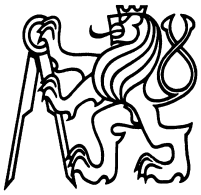
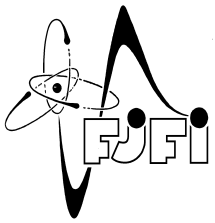


České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Strategie proti technické analýze

Counterstrategy to technical analysis

Bakalářská práce

Autor: **Peter Hron**

Vedoucí práce: **RNDr. Martin Šmíd, PhD.**

Akademický rok: 2020/2021

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

Katedra: matematiky

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Peter Hron
Studijní program:	Aplikace přírodních věd
Studijní obor:	Matematické inženýrství
Studijní zaměření:	Aplikované matematicko-stochastické metody
Název práce (česky):	Strategie proti technické analýze
Název práce (anglicky):	Counterstrategy to Technical analysis

Pokyny pro vypracování:

1) Posluchač se seznámí s prvky přibližného dynamického programování*, což je metoda matematické optimalizace spočívající v postupné aproximaci hodnotové funkce.
2) Posluchač si dále vybere jednu z metod technické analýzy, což je soubor heuristických pravidel pro obchodování na finančních trzích.
3) Konkrétně v rámci simulačního prostředí, používaného na školícím pracovišti, vybranou metodu technické analýzy (v C++) implementuje.
4) Pomocí některé metody přibližného dynamického programování navrhne a implementuje protistrategii, pro niž následně pomocí simulace odhadne střední výnos a riziko.
5) *Přibližné dynamické programování (approximate dynamic programming) je moderní numerická metoda, spočívající v postupné aproximaci hodnotové funkce. Její základní verze se snaží odhadnout optimální politiku pro každý prvek stavového prostoru, nastává zde však trojí prokletí dimenzionality: stavového prostoru, rozhodovacího prostoru a náhodného prvku. Některé úlohy (mezi něž bohužel patří právě ty finanční), navíc nelze rozumně s konečným stavovým prostorem formulovat: toto vše jsou výzvy, kterým bude muset posluchač čelit a jejich úspěšné překonání může být cenným výsledkem práce.

Doporučená literatura:

1) O. Guéant, The Financial Mathematics of Market Liquidity: From Optimal Execution to Market Making. Chapman & Hall, 2016.

2) W. B. Powell, Approximate Dynamic Programming: Solving the curses of dimensionality. John Wiley & Sons, 2011.

3) M. Šmíd, Estimation of zero-intelligence models by L1 data. Quantitative Finance 16(9), 2016, 1423-1444. DOI: 10.1080/14697688.2016.1149612

4) M. Šmíd, Asynchronous Risk-Averse Decision Model of Trading on a Limit Order Market (conference presentation). The 8th HF Conference, Stevens Institute, Hoboken, NJ, June 27, 2019.

Jméno a pracoviště vedoucího bakalářské práce:

RNDr. Martin Šmíd, PhD.
Ústav teorie informace a automatizace, Akademie věd ČR, Pod Vodárenskou věží 4, 182 00, Praha 8

Jméno a pracoviště konzultanta:

Datum zadání bakalářské práce: 31.10.2020

Datum odevzdání bakalářské práce: 7.7.2021

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 30.10.2020

.....
garant oboru

.....
vedoucí katedry

.....
děkan

Poďakovanie
V prvom rade by som sa rád poďakoval vedúcemu práce, Martinovi Šmídovi, za jeho venovaný čas, cenné rady, ústretový prístup a dôsledné vedenie pri tvorbe práce.

Záverom svojho štúdia vyjadrujem tiež veľkú vďaku celej mojej rodine a kruhu priateľov za ich bezhraničnú podporu nielen pri tvorbe tohto textu, ale aj v ďalších náročných obdobiach počas uplynulých troch rokov.

Čestné prehlásenie

Prehlasujem, že som prácu v jej plnom rozsahu vypracoval samostatne pod vedením školiteľa a s pomocou uvedenej použitej literatúry.

V Prahe dňa 7.7.2021 Peter Hron

Název práce: **Strategie proti technické analýze**

Autor: Peter Hron

Obor: Matematické inženýrství

Zaměření: Aplikované matematicko-stochastické metody

Druh práce: Bakalářská práce

Vedoucí práce: RNDr. Martin Šmíd, PhD., Ústav teorie informace a automatizace AV ČR

Abstrakt: Práca poskytuje základný úvod do teórie dynamického a približného dynamického programovania s dôrazom na využitie daných metód v oblastiach finančných trhov. Na modeli zjednodušeného obchodovania s jednou akciou sa venuje analýze profitability troch metód technickej analýzy (moving average convergence-divergence, relative strength index a ease of movement) a tiež stratégie založenej na dynamickom programovaní. Následne pre model vysokofrekvenčného trhu implementujeme stratégiu technickej analýzy MACD a navhrneme a implementujeme stratégiu využívajúcu metódy približného dynamického programovania. Profitabilitu týchto dvoch stratégii porovnáme simuláciou.

Klíčová slova: dynamické programovanie, približné dynamické programovanie, simulátor finančného trhu, technická analýza

Title: **Counterstrategy to technical analysis**

Author: Peter Hron

Abstract: The thesis introduces the fundamentals of the theory of dynamic and approximate dynamic programming with emphasis on applications of the mentioned methods in the field of financial markets. Using a simplified model of 1-stock trading, it offers an analysis of profitability of three methods of technical analysis (moving average convergence-divergence, relative strength index and ease of movement) and also of a devised investment strategy based on dynamic programming. Subsequently, we use a model of high-frequency trading market to implement the MACD technical analysis indicator and devise and implement another strategy using the approximate dynamic programming framework. By way of simulation, we compare the profitability of the two strategies.

Keywords: dynamic programming, approximate dynamic programming, financial market simulator, technical analysis

Obsah

Úvod

Pojednanie o tom, ako spoľahlivo zarobiť na finančnom trhu, je predmetom skúmania rôznych vedeckých odvetví už od vzniku kapitálových trhov v 17. storočí. Dnes je investovanie na bežných finančných trhoch prevažne automatizované pomocou programov, ktoré istým spôsobom algoritmicky obchodujú na trhu. Potreba vylepšovať tieto algoritmy, ale aj hľadať iné spoľahlivé a profitabilné metódy pre investovanie obchodníkov viedli rôznymi smermi. Jedným zo smerov je technická analýza ako súbor heuristických metód založených na sledovaní opakujúcich sa trendov. Tieto metódy sú matematicky aj ekonomicky vedecky nepodložené, za svoju už takmer storočnú históriu si však našli množstvo podporovateľov, ktorí ich používaním zarobili množstvo peňazí. Na druhej strane majú minimálne toľko odporcov, a tak diskusia o ich kvalite zďaleka nie je ukončená.

Ďalšou z možností, ako obchodovať na trhu, je nepochybne využitie matematických metód. Exaktná matematika je na nepredvídateľné prostredie kapitálových trhov len ťažko aplikovateľná, uplatnenie si ale nájdu štatistické a stochastické metódy. Nakoľko bežnou stratégiou obchodníka na trhu je maximalizovať zisk, prirodzene sa hlásia o slovo aj metódy matematickej optimalizácie. Jedna z nich, dynamické programovanie, poskytuje od svojho vzniku v 50. rokoch 20. storočia kostru pre riešenie problémov z oblasti operačného výskumu, s priemyselnými aplikáciami od optimálneho priraďovania vodičov kamiónov k zákazkám až po riadenie účtovníctva veľkých firiem.

V práci sa budeme venovať pokusom o aplikáciu týchto pre finančný trh neštandardných metód k hľadaniu investičných stratégii. Na začiatku uvádzame základný prehľad o matematických metódach, ktoré v ďalších kapitolách aplikujeme na finančné problémy. Definujeme základné pojmy dynamického a približného dynamického programovania, opisujeme známe používané algoritmy pre riešenia problémov a diskutujeme pozítíva aj nedostatky ich využitia. V druhej kapitole využijeme dynamické programovanie k vytvoreniu dvoch netriviálnych investičných stratégii pre jednoduché obchodovanie s 1 akciou. Pomocou implementácie v programovacích prostrediach sa pokúsime o analýzu profitability týchto navrhnutých stratégií a tiež stratégií založených na troch klasických indikátoroch technickej analýzy. V závere práce sa presunieme na simulátor vysokofrekvenčného trhu, kde implementujeme indikátor technickej analýzy MACD a navrhneme a implementujeme vlastnú stratégiu založenú na približnom dynamickom programovaní. Simuláciou tohto trhu s viacerými účastníkmi porovnáme ich profitabilitu.

