
- Zapomínací bránou (Forget gate) - působí jako vymazávací prvek v případě zjištění nepotřebné informace.

Učení LSTM sítí pak většinou probíhá za pomocí adaptovaného učícího algoritmu Back-propagation sítě. Na rozdíl od ostatních zpětnovazebních sítí zde nedochází k nechtěnému cyklení zpětně distribuovaných chyb a návratové chyby jsou promítány do funkce bran a tedy správně určují kdy, co a jak dlouho má být zadržováno v pamětech jednotlivých bloků.

# Kapitola 4

# Analýza problému

Zatímco předchozí kapitola se zabývala hlavně teoretickými základy pozadí, zde bude probráno několik konkrétních případů měn, směnáren a nástrojů pro vytvoření lepšího přehledu o minulém i aktuálním fungování některých prvků v tomto prostředí.

## 4.1 Analýza konkrétních měn

K analýze konkrétních měn byly vybrány hlavní zástupci tohoto segmentu trhu, tedy jmenovitě Bitcoin, Litecoin a Ripple. Dalšími aktuálně významnými měnami by například byly Stellar, DogeCoin nebo třeba Quarkcoin avšak popisovat celou řadu měn, které jsou zrovna v danou chvíli populární, není smyslem této práce.

### 4.1.1 Bitcoin

Bitcoin je decentralizovaný platební systém využívající klient-klient (P2P) komunikace navržený 31. října 2008 člověkem nebo skupinou označující se jako Satoshi Nakamoto [22]. Její základní jednotkou je jeden Bitcoin (BTC) aktuálně dělitelný na osm desetinných míst. Maximální počet vytvořených Bitcoinů je stanoven na 21 milionů avšak vzhledem k tomu, že Bitcoiny mohou být nenávratně ztraceny, odhaduje se, že minimálně 30% Bitcoinů bude již napořád mimo oběh.


Obrázek 4.1: Logo používané na reprezentaci Bitcoin měny [4].

Bitcoin je určen jako náhrada měn s nuceným oběhem, kde hodnota Bitcoinu je určena pouze nabídkou a poptávkou nikoli regulací. Bitcoin je pravděpodobně jedna z nejrozšířenějších alternativních měn s tržní kapitalizací 4 až 9 mld. USD (v roce 2014) a používá se jako symbol alternativních měn.

Svou nezávislost si Bitcoin udržuje sítí klient-klient (tedy bez centrálního serveru) a otevřenými zdrojovými kódy, kde si každý může přečíst implementaci jednotlivých částí

protokolu a navrhnout změny. Případné změny jsou pak schvalovány konsensem developerů. Nové Bitcoiny jsou emitovány takzvaným minerům za řešení kryptografických úkolů spojených s ověřením transakcí. Samotná hodnota odměn se většinou pohybuje na úrovni ceny energie spotřebované na výpočet řešení problému.

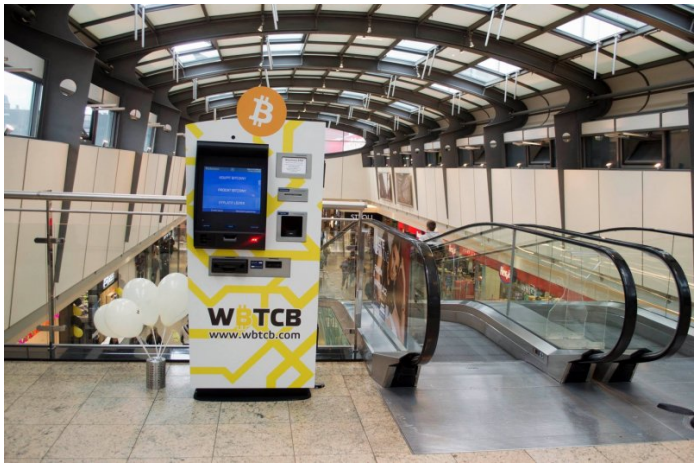
Zabezpečení Bitcoinové sítě a „peněženek“ uživatelů je prováděno pomocí asymetrických 256ti bitových klíčů - veřejný/privátní, kdy si každý uživatel může vygenerovat libovolný počet peněženek. Pravděpodobnost kolize klíčů (tedy přístup do peněženky někoho jiného) je udávána jako $2^{160}$ tedy zhruba $1.4 * 10^{48}$ - jeden a půl oktiliónu. Bezpěčnost Bitcoinu tedy prakticky není ohrožena útokem na samotný protokol a terčem útoků se tedy stávají lidé, kteří si špatně zabezpečují počítač či jiné medium s přítomností Bitcoinů.

Samotná síť byla spuštěna v lednu roku 2009 a v průběhu roku 2010 byla provedena první platba za reálné zboží - člověk s označením „laszlo“ si koupil dvě pizzy za 10,000 BTC [6]. V lednu 2011 pak Bitcoin dosáhl parity s americkým dolarem a největšího dosavadního (květen 2016) maxima Bitcoin dosáhl 29. listopadu 2013 s hodnotou 1,242 USD za jeden BTC na burze „Mt. Gox“ [18]. Vývoj ceny dle burzy „Bitstamp“ za poslední dva roky lze vidět na obrázku 4.2.


Obrázek 4.2: Graf vývoje ceny USD-BTC (vlevo cena v USD za jeden Bitcoin, dole rozsah červen 2012 až květen 2016) [5].

Navzdory klesajícímu trendu roku 2014 přijímalo Bitcoin stále více obchodníků a to včetně softwarového giganta Microsoft, výrobce počítačů Dell či obchodních řetězců Overstock, Neweegg nebo Monoprix.

Zároveň se začali v roce 2014 a 2015 rozšiřovat automaty umožňující nákup Bitcoinů ve fyzickém prostředí. Příkladem takového automatu je například stroj v nákupním centru Vaňkovka v Brně – obrázek 4.3. Síť takovýchto automatů se neustále rozšiřuje a jejich lokace se dají zjistit například z internetových seznamů jako Coin ATM Radar [7].




Obrázek 4.3: Automat na nákup Bitcoinů umístěný v Brně v OC Vaňkovka [7].

### 4.1.2 Litecoin

Litecoin byl vytvořen jako doplněk či dokonce náhrada měny Bitcoin. Většina principů (i zdrojových kódů) je převzata z měny Bitcoinu avšak přírůst nových Litecoinů je limitován nastavováním obtížnosti objevování na konstantní počet snižující se o polovinu každé čtyři roky. Finální počet Litecoinu by měl být 84 milionů. [30]


Obrázek 4.4: Logo používané na reprezentaci Litecoin měny [30].

Ke dni 29. prosince 2014 má jeden Litecoin (LTC) hodnotu 2,70 USD a tedy pro běžné denní transakce se jeví jako přátelštější měna než Bitcoin, kde se kvůli vysoké ceně transakce zpravidla odehrávají v desetinových či dokonce setinových částech Bitcoinu.

Litecoiny je možné využít jako výměnné medium či k nákupu v určitých obchodech (na rozdíl od Bitcoinu se jedná zpravidla o malé obchody) jejich výběr je dostupný například na [19].

### 4.1.3 Ripple

Ripple je protokol umožňující směnu a převod různých měn a komodit mezi uživateli za minimální poplatky. Ripple byl vytvořen a je spravován firmou Ripple Labs a síť běží na distribuované soustavě serverů avšak zdrojový kód protokolu je otevřený.

Vlastní měnou pro transakce je jednotka XRP pohybující se na zhruba na hodnotě 42 XRP za jeden USD (ke 29. prosinci 2014). Samotné XRP byly předgenerovány firmou Ripple labs v množství 80 miliard (z toho zatím jen 55 miliard je určeno pro distribuci do oběhu) a nyní jsou distribuovány zájemcům za stanovenou sumu.


Obrázek 4.5: Logo používané na reprezentaci ripple [32].

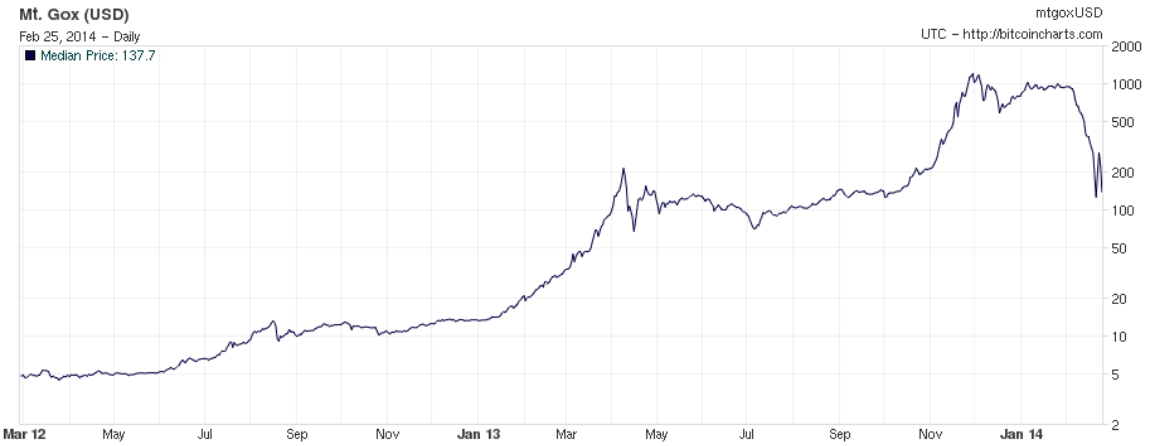
## 4.2 Analýza internetových směnáren

Jak již bylo zmíněno dříve, tak prostředí elektronických směnáren je velmi rozličné a pro lepší uvedení do reality si uvedeme několik významných případů elektronických směnáren a to konkrétně jednu z nejvýznamnějších směnáren v historii Bitcoinu - směnárnu Mt. Gox a poté několik dalších směnáren, které měly nebo stále mají vliv na vývoj alternativních měn.

### 4.2.1 Mt. Gox

Směnárna Mt. Gox byla založena 18. července 2010 programátorem Jed McCaleb a následně ji 6. března 2011 koupil Mark Karpelès v internetové komunitě figurující jako „MagicalTux“.

Směnárna Mt. Gox se i přes různé legální a finanční problémy stala největší výměnnou sítí Bitcoinů a v roce 2013 přes ní probíhalo 70% všech Bitcoinových transakcí. Na této směnárně se také v roce 2013 prodávaly Bitcoiny za historicky nejvyšší hodnotu a to 1,242 USD za jeden Bitcoin.




Obrázek 4.6: Graf vývoje ceny na směnárně Mt. Gox před jejím uzavřením. (Na levé straně logaritmická stupnice ceny jednoho BTC za USD, dole roky 2012 až 2014) [31] k 6.1. 2015.

#### Bankrot

Ke konci roku 2013 někteří uživatelé začínali hlásit problémy při snaze převést peníze ze směnárny pryč. 7. února 2014 pak byly oficiálně zastaveny možnosti převést Bitcoiny mimo směnárnu a 10. února vyšla oficiální zpráva, že tak bylo učiněno kvůli údajné chybě v Bitcoinovém softwaru [11].

17. a 20. února následovaly další tiskové správy od směnárny Mt. Gox oznamující, že na problému se stále pracuje a že lidé svoje peníze dostanou zpět. 23. února Mark Karpelès odstoupil z předsednictva neziskové organizace Bitcoin Foundation [28], která má misi: „Standardizovat, chránit a propagovat využívání měny Bitcoin pro prospěch uživatelů na celém světě“.

Den poté, 24. února 2014, nejprve směnárna Mt. Gox pozastavila veškeré obchodování a o pár hodin později přestaly fungovat i jejich internetové stránky. Zároveň se na veřejnost dostal údajný interní dokument odhalující jako problém chybějících skoro 800 000 Bitcoinů (v té době ekvivalent zhruba 480 mil. USD).

Směnárna Mt. Gox se z těchto problémů již nikdy nedostala a v následujících dnech už jen oficiálně zažádala o ochranu před věřiteli, nejprve 28. února v Japonsku a 9. března v USA.

### 4.2.2 BTC China

Směnárna BTC China je aktuálně (leden 2015) jedna z největších směnáren s virtuálními měnami a to zejména Bitcoinem. Je to první Čínská směnárna založená v červnu roku 2011 a obchodovat se na ní dá ve třech párech: Bitcoin-Yen, Litecoin-Yen a Bitcoin-Litecoin.

Dvacetičtyřhodinový objem směňovaných měn se pohybuje okolo 120 tisíc Bitcoinů a 110 tisíc Litecoinů (tedy dle aktuálního kurzu zhruba 34 mil. USD za den). Předpokládá se, že majorita uživatelů jsou čínské národnosti a to i přes to, že Bitcoiny jsou v Číně pro obchodní transakce zakázány.

#### Datový přístup

Server BTC China nabízí možnost strojového přístupu přes autentizované API (application program interface) s možností jak sledování aktuálního obchodu, tak i možností zadávat transakce.

### 4.2.3 BTC-e

Směnárna BTC-e byla spuštěna 7. srpna 2011 a je zaměřena primárně na anglicky a rusky mluvící komunitu. Ke směně je dostupných 23 měnových párů včetně Bitcoin ku Americkému dolaru, Ruskému rublu či Euru.

Dvacetičtyřhodinový objem směňovaných měn se většinou pohybuje zhruba na 12 tisících BTC (tedy 3 mil. USD).

#### Datový přístup

API rozhraní serveru BTC-e umožňuje strojový přístup k aktuálním obchodům a umožňuje i zadávání transakcí, avšak již není možné zadávat pokyny k přesunu měn mimo server.

### 4.2.4 Cryptsy

Směnárna Cryptsy byla založena 20. května 2013 na Floridě v USA společností „Project Investors, Inc.“. Její dvacetičtyřhodínový obrat byl oproti výše zmíněným výrazně menší a to zhruba 1 200 BTC (300 tisíc USD) avšak nabízela k obchodování 472 párů převážně alternativních měn, čímž na trhu plnila hlavní převodní směnárnu pro spekulativní obchody do menších měn a převod zdrojů zpět do stabilnější části sektoru.

#### Datový přístup

API rozhraní serveru Cryptsy, podobně jako server BTC-e umožňovala přístup k aktuálním datům a možnost zadávat transakce. Díky tomu se stala populární pro řadu obchodovacích skriptů.

#### Uzavření

Server Cryptsy byl velmi populární mezi jak novými uživateli, tak dlouhodobými obchodníky, jeho osud se dá shrnout v pár bodech.

- 7. listopadu 2015 byl na serveru **redit** [23] zveřejněn článek uživatele pod jménem Otohs popisující bezdůvodné útrapy při snaze převést větší množství peněz v měně Bitcoinů ven ze serveru.
- 22. listopadu 2015 server vydal oficiální prohlášení o interní chybě, kvůli které musí dočasně zmrazit všechny měny.
- 24. listopadu 2015 vydali další prohlášení oznamující údajný útok na server v podobě distribuovaného odepření služby (DDoS), na základě kterého není možné přistupovat k některým stránkám serveru.
- Prosinec 2015 - na řadě serverů zabývajících se zprávami ze světa elektronických měn se objevují spekulace a varování o možném pádu serveru. Zároveň lidé upozorňují, že server po celou dobu stále přímá příchozí platby, umožňuje vnitřní transakce, ale neprovádí žádné odchozí platby.
- 5. ledna 2016 jsou pozastaveny už i vnitřní transakce.
- 9. ledna 2016 byla medii zveřejněna informace, že kanceláře serveru Cryptsy jsou prázdné a zaměstnanci k nedohledání.
- 13. ledna 2016 byla skupinou uživatelů podána hromadná žaloba v hodnotě škody 5 miliónu Amerických dolarů s návrhem na zmražení majetku firmy a exekuci.