Kapitola 1

Dynamické programovanie

1.1 Motivácia a historický vývoj

Optimalizačné úlohy ľudstvo sužujú už od nepamäti. V princípe triviálne myšlienkové postupy, ako zvládať bežné situácie, či ťažkosti, ktoré nám život stavia do cesty, často kulminujú práve k formulácii úloh, pri ktorých je potrebné maximalizovať či minimalizovať rôzne hodnoty. Už bežné prírodné úkazy, ako napríklad správanie sa sťahovavých vtákov pri migrácii do teplejších oblastí, javia znaky optimalizácie, kde vtáky kombináciou zmyslov, vrodených reflexov či odpozorovaného správania za chodu analyzujú informácie o okolí a optimalizujú rýchlosť, výšku či formáciu letu, aby sa do svojich destinácii dostali čo najrýchlejšie a s čo najmenšou spotrebou energie. Táto vtáčia stratégia dala názov algoritmu (Migrating birds optimization), ktorého rôzne formy sa využívajú v riešeniach hneď niekoľkých optimalizačných problémov. Niektoré z nich, napr. [Knapsack MBO] už s letom vtákov majú pramálo spoločného. Aj nesťahované vtáky majú k optimalizácii očividne veľmi blízko, kde niektoré druhy jej princípy využívajú k znižovaniu energetickej náročnosti lovu, a to kombináciou úsporného letu a skokov po stromoch. Mechanike a energetike tohto optimalizovaného pohybu sa venuje napríklad [Birds]. Článok [HoO] zasa opisuje rôzne ďalšie prírodné a kultúrne úkazy, kde všade sa aj bez najmenšej zmienky o matematike vyskytujú problémy a riešenia súvisiace s optimalizáciou; niektoré z nich, verím, stoja za zmienku v nasledujúcich odstavcoch.

Nepochybne zaujímavý je princíp pavúčieho lovu, kde sa pavúk vydá ku koristi zaseknutej v pavučine istou netriviálnou cestou po vláknach pavučiny - podľa názoru niektorých biológov práve tou najkratšou. Tu je analógia so slávnou skupinou optimalizačných problémov najkratšej cesty už na pohľad zrejmá.

Aj v starovekej čínskej literatúre nachádzame zaujímavé príklady optimalizácie. Keď podľa príbehu kráľ oblasti Qi vyzval na konské preteky jedného zo svojich subjektov, chovateľ Tian Gi využil k porazeniu kráľa jednoduchú optimalizačnú stratégiu: keď po rozbehu videl, že najrýchlejší kráľov kôň je rýchlejší ako jeho najrýchlejší, a rovnako druhý aj tretí najrýchlejší sú rýchlejší ako jeho druhý a tretí, musel pred finálnymi pretekmi porozmýšľať. Do troch pretekov mal postaviť tri kone: jedného na súboj s najrýchlejším kráľovým, druhého na súboj s 2. najrýchlejším kráľovým, a tretieho na súboj s 3. najrýchlejším kráľovým. Víťazstvo v 2 z 3 pretekov tak dosiahol jednoducho: obetoval prvý

pretek, kde s najrýchlejším kráľovým postavil do súboja svojho 3., teda najpomalšieho. V druhom preteku postavil svojho najrýchlejšieho, ktorý bol schopný poraziť kráľovho 2. najrýchlejšieho, rovnako ako v treťom preteku Tianov 2. najrýchlejší porazil kráľovho 3. najrýchlejšieho. Tian Gi dokázal využiť rozhodovací proces vo svoj prospech a zo situácie s trochou rozumu vyťažil maximum optimálnou stratégiou. Aj dnes dynamickým programovaním hľadáme optimálne stratégie, veľmi často spôsobom principiálne podobným tomu, aký použil Tian Gi dávno pred zrodom matematickej optimalizácie a to analýzou možností, ktoré nastanú našimi rozhodnutiami a následným výberom tých rozhodnutí, ktoré vedú k optimálnemu riešeniu celého problému.

Vyššie uvedené príklady (a množstvo ďalších) demonštrujú, že optimalita je nepochybne prirodzenou súčasťou prírody. Je fundamentálnym faktorom v prírodných procesoch, riadi správanie a život zvierat aj človeka pri sociálnych či biologických aktivitách. Ľudia sa implicitne aj explicitne optimalizačným problémom venujú tisícročia, avšak prístupy, ktoré pri riešení používali, sa principiálne líšia s tými dnešnými. V minulosti k hľadaniu optimálnych stratégii neboli k dispozícii simulácie a dodnes nie je úplne presvedčivo vysvetlené, ako vývoj optimalizačných metód vlastne prebiehal. Ak sa tri obce nevedeli rozhodnúť, v ktorej z nich postaviť školu pre deti tak, aby to mali deti čo najbližšie zo všetkých troch, bol na svete optimalizačný problém, ktorý sa dal simulovať hneď niekoľkými spôsobmi. Za jeden zo zaujímavých sa dá považovať využitie gravitácie: vytvorené sú tri diery v stole (alebo podobnej horizontálne stojacej doske), ktoré svojím rozložením kopírujú pôdorys umiestnenia obcí. Do dier sa z hornej časti stola prestrčia tri kusy povrazu. Konce povrazov nad doskou sú zviazané do jedného uzla a na dolné konce všetkých troch je umiestnené závažie rovnakej hmotnosti. Keď sa tento systém vplyvom gravitácie dostane do rovnováhy, poloha uzla na stole je miestom optimálneho umiestnenia školy. Optimalizačný problém takéhoto jednoduchého rázu sa ale samozrejme dá kvantifikovať aj matematicky. Ešte predtým, než sa dostaneme k optimalizačnému formalizmu, zhrnime pre úplnosť v nasledujúcich odstavcoch v rýchlosti historický vývoj dynamického programovania, ako je opísaný v [HistofDP].

Koncept dynamického programovania vznikol na prelome 40. a 50. rokoch 20. storočia v práci Richarda Bellmana[1]. V inštitúte RAND[2], kde v oddelení pre matematiku pracoval od roku 1949, sa spolu s kolegami v tomto období venovali rôznym aplikačným problémom teórie hier. Jeden z dôležitých problémov bola optimálna stratégia výberu cieľov, napríklad vo vojnových stavoch pri voľbe strategických objektov nepriateľa, ktoré majú byť zasiahnuté navigovanými raketami. V tej dobe sa k optimalizačným úlohám tohto typu používala teória variačného počtu, ktorá ale často nie je použiteľná pri diskrétnych stavových priestoroch či prípadoch, kde pravdepodobnostné funkcie nie sú diferencovateľné. Bellman na tieto problémy nasadil rôzne algoritmy „hrubej sily“, ktoré ho postupne doviedli k objaveniu techník, ktoré pri riešení postupujú istou formou rozumne zvolenej rekurzie. Tento postup formuloval pre stochastické problémy s nasledujúcimi vlastnosťami:

1. stavy systému je možné opísať malým množstvom parametrov;

[1]Richard Ernest Bellman (26.8.1920 - 19.3.1984) bol americký aplikovaný matematik

[2]RAND Corporation (založený 1948) je spoločnosť a think tank zaoberajúci sa vedou a výskumom pre Ozbrojené sily USA

2. v každom stave sa vykoná rozhodnutie, ktorého transformuje aktuálny stav na nasledujúci, zasa opísateľný podobnými parametrami;

3. historický vývoj systému nijako neovplyvňuje budúcnosť, t.j. istá forma Markovovej vlastnosti.

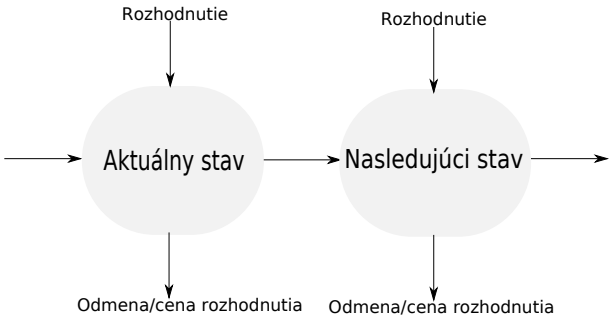
Túto techniku nazval Bellman „princípom optimality“, ktorá pod optimálnou stratégiou rozumie takú, ktorá je optimálna v každom kroku, teda nezávisle na tom, aké rozhodnutia sa urobia v predošlých časoch. Jediné, na čom rozhodovacia stratégia v danom kroku závisí, je aktuálny stav, v ktorom sa systém nachádza. Tento stav tak v sebe nesie všetky informácie, ktoré v danom kroku potrebujeme na vykonanie rozhodnutí, čo je myšlienka, s ktorou sa dynamické programovanie pasuje dodnes. Práve množina týchto stavov, tzv. stavový priestor, si však so sebou nesie jedno z „prekliatí dimenzionality (curses of dimensionality)“, čo je ďalší pojem pochádzajúci od Bellmana vyjadrujúci fakt, že v drvivom množstve modelovaných problémov je množina prípustných stavov enormne veľká, čo spôsobuje nárast časovej aj pamäťovej náročnosti možných algoritmov na riešenie. Aj preto je aj v dnešnej dobe takmer nemožné vyriešiť „veľké“ problémy dynamického programovania a vo väčšine prípadov sa tak používa istá forma aproximácie.

Moderné problémy dynamického programovania sú teda prevažne o nájdení „rozumnej“ formy opisu problému, pod čím rozumieme úlohu čo najlepšie zvoliť stavový systém tak, aby riešiteľa nezaskočilo prekliatie dimenzionality. K tomuto sa od polovice minulého storočia vymyslelo množstvo techník a algoritmov, prekliate dimenzionality tu však pri enumeratívnych matematických problémoch, ktorých riešenia v istom zjednodušenom zmysle vyžadujú prejsť všetky možností, stále bude. Práve preto sa aj v dnešnej dobe superrýchlych počítačov a pokročilých výpočtových metód stále javí byť najlepším postupom kombinácia istých foriem aproximácie, selektívnych výpočtov a tzv. sedliackeho rozumu - t.j. heuristického pozorovania skúseného oka človeka či prípadných modelov umelej inteligencie.

Záverom tejto sekcie sa isté patrí objasniť názov kapitoly ako taký - prečo si Bellman zvolil práve názov dynamické programovanie? Podľa spomienok Bellmana a jeho blízkych kolegov z [HistofDP] to vzniklo ako krycie meno výskumu, ktorému sa v RAND-e venovali. Začiatkom 50. rokov bol ministrom obrany Spojených štátov Robert Lovett[3], a on ani jeho jeho nástupca Charles Wilson[4] neboli fanúšikmi teoretického vedeckého výskumu. Bellman tak potreboval fakt, že sa venuje matematickému výskumu, pred vysokými predstaviteľmi utajiť - RAND Corp. bol financovaný, a de facto aj riadený ministerstvom obrany - a tak prišiel k názvu dynamické programovanie. Vzniklo to spojením dvoch termínov: programovanie, ako zobecnenie pojmu rozmýšľania, plánovania, robenia rozhodnutí, čo bolo v princípe to, o čo sa vo výskume snažili; zatiaľ čo prídavné meno dynamický dobre opisovalo formu, akou sa samotné riešienie vykonávalo - rôznorodé, adaptívne, v časových krokoch. Rolu tiež hral fakt, že k prídavnému menu dynamický (dynamic) by človek len ťažko hľadal negatívny podtón, a tak týmto výberom zabili hneď dve muchy jednou ranou.

[3]Robert Abercrombie Lovett (14.9.1895 – 7.5.1986) bol americký vojak a politik, 4. minister obrany USA (1951 - 53)

[4]Charles Erwin Wilson (18.7.1890 – 26.9.1961) bol americký inžinier, businessman a politik, 5. minister obrany USA (1953 - 57)



Obr. 1.2.1: Princíp rozhodovacieho procesu medzi stavmi

Aj preto v roku 1953 v RAND-e vyšiel prvý report o dynamickom programovaní práve s týmto názvom (An Introduction to the Theory of Dynamic Programming), a v roku 1957 nasledovala kompletná kniha [Bellman], ktorá dodnes poskytuje kostru pre výskum v dynamickom programovaní.

1.2 Typické problémy dynamického programovania

[Puterman] a [Powell] vymenúvajú množstvo modelových úloh a problémov z matematickej teórie, ale aj reálnej praxe, kde sa riešenie dá modelovať a hladať pomocou dynamického programovania. Uveďme aspoň niektoré ich základné črty. Optimalizačné úlohy s rozhodnutiami vznikajú v oblastiach ľudskej činnosti od investovania na finančnom trhu, cez priemyselnú výrobu až po management národných hospodárstiev svetových veľmocí. Vyskytnú sa ale aj pri voľnočasových aktivitách, s mnohými aplikáciam i v hrách ako je backgammon či bridž. Všetky tieto modelové problémy majú charakteristiky robenia rozhodnutí, následným pozorovaním zmien a nového stavu po vykonaní rozhodnutia a následne robenie ďalších rozhodnutí s použitím novozistených informácii. Patria do skupiny tzv. sekvenčných rozhodovacích procesov (sequential decision problems), ktorých formulácia je často zrejmá z podstaty modelu, ale riešenie je zvyčajne vysoko netriviálne a jeho charakter neodhadnuteľný z formulácie problému ako takého.

Inžinierske a ekonomické problémy sú zvyčajne formulované so spojitými stavovými aj rozhodovacími priestormi, zatiaľ čo operačný produkuje modely s diskrétnymi aj spojitými parametrami. V oboch prípadoch sa však jedná o formuláciu modelu v podobe rekurzívnych rovníc, ktorých premenné sú prvky stavového priestoru vyjadrujúce závislosť systému na celej doterajšej histórii v istej skompaktnenej podobe. V každom stave sa po zvolení rozhodnutia dostaneme do ďalšieho stavu. Rozhodnutia vykonávame podľa toho, čo ich vykonanie prinesie - napríklad pri investovaní na finančnom trhu nás rozhodnutie „kúpiť akciu“ bude stáť pomyselnú cenu akcie v danom čase. Podobne pri rozhodnutí „predať akciu“ zinkasujeme odmenu v podobe predajnej ceny. Toto postupné rozhodovanie na základe dostupných informácii je hnacou silou dynamického programu.

Spomeňme bodovo zopár špecifických, ale stále veľmi širokých tried problémov, kde sa hojne využíva dynamické programovanie.

1. **Distribúcia rozpočtov.** Pre daný fixný rozpočet hľadáme optimálne rozloženie zdrojov na aktivity, ktoré prinášajú zisk alebo stratu prípadne závisiac na tom, koľko do danej aktivity investujeme. Každá spoločnosť potrebuje pracovať so svojim rozpočtom tak, aby minimalizovala stratu, resp. maximalizovala zisk za nejakých podmienok (medzi ktoré môže patriť napríklad práve fixná hodnota rozpočtu, príležitosti na investície, atď.). Kandidáti na politické funkcie potrebujú alokovať zdroje na reklamu, chod kampane či ďalšie výdavky, alebo sa rozhodovať ktoré špecifické miesta v štáte navštíviť a koľko času tam stráviť.

2. **Skladovanie.** Majme fixný obmedzený skladovací priestor (sklad potravín či surovín, ale aj napríklad nabitý batériový článok v ktorom „skladujeme" energiu), v ktorom je potrebné rozhodovať, koľko surovín či energie uskladniť, a naopak, koľko voľného priestoru sa nám oplatí ponechať. Rozhodnutia robíme na základe vplyvu vonkajších parametrov, ako sú aktuálne ceny surovín či energie, koľko miesta majú k dispozícii konkurenčné sklady, a podobne.

3. **Predaj a nákup na finančnom trhu.** Oplatí sa za daných podmienok (cena, rozpočet,...) kúpiť akcie či komodity na finančnom trhu? Kedy vlastnené akcie naopak predať? Finančná problematika trhu s racionálnymi agentmi je práve tématikou podstatnej časti práce.

4. **Cenotvorba (pricing).** Ako spomíname vyššie, dynamické programovanie je použiteľné v modelovaní správania sa na trhu. Neprekvapí teda, že jeho obdoby nájdu uplatnenie v pricingu akcií či opcií, nakoľko cena je silne ovplyvnená správaním sa agentov na trhu.

5. **Dynamická alokácia.** K dispozícii majme fixný počet zdrojov. Za daných podmienok je potom úlohou rozhodnúť, ako optimálne alokovať tieto zdroje k rôznym aktivitám, aby sa maximalizoval istý úžitok. Iste nezaškodí predstava napríklad ľudských zdrojov - počet vodičov nákladných áut, ktorých máme k dispozícií na odvoz či prívoz tovaru. Niektorí z dostupných vodičov môžu mať povolenie šoférovať len v istých krajinách, zatiaľ čo všetci môžu byť na ceste len istý počet hodín v týždni. Na základe týchto (alebo rôzných ďalších) podmienok rozhodnime, ako optimálne alokovať vodičov k jednotlivým zákazkám, aby sa maximalizoval profit spoločnosti, ale zároveň boli dodržané pracovné hodiny vodičov a rešpektované ich povolenia.

Záverom pre názornosť explicitne definujme a naznačme riešenie slávneho problému, v ktorom zreteľne zaznejú už spomínané techniky dynamického programovania.

1.2.1 Problém najkratšej cesty

Problém najkratšej cesty je slávnym problémom teórie grafov, v ktorom hľadáme cestu medzi dvoma vrcholmi grafu tak, aby táto cesta bola v istom zmysle najkratšia. V rýchlosti uveďme základné pojmy teórie grafov, aby bolo možné sa v terminológii zorientovať.