Následkem probíhajícího soudu vyšlo najevo, že server utrpěl krádež 10 tisíce bitcoinů už v červenci roku 2014 v té době v hodnotě 6 miliónů Amerických dolarů. O ztrátě však neinformoval a snažil se doplnit chybějící finance riskantními obchody s majetkem uživatelů.

Celková škoda stále nebyla vyčíslena. Pro měnu Bitcoin tyto události neměly větší dopad, protože celkový poměr ztracených bitcoinů nebyl zas tak velký. Pro řadu malých měn, kde server Cryptsy byl prakticky jediný server, kde se s nimi dalo obchodovat, byly však tyto události likvidační.

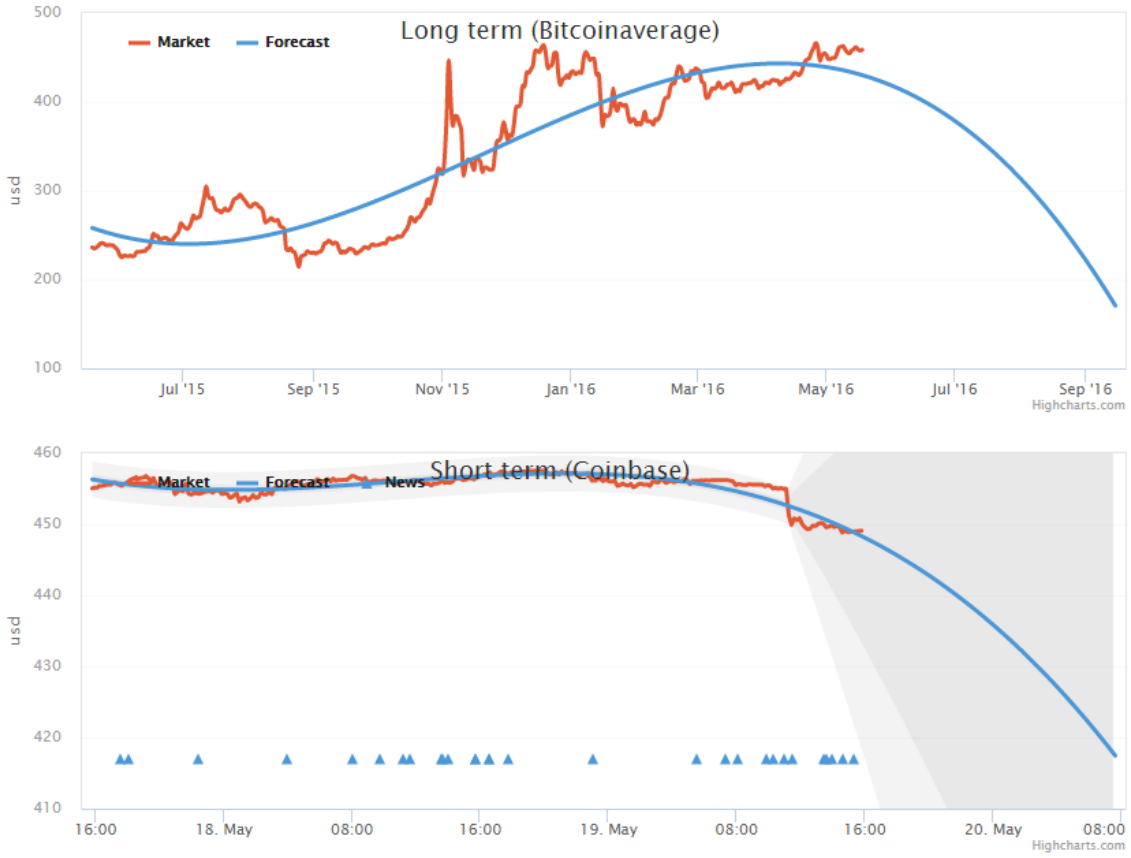
## 4.3 Dostupné nástroje

Vzhledem k aktuálnímu rozšíření alternativních měn je samozřejmostí i řada různých nástrojů a prostředků pro uživatele a obchodníky s těmito měnami. Jedná se hlavně o základní informační nástroje ale i obchodní systémy.

### 4.3.1 Informační nástroje

Virtuální měny jsou založeny na principu práce s informacemi a není tedy nic překvapivého, že v komunitě lidí, kteří s nimi pracují, je řada vývojářů, kteří tvoří nástroje generující různé grafy a výpisy pro prezentaci dalších získaných poznatků, které nemusejí být na první pohled vidět. Takovýchto nástrojů existuje celá řada, jak již bylo možné vidět například na obrázku 3.3 ze serveru CryptoCoin Charts, který v pravidelných intervalech generuje informace o dostupných měnách a směnárnách a umožňuje řadu různých grafických reprezentací jejich poměrů.

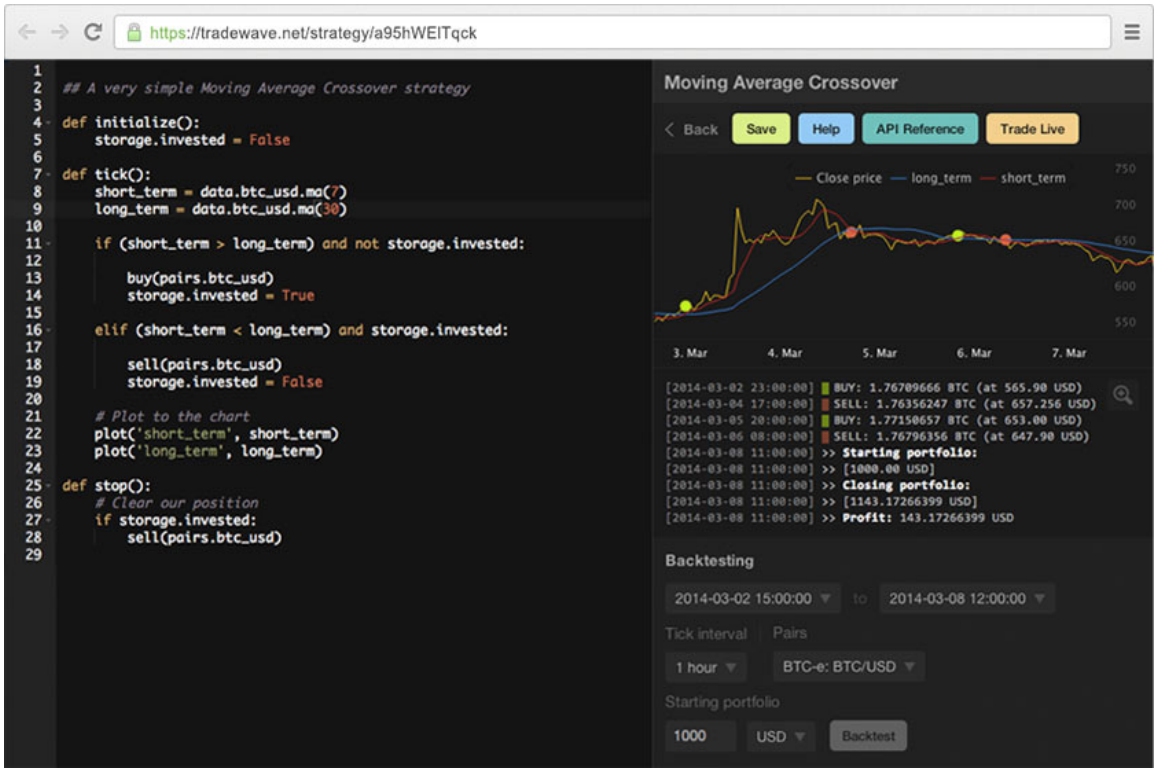
Dokonce jsou již dostupné nástroje generující predikce vývoje jako například server Bitcoin Forecast, který generuje krátkodobé a dlouhodobé předpovědi vývoje měny Bitcoin (na obrázku 4.7) a další verze stejného serveru s názvem Altcoin Forecast predikující čtyři menší měny.




Obrázek 4.7: Krátkodobá a dlouhodobá predikce vývoje ceny měny Bitcoin serverem Bitcoin Forecast [1].

### 4.3.2 Obchodní nástroje

Jedním ze zaběhnutých obchodních nástrojů je například komerční sytém od firmy Tradewave, který umožňuje automatické obchodování na několika burzách alternativních měn. Je zde možné například nastavit příkazy na různé typy transakcí či vytvořit si vlastní rozhodovací algoritmus pro obchodování. Problémem však zůstává trénování algoritmů, kde je Tradewave značně limitovaný a uživatel je nucen platit měsíční poplatky i když žádné transakce neprovádí.




Obrázek 4.8: Možnosti nastavení automatického obchodního systému Tradewave [2].

Samozřejmě existuje i řada dalších nástrojů, ale jejich výčet by byl příliš široký pro rozsah této práce. Navíc v době, kdy se tato práce dostane do rukou čtenáře, bude minimálně polovina z nich již nefunkčních či silně zavádějících. Hlavním problémem těchto nástrojů však zůstává, že jsou velice úzce specializované a neumožňují uživateli volnost, kterou poskytuje systém navrhovaný v rámci této práce.

# Kapitola 5

# Vlastní návrhy řešení

Jak již bylo popsáno v předchozích kapitolách, trh elektronických měn je velmi nestabilní a to z pohledu nejen jednotlivých komodit – elektronických měn, ale i z pohledu obchodovacích míst – elektronických směnáren.

Hlavní prioritou řešení této práce jsou tedy možnosti vysoké variability a přizpůsobivosti. Aplikace musí umět rychle reagovat na změny trhu a musí mít možnosti jak být přizpůsobena novým podmínkám v co nejkratším čase.

Součástí této práce jsou tedy dvě aplikace pracující s daty na alternativních měnových trzích. První aplikace se zabývá „výměnným obchodem“ (5.2) založeným na souběžném provedení více transakcí, čímž v podstatě eliminuje problémy dlouhodobého vývoje kurzu a využívá krátkodobých výchylek způsobených decentralizací. Druhá aplikace se pak zabývá více tradiční predikcí kurzu (5.5) na základě historických i aktuálních hodnot a především sledování vzájemných vztahů mezi jednotlivými burzy.

Pro obě aplikace je důležitý zdroj dat. Vzhledem k tomu, že informace o aktuálních kurzech a transakcích nejsou nijak standardizovány, je součástí této práce i knihovna zabývající se sběrem a interpretací dostupných dat.

## 5.1 Knihovna sběru dat

Samotný sběr musí probíhat v duchu práce a to s možností variability, tak aby uživatel měl možnost vstupovat na nové trhy s jednoduchostí a rychlostí převyšující možnosti zakládání nových směnáren. Přidání nové směnárny či měny by tedy mělo být časovou záležitostí minut, ne-li vteřin.

Knihovna zabývající se sběrem dat je koncipována jako co nejobecnější v rámci možností dolování dostupných dat z různých serverů a nachystaná na případná rozšíření vyplývající z nových faktů. Zpracovávané informace na jednotlivých serverech se pak dají rozdělit do dvou kategorií a to informace určené pro strojové zpracování (kapitola 5.1.1) a informace určené pro uživatele (kapitola 5.1.2).

Za účelem splnění výše zmíněných požadavků byla zvolena implementace knihovny v jazyce Python (2.7). Jazyk Python umožňuje snadné prototypování, možnosti změn za běhu a velmi snadnou přenositelnost na různé platformy a je tak ideální pro řešení daného problému.

### Popis knihovny

Struktura implementované knihovny vypadá následovně:

```
python
└── lib
    └── market
        ├── market.py
        ├── utils
        │   ├── json_resolver.py
        │   └── html_resolver.py
        └── markets
            ├── server1.py
            ├── server2.py
            └── ...
```

Hlavní třídou je soubor **market.py**, který slouží jako nadřazená struktura jednotlivých serverů – směnáren. Tato třída je pak děděna jednotlivými třídami serverů **markets/***server***.py**, kde je požadováno vyplnit:

- *name* - Jméno, jak bude server označován.
- *base_api* - Url adresa, odkud se budou dolovat data.
  Například pro server BTC China bude adresa:
  **base_api = "https://data.btcc.com/data/orderbook?market=**%*s***&limit=1"**,
  kde %*s* bude nahrazeno kódy jednotlivých párů měn.
- *codes* - Seznam měnových párů na serveru a jejich kódové označení pro zpracovávaný server, tedy například pro výše zmiňovaný server to bude:
  **codes** = {
  "**BTC-CNY**": '*cnybtc*',
  "**LTC-BTC**": '*btcltc*',
  "**LTC-EUR**": '*cnyltc*'
  }
- *template* – Zdrojový řetězec, či struktura udávající jak číst informace z konkrétního serveru. Podrobný popis tohoto parametru bude následovat v kapitolách 5.1.1 a 5.1.2.
- *data_type* – Parametr udávající formát dat. Parametr není povinné určovat a pokud není nastaven, považuje se za hodnotu "**JSON**", tedy za strojově zpracovatelná data stejnojmenné struktury.

Po řádném předchozím nastavení všech požadovaných parametrů stačí vložit novou třídu do libovolného skriptu a následně zavolat její metodu **.get_current_market_info(***from***,***to***)**. Tato metoda přijímá dva parametry *from* a *to* určující, jaké informace požadujeme, tedy z jaké měny do které.

Pokud daný server nemá měnový pár definovaný, knihovna vrátí prázdné pole. Pokud je na serveru definovaný pár v opačném směru, je vypsáno varování a knihovna následně vypočítá požadované kurzy z opačného páru a převede velikosti transakcí do druhé měny. Úspěšně zjištěná data jsou pak vracena ve struktuře:

{
    "**sell**": {
        "**price**": *price*,
        "**quant**": *quanity*
    },
    "**buy**": {
        "**price**": *price*,
        "**quant**": *quanity*
    }
}

Pokud to server vyžaduje, je možné také v definované třídě změnit chování metod:

- **__init__()** – Metoda volaná při inicializaci. Zde lze například stanovit změny v nastavení na základě nových hodnot, či zahájit sériový protokol komunikace.
- **unauthenticated_request**(*pair_code*) – Volání dotazu na server. Toto volání může být potřeba přepsat například v případě, kdy server vyžaduje speciální typ komunikace.
- **get_data**(*currency_pair*) – Vlastní metoda procesu získáváni dat.

Vytváření nových šablon serverů nemusí být vždy zcela intuitivní a z tohoto důvodu knihovna obsahuje možný parametr **debug**, který při nastavení na logickou hodnotu *True* vypisuje interní informace o procesu získávání dat. Tento parametr lze nastavit jak v jednotlivých záznamech pro vlastní servery, tak globálně ve třídě *market.py*.