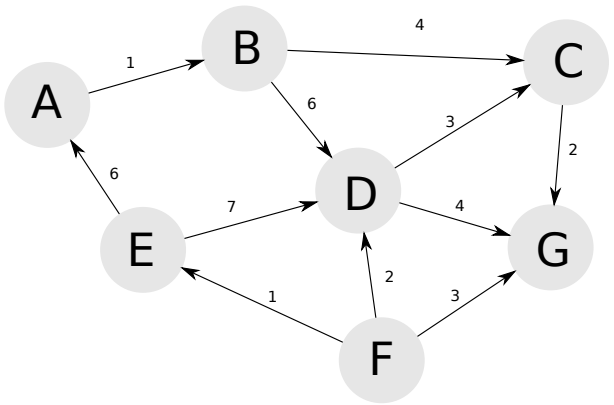
Grafom G rozumieme usporiadanú dvojicu (V, E) neprázdnych konečných množín, kde V nazývame množinou vrcholov a E nazývame množinou hrán - dvojíc vrcholov, medzi

ktorými je možné prechádzať. Podľa charakteru množiny E - obsahovať môže usporiadané, resp. neusporiadané dvojice vrcholov - rozdeľujeme grafy na orientované (prechádzať hranou je možné len smerom z prvého do druhého vrcholu usporiadanej dvojice), resp. neorientované (prechádzanie hranou nie je obmedzené smerom). Graf ďalej nazveme ohodnotený, ak je k dispozícii neprázdna množina W tzv. váh, pomocou ktorých každej hrane $e \in E$ priradíme reálne číslo $w \in W$. Problém najkratších ciest môžeme ďalej formulovať pre rôzne typy grafov, z ktorých vyplynú rôzne typy riešiacich algoritmov. Pre ilustráciu zvoľme orientovaný ohodnotený graf, pohyb v ktorom môže napríklad modelovať pohyb chodca po (v našom prípade jednosmerných) chodníkoch (hranách) medzi križovatkami (vrcholmi) veľkomesta. Váhy jednolivých hrán môžeme chápať ako čas potrebný na prejdenie jednotlivej hrany.

Riešenie problému teda bude hľadať najkratšiu cestu medzi zvolenou začiatočnou a_0 a koncovou križovatkou a_k, to jest množinu križovatiek $A = (a_0, a_1, \ldots, a_k)$ z V, ktoré postupne navštívime pri prechádzaní chodníkmi $c_i = (a_i, a_{i+1}) \in E$ pre $i \in \{0, \ldots, k-1\}$. Ak označíme $t(u, v)$ váhu hrany (u, v), teda čas potrebný na prejdenie chodníkom z križovatky u do križovatky v, úlohou je minimalizovať celkový čas na prejdenie tejto cesty v tvare

$$T = \sum_{i=0}^{k-1} t(a_i, a_{i+1}).$$

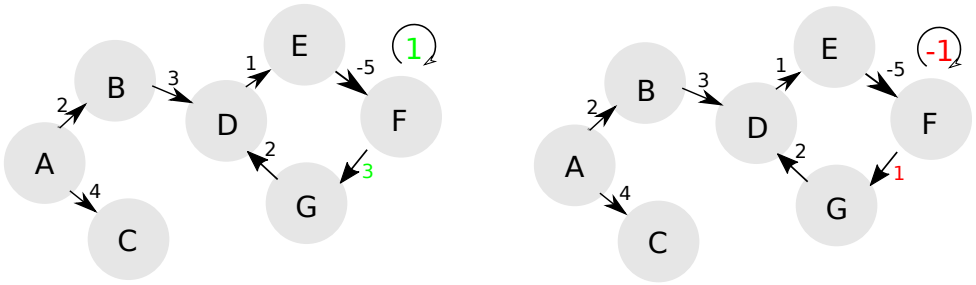
Pred naznačením riešenia diskutujme dve prirodzené komplikácie. Prvou z nich je fakt, že doteraz sme mlčky predpokladali, že spomínaná najkratšia cesta existuje. Ako naznačuje Obrázok 1.2.2., v obecnom orientovanom ohodnotenom grafe nie je v žiadnom prípade zaručené, že existuje vôbec nejaká cesta medzi dvomi ľubovoľne zvolenými vrcholmi. Aj kvôli zachyteniu tejto možnosti v prvom kroku algoritmu riešenia volíme za počiatočné hodnoty dĺžky cesty nekonečno (v praxi veľmi veľké čísla, často nazývané „big M“ - veľké M), ako bude upresenené v algoritme.



Obr. 1.2.2: Cesta po grafe z vrcholu A do vrcholu E neexistuje. Hľadanie tej najkratšej teda pochopiteľne nedáva zmysel.

Problémy s existenciou riešenia môžu tiež vzniknúť v grafoch, ktorých hrany môžu mať váhy negatívnych hodnôt. K objaseneniu tohto problému definujme pojem cyklu v grafe.

Majme graf $G = (V, E)$ a množinu vzájomne rôznych vrcholov $\{v_1, \ldots, v_k\} \subset V$. Ak existujú hrany $h_i = (v_i, v_{i+1}) \in E$ pre všetky $i = 1, \ldots, k-1$ a tiež $h_k = (v_k, v_1) \in E$, potom postupnosť $(v_1, h_1, v_2, h_2, \ldots, h_{k-1}, v_k, h_k, v_1)$ nazveme cyklom v G. Graf obsahujúci cyklus nazveme cyklický. V cyklických grafoch, ktoré obsahujú tzv. cyklus so záporným súčtom váh, to jest $\sum_{i=1}^{k} h_k < 0$, môže pri probléme najkratšej cesty nastať prirodzený problém: najkratšia cesta sa môže vyskytnúť v cykle, ktorého postupným prechádzaním sa súčet váh znižuje. Zrejme teda „najkratšia možná cesta“ bude v takomto prípade dĺžky $-\infty$, a to po cyklickom nekonečnom prechádzaní vybraného cyklu. Takýto príklad ilustruje Obrázok 1.2.3.



Obr. 1.2.3: Cyklický graf naľavo obsahuje cyklus s pozitívnym súčtom váh. Tento cyklus tak neprekáža typickému algoritmu hľadania najkratšej cesty z A do E. Graf napravo obsahuje cyklus s negatívnym súčtom váh, a v tomto grafe najkratšia z A do E cesta nie je prístupná (klesá do $-\infty$).

Matematicky tento prípad môžeme považovať za triviálny a nie veľmi zaujímavý; pripomeňme však, že problémy najkratšej cesty vznikali v rôznych oblastiach praxi, kde negatívne váhy hrán v modeloch sú veľmi bežné. Za všetky spomeňme jeden príklad: model bankového účtu ako grafu, kde presun medzi vrcholmi modeluje pohyb stavu prostriedkov na účte, ktorého zmeny môžu byť nepochybne kladné aj záporné.

S riešeniami podobných modelov bol taktiež obojstranne prepojený vývoj výpočtovej techniky. Kvôli neochote výpočtovej techniky pracovať s nekonečnami tak bolo nutné funkcionalitu riešení pre tieto prípady algoritmicky ošetriť. Napríklad známy Djikstrov algoritmus je v priemere rýchlejší než Bellman-Fordov, ale na rozdiel od druhého spomenutého zlyhá v prípade záporných cyklov v grafe.

Zostaňme však pre jednoduchosť pri tom najjednoduchšom prípade chodca brázdiaceho jednosmerné ulice veľkomesta, teda orientovaného, ohodnoteného grafu s kladnými váhami hrán. Posuňme sa v značení od teórie grafov a skúsme situáciu modelovať jazykom dynamického programovania. Predpokladajme, že chodec začína na križovatke s a jeho destináciou je križovatka d. Označme:

S = množina všetkých možných stavov (križovatiek), v ktorých sa môže chodec nachádzať

$t(u, v)$ = čas potrebný na prejdenie chodníka z križovatky u do križovatky v

$S_o(u)$ = množina križovatiek, do ktorých sa dá prejsť **z** križovatky u

Chodec v začiatku stojí na križovatke s. Môže si vybrať, na ktorú križovatku prejsť v prvom kroku, teda volí nejaké $v \in S_o(s)$. Ak chce voliť optimálne, zrejme musí voliť v tak, že aj nasledujúci čas cesty z v do destinácie d je najnižší. Ktoré v to je, samozrejme nevie (inak by úloha bola bezpredmetná), môže ale hádať. Skúsi teda hádať iteratívne - nájde si počiatočný odhad minimálneho času potrebného na cestu z v do destinácie d, označme tento odhad $l_0(v)$. Aký odhad voliť na začiatku, ak nemáme žiadne informácie? Z dôvodov spomínaných vyššie, voliť budeme ∞ resp. nejaké dostatočne veľké číslo. Tento odhad najkratšieho času potrebného na prejdenie cesty z v do d porovnáme s najkratším časom, v ktorom by chodec mohol d dosiahnuť, ak by sa v prvom kroku vydal po nejakej dostupnej ceste do nejakej dostupnej križovatky. Nakoľko ale nemáme žiadne informácie o tom, ktorá cesta je v prvom kroku optimálna, musíme porovnať všetky možnosti. Iterujeme ďalej, vždy v kroku n nájdeme optimálne časy ako opäť minimálnu hodnotu času potrebného na prejdenie zvoleného chodníka + následnej cesty už z križovatky, do ktorej sa zvolenou cestou dostaneme. Takto odhadujeme hodnoty $l_n(v)$, až kým nie sú optimálne (t.j. prestanú sa meniť s rastúcim n). Kostra algoritmu by tak mohla vyzerať nasledovne (bližšie [Powell], str. 46-47).

Algoritmus 1.1 Riešenie úlohy najkratšej cesty

1. Inicializujem odhady
$$l_0(v) = \begin{cases} \infty & v \neq d; \\ 0 & v = d. \end{cases}$$
a kladiem na začiatok $n = 1$.

2. Spočítam pre všetky možné križovatky $u \in S$
$$l_n(u) = \min_{v \in S_0(u)} \Big(t(u,v) + l_{n-1}(v) \Big)$$

3. Ak je $l_n(u) < l_{n-1}(u)$, kladiem $n \leftarrow n+1$ a pokračujem krokom 2. Inak končím.

Analogický algoritmus by zrejme šiel vytvoriť pohľadom odzadu: počínajúc v cieľovej križovatke d postupným výpočtom najkratšej cesty z križovatiek, z ktorých je možné sa dostať do d, a rekurzívne ďalej z ďalších dostupných, podľa logiky už opísanej vyššie opačným smerom. V nasledujúcich kapitolách vyplynie, že práve tento spätný pohľad je pre dynamické programovanie veľmi prirodzený.

Je vhodné upozorniť, že ani jeden z týchto algoritmov fungujúcich len ako intuitívny príklad dynamického programu nie je zďaleka z implementačného hľadiska optimálny medzi množstvom riešení tohto extenzívne študovaného problému. Hlbšie štúdium problému najkratšej cesty samozrejme nie je zámerom práce, podrobnejšiu analýzu čitateľ nájde už v spomínanej literatúre [Powell], ale aj v množstve ďalších publikácii venujúcich sa práve algoritmom podobného rázu. Nakoľko algoritmy tohto typu využíva napríklad takmer každé navigačné zariadenie, neprekvapí, že ich štúdium a zlepšovanie je aj v súčasnosti stále dôležité.

Tak ako v riešení uvedenom vyššie, diskrétny stavový priestor rôznorodých problémov si dokáže nájsť analógiu v množine vrcholov grafu. Aj preto je problém najkratšej cesty pre štúdium dynamického programovania nesmierne zaujímavy - Powell podotýka, že takmer každý deterministický diskrétny dynamický program je možné modelovať ako problém najkratšej cesty v istom dobre zvolenom grafe.

Pri riešení si ešte všimnime typické znaky dynamického programu, ktoré opíšeme detailnejšie v nasledujúcich podkapitolách:

- **delenie problému na menšie podproblémy:** hľadanie najkratšej cesty z križovatky s do d sme postupne vyjadrili ako hľadanie najkratšej cesty z križovatky v, ktorá už je o krok bližšie k d, a rekurzívne ďalej
- **rekurzia využívajúca nové informácie v každom kroku:** pri „odhadovaní“ optimálnych časov využívame fakt, že už poznáme odhad v predošlej iterácii
- **optimálna subštruktúra riešenia - optimálny algoritmus je optimálny v každom kroku:** ľubovoľná podmnožina najkratšej cesty je opäť najkratšia „podcesta“ (ak by nebola, medzi istými križovatkami i a j by musela existovať nejaká kratšia, a jej spojením so zvyškom pôvodnej najkratšej cesty dostávame celkovú kratšiu cestu; spor)

1.3 Značenie a základné pojmy

Dynamické programovanie je podľa samotného Bellmana (viď [Bellman], Predslov) framework na riešenie problémov vyplývajúcich zo štúdia tzv. viackrokových rozhodovacích procesov (multistage decision processes). Takýto proces, a teda objekt skúmania dynamického programovania, je spravidla modelovaný pomocou nasledujúcich parametrov.

1. **Stavový priestor.** Zvyčajne značený S (z angl. state space), táto množina zahrňa všetky možné stavy, v akých sa skúmaný systém môže nachádzať. Množina S môže byť konečná, nekonečná (spočítateľná aj nespočítateľná); podobne môže byť diskrétna alebo spojitá. Kľúčovým pre dynamické programovanie je predpoklad, že stav v danom okamihu zahŕňa kompletnú informáciu o historickom vývoji systému - podľa už spomínanej analógie Markovovej vlastnosti rozhodovacích procesov je aktuálny stav všetko, čo potrebujeme k určeniu optimálnej stratégie v danom kroku.
2. **Rozhodnutia a priestor rozhodnutí.** Zvyčajne značené a (z angl. action), resp. A (z angl. action space), rozhodnutia sú premenné, ktoré kontrolujeme. V každom stave je priestor rozhodnutí závislý na danom stave, a voľba toho správneho rozhodnutia zo všetkých možných je kľúčovou časťou dynamického programovania.
3. **Zisková/nákladová funkcia.** Často C (z angl. cost funcion). Rozhodnutie, ktoré v danom stave vykonáme, má dôsledky. Tieto dôsledky - zisk, stratu či náklady spôsobené rozhodnutím - vystihuje zisková funkcia.

4. **Vonkajšie (exogénne) zmeny a informácie.** Opisujú informácie, ktoré systém ovplyvňujú zvonka - nedokážeme ich kontrolovať, a ich prínos je tak modelovaný ako náhodný.

5. **Funkcia prechodu medzi stavmi.** V každom stave vykonávame rozhodnutia, ktoré ovplyvňujú to, ako sa systém vyvinie po danom rozhodnutí. Funkciu, ktorá definuje túto závislosť zmeny stavu systému na prístupných premenných, ale aj náhodných informáciách, nazývame funkcia prechodu, a zvyčajne značíme T (z angl. transition function), prípadne S^M(z angl. model function).

6. **Hodnotová funkcia.** Značená rôzne, ale často V (z angl. value function), opisuje „hodnotu“ v danom stave. Je blízko analogická k tzv. účelovej funkcii, ktorú poznáme z optimalizácie a ktorej hodnoty sa v danom probléme snažíme maximalizovať resp. minimalizovať.

Dynamické programy modelujeme v istej postupnosti krokov, spravidla časových. Všetky vyššie uvedené objekty teda budeme parametrizovať parametrom t, ktorý bude vyjadrovať, v akom kroku sa nachádzame. Rozumieme teda značeniu x_t ako hodnote premennej x v kroku (čase) t. Pri spojitých parametroch je typické písať t ako argument, napr. $x(t)$.

Venujme ďalej samostatnú podkapitolu každému z týchto kľúčových pojmov, kde detailnejšie opíšeme potrebnú terminológiu a na príklade modelových problémov názvoslovie ilustrujeme. Podkapitoly logicky nasledujú [Powell], Kap. 5, odkiaľ je aj čerpaná inšpirácia väčšiny príkladov.

1.3.1 Stavy a stavový priestor

Powell definuje stav systému ako (dimenzionálne) najmenšiu premennú vystihujúcu historický vývoj systému, ktorej znalosť je v každom okamihu **nutná a postačujúca** k jednoznačnému určeniu zvyšných definičných parametrov systému - t.j. funkcie prechodu, ziskovej funkcie a priestoru rozhodnutí. Stav v čase t (s_t) je teda súbor informácii, ktoré v danom čase potrebujeme poznať, aby sme dokázali modelovať vývoj systému v čase $t+1$ a neskôr. Podobne, ak neplynú z danej stavovej premennej všetky potrebné informácie na výpočet parametrov (priestoru rozhodnutí, ziskovej funkcie a funkcie prechodu), táto stavová premenná nie je kompletná. Tento fakt je dobrým prostriedkom pre overenie, či daný systém modelujeme na správnom stavovom priestore. Ak isté dáta zo stavových premenných nikde pri výpočte ostatných parametrov systému nevyužijeme, tieto dáta je vhodné zahodiť a zmenšiť tak potrebný stavový priestor.

Výber stavového priestoru nie je v žiadnom prípade jednoznačný. Aj preto vyššie uvádzame (inšpirovaní Powellom, str. 179), že stav systému považujeme za „dimenzionálne najmenší“. Pri modelovaní rôznych situácii sa javí, že k modelovaniu budúcnosti systému potrebujeme poznať celú históriu dynamiky systému. To je väčšinou nepraktické. Nakoľko stavové premenné používame v rekurzívnych výpočtoch, je veľmi dôležité ich voliť tak, aby boli čo najkompaktnejšie. Taktiež pre veľkú triedu modelových úloh skutočne nepotrebujeme poznať celý historický vývoj.