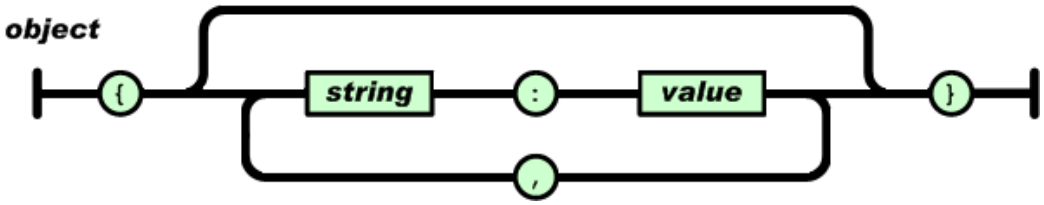
Samotné zpracování dat pak probíhá, jak již bylo zmíněno výše, dle typu dat. Zpracování jednotlivých typů je definováno na základě účelu a struktury podoby, ve kterých jsou data reprezentována. Samotné kódy pro jednotlivé zpracování jsou pak obsaženy v knihovně ve složce **utils**.

### 5.1.1 Strojově zpracovatelné informace

První kategorií dat jsou takzvané strojově zpracovatelné informace, tedy jsou to informace předem určené pro zpracování aplikacemi či skripty. Tyto informace bývají distribuovány pomocí aplikačních rozhraní – API a to většinou v podobě odpovědí na předem definované http dotazy. Nejčastější formou odpovědi pak bývá JSON soubor obsahující objekt s daty.

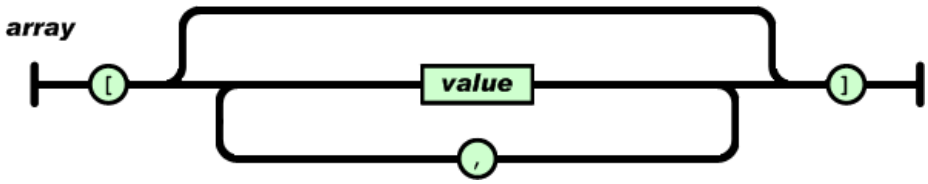
#### Zpracování JSON objektů

JSON objekty mají relativně jednoduchou avšak velmi silně vyjadřovací strukturu. Každý server si vybírá jiný způsob vyjádření a je tedy potřeba vytvořit systém snadného nalezení relevantních informací. Nejčastějším způsobem je uložení dat do objektu (obrázek 5.1).



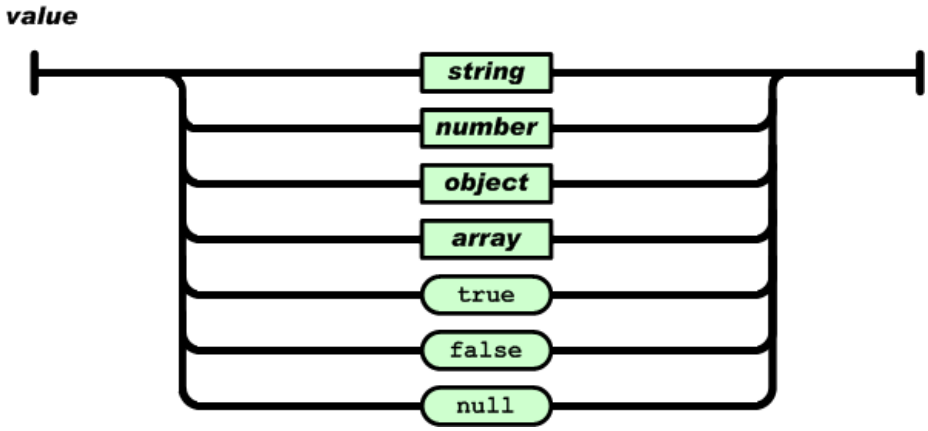

Obrázek 5.1: Struktura JSON objektu [17]

Uvnitř objektu je pak standardně k nalezení pole hodnot (obrázek 5.2)




Obrázek 5.2: Struktura JSON pole [17]

obsahující často buď přímo hodnoty (obrázek 5.3) nebo znovu objekty.




Obrázek 5.3: Možnosti JSON hodnoty [17]

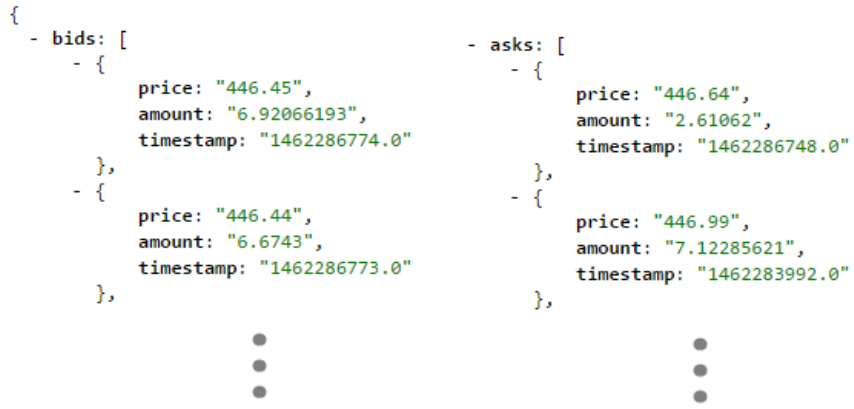
Zpracovávající skript se tedy musí umět probrat skrze tyto struktury a najít požadované hodnoty. Z tímto účelem byl v rámci této práce zvolen systém vzorových předloh definovaných na stejných principech JSON jako zdrojová data.

Vezmeme-li tedy například data z odpovědi serveru Bitfinex (`https://api.bitfinex.com/v1/book/BTCUSD`) na obrázku 5.4, můžeme z nich získat data za pomoci předlohy na obrázku 5.5, kde:

- Prefix **A_** respektive **B_** znamená data pro nabídku (ask) či poptávku (bid).
- Řetězce **PRICE** pak značí cenu za jednotku (kurz) a **AMOUNT** dostupné množství.

- Postfix `_N` pak značí formát jako číslo a `_T` formát jako text.

Tedy předlohy, která říká, že se má vzít objekt s názvem „bids“ a z něj položka – což je opět objekt obsahující aktuální kurz poptávky pod názvem „price“ a velikost této poptávky pod názvem „amount“. Celý proces se pak opakuje pro variantu s nabídkou – „asks“.




Obrázek 5.4: Data ze serveru Bitfinex.

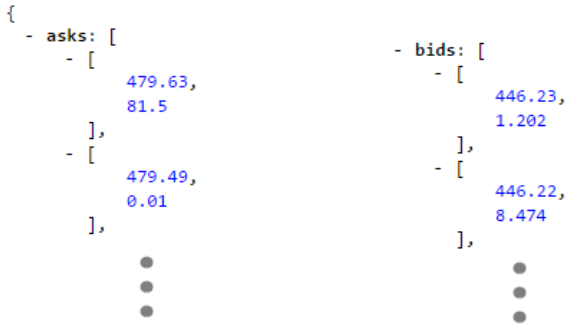
```
{
    "bids": [
        {
            "price": "B_PRICE_T",
            "amount": "B_AMOUNT_T"
        }
    ],
    "asks": [
        {
            "price": "A_PRICE_T",
            "amount": "A_AMOUNT_T"
        }
    ]
}
```

Obrázek 5.5: Šablona pro data ze serveru Bitfinex.

Obdobně například pro data ze serveru OKCoin (https://www.okcoin.com/api/v1/depth.do?symbol=btc_usd) na obrázku 5.6 s tím rozdílem, že nabídka je zde řazena od největší k nejmenší. Pokud tedy chceme získat informace o nejnižší nabídce, je potřeba získat poslední položku v seznamu, k čemuž slouží v šabloně (obrázek 5.7) sekvence `"...",` , která zaručí, že dekódovací skript přeskočí všechny záznamy až na vyžadovaný počet na konci.

### 5.1.2 Zpracování informací z textu

Sekundárním zdrojem informací jsou data ze stránek určené pro běžného čtenáře, tedy data která nebyla určena pro strojový přístup. Tuto variantu je potřeba aplikovat u serverů,




Obrázek 5.6: Data ze serveru OKCoin.

```
{
    "asks": [
        "...",
        ["A_PRICE_N", "A_AMOUNT_N"]
    ],
    "bids": [
        ["B_PRICE_N", "B_AMOUNT_N"],
        "..."
    ]
}
```

Obrázek 5.7: Šablona pro data ze serveru OKCoin.

které nebyly schopny zveřejnit svoje aplikační rozhraní či u serverů, které svoje rozhraní schovávají ze strachu přetížení dotazy od skriptů.

V těchto případech se knihovna vytvořená pro tuto práci maskuje jako běžný lidský uživatel. Tedy navštíví na pozadí stránku pod hlavičkou běžného prohlížeče a stáhne celý její obsah a následně v textu vyhledá požadované hodnoty.

Vyhledávání požadovaných hodnot probíhá za pomoci formátovacích značek HTML jazyku, které jsou nejčastěji využívány pro reprezentaci stylu stránky. Script nejčastěji nejprve najde nejbližší specificky označený prvek na stránce nejblíže k hledaným údajům a následně postupuje po struktuře dokumentu k cíli.

Na příkladu výřezu stránky serveru BTC-e `https://btc-e.com/exchange/btc_usd` (obrázek 5.8) můžeme vidět takovou strukturu. Script v takovém případě:

- nejdříve vyhledá blok **div** s identifikátorem **id='orders_1'**,
- následně přistoupí k prvnímu dceřinému bloku **table**,
- z něhož přejde na dceřiný blok **tr**, který v tomto případě obsahuje hlavičky a je tedy potřeba přistoupit k jeho následovníku, kde
- příkazem **next_sibling** skript přejde na správný blok dat.
- Následně se opět zanoří do dceřiného bloku **td**

- a přečte obsah příkazem **`text`**.

Celý proces tedy záleží na struktuře stránky, avšak je snadno definovatelný v rámci deklaračních tříd skriptu. Kompletní vzor pro opakované získávání všech relevantních dat na serveru BTC-e pak lze vidět na obrázku 5.9.

```
<div id='orders_1' style='overflow-y: auto; overflow-x: hidden; max-height: 485px;'>
    <table class='table' style='width: 99%'>
        <tr>
            <th>price</th><th>BTC</th><th>USD</th>
        </tr>
        <tr class='order'>
            <td>447.799</td>
            <td>0.05032359</td>
            <td>22.53485327</td>
        </tr>
        <tr class='order'>
            <td>447.8</td>
            <td>6.52899888</td>
            <td>2923.68569846</td>
        </tr>
```

Obrázek 5.8: Výřez zdrojového kódu stránky serveru BTC-e

```
{
    "A_PRICE_T":
        [["div", {"id": "orders_1"}],
         "table",
         "tr",
         "next_sibling",
         "td",
         "text"],
    "A_AMOUNT_T":
        [["div", {"id": "orders_1"}],
         "table",
         "tr",
         "next_sibling",
         "td",
         "next_sibling",
         "text"],
    "B_PRICE_T":
        [["div", {"id": "orders_2"}],
         "table",
         "tr",
         "next_sibling",
         "td",
         "text"],
    "B_AMOUNT_T":
        [["div", {"id": "orders_2"}],
         "table",
         "tr",
         "next_sibling",
         "td",
         "next_sibling",
         "text"],
}
```

Obrázek 5.9: Vzor pro dolování dat ze serveru BTC-e.

### 5.1.3 Demonstrace

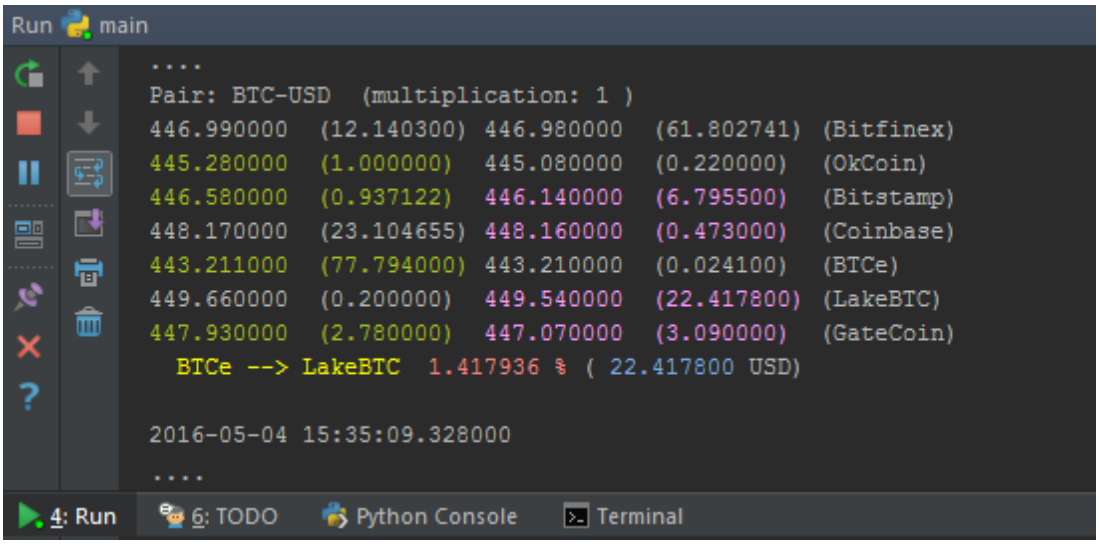
Jednou z elektronických součástí práce je i krátká demonstrace monitorovacího skriptu pojmenovaná **monitor_demo.py**.

```
python
└── demo
    └── monitor
        └── monitor_demo.py
```

Jedná se o vytvoření instance třídy **Monitor** a nastavení základních parametrů:

- **currencies** – Seznam měn, které se mají sledovat.
- **main_currency** – Určení hlavní měny, se kterou se budou vytvářet páry.
- **markets** – Seznam serverů, na kterých se má sledování provádět.

Spuštění hlavního cyklu metodou **.run_main_cycle()** pak způsobí cyklus dotazů pro všechny měnové páry na všech serverech, kde jsou tyto páry definované. Třída **Monitor** také rovnou v každém cyklu měnového páru automaticky zvýrazní nákupní ceny, které jsou nižší než prodejní na jiném serveru a prodejní ceny, které jsou vyšší než některé z nákupních. Na konci každého takového cyklu pak případně zapíše nejvýhodnější přesun a jeho maximální kapacitu. Příklad takového průběhu lze vidět na obrázku 5.10.




Obrázek 5.10: Výpis monitorovacího skriptu.