Ilustruje pojem stavového priestoru na príklade. Vo finančných problémoch, akým sa práca venuje, je výber rozumného stavového priestoru kľúčová časť riešenia. Predstavme

si problém, v ktorom modelujeme vývoj ceny akcie na finančnom trhu, k čomu využívame historické ceny (napríklad ceny akcie pri večernom zatvorení burzy v posledných n dňoch). Chceme teda vždy v nový deň vypočítať odhad ceny akcie v nasledujúcom období. Pre tieto účely sa využívajú rôzne typy kĺzavých priemerov, porovnajme teda dva najpoužívanejšie z nich.

Definícia. Kĺzavým priemerom rozumieme postupný výpočet priemerov podpostupností istej postupnosti čísel (napr. časového radu). Majme teda časový rad $p_1, ..., p_n$. Pre našu ilustráciu spomeňme **jednoduchý (aritmetický)** a **exponenciálny** kĺzavý priemer.

Jednoduchým kĺzavým k-priemerom v úseku n rozumieme nevážený aritmetický priemer posledných k hodnôt, t.j.

$$SMA_k = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i$$

Exponenciálnym kĺzavým priemerom rozumieme súbor $(E_t)_{t\in\{1,...,n\}}$

$$E_t = \begin{cases} p_1 & t = 1 \\ \gamma p_t + (1-\gamma)E_{t-1} & 1 < t \leq n. \end{cases}$$

Faktor $\gamma \in (0, 1)$ volíme zvyčajne podľa toho, aký časový úsek chceme priemerovať. V prípade konštantných úsekov dĺžky T sa často volí $\gamma = \frac{2}{T+1}$.

Pri modelovaní jednoduchým priemerom bude naša predpoveď budúcej ceny v kroku n tvaru

$$\hat{p}_n = \frac{1}{k} \sum_{i=1}^{k} p_{n-i+1}$$

a teda potrebujeme k výpočtu poznať k historických cien $p_n, ..., p_{n-k+1}$. Stavový priestor by tak obsahoval stavovú premennú tvaru $s_n = (p_n, ..., p_{n-k+1})$.

Pri modelovaní exponenciálnym priemerom je predpoveď v kroku n

$$\hat{p_n} = \gamma p_n + (1-\gamma)\hat{p}_{n-1}$$

potrebujeme teda len stavovú premennú $s_n = \hat{p_n}$.

Je zrejmé, že použitím exponenciálneho priemeru „ušetríme“ na veľkosti stavovej premennej, čo môže byť pri náročnejších výpočtoch výhodné. Na druhej strane, v našich výpočtoch môžeme skutočne potrebovať práve použitie aritmetického priemeru, v ktorom prípade už samozrejme nutne musíme voliť väčší stavový priestor. V oboch prípadoch sme ale našli stavové premenné dimenzionálne najmenšie - ak predpokladáme, že ďalší vývoj nejako závisí na všetkých spomínaných cenách, pri vynechaní ktorejkoľvek z nich už by výpočty nešli realizovať.

Stavy budeme ďalej (ak práve nebude uvedené inak) značiť malými písmenami s, s_t,zatiaľ čo stavové priestory veľkým S. Prirodzene teda bude $s, s_t \in S$.

1.3.2 Rozhodnutia a priestor rozhodnutí

Pointou dynamického programovania je výber správnych rozhodnutí v čase. Pojem rozhodnutia môže mať v rôznych modelových situáciách skutočne rôzne významy, ale všetky sú založené na tej istej podstate - na základe dostupných informácii vyberáme z dostupných možností tú optimálnu, kde pojem optimality je definovaný vopred (maximalizáciou či minimalizáciou danej účelovej funkcie). Rozhodnutie teda môže byť pre chodca riešiaceho problém najkratších ciest mestskými chodníkmi názov križovatky, do ktorej sa zo svojej pozície vydá, zatiaľ čo pre obchodníka na finančnom trhu, ktorý vo svojom portfóliu drží akcie, rozhodnutím bude akcie držať ďalej, predať ich, alebo naopak nakúpiť nové.

Rozhodnutia budeme značiť malými písmenami a, a_t, zatiaľ čo priestory rozhodnutí A, A_t. Je nutné podotknúť, že priestor rozhodnutí sa v závislosti na stave mení, je teda $A_t = A_t(s_t) \subset A$, kde A značíme zjednotenie $A = \bigcup_t A_t$.

Pri stochastických problémoch nie sme vždy schopní nájsť presnú postupnosť rozhodnutí, ktorá vedie k optimálnemu cieľu, pre všetky prvky stavového priestoru. Vo finančných aplikáciách to môže byť spôsobené napríklad enormnou veľkosťou stavového priestoru. Princípom dynamického programovania je hľadať politiku v podobe rozhodnutí ako reakcií na stav (a prípadné náhodné vplyvy), ktorá je v istom zmysle výhodná - v jednoduchých úlohách sme často schopní nájsť priamo tú optimálnu politiku; nie je to však vždy možné.

Definícia. Optimálna politika je pravidlo, resp. súbor pravidiel, pomocou ktorých je presne a jednoznačne určené rozhodnutie podľa informácii určených daným stavom. Matematicky môžeme politiku definovať ako funkciu P na stavovom priestore, ktorá pre daný stav vráti optimálne rozhodnutie, teda

$$P : S \to A;\ P(s_t) = a_t^* \in A_t(s_t).$$

Pre ilustráciu vezmime opäť obchodníka na finančnom trhu, ktorý sa snaží optimalizovať svoje portfólio. Bude tak hľadať optimálny počet akcií, ktorý si má na svojom účte držať. Nech momentálne vlastní s_t akcií, a označme a_t jeho rozhodnutie v danom čase ako množstvo akcií, ktoré predá, aby sa priblížil optimálnemu množstvu. Modelujme jeho správanie triviálnou politikou:

$$a_t = P(s_t) = \begin{cases} m & \text{ak } s_t > M \\ 0 & \text{inak} \end{cases}.$$

Inými slovami, bude predávať fixný počet akcií m, ak momentálne vlastní viac než fixný počet M. Úlohou obchodníka tak bude nájsť pomocou maximalizácie nejakej vybranej hodnotovej funkcie optimálne parametre (m, M), ktorými danú politiku (a ňou aj stav svojho portfólia) optimalizuje.

1.3.3 Zisková funkcia

Pre modelovanie dynamického programu je kľúčová vlastnosť, že pri vývoji systému existuje akási miera toho, ako dobre na tom v danej situácii sme. Je preto dôležité kvan-

tifikovať, ako sa táto miera mení, keď sa pohybujeme medzi stavmi a vykonávame rozhodnutia. Zisk, prípadne stratu vyplývajúcu z vykonávania rozhodnutí v daných stavoch opisujeme tzv. **ziskovou funkciou.**

Definícia. Ziskovou funkciou rozumieme súbor funkcií $C_t : S \times A \to M$, $t \in T$ (pre využitie optimalizačných techník prevažne modelujeme M ako číselnú množinu $M \subset \mathbb{R}$) opisujúcu zisk (odmenu, cenu, ...), ktorú získame, ak v stave $s_t \in S$ vykonáme rozhodnutie $a_t \in A_t(s_t)$. Túto hodnotu je zvykom značiť s indexom, t.j. $C_t(s_t, a_t)$, pre ozrejmenie závislosti na informáciách zo stavovej premennej s_t.

V ilustrácii nášho obchodníka teraz predpokladajme, že obchodník na burze miesto akcií nakupuje suroviny (napríklad diamanty), ktoré ďalej distribuuje svojim zákazníkom (klenotníctvam). Predpokladajme najprv, že jeho zákazníci majú týždenne v týždni t fixnú spotrebu R_t diamantov (dopyt), ktorú obchodníkovi oznámia vopred. Diamanty obchodník klenotníctvam predáva za cenu p_t^d, ktorá je vždy fixne daná trhovou cenou p_t, za ktorú nakupuje, a nejakou ziskovou maržou obchodníka (napr. 25%). Tiež má obchodník fixné náklady na prepravu tovaru nakúpeného na burze (palivo a plat vodiča), tie označme f. Ďalej označme

$s_t =$ množstvo diamantov, ktoré má obchodník dispozícii v týždni t (pred nákupom nových)
$a_t =$ množstvo diamantov, ktoré sa obchodník rozhodne nakúpiť v týždni t

Zisková funkcia obchodníka po nákupe a následnom predaji tovaru tak bude tvaru

$$C_t(s_t, a_t) = \begin{cases} p_t^d \min\{R_t, s_t + a_t\} - p_t a_t - f & \text{ak } a_t > 0 \\ p_t^d \min\{R_t, s_t\} & \text{ak } a_t = 0. \end{cases}$$

Podotknime, že tento problém by sa na prvý pohľad dal riešiť aj iným, možno prirodzenejším riešením. Nakoľko obchodník dopredu pozná hodnoty R_t, ktoré je schopný ďalej distribuovať a predať za zisk, dávalo by zmysel nakúpiť na burze len dané množstvo R_t diamantov, a mať tak rozhodovanie triviálne. V každom týždni by tak predal všetky nakúpené diamanty. Táto stratégia však nemusí byť optimálna práve kvôli fixným nákladom f: obchodníkovi sa totiž oplatí nakupovať naraz väčšie množstvá diamantov, a tým ušetriť na prevoze. V prípade, že by tieto náklady nemal, optimálnou stratégiou by nepochybne bolo nakupovať práve množstvá R_t, vďaka čomu by problém žiadne dynamické programovanie nevyžadoval.