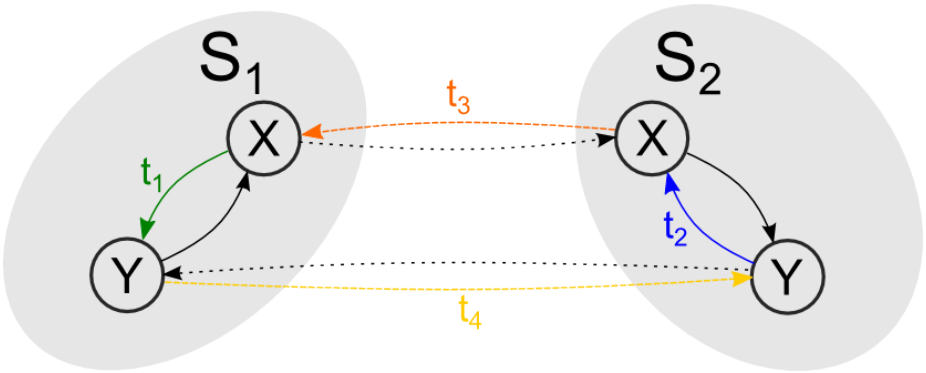
## 5.2 Výměnný obchod

Využít integrované informace od jednotlivých serverů lze několika způsoby, jedním z nich je způsob pro účel práce označený jako „výměnný obchod“, který funguje na principu využití aktuálních informací o stavu na různých burzách – směnných serverech. Tento proces není ovlivňován dlouhodobými trendy měn a využívá čistě krátkodobé volatility a neschopnost rychlého přizpůsobení decentralizováného trhu na změnu.

Výměnný obchod funguje na principu současné směny finančních párů na dvou různých burzách. Tedy konkrétně směny na páru $X \Rightarrow Y$ na jedné burze a $Y \Rightarrow X$ na burze druhé tak, že celkové množství konkrétních měn zůstane stejné či je vyšší (operace $t_1$ a $t_2$ na obrázku 5.11). Zjednodušené rovnice transakcí pak tedy mohou vypadat jako:

$$t_1 : \Delta M_1^{y'} = \frac{\Delta M_1^x}{R_1^{ask}}, \tag{5.1a}$$

$$t_2 : \Delta M_2^{x'} = \Delta M_2^y * R_2^{bid}. \tag{5.1b}$$




Obrázek 5.11: Znázornění výměnného obchodu.

Kde $\Delta M_1^x$ značí změnu množství měny $X$ na serveru 1, $R_1^{ask}$ značí nabídkovou výši kurzu na serveru 1, $R_2^{bid}$ značí poptávkovou výši kurzu na serveru 2.

Pro splnění podmínky, aby celkové množství zůstalo stejné či vyšší, tedy musí platit nerovnice:

$$\Delta M^{y'} \geq 0 \tag{5.2a}$$
$$\Delta M^{x'} \geq 0 \tag{5.2b}$$

kde:

$$\Delta M^{y'} = \Delta M_1^{y'} - \Delta M_2^{y} \tag{5.3a}$$
$$\Delta M^{x'} = \Delta M_2^{x'} - \Delta M_1^{x} \tag{5.3b}$$

a tedy:

$$\Delta M_1^{y'} \geq \Delta M_2^{y} \tag{5.4a}$$
$$\Delta M_2^{x'} \geq \Delta M_1^{x} \tag{5.4b}$$

Po dosazení vzorce 5.1 do nerovnic 5.4 a následném zjednodušení samozřejmě dostaneme základní podmínku, kdy obchod může proběhnout:

$$R_1^{ask} \leq R_2^{bid}. \tag{5.5}$$

Potencionální zisk $z_p$ pak vypočteme jako:

$$z_p = \frac{R_2^{bid}}{R_1^{ask}} - 1. \tag{5.6}$$

Nyní tedy byly vyjádřeny rovnice prvních dvou transakcí, po těchto transakcích se nám však hromadí měna $Y$ na první burze a měna $X$ na druhé burze. Je tedy potřeba provést další transakce $t_3$ a $t_4$ přesouvající přebytky měny na jedné burze do burzy druhé. Tyto transakce tedy slouží čistě jako přesun, jejichž hlavním faktorem bude časová prodleva, která nám však již nevadí a jejich rovnice tedy můžeme vyjádřit jako:

$$t_3 : \Delta M_2^{y'} = \Delta M_1^{y} \tag{5.7a}$$

$$t_4 : \Delta M_1^{x'} = \Delta M_2^x. \tag{5.7b}$$

Jelikož již máme definován kompletní oběh měny přes transakce $t_1$ až $t_4$ můžeme spočítat celkový zisk $\pi$ jako:

$$\pi = M * z_p, \tag{5.8}$$

kde $M$ je celkové množství vstupního kapitálu pro nákup. Po dosazení tedy dostaneme:

$$\pi = M * (\frac{R_2^{bid}}{R_1^{ask}} - 1). \tag{5.9}$$

Tedy například na obrázku 5.10 lze vidět, že je možné na serveru BTCe nakoupit měnu BTC za 443.211 USD a zároveň prodat měnu BTC za 449.540 USD na serveru LakeBTC a realizovat tak potencionální zisk $z_p = 1.42\%$

### 5.2.1 Poplatky

Nejprve je však od potencionálního zisku potřeba odečíst kumulativní poplatky obou serverů, které se skládají z procentuální hodnoty transakce $f$ a případně minimální hodnoty poplatku $F^{min}$ či fixního poplatku $F^{fix}$. Rovnice transakcí tedy budou vypadat jako:

$$t_1 : \Delta M_1^{y'} = \frac{\Delta M_1^x}{R_1^{ask}} * (1 - f_1) - F_1, \tag{5.10a}$$

$$t_2 : \Delta M_2^{x'} = \Delta M_2^y * R_2^{bid} * (1 - f_2) - F_2. \tag{5.10b}$$

Rovnice zisku 5.9 se změní na:

$$\pi = M * z_p - (F_1 + F_2), \tag{5.11}$$

kde:

$$z_p = \frac{R_2^{bid}}{R_1^{ask}} - 1 - f_1 - f_2. \tag{5.12}$$

Například tedy server BTCe si účtuje 0.2% z transakce a server LakeBTC 0.02% až 0.25% z transakce (na základě výše provedených obchodů za delší období), čímž se výše zmíněný konkrétní obchod stává méně výhodný avšak stále pozitivně realizovatelný.

### 5.2.2 Kapacita transakce

Protože tyto obchody jsou většinou prováděny na menších serverech s většími výchylkami kurzu, jsou transakce často velmi limitovány dostupnou kapacitou měny a zadanými výšemi transakcí zadavatelem.

Kapacita možného našeho prodeje $C_1^{sell^x}$ měny $x$ na serveru 1 je tedy omezená kapacitou nabídky transakce zadavatele $M_1^{sell^y}$ a vlastními dispozicemi měny $M_1^x$ na serveru 1 a obdobně pro kapacitu možného našeho prodeje $C_2^{sell^y}$ měny $y$ na serveru 2 jsme omezeni kapacitou transakce zadavatele poptávky $M_2^{buy^y}$ a vlastní dispozicí měny $M_2^y$.

$$C_1^{sell^x} = min\Big(M_1^{sell^y} * R_1^{ask}, M_1^x\Big) \tag{5.13a}$$

$$C_2^{sell^y} = min(M_2^{buy^y}, M_2^y). \tag{5.13b}$$

Pro zachování podmínky stejného nebo vyššího celkového množství obou měn $x$ i $y$ potřebujeme tyto hodnoty integrovat. Maximální množství měny tedy můžeme spočítat několika možnostmi, kde například při preferenci maximálního zisku na měně $X$ tedy při vyšší výhodnosti kurzu $R_2^{bid}$, kdy chceme dosáhnout co nejmenšího přebytku $\Delta M^y$. Hodnoty pak vypadají jako (zapsáno bez poplatků):

$$\Delta M_1^x = min(C_1^{sell^x}, C_2^{sell^y} * R_1^{ask}), \tag{5.14a}$$

$$\Delta M_2^y = \frac{\Delta M_1^x}{R_1^{ask}}, \tag{5.14b}$$

nebo maximalizaci na měně $Y$ (při preferencích pro kurz $R_1^{ask}$):

$$\Delta M_2^y = min(\frac{C_1^{sell^x}}{R_2^{bid}}, C_2^{sell^y}), \tag{5.15a}$$

$$\Delta M_1^x = \Delta M^y * R_2^{bid}, \tag{5.15b}$$

Nejčastější případ však nastává, když jde čistě o maximalizaci zisku nezávisle na měnách a tedy:

$$\Delta M_1^x = max\Big(min(C_1^{sell^x}, C_2^{sell^y} * R_1^{ask}), min(C_1^{sell^x}, C_2^{sell^y} * R_2^{bid})\Big), \tag{5.16}$$

$$\Delta M_2^y = max\Big(min(\frac{C_1^{sell^x}}{R_1^{ask}}, C_2^{sell^y}), min(\frac{C_1^{sell^x}}{R_2^{bid}}, C_2^{sell^y})\Big), \tag{5.17}$$

### 5.2.3 Výměna mezi servery

Dalším problémem je, že k obchodům dochází na dvou různých serverech a převod měn mezi servery může trvat klidně i několik hodin ne-li až dnů. Pro uskutečnění takovýchto obchodů je tedy potřeba mít na obou zúčastněných serverech dostatečné obnosy. Případné přebytky a nedostatky konkrétních měn na konkrétních serverech je pak možné přesouvat mezi servery po splnění transakce, kdy už nejde o čas (operace $t_3$ a $t_4$ na obrázku 5.11), avšak tyto přenosy mimo server jsou zpravidla znovu zpoplatněny a často i omezeny minimální výší.

$$t_3 : \Delta M_2^{y'} = \Delta M_1^y * (1 - f_1^o) - F_1^o \tag{5.18a}$$

$$t_4 : \Delta M_1^{x'} = \Delta M_2^x * (1 - f_2^o) - F_2^o \tag{5.18b}$$

Záleží tedy na konkrétní situaci a to hlavně na četnosti a směrech výkyvů kurzů daných serverů, tedy šance, zda se v dohledné době objeví možnost udělat zpětný výdělečný obchod, či je výhodnější zaplatit poplatky za mezi-serverovou výměnu, aby jsme měli k dispozici prostředky na uskutečňování dalších transakcí stejným směrem.

### 5.2.4 Rizika

Pro co nejlepší využití této metody je důležité sledovat a mít finanční rezervy na co nejvíce různých serverech. Čím menší server tím náchylnější je k výrazným výkyvům mimo kurz oproti ostatním a tím ziskovější se stávají takovéto výměnné transakce, které pak mohou dosahovat i desítky procent.

Připravenost realizovat tyto obchody tedy znamená uložení financí na mnoho různých serverů, kde řada z nich je nestabilní a kde většinu času pouze 'čekají' na vhodný obchod. Riziko krachu konkrétního serveru pak tedy mnohokrát převyšuje případnou ziskovost obchodů zahrnující tento server.

### 5.2.5 Udržovatelnost

Část rizika se dá odstranit pečlivým monitorováním jednotlivých serverů a případnými přesuny finančních zdrojů. Vzhledem k rozličným druhům programových přístupů k jednotlivým serverům však toto vyžaduje velké množství úprav jak ve skriptech monitorujících data, tak hlavně ve skriptech provádějících transakce.

Vzhledem k tomu, že většina serverů má omezený aplikační přístup k provádění transakcí nejen různými bezpečnostními prvky, ale často i zpoplatněním, tak není součástí této práce knihovna k provádění samotných transakcí. Takováto univerzální knihovna by byla velmi rychle zastaralá a zbytečně rozsáhlá. Mnoho serverů již nabízí vlastní knihovny pro komunikaci s konkrétním serverem vyhovující jejich protokolu komunikace. Tyto knihovny se dají zpravidla pak jednoduše připojit na skripty implementované v rámci této práce.

Prioritou této práce tedy je umožnit uživateli vytvořit si přehled o možnostech a případně implementovat si vlastní vazby na servery dle svého výběru.

### 5.2.6 Implementace

Součástí této práce je i demonstrační skript provádějící potřebné výpočty zmíněné v textu výše. Demonstrační soubory jsou uloženy ve složce **demo/switchtrade**

```
python
└── demo
    └── switchtrade
        ├── switchtrade_demo.py
        └── utils
            ├── switchtrade_operation.py
            └── trading_market.py
```

Pro spuštění výpočtu stačí do monitorovacího skriptu vložit volání funkce:
**trade_exchange**(*sell_market*, *buy_market*, *currency_1*, *currency_2*, *sell_values*, *buy_values*)
(definované v souboru **switchtrade_operation.py**) kde:

- *sell_market* je instance třídy **Market** rozšířenou o třídu **TradingMarket**, která má specifikované interní množství první měny *currency_1*
- *buy_market* je instance třídy **Market** rozšířenou o třídu **TradingMarket**, která má specifikované interní množství druhé měny *currency_2*
- *currency_1* je měna, kterou

  budeme prodávat na prvním marketu *sell_market* za měnu *currency_2* a

  nakupovat na druhém marketu *buy_market* za měnu *currency_2*.
- *currency_2* analogicky k *currency_1* v opačném směru

- *sell_values* jsou hodnoty vrácené voláním metody:
  **.get_current_market_info(***currency_1***,** *currency_2***)** na instanci *sell_market*.
- *buy_values* jsou hodnoty vrácené voláním metody:
  **.get_current_market_info(***currency_1***,** *currency_2***)** na instanci *buy_market*.

Součástí elektronického obsahu práce je i demonstrační soubor **switchtrade_demo..py**, který ukazuje základní nastavení pro demonstraci procesu výměnného obchodu. Třída **SwitchTrade** rozšiřuje monitorovací skript a provádí volání samotné metody **trade_exchange(. . . )** na závěr vyhodnocení monitorovacího cyklu (metoda **process_cycle(***currency***)**).

Výpis demonstračního skriptu je možné vidět na obrázku 5.12, kde si lze všimnout limitování transakcí vlastním obnosem $C_1^{sell^x} = 1500$ USD v případě první transakce a limitování druhé transakce velikostí požadavku na prodej měny BTC zadavatelem $C_2^{sell^y} = 4.59$ BTC.

Optimální velikosti transakcí pro maximalizaci zisku dle rovnic 5.16 pak vycházejí na $\Delta M_1^x = 1500$ USD a $\Delta M_2^y = 3.32$ BTC a realizované obchody by v tomto případě měly nulový celkový zisk na měně BTC, ale zisk 24.16 na měně USD.

## 5.3 Dlouhodobé data

V předchozí sekci 5.1 jsme definovali a sestrojili knihovnu na sběr dat, tyto data je možné využít na řadu operací, jednou z nichž je i demonstrace využití v sekci 5.2, avšak pro řadu aplikací je vhodné využívat nejenom aktuální, ale i historické hodnoty.

Z toho důvodu tato práce dále využívá databázi historických hodnot. Záznamy jsou ukládány v databázi MySQL a kopírují schéma používané v předchozích skriptech této práce, tedy konkrétně:

- *id* – Unikátní číslo vyjadřující pořadí zápisu záznamu do databáze.
- *server* – Zdrojový server, ze kterého byl záznam pořízen.
- $c\_from$ – První měna obchodovaného páru.
- *c_to* – Cílová měna obchodovaného páru.
- *ask* – Nabídková cena měny *c_to* za jednotku $c\_from$.
- *bid* – Poptávková cena měny *c_to* za jednotku $c\_from$.
- *ask_amount* – Výše poslední nabídky v měně $c\_from$
- *bid_amount* – Výše poslední poptávky v měně $c\_from$
- *time* – Časová známka získání údajů.

```
Pair: BTC-USD  (multiplication: 1 )
457.900000  (0.100000)  457.890000  (6.006387)  (Bitfinex)
455.450000  (0.400000)  455.350000  (1.055000)  (OkCoin)
456.800000  (4.083460)  456.020000  (2.144970)  (Bitstamp)
459.990000  (0.100200)  459.960000  (4.591850)  (Coinbase)
452.442000  (25.048756) 451.710000  (5.600000)  (BTCe)
458.390000  (0.108400)  458.290000  (0.000300)  (LakeBTC)
459.500000  (0.100000)  456.800000  (1.414000)  (GateCoin)
  BTCe --> Coinbase  1.661649 % ( 4.591850 BTC)
BTCe
     M1x    (my balance): 1500.00 USD
     R1ask  (exch.rate):  451.54 USD/BTC  (452.44 - 0.20% fee)
     M1sellY(transaction):25.05 BTC

Coinbase
     M2y    (my balance): 20.00 BTC
     R2bid  (exch.rate):  458.81 USD/BTC  (459.96 - 0.25% fee)
     M2buyY(transaction):4.59 BTC

(t1 capacity) C1sellX = min(25.048756 * 451.54, 1500.00 )    = 1500.00 USD
(t2 capacity) C2sellY = min(4.591850 , 20.0000 )             = 4.591850 BTC

dM1x = 1500.000000
dM2y = 3.321986

Transaction 1 at BTCe
     sell 1500.00 USD
     buy  3.321986 BTC
     at   452.4420 USD/BTC
     fee  3.0000 USD

Transaction 2 at Coinbase
     buy  1524.16 USD
     sell 3.321986 BTC
     at   459.9600 USD/BTC
     fee  3.8104 USD

Profit
     24.1608  USD
     0.000000  BTC

2016-05-09 11:04:37.530000
```

Obrázek 5.12: Výpis skriptu vypočítávajícího výměnné transakce.

Součástí digitálního obsahu práce je i soubor *records_structure.sql* obsahující přesnou definici schématu tabulky a soubor *records_data.sql* obsahující zálohu dat získaných během tvorby této práce.

```
server
└── database
    ├── records_structure.sql
    └── records_data.sql
```

### 5.3.1 Sběr dat

Samotný sběr dat a jejich ukládání do databáze lze snadno provádět za pomocí vytvořené knihovny **DbHandler**.

```
python
└── lib
    └── db_handler
        └── db_handler.py
```

Hlavní třídu knihovny stačí inicializovat typem *ConnectionType.direct* a následně na ní zavolat metodu:

**.connect_to_db(***server***,** *user***,** *password***,** *database***)**

s parametry na připojení k databázi. Takto připravenou třídu je možné vložit do poděděné třídy typu **Monitor** a při přepsání či doplnění její funkce **process_values(...)** snadno vzniká kompletní monitorovací skript, který všechny potřebné informace může ukládat do databáze voláním funkce:

**.db_insert(***server***,***c_from***,***c_to***,***ask***,***bid***,***ask_amount***,***bid_amount***)**,

kde parametry odpovídají výsledným hodnotám v databázi.

V elektronické příloze této práce jsou přiloženy dva skripty demonstrující tuto činnost, kde skript s názvem *db_save_direct_demo.py* demonstruje výše zmíněné operace a skript *db_save_remote_demo.py* demonstruje možnost vzdáleného ukládání záznamů na server pomocí http volání.

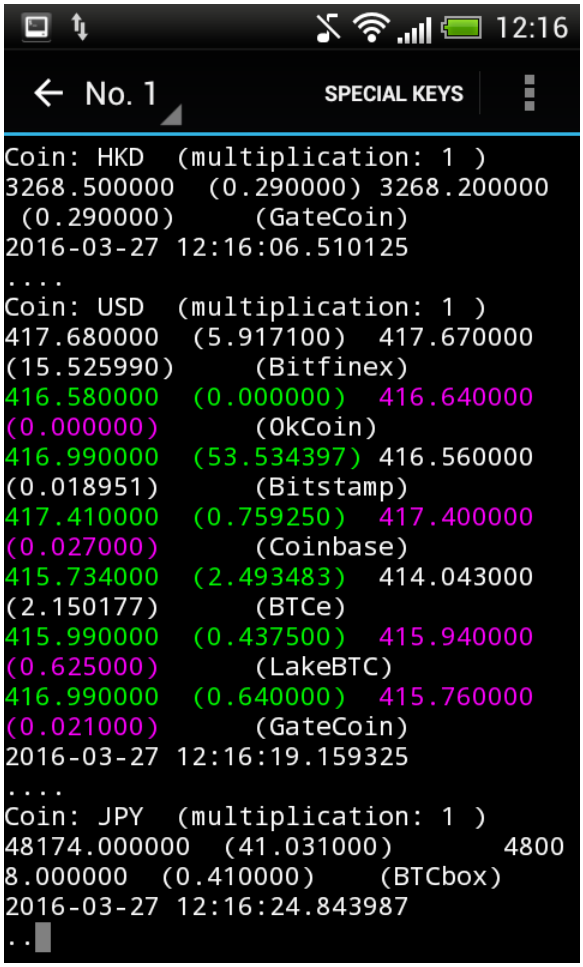
```
python
└── demo
    └── db_storage
        ├── db_save_direct_demo.py
        └── db_save_remote_demo.py
```

Vzdálené ukládání je možné pomocí volání metod serveru implementovaného v rámci této práce a zmíněné dále v textu. Skript je, mimo jiné, i za tímto účelem optimalizován, tak aby bylo ho možné spouštět na různých platformách včetně mobilních zařízení například využívající systém Android 2.1+ (obrázek 5.13).

Pro využití vzdáleného ukládání stačí inicializovat výše zmíněnou třídu **DbHandler** s parametrem *ConnectionType.remote* a následně nastavit adresu serveru voláním metody:

**.set_api(***server_url***)**.

Předávání hodnot na ukládání již pak probíhá analogicky funkcí **db_insert(...)** stejně jako u předchozí varianty.




Obrázek 5.13: Výpis skriptu běžícího na mobilním zařízení, který sbírá data a následně je odesílá na vzdálený server.

### 5.3.2 Implementace serveru

Součástí práce je tedy i jednoduchý server vytvořený v jazyce PHP, jehož hlavním úkolem je tvořit mezičlen jednotlivých aplikací a databáze zmíněné v předchozí kapitole. Implementace serveru je součástí elektronické přílohy této práce a její struktura vypadá následovně:

```
server
└── backend
    ├── api.php
    ├── config.php
    └── mysql_communicator.php
```

Soubor **config.php** slouží, jak už název napovídá, k nastavení konfigurace, která je oddělená od zbytku kódu pro usnadnění přenositelnosti. Zejména jsou zde definované autorizační prvky pro připojení k serveru a případné bezpečnostní omezení pro přístup jak zapisujících, tak čtecích aplikací.

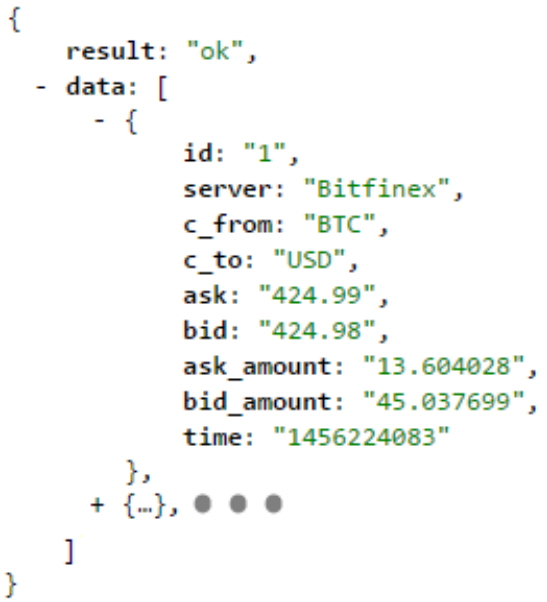
### Vkládání dat

Soubor **api.php** provádí samotné aplikační rozhraní. Jeho hlavní zápisovou funkcí je požadavek s parametrem $request = put$. Jedná se o zrcadlení funkce **db_insert(...)** implementované v rámci knihovny **DbHandler**, tedy proces uložení hodnot pod názvy *server*, $c\_from$, $c\_to$, *ask*, *bid*, $ask\_amount$ a $bid\_amount$ do databáze. V případě stanovení nutnosti autentizace k zápisu je také potřeba připojit parametr *token* s odpovídající hodnotou. Příkladem takovéhoto požadavku tedy může být http dotaz na server:

.../**api.php?request**=$put$**&**
**server**=$Bitfinex$**&**
**c_from**=$BTC$**&**
**c_to**=$USD$**&**
**ask**=453.5**&**
**bid**=453.42**&**
**ask_amount**=14.88**&**
**bid_amount**=12.76

### Získávání dat

Základní metodou pro získávání dat od serveru je pak požadavek s parametrem $request = get$, který má schopnost vrátit všechny aktuální záznamy z databáze ve formátu JSON (obrázek 5.14). Součástí každého vráceného záznamu je kromě vložených informací také pole *time*, které obsahuje časovou značku pořízení záznamu ve formátu unixového času.




Obrázek 5.14: Výpis dat ze serveru s integrovanými záznamy.

Vzhledem k celkovému počtu záznamů v databázi by však plné vracení všech záznamů znamenalo neúměrné nároky na server a síťovou komunikaci. Počet záznamů je proto možné limitovat parametrem $RECORD\_LIMIT$ v souboru **config.php**, který je v základu nastaven na limitování počtu vrácených položek na 10 000 záznamů.

Pokud tedy budeme chtít od serveru získat více záznamů než je povoleno limitem, můžeme posílat opakované požadavky s parametrem *request* = *update* a nastavením parametru *id* na hodnotu posledního záznamu, který již máme. Například tedy dotazy jako:
.../**api.php?request**=*update*&
**id**=0
.../**api.php?request**=*update*&
**id**=10000
atd.

Další podporované funkce serveru jsou na dotazy *request* = *get_unique* a *request* = *get_aggregated_data*. Na dotaz s parametrem *get_unique* server vrátí pole hodnot obsahující všechny unikátní kombinace názvu serveru (ze kterého byly záznamy pořízeny) a jejich měnových párů. Výsledek zároveň obsahuje agregované hodnoty, kterými jsou průměry všech polí záznamů.

Dotaz s parametrem *get_aggregated_data* pak umožňuje získat agregované hodnoty pro různá časová období. Základními parametry jsou *timestamp* a *period*, kde parametr *timestamp* určuje od kdy chceme záznamy (parametr je zase časová známka v unixovém formátu) a parametr *period* určuje podle jaké časové periody chceme záznamy agregovat. Možné hodnoty časové periody jsou $xxYY$, kde $xx$ je číslo udávající počet a $YY$ je zkratka určující jednotky z:

- **YY** – Rok
- **MM** – Měsíc
- **DD** – Den
- **hh** – Hodina
- **mm** – Minuta
- **ss** – Vteřina

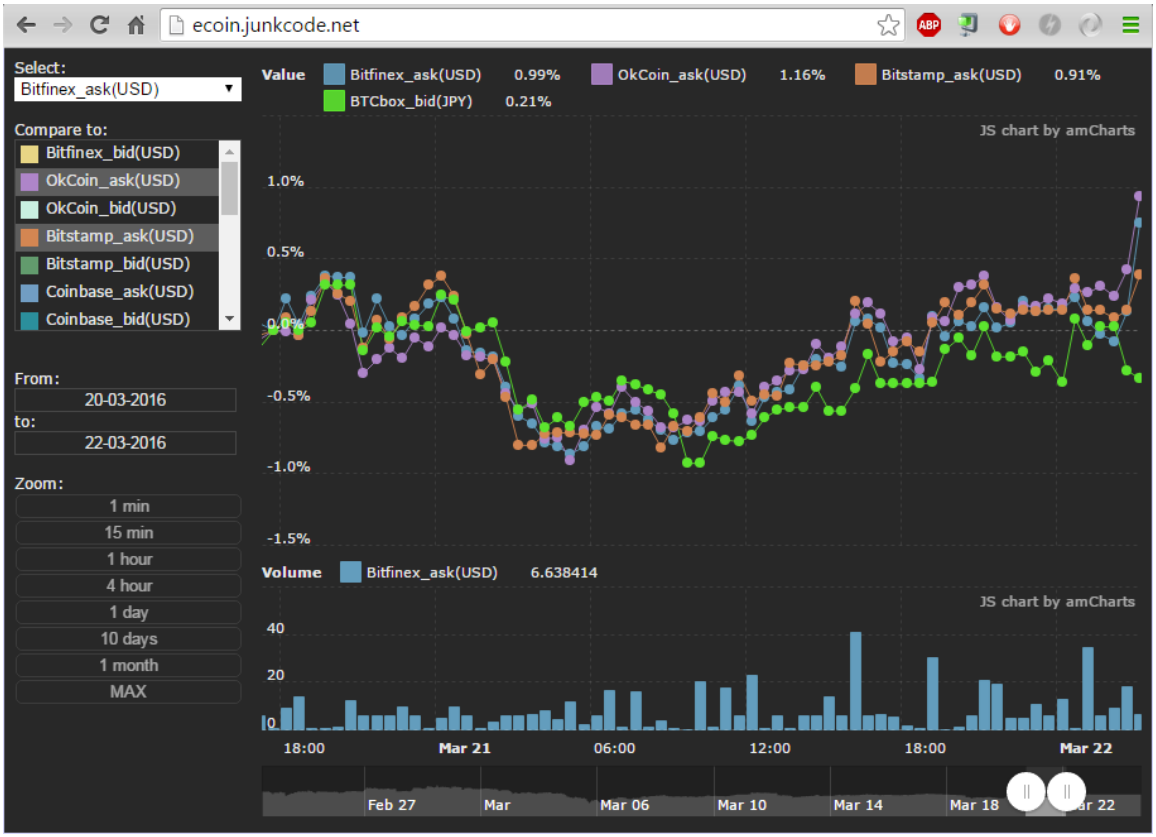
Záznamy je dále možné omezit parametrem *timestamp*2, který určuje konec časového okna ze kterého chceme záznamy. Server navíc umožňuje další parametr *context*, který přijímá stejné hodnoty jako parametr *period* a určuje velikost agregace dat mimo vyhrazené okno, tedy například při zavolání:
.../**api.php?request**=*get_aggregated_data*&
**timestamp**=1459799817
**timestamp2**=1460050746
**period**=*mm*
**context**=*DD*
server vrátí:

- **denní** agregované záznamy od počátku databáze až do 5. dubna 2016,
- za nimi následují **minutové** agregované záznamy od 5. do 6. dubna 2016 a
- následně jsou připojeny znovu **denní** agregované záznamy od 6. dubna 2016 do konce databáze.

### 5.3.3 Centrální sklad dat

Pro lepší uživatelskou kontrolu a přehled je server vybaven i rozhraním umožňujícím zobrazení jednotlivých vývojů kurzů i jejich vzájemné postavení. Klientské rozhraní je založeno na zobrazovací knihovně **amCharts** od stejnojmenné společnosti a je modifikováno pro spolupráci se serverem definovaným v předchozí sekci.