1.3.4 Exogénne informácie

Na dynamiku systému môžu okrem vykonaných rozhodnutí vplývať aj rôzne vonkajšie (exogénne) faktory. V množstvách aplikácií, kde je modelovaný systém nedeterministický, je práve vhodná implementácia exogénnych procesov do riešenia kľúčová. Predstavme si cenu akcie na trhu, ktorá sa v čase vyvíja istým náhodným spôsobom. Najjednoduchší prípad takéhoto vývoja môžeme zachytiť napríklad zmenou v jednotlivých krokoch

$$p_{t+1} = p_t + \delta_{t+1}$$

kde δ_{t+1} je náhodná veličina, ktorej hodnotu pri vykonávaní rozhodnutia v čase t nepoznáme. Môžeme mať napríklad k dispozícii jej pravdepodobnostné rozdelenie, ale v rôznych aplikáciách, kde nie je ani toto k dispozícii, sa musíme uspokojiť s odhadmi.

Obecne značíme náhodné informácie W_t v tvare

$W_t =$ súbor exogénnych informácii, ktoré sú dostupné **po** vykonaní rozhodnutia v čase t

W_t tak môže byť rôznych tvarov, zvyčajne je však možné modelovať prichádzajúce informácie ako vektor, ktorého zložky sú náhodné veličiny opisujúce jednotlivé druhy nových informácii.

Predstavme si teda, že náš obchodník z predošlej podkapitoly podpíše zmluvu s novým klenotníctvom, ktoré je na trhu krátko a tak presne nedokáže určiť svoju týždennú spotrebu. Obchodníkovi tak poskytne len približné informácie o tom, aký dopyt môže od nich očakávať. Tento dopyt modelujme ako exogénnu premennú náhodnou veličinou G_{t+1}, kde indexom podotýkame, že táto hodnota nie je v týždni t známa, a obchodník tak musí svoj nákup vykonať bez jej presnej znalosti (k dispozícii ale môže mať napríklad informáciu o pravdepodobnostnom rozdelení). Už známa zisková funkcia z príkladu vyššie tak musí v tomto prípade zohľadniť fakt o neistote, a to v podobe strednej hodnoty:

$$C_t(s_t, a_t) = \mathbb{E}\big[p_t^d \min\{G_{t+1}, s_t + a_t\} - p_t a_t - \mathbb{I}_{\{a_t > 0\}}(a_t) f\big],$$

kde pre zjednodušenie zápisu značíme $\mathbb{I}_M$ charakteristickú funkciu javu M. Definovali sme tak niečo, čo by sa dalo nazvať „funkcia očakávaného zisku".

1.3.5 Funkcia prechodu

Vývoj systému je opísaný pohybom po stavovom priestore. Funkcia prechodu vystihuje, ako tento pohyb závisí na historickom vývoji (prostredníctvom aktuálneho stavu), na rozhodnutí, pre ktoré sa v danom kroku rozhodneme, a prípadne na exogénnych informáciách, ktoré systém ovplyvnia už po vykonaní tohto rozhodnutia. V rôznych modeloch je možné túto závislosť vyjadriť rôznymi spôsobmi, pre naše potreby vyslovme však matematicky čo najobecnejší predpis.

Definícia. Majme proces $(s_t)_{t \in T}$ definovaný na stavovom priestore S a s priestorom rozhodnutí A, prípadne s priestorom možných exogénnych informácii W. Funkciou prechodu nazveme funkciu $S^M : S \times A \times W \to S$, ktorej predpis jednoznačne určuje prechod medzi stavmi s_t a s_{t+1}v zmysle

$$s_{t+1} = S^M(s_t, a_t, W_t).$$

Predstavme si teda znovu problém nášho obchodníka s diamantmi. Už so známym značením môžeme v jeho prípade očakávať, že stav s_t, ktorým sme značili množstvo diamantov, ktoré má momentálne na sklade, sa bude vyvíjať v závislosti na tom, koľko diamantov a_t sa rozhodne nakúpiť, a aký dopyt R_t budú aktuálne jeho zákaznici mať. Teda

$$s_{t+1} = S^M(s_t, a_t, R_t) = \max\{s_t + a_t - R_t, 0\}.$$

V prípade nového klenotníctva s nedeterministickým dopytom môžeme písať

$$s_{t+1} = S^M(s_t, a_t, G_{t+1}) = \max\{s_t + a_t - G_{t+1}, 0\}$$

kde opäť G_{t+1} je náhodná veličina.

1.3.6 Hodnotová funkcia a rovnice optimality

Cieľom optimalizačných úloh je spravidla maximalizovať či minimalizovať istú funkciu. V optimalizačnej teórii túto funkciu väčšinou nazývame účelová funkcia. V dynamickom programovaní potom pojmom hodnotová funkcia myslíme funkciu, ktorá priradzuje jednotlivým prvkom stavového priestoru práve riešenie rovníc optimality (t.j. hodnoty objektívnej funkcie v bodoch optima). Pripomeňme, že úlohou dynamického programovania je nájsť optimálnu politiku, to jest funkciu, ktorá jednotlivým stavom priradí optimálne rozhodnutia. Stále sme ale nedefinovali, čo presne pre nás znamená pojem optimality, čo bude zámerom nasledujúcich pár odstavcov.

Výhodnosť rozhodnutia je definovaná, ako spomíname pri uvádzaní názvoslovia, odmenou (prípadne ziskom/stratou) vyplývajúcou z daného rozhodnutia. Tú opisuje zisková funkcia, a tak hľadanie optimálnej politiky môžeme pre deterministické úlohy v obecnom tvare formulovať ako riešenie maximalizačnej úlohy pre súčet ziskov v jednotlivých časových úsekoch, t.j.

$$\max_{\{a_t\}_{t=0}^{T}} \left(\sum_{t=0}^{N} \gamma^t C_t(s_t, a_t)) \right) = \max_{P \in \mathbb{P}} \left(\sum_{t=0}^{N} \gamma^t C_t(s_t, P_t(s_t)) \right) \tag{1.3.1}$$

kde názorne maximalizujeme podľa možných politík $P \in \mathbb{P}$. Konštanta $\gamma \in (0, 1]$ je tzv. **discount factor** modelujúci fakt, že výška odmeny/zisky/straty vyplývajúca z rozhodnutí môže byť závislá na tom, v akom časovom kroku (ako ďaleko v čase) sa aktuálne nachádzame. Dĺžka časového horizontu N je obecne z $\mathbb{N} \cup \{\infty\}$, a prítomnosť discount factoru je v príkladoch s nekonečným horizontom dôležitá pre konvergenciu sumy. Aj v modeloch s konečným horizontom si však discount factor nájde využitie: napríklad vo finančných aplikáciách, kde hodnotová funkcia často vystihuje istý obnos prostriedkov (napr. peňazí), ktoré vďaka vykonaniu rozhodnutia obdržíme, je vhodné túto skutočnosť zachytiť tým, že hodnota istého obnosu prostriedkov nadobudnutých v budúcom čase je pre nás z dnešného pohľadu nižšia, ako hodnota toho istého obnosu prostriedkov dnes. Naopak v prípadoch, kedy k tomuto „diskontovaniu" nedochádza, použijeme tvar s $\gamma = 1$.

Preformulujme úlohu 1.3.1 na tvar, v ktorom bude podstata úlohy viac zrejmá. Oproti hľadaniu optimálnej politiky ako celku (t.j. riešeniu celej úlohy naraz) pre nás bude iste jednoduchšie hľadať optimálne rozhodnutia postupne v daných stavoch, ktoré v časových úsekoch prechádzame. Ak sa nachádzame v stave $s_t \in S$ a vykonáme rozhodnutie $a_t \in A_t$, podľa funkcie prechodu sa dostaneme do stavu $s_{t+1} = S^M(s_t, a_t)$. Ak označíme hodnotu v našom stave $V_t(s_t)$, hodnota v novom stave bude $\gamma V_{t+1}(s_{t+1})$. Zároveň po vykonaní rozhodnutia získavame „odmenu" $C_t(s_t, a_t)$. Ak teda v stave s_t chceme vybrať optimálne rozhodnutie, vyberáme to, ktoré maximalizuje hodnotu $\gamma V_{t+1}(s_{t+1}) + C_t(s_t, a_t)$.