Všechny soubory potřebné pro zprovoznění vlastního serveru jsou přiloženy v rámci digitálního obsahu práce a zároveň je spuštěna online dostupná verze na adrese `http://ecoin.junkcode.net/` (obrázek 5.15).




Obrázek 5.15: Zobrazení dat v centrálním skladu.

Kompletní klientské prostředí je umístěné na přiloženém digitálním obsahu jako:

```
server
└── frontend
    ├── intex.html
    ├── js
    │   ├── config.js
    │   ├── chart_manager.js
    │   └── server_communicator.js
    └── lib
        └── amChartsfiles
```

Soubor **index.html** je základní stránkou načítající knihovnu, funkční skripty a styly stránky. Zde je možné připojit případné další funkce.

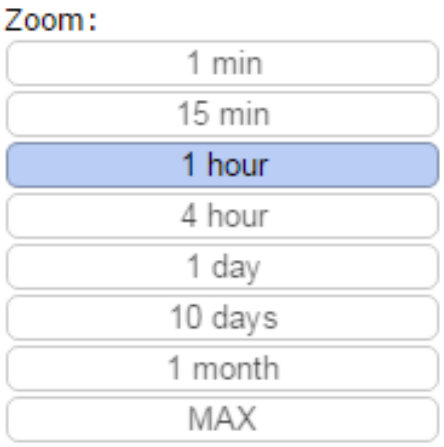
Soubor **config.js** následně stanovuje základní nastavení a to především:

- $SERVER\_PATH$ – Adresa serveru, ke kterému se klientské rozhraní bude připojovat.
- $AUTOREFRESH\_ENABLED$ – Povolení možnosti automatické aktualizace zobrazovaných dat, jež způsobí periodické načítání nových informací ze serveru.
- $AUTOREFRESH\_INTERVAL$ – Interval četnosti načítání nových dat.
- $AUTOREFRESH\_TRIGGER$ – Velikost odstupu uživatelského okna od konce datové řady, kdy bude automatická aktualizace spuštěná.
- $AUTOREFRESH\_PERIODS$ – Granularita dat, na které se obnova bude vztahovat.

Soubor **chart_manager.js** působí jako hlavní nastavovací a komunikační prvek se zobrazovací knihovnou a jsou v něm definovány postupy reakcí na jednotlivé vstupy od uživatele.

Soubor **server_communicator.js** zastupuje poslední článek v komunikaci se serverem, kdy funkce v tomto souboru mají za úkol vzít parametry požadovaných dat, zjistit je od serveru a vrátit zobrazovací knihovně naformátované záznamy. Výměnou tohoto souboru může být snadno dosaženo připojení na jiný zdroj dat.

Uživatelské rozhraní datového skladu tedy nejenom umožňuje zobrazovat data za určité období výběrem časového okna (například tlačítky v levé nabídce či přímo na spodní časové ose), ale umožňuje i plynulé zobrazování nových (přibývajících dat). Při zobrazení delších časových úseků jsou data agregována pro lepší přehlednost a to na několik různých hodnot period v závislosti na časové velikosti zobrazeného okna a velikosti okna prohlížeče.




Obrázek 5.16: Základní možnosti nastavení velikosti okna zobrazovaných dat.

## 5.4 Zpracování dat

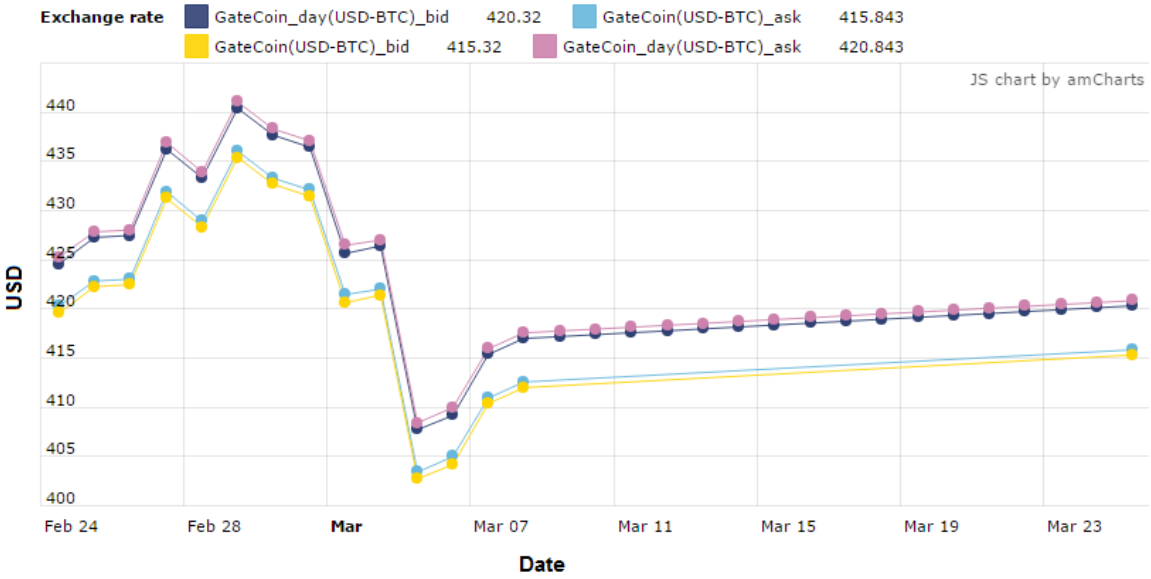
V předchozích sekcích jsme si popsali jak data posbírat od jednotlivých serverů a uložit do centrální databáze. Takto posbírané data však mají nevýhodu, že jsou většinou v nepravidelných intervalech a obsahují mezery způsobené jak výpadky serverů, tak výpadky spojení či chybami klienta.

Pokud tedy chceme mít ke zpracování data v pravidelných intervalech, tak jak to vyžaduje řada algoritmů, musíme je vyfiltrovat a doplnit. K tomuto účelu je jako součást práce vytvořená knihovna **DataWorks**, jež funguje jako rozšíření knihovny **DbHandler**.

```
python
└── lib
    └── db_handler
        ├── db_handler.py
        ├── data_works.py
        └── utils
            └── data_filter.py
```

### 5.4.1 Doplnění dat

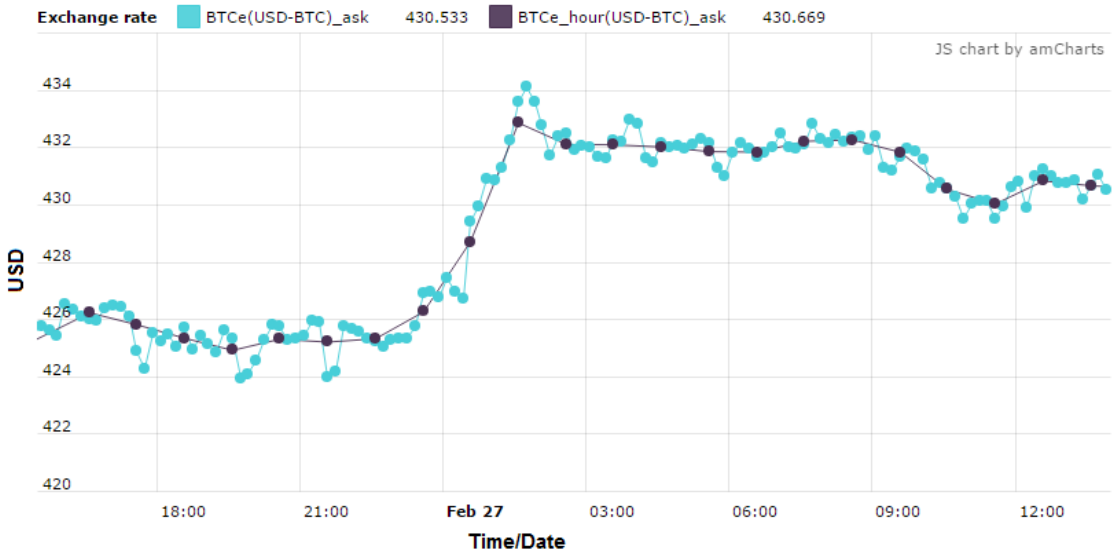
Základní požadovanou operací je tedy doplnění dat o chybějící hodnoty. Tyto data jsou doplněna na základě lineární aproximace hodnot podle dvou nejbližších hodnot (okrajových hodnot výpadku). Příklad takového doplnění lze vidět na obrázku 5.17, kde jsou chybějící hodnoty údajů ze serveru GateCoin ve druhé polovině řady vypočteny uměle (na obrázku 5.17 jsou doplněné datové řady záměrně zvýšeny o hodnotu 5 USD pro lepší přehlednost).




Obrázek 5.17: Ukázka doplnění chybějících dat.

### 5.4.2 Vyhlazení dat

Druhou požadovanou funkcí je urovnání časových rozestupů mezi jednotlivými záznamy, či případně i vyhlazení a snížení celkového počtu záznamů. Vyhlazení dat spočívá v průměrování hodnot za určité období a dochází tak k odstranění lokálních výkyvů a lepší viditelnosti dlouhodobých trendů. Příklad vyhlazení lze vidět na obrázku 5.18, kde jsou data ze serveru BTCe vyhlazena do hodinových intervalů.




Obrázek 5.18: Základní možnosti nastavení velikosti okna zobrazovaných dat.

### 5.4.3 Demonstrace použití

Demonstrace použití knihovny na čištění dat **DataWorks** lze vidět ve skriptu **data_filling.py**, který demonstruje základní volání této knihovny.

```
python
└── demo
    └── data_cleaning
        ├── data_filling.py
        └── data_filling_formats.py
```

Knihovna se skládá ze třídy, kterou stačí instanciovat a nastavit totožně jako třídu **DbHandler** a poté stačí zavolat zabalující metodu **.fill_data(***name***,** *interval***)**, která provede následující kroky:

- Kontrola zda-li existuje v databázi tabulka s názvem *name*.

  Pokud tabulka neexistuje, je vytvořena.

  Pokud tabulka existuje, je vypsáno varování a původní data tabulky jsou posléze smazána.

- Postupné procházení dat kde:

  dochází k vypočítávání průměru z hodnot, které jsou častější než interval *interval* a

  doplňování hodnot, které jsou vzdáleny od sousedních více než jednotku *intervalu*.

- Vypočtené hodnoty jsou pak uloženy do vlastní tabulky pod jménem *name*.

Takto zpracovávané hodnoty jsou v základu ukládány do tabulky databáze, odkud jsou zdrojové hodnoty. Mnohdy se však může stát, že hodnoty jsou žádané v jiném formátu či nepotřebujeme zpracovat veškeré hodnoty od všech serverů a tak metoda **.fill_data(...)** nabízí volitelné parametry:

- **filetype=***type* – Nastavení typu výstupu:

  *MySQL* – Základní hodnota zápisu do databáze.

  *CSV* – Hodnoty budou zapsány do souboru formátu CSV.

  *JSON* – Hodnoty budou zapsány do souboru formátu JSON.

  *RETURN* – Hodnoty jsou vráceny ve struktuře pro další zpracování.

- **data_filter** – který může mít následující nastavení:

  **time_from=***x* – Nastavení časového omezení $x$ odkdy se mají záznamy zpracovávat.

  **time_to=***x* – Nastavení časového omezení $x$ posledního záznamu na zpracování.

  **server=***server* – Nastavení filtru jen na některé servery.

  **c_from=***XXX* – Nastavení, po kterém se budou zpracovávat jen záznamy, jimž odpovídá první měna z páru.

  **c_to=***XXX* – Nastavení, po kterém se budou zpracovávat jen záznamy, jimž odpovídá druhá měna z páru.

Jak již bylo zmíněno výše, všechny tyto dodatečné parametry jsou nepovinné a jejich použití je ukázáno v demonstračním skriptu **data_filling_formats.py**.

## 5.5 Predikce kurzu

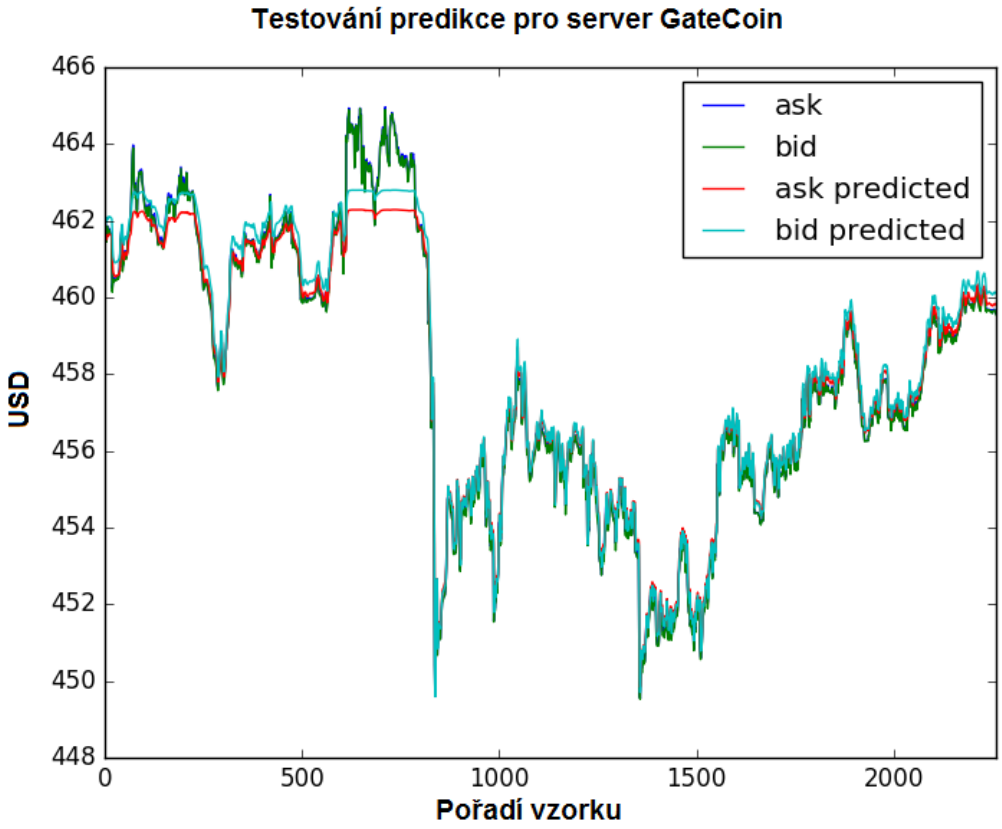
Datové funkce popsané v předchozí sekci nám již generují soubory dat se záznamy, které mají časově rovnoměrně rozdělené intervaly a jsou doplněny o chybějící hodnoty. Takovéto data už se dají relativně snadno využít na trénování predikčních modelů.

V rámci této práce je tedy implementovaná i základní knihovna pro predikci kurzu na základě neuronových sítí a to zejména za využití optimalizovaných operací definovaných v knihovně **Theano 0.8.0**, práci neuronových vrstev z knihovny **Keras 1.0.1** a práce s datovými sadami z knihovny **Pandas 0.18.0**.

Knihovna **PredictionModel** implementovaná v rámci této práce se tedy snaží zapouzdřit výše zmíněné a poskytnout jak uživateli, tak případným dalším nástrojům snadnou práci s výkonnými funkcemi. Základem knihovny je soubor **prediction_model.py**.

```
python
└── lib
    └── prediction
        ├── prediction_model.py
        └── utils
            └── data_manager.py
```

Třída **PredictionModel** efektivně umožňuje vytvářet, trénovat i testovat různé modely. V základu třída používá zpětnovazební neuronové sítě *LSTM* popsané v kapitole 3.2.4. Zároveň je možné pomocí této třídy nechat vykreslovat základní grafy testovacích dat vizuálně demonstrující stupeň naučení (obrázek 5.19).




Obrázek 5.19: Příklad grafu vygenerovaného třídou PredictionModel znázorňující rozdíl mezi cílovými a předpovídanými hodnotami.

Tato třída také umožňuje ukládání a načítání modelů a jejich vah a lze tedy vytvořit nástavbu umožnující soutěžení algoritmů a jejich verzí a tedy i snadnou možnost rozšíření knihovny o genetické algoritmy pracující nad neuronovými sítěmi.

### 5.5.1 Demonstrace jednoduchých LSTM modelů

Základní využití je demonstrováno ve skriptu **simple_lstm.py** uloženém na digitální příloze ve složce **demo/predictions**:

```
python
└── demo
    └── predictions
        ├── simple_lstm.py
        ├── double_lstm.py
        └── training_speed_comparison.py
```

Tento skript demonstruje základní použití knihovny a to prací s instancí třídy **PredictionModel** a třídy **DataManager**, která je rozšířením třídy **DbHandler** popsané v sekci 5.3.1 implementující potřebné funkce navíc.

Poté, co jsou potřebné data načtena z databáze, jsou data posunuta v ose $Y$ pro rychlejší trénování sítě $LSTM$ a následně jsou rozdělena na trénovací a testovací množiny sekvencí dat a jejich cílů funkcí:
**train_test_data_split**(*data_series* [**,** *test_size*] [**,** *shuffle_train_data*] [**,** *batch_size*]), kde:

- **data_series** – Jsou vstupní řady dat.
- A volitelné parametry:

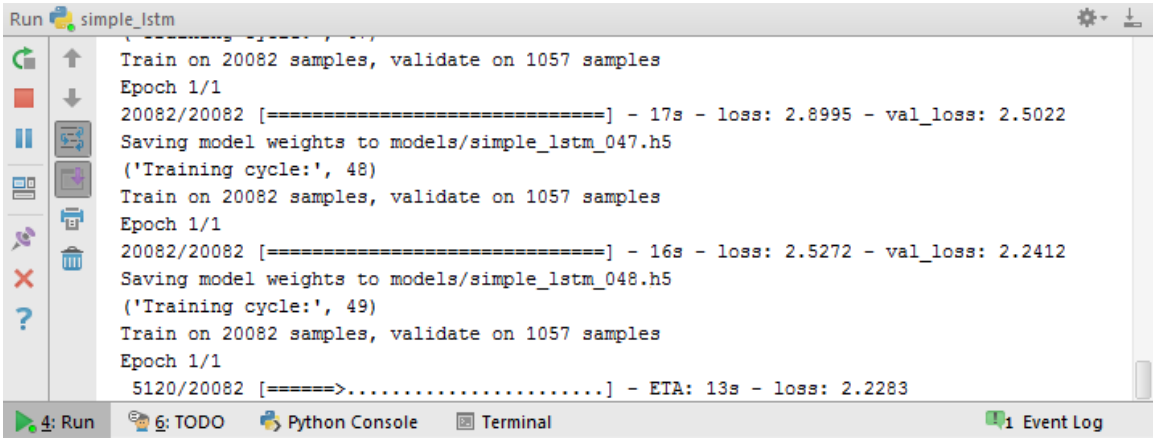
  **test_size** – Poměr velikosti trénovací a testovací množiny. Základní hodnota je nastavena na 0.1 – tedy 10% dat je použito na testování.

  **shuffle_train_data** – Určuje zdali má být trénovací množina dat zamíchaná. Základní hodnota je nastavena na míchání, avšak to může způsobovat problémy, pokud chceme vytvářet i vizualizace testovacích dat.

  **batch_size** – Určuje velikost sekvence dat, na základě které se bude předpovídat další hodnota. Základní hodnota je nastavena na 100 jednotek, tedy síť vždy dostane sekvenci 99 vstupů a má za úkol odhadnout hodnotu/hodnoty na pozici 100.

Takto zpracované data jsou následně předána modelu sítě metodou:
**.set_data**(*train_data*, *test_data*, *mean_shift*), kde parametry *train_data* a *test_data* jsou množiny získané předchozí funkcí a parametr *mean_shift* pak určuje posun hodnot pro správnou vizualizaci testovacích predikcí.

Takto nastavený model, pak již stačí vytvořit a spustit. Průběh výstupu skriptu **simple_lstm.py** pak lze vidět na obrázku 5.20, zároveň skript vygeneruje *json* strukturu reprezentující aktuální model a v každém kroku skript vygeneruje grafy pro vizuální kontrolu (například obrázek 5.19 z trénovacího cyklu 48) a uloží aktuální váhy vnitřních vazeb pro případné pozdější načtení.




Obrázek 5.20: Výstup demonstračního skriptu *simple_lstm.py*.

### 5.5.2 Složitější modely

Pokud chceme použít knihovnu s jakoukoli jinou strukturou, můžeme jednoduše převzít třídu **PredictionModel** a nahradit obsah metody **.create_model()**, jak je demonstrováno ve skriptu **double_lstm.py**.

Konkrétně se jedná o přepsání metody vytvářející samotný model, tedy v tomto případě vytvoření vrstev:

- 1. LSTM vrstva:
  **model.add(LSTM(***input_dim* **=2,** *output_dim* **=8,** *return_sequences* **=True))**

  Jejímž vstupem jsou samotná dvourozměrná data,

  výstupem jsou osmi-rozměrné vektory a

  parametrem *return_sequences* nastavujeme, že chceme aby vrstva vracela vektory pro všechny stupně řady.

- 2. LSTM vrstva:
  **model.add(LSTM(***input_dim* **=8,** *output_dim* **=16,** *return_sequences* **=False))**

  Vstupy druhé vrstvy jsou sekvence osmi-rozměrných vektorů vrstvy předchozí a

  výstupem je 16-rozměrný vektor.

  Parametrem *return_sequences* je zde nastaveno, aby se vracel již jen vektor po posledním vstupu sekvence.

- Finální perceptronů: **model.add(Dense(***input_dim* **=16,** *output_dim* **=2))**

  Tato vrstva přebírá 16-ti rozměrný vektor z druhé LSTM vrstvy a generuje na základě něj výsledný dvourozměrný výstup – predikci další hodnoty.

Další zacházení s modelem již může být naprosto totožné (jak demonstruje výše zmíněný skript **double_lstm.py**) a využít libovolné funkce předefinované třídou **PredictionModel** či je možné provést libovolné další úpravy.

### 5.5.3 Práce s modely

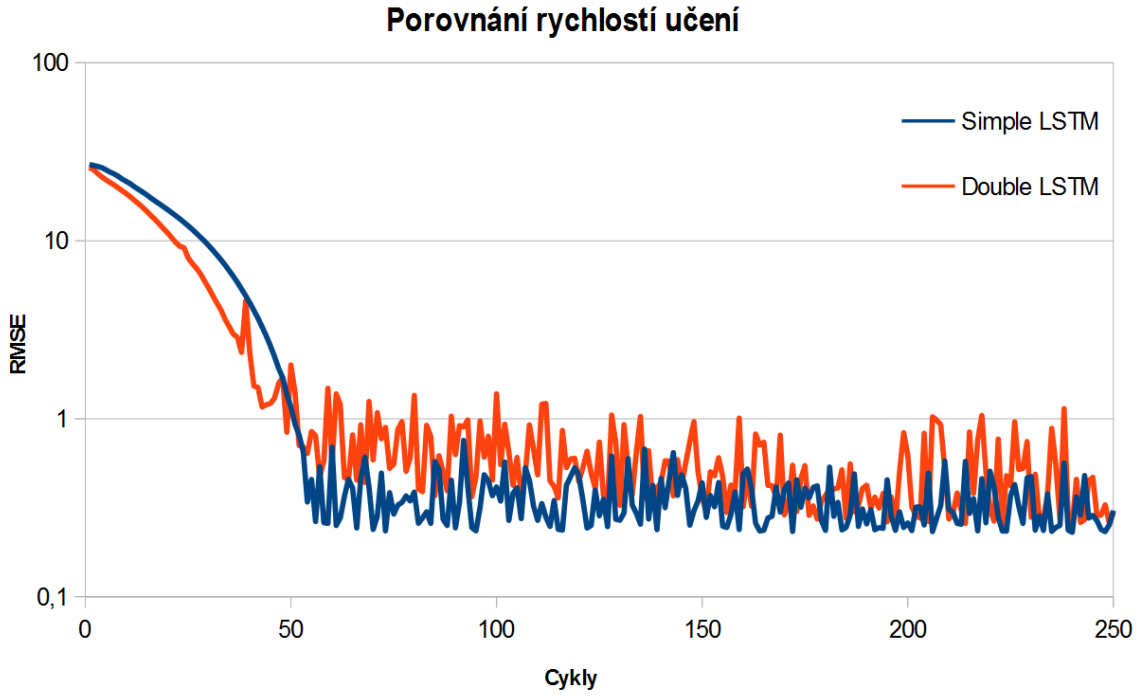
Jak lze vidět ve výstupu obou předchozích skriptů, v každém kroku jsou ukládány váhy jednotlivých prvků v rámci sítě a protože máme zároveň zapnuté ukládání struktury modelů při vytváření, můžeme tyto modely snadno zase načíst a pracovat s nimi.

Využití uložených modelů a vah z jednotlivých kroků si můžeme demonstrovat v rámci skriptu **training_speed_comparison.py**. Ten, jak již název napovídá, napomáhá porovnávat rychlost učení jednotlivých modelů, konkrétně v tomto případě předchozích modelů, což jsou jednovrstvá LSTM síť o 20 $LSTM$ blocích a dvouvrstvá LSTM síť s 8 $LSTM$ bloky v první vrstvě a 16 v druhé.

Tento skript tedy znovu pro jednoduchost načte testovací data podobně jako skripty předchozí avšak už z nich bude využívat jen testovací část dat. Následně skript načte jednotlivé modely a spustí cyklus testování.

V každém cyklu testování jsou do modelů načteny váhy, které byly produktem cyklu učení stejného pořadí a následně je spuštěna série predikcí na testovacích vstupních datech. Výsledky se pak porovnají s požadovanými cílovými hodnotami metodou nejmenších čtverců.

Na závěr po otestování všech požadovaných cyklů definovaných modelů jsou pak výsledky zapsány do *.csv* souboru pro porovnání, ze kterých je vygenerován graf 5.21.




Obrázek 5.21: Graf porovnání rychlosti učení dvou různých modelů využívajících LSTM vrstvy.

Z grafu je patrné, že průměrná chyba počítaná metodou nejmenších čtverců (RMSE) se při učení modelu používajícího dvě vrstvy LSTM ze začátku snižuje rychleji, ale následně je méně stabilní. U modelu pouze s jednou vrstvou LSTM zase pro změnu jde snižování chyby pomaleji, ale zato je učení sítě stabilnější.

### 5.5.4 Predikce budoucího vývoje

Nyní jsme si již popsali vytvoření kompletních nástrojů, od shromáždění a přípravu dat po tvorbu predikčních modelů a jejich porovnávání a můžeme tedy přistoupit k samotné predikci budoucích hodnot, která je demonstrovaná ve skriptu **future_prediction.py**.

```
python
└── demo
    └── future_predictor
        └── future_prediction.py
```

Tento skript se skládá ze třídy dědící třídu **Monitor** popsanou v sekci 5.1.3, která je následně upravena:

- Pro zjednodušení je zapnuto pouze monitorování měny $USD$.
- V rámci inicializace třídy (metoda **__init__()**) je mimo jiné:

  Vytvořena instance třídy **DataManager** popsané v sekci 5.5 a následně jsou vytvořeny a do databáze uloženy 5-ti minutové vyhlazené data. Zároveň je také vytvořena prázdná tabulka pro budoucí predikce.

  Pro zjednodušení je vytvořen datový filtr, který omezí zpracovávané záznamy na jeden ze serverů.

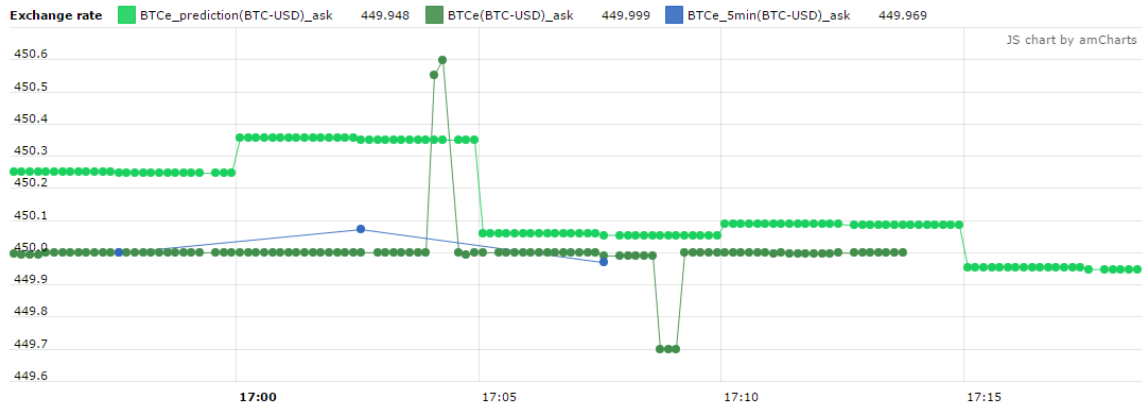
  Na závěr inicializace je načten samotný model predikční sítě třídy **PredictionModel**.
- Průběh monitorování je pak změněn v metodě **process_values(...)**, kdy si zjištěné hodnoty ukládáme do databáze čistých hodnot podobně jako je zmíněno v sekci 5.3.1.
- Po dokončení každého cyklu zjištění nových hodnot následně provedeme v přepsané metodě **process_cycle(...)**:

  Aktualizaci tabulky vyhlazených dat voláním metody **fill_data(...)** (s parametrem *update* =**True**, protože nám jde jen o aktualizaci na základě nových hodnot).