Dostávame tak tvar, ktorý vlastne hovorí, že každý dynamický program je možné modelovať ako rekurziu, ktorá dáva do súvislosti hodnotu v danom stave s hodnotou v

stave nasledujúcom. Pri deterministických modeloch, s už uvádzaným značením môžeme túto rekurziu obecne písať v tvare

$$V_t(s_t) = \max_{a_t \in A_t} \big(C_t(s_t, a_t) + \gamma V_{t+1}(s_{t+1})\big) = C_t(s_t, a_t^*) + \gamma V_{t+1}(s_{t+1}) \tag{1.3.2}$$

kde sme označili $s_{t+1} = S^M(s_t, a_t^*)$ už pre vykonané optimálne rozhodnutie (ak existuje)

$$a_t^*(s_t) = \arg\max_{a_t \in A_t} \big(C_t(s_t, a_t) + \gamma V_{t+1}(s_{t+1})\big).$$

Rovnica 1.3.2, s ktorou sa vo väčšine literúry stretávame pod názvom **Bellmanova rovnica**, resp. **rovnica optimality**, je kľúčovou myšlienkou dynamického programovania. Vyskytuje sa v rôznych tvaroch, pre naše potreby uvedieme ešte pár z nich.

Zorientujme sa v zápise, ktorý sme zaviedli. Maximum je počítané cez všetky možné rozhodnutia a_t prípustné v danom stave. Ďalej vidíme už známe značenie $C_t(s_t, a_t)$ pre ziskovú funkciu. V každom prvku stavového priestoru podľa vyššie uvedeného počítame hodnotu $V_t(s_t)$, čím definujeme funkciu $V : S \to \mathbb{R}$. Funkciu V nazývame hodnotová funkcia. Rozhodovací proces sa tak riadi práve hodnotami tejto funkcie.

Implicitne vo vyššie uvedených rovniciach sa skrýva fakt, že v kroku t potrebujeme na výpočet hodnotovej funkcie hodnotu V_{t+1}. V problémoch s konečným horizontom, ktorým sa v nasledujúcich odstavcoch budeme venovať, teda riešenie takéhoto dynamického programu funguje rekurzívne odzadu. Už na začiatku tak potrebujeme nutne poznať hodnoty na konci procesu, $V_N(s_N)$, a to pre všetky možné stavy s_N. Tieto hodnoty nazveme finálne hodnoty (terminal values, resp. terminal valuation). Ich špecifické tvary sú, samozrejme, v rôznych problémoch diametrálne odlišné. Charakter tohto postupu teda pracuje s predpokladom, že tieto hodnoty dokážeme z charakteru úlohy vyčítať, alebo ich získať iným spôsobom. Ak to nie je možné, používajú sa rôzne odhady, prípadne sa volia $V_N(s_N) = 0$, čím je taktiež zdôraznené, že tieto hodnoty nie sú pre charakter riešenia problému kľúčové. Úlohou DP je totiž nájsť optimálnu politiku pre rozhodovanie v blízkej budúcnosti, teda taktiku, ako robiť rozhodnutia $a_0, \ldots, a_t$ v nejakých časových úsekoch $t = 0, \ldots, T < N$.

Spomeňme ešte ďalšie formulácie vyššie uvedených rovníc. Problémy finančnej praxe sú zvyčajne formulované stochasticky. Vývoj dynamického systému ako je napríklad finančný trh, je ovplyvnený veľkým množstvom exogénnych faktorov, ktorých budúci dopad pri robení rozhodnutí v aktuálnom čase nepoznáme. Tieto stochastické exogénne premenné môžeme však podobne ako vyššie modelovať pomocou ich pravdepodobnostného rozdelenia.

Predstavme si teda opäť raz nášho obchodníka, ktorý obchoduje na burze s diamantmi. Ako v predošlej sekcii, vždy medzi časovými úsekmi nakúpi istý počet diamantov, aby pokryl dopyt svojich odberateľov. Tentokrát pre ilustráciu predpokladajme, že dodáva diamanty novým klenotníctvam, ktorých presný dopyt vopred nepozná. Označme tak dopyt D ako náhodnú veličinu, pre jednoduchosť s diskrétnym oborom hodnôt. V tomto prípade bol ale obchodník o niečo šikovnejší, a dobrým prieskumom trhu zistil aspoň približné pravdepodobnostné rozdelenie dopytov $(\mathbb{P}[D = d])_{d \in \mathbb{N}}$. Ak označíme s_t množstvo diamantov, ktoré aktuálne skladuje a pripomenieme si funkciu prechodu v tvare z predošlej sekcie

$$s_{t+1} = S^M(s_t, a_t, D_{t+1}) = \max\{s_t + a_t - D_{t+1}, 0\},$$

využitím pravdepodobnostného rozdelenia veličiny D resp. D_{t+1} dokáže obchodník spočítať aspoň pravdepodobnosť, že v nasledujúcom úseku bude mať na sklade k dispozícii istý počet diamantov. Presnejšie vie nájsť pravdepodobnosti

$$\mathbb{P}[s_{t+1} = s] = \begin{cases} \mathbb{P}[D = s_t + a_t - s] & \text{ak } 0 < s \leq s_t + a_t \\ \sum_{d=s_t+a_t}^{\infty} \mathbb{P}[D = d] & \text{ak } s = 0 \\ 0 & \text{ak } s > s_t + a_t \end{cases}$$

Posledná možnosť zrejme objasňuje fakt, že stav diamantov na sklade v ďalšom časovom úseku nemôže byť vyšší než množstvo súčet starého stavu s_t a nového nakúpeného množstva a_t (dopyt je vždy nezáporný).

Pripomienka 1. Viditeľne, vyššie uvedené pravdepodobnosti sú závislé na s_t aj a_t, čo dáva intuíciu značeniu

$$\mathbb{P}[s_{t+1} = s] = \mathbb{P}[s_{t+1}|s_t, a_t]$$

ako pravdepodobnosti, že skončíme v stave s_{t+1}, ak sa nachádzame v stave s_t a vykonáme rozhodnutie $a_t \in A_t(s_t)$. Toto značenie budeme aj napriek jeho matematickej nekorektnosti ďalej používať, k bližšej diskusii sa dostaneme v nasledujúcich odstavcoch.

Pre diskrétne náhodné veličiny W môžeme teda modifikovať deterministickú Bellmanovu rovnicu 1.3.2 na nedeterministický, tzv. štandardný tvar

$$V_t(s_t) = \max_{a_t \in A_t} \left(C_t(s_t, a_t) + \sum_{s \in S} \mathbb{P}[s_{t+1} = s|s_t, a_t] V_{t+1}(s) \right) \tag{1.3.3}$$

ekvivalentne môžeme písať

$$V_t(s_t) = \max_{a_t \in A_t} \left(C_t(s_t, a_t) + \mathbb{E}\left[V_{t+1}(s_{t+1}(s_t, a_t, W_{t+1})) \middle| s_t, a_t \right] \right) \tag{1.3.4}$$

$$= \max_{a_t \in A_t} \left(C_t(s_t, a_t) + \mathbb{E}\left[V_{t+1}(s_{t+1}) \middle| s_t, a_t \right] \right) \tag{1.3.5}$$

kde v druhom riadku značíme už implicitne ako funkčnú hodnotu $s_{t+1} = S^M(s_t, a_t, W_{t+1})$. V tomto tvare je vďaka strednej hodnote stochastická povaha problému viac zrejmá, je ale potrebné si matematicky ujasniť značenie. Zápis $s_{t+1} = S^M(s_t, a_t, W_{t+1})$ klasicky vypovedá o **funkčnej závislosti**, zatiaľ čo stredná hodnota podmienená s_t a a_t znázorňuje **závislosť pravdepodobnostného rozdelenia** náhodnej veličiny s_{t+1}(t.j. veličiny W_{t+1}) na týchto premenných. Rozhodnutie a_t síce nie je náhodnou veličinou, a teda podmieňovanie nie je matematicky korektné, tento zápis sa ale (nielen) v optimalizačných kruhoch ujal, a tak ho budeme používať v tvare vyššie. Upozorňujeme, že tento zápis chápeme v intuitívnom zmysle podľa Pripomienky 1.

Ďalej je bežný aj zápis s nepodmienenou strednou hodnotou

$$V_t(s_t) = \max_{a_t \in A_t} \left(C(s_t, a_t) + \mathbb{E}\left[V_{t+1}(s_{t+1}(s_t, a_t, W_{t+1})) \right] \right) \tag{1.3.6}$$

V tomto prípade podotýkame, že v tomto zápise sa implicitne skrýva skutočnosť, že **pravdepodobnostné rozdelenie** exogénnej premennej W_{t+1} (a teda rozdelenie s_{t+1}) **nezávisí** na stavovej