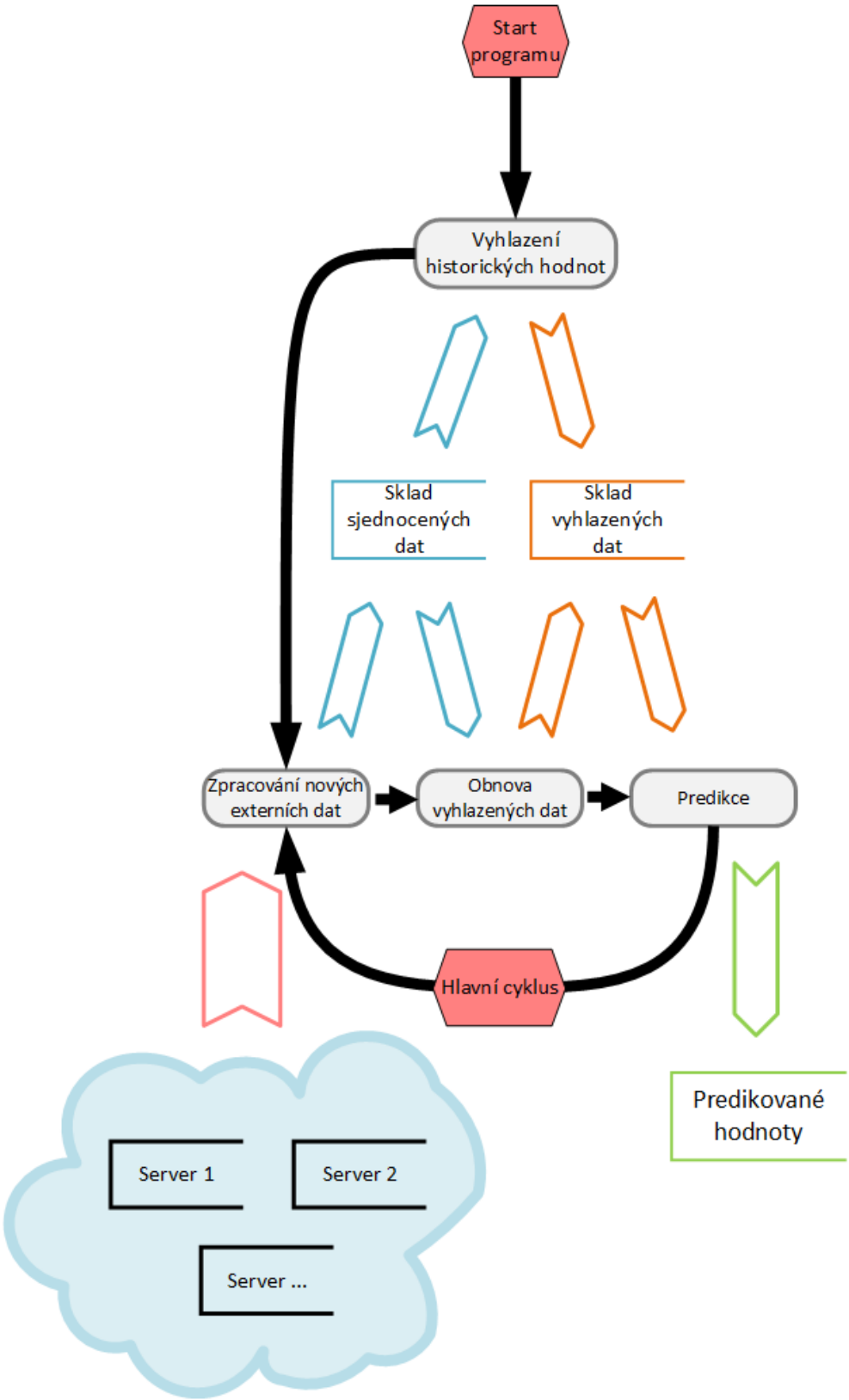
  Následně již můžeme zkontrolovat, zdali máme dostatek vyhlazených historických hodnot a provést na nich novou predikci.

  Predikce je poté pro účel demonstrace uložena do databáze do vlastní tabulky.

Porovnání predikčního skriptu lze pak vidět na obrázku 5.22, kde jsou zároveň vyobrazeny hodnoty na zdrojovém serveru a vyhlazené hodnoty, ze kterých se odhaduje predikce. Celkový běh skriptu je pak znázorněn na obrázku 5.23.




Obrázek 5.22: Ukázka 5-ti minutové predikce vývoje ceny na serveru BTCe pro čas 17:13.

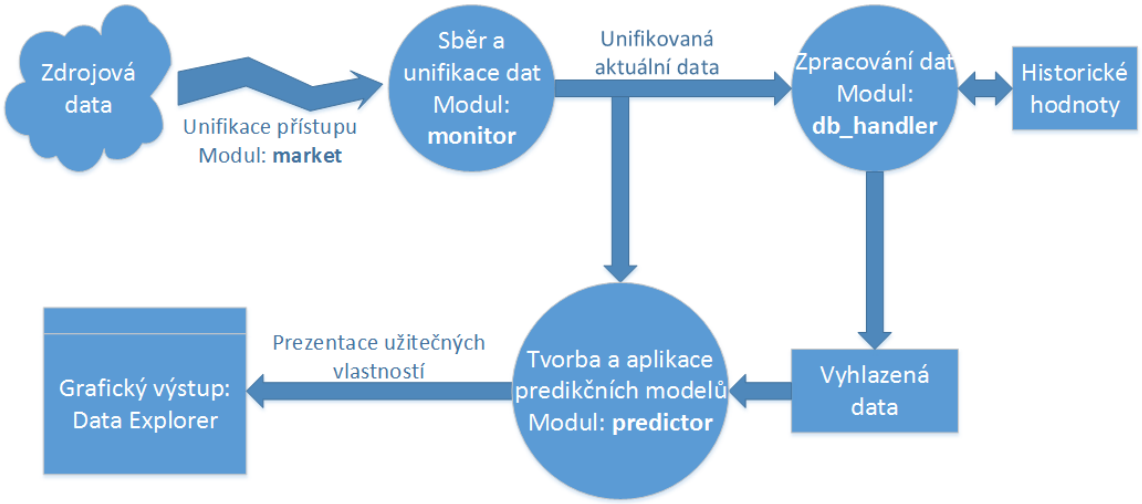



Obrázek 5.23: Znázornění postupu skriptu predikujícího hodnoty.

## 5.6 Přehled systému

Předchozí sekce popisují implementaci jednotlivých modulů, které mohou tvořit komplexní systémy. Jednotlivé moduly jsou navrženy jako samostatné avšak navazují na sebe a okolí následovně:

- Modul **market**, jehož vstupem jsou datové vzory jednotlivých serverů a výstupem jsou unifikované metody přístupu k datům.
- Modul **monitor**, jehož vstupem jsou jednotlivé instance hlavní třídy modulu **market** a výstupem jsou unifikovaná data připravena ke zpracování dle potřeb uživatele.
- Modul **db_handler**, jehož vstupem jsou příchozí data a výstupem jsou komplexní datové struktury v rozličných databázových formátech pro zpracování dle potřeby.
- Modul **predictor**, jehož vstupem jsou sekvence dat a výstupem jsou odhadované budoucí hodnoty na základě odhalených trendů.




Obrázek 5.24: Přehled implementovaných modulů a jejich vazeb.

# Kapitola 6

# Závěr

V této práci byla rozebrána problematika zpracování a využití informací na trzích alternativních měn a následně byl navržen a implementován modulární systém splňující kriteria akademické platformy umožnující snadnou a rychlou adaptaci na řadu různorodých problémů nejenom v oblasti alternativních měn.

Splnění požadované funkcionality bylo provedeno implementací čtyř hlavních modulů v jazyce *Python* 2.7, jejichž implementace je shrnuta v sekci 5.6.

Moduly byly doplněny serverovou strukturou **Data Explorer**, implementovanou na jazycích *PHP* a *JavaScript*, která umožňuje přehledné sledování stavu vývoje dat v databázi *MySQL* a umožňuje i vzdálený přístup k datovým strukturám pro závislé i nezávislé aplikace.

Všechny moduly byly navrženy a implementovány s důrazem na modularitu a možnost přizpůsobení se různým jiným problémům a potřebám. Například modul **db_handler** implementuje možnosti výstupu zpracovaných dat do formátů $*SQL$ databází, či souborových úložišť *JSON*, *CSV* nebo programového výstupu, který si uživatel může snadno konvertovat do libovolného potřebného formátu.

Modulární systém je také demonstrován řadou demonstračních aplikací ukazujících možnosti a použití jednotlivých modulů a to zejména demonstrací takzvaného „Výměnného obchodu“ (v sekci 5.2) demonstrující rychlé využití zpracovávaných rozličných dat a demonstrací „Predikce“ (v sekci 5.5.4) demonstrující kontinuální zpracování hodnot a na jejich základě predikování hodnot budoucích.

# Literatura


[1] Bitcoin Forecast. [Online; navštíveno 19.05.2016].
URL http://www.bitcoinforecast.com/

[2] Tradewave, Inc. [Online; navštíveno 20.04.2016].
URL https://tradewave.net/features

[3] Australian Securities and Investments Commission (ASIC): Inflating the share price. 2014, [Online; navštíveno 11.1.2015].
URL https://www.moneysmart.gov.au/scams/investment-scams/inflating-the-share-price

[4] Bitboy: Bitcoin Logo. 2014, [Online; navštíveno 30.12.2014].
URL http://www.canbike.org/information-technology/bitcoin-logo-png-svg.html

[5] Bitcoincharts.com: Bitcoin Charts. 2014, [Online; navštíveno 26.12.2014].
URL http://bitcoincharts.com/charts/

[6] Bitcoinmoney.com: Most expensive pizzas Papa Johns has ever delivered. 2011, [Online; navštíveno 29.12.2014].
URL http://bitcoinmoney.com/post/3109720916/laszlos-pizza

[7] Coin ATM Radar: Bitcoin ATM machine. [Online; navštíveno 19.05.2016].
URL https://coinatmradar.com/bitcoin_atm/223/bitcoin-atm-general-bytes-brno-galerie-vankovka-brno/

[8] CoinMarketCap: Crypto-Currency Market Capitalizations. 2014, [Online; navštíveno 6.1.2015].
URL http://coinmarketcap.com/currencies/volume/24-hour/

[9] CryptoCoin Charts: Cryptocurrency Exchanges / Markets List. 2014, [Online; navštíveno 6.1.2015].
URL http://www.cryptocoincharts.info/markets/info

[10] Cryptonator: Cryptocurrency online exchanges. 2014, [Online; navštíveno 6.1.2015].
URL https://www.cryptonator.com/exchange

[11] Dougherty, C.: Bitcoin Price Plunges as Mt. Gox Exchange Halts Activity. *Bloomberg*, 2014.
URL http://www.bloomberg.com/news/2014-02-07/bitcoin-price-falls-as-mt-gox-exchange-halts-activity.html

[12] Fuller, N.: The 'Most Successful' Price Action Trader in History: Munehisa Homma. 2013.
URL http://www.learntotradethemarket.com/forex-articles/most-successful-price-action-trader-in-history-munehisa-homma

[13] Gurney, K.: *An introduction to neural networks*. London: UCL Press, 1997, ISBN 1857285034.

[14] Hannun, A.; Case, C.; Casper, J.; aj.: Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[15] Hochreiter, S.; Schmidhuber, J.: Long short-term memory. *Neural computation*, ročník 9, č. 8, 1997: s. 1735–1780.

[16] Howbert, J.: Introduction to Machine Learning - Classification / Regression. 2012.
URL http://courses.washington.edu/css490/2012.Winter/lecture_slides/14_neural_networks_2.pdf

[17] International", E.: Standard ECMA-404 The JSON Data Interchange Format. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf, 2013, [Online; navštíveno 03.05.2016].

[18] Kitco News: 2013: Year Of The Bitcoin. *Forbes*, 2013.
URL http://www.forbes.com/sites/kitconews/2013/12/10/2013-year-of-the-bitcoin/

[19] Litecoin.info: Goods merchants. 2014, [Online; navštíveno 29.12.2014].
URL https://litecoin.info/Service_directory#Goods_merchants

[20] Maltarollo, V. G.; da Silva, A. B. F.; Honório, K. M.: *Applications of artificial neural networks in chemical problems*. INTECH Open Access Publisher, 2013.

[21] Morris, G.: *Candlestick charting explained : timeless techniques for trading stocks and futures*. New York: McGraw-Hill, 2006, ISBN 007146154X.

[22] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System.
URL https://bitcoin.org/bitcoin.pdf

[23] Otohs: Cryptsy Exchange - Warning, unable to withdrawal funds, now one month plus. https://www.reddit.com/r/Bitcoin/comments/3rvl2a/cryptsy_exchange_warning_unable_to_withdrawal/, 2015, [Online; navštíveno 03.05.2016].

[24] P2P Foundation: Network Topology. 2008, [Online; navštíveno 10.1.2015].
URL http://p2pfoundation.net/Network_Topology

[25] Pearlmutter, B. A.: Gradient calculations for dynamic recurrent neural networks: a survey. *Neural Networks, IEEE Transactions on neural networks*, ročník 6, č. 5, 1995: s. 1212–1228.

[26] Rumelhart, D. E.; McClelland, J. L.: Learning and Relearning in Boltzmann Machines. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, 1996: s. 282–317.

[27] Sak, H.; Senior, A.; Rao, K.; aj.: Google voice search: faster and more accurate. http://googleresearch.blogspot.cz/2015/09/google-voice-search-faster-and-more.html, 2015, [Online; navštíveno 01.05.2016].

[28] SIDEL, R.: Mt. Gox Resigns From Bitcoin Foundation. *The Wall Street Journal*, 2014.
URL
http://www.wsj.com/articles/SB10001424052702303426304579401883794330454

[29] University of Nebraska–Lincoln: Artificial neural networks — University of Nebraska–Lincoln, Wikipedia. http://cse-wiki.unl.edu/wiki/index.php?title=Artificial_Neural_Networks&oldid=23280, 2016, [Online; navštíveno 29.04.2016].

[30] Wikipedie: Litecoin. 2014, [Online; navštíveno 29.12.2014].
URL http://en.wikipedia.org/wiki/Litecoin

[31] Wikipedie: Mt. Gox. 2015, [Online; navštíveno 7.1.2015].
URL http://en.wikipedia.org/wiki/Mt._Gox

[32] Wikipedie: Ripple (payment protocol). 2015, [Online; navštíveno 6.1.2015].
URL http://en.wikipedia.org/wiki/Ripple_%28payment_protocol%29

[33] Wikipedie: Artificial neural network. http://en.wikipedia.org/w/index.php?title=Artificial%20neural%20network&oldid=716927100, 2016, [Online; navštíveno 26-April-2016].

[34] Wikipedie: Artificial neuron. http://en.wikipedia.org/w/index.php?title=Artificial%20neuron&oldid=709135736, 2016, [Online; navštíveno 27-April-2016].

[35] Wikipedie: Boltzmann machine. http://en.wikipedia.org/w/index.php?title=Boltzmann%20machine&oldid=717223720, 2016, [Online; navštíveno 01-May-2016].

[36] Wikipedie: Hopfield network — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Hopfield%20network&oldid=717991382, 2016, [Online; navštíveno 01.05.2016].

[37] Wikipedie: Long short-term memory. http://en.wikipedia.org/w/index.php?title=Long%20short-term%20memory&oldid=706061510, 2016, [Online; navštíveno 01.05.2016].

[38] Wikipedie: Perceptron.
https://cs.wikipedia.org/w/index.php?title=Perceptron&oldid=12313597, 2016, [Online; navštíveno 29.04.2016].

# Příloha A

# Obsah CD

- ./lib/ – Složka s knihovnou modulů.
- ./server/ – Složka s datovým serverem.
- ./demo/ – Složka s demonstračními skripty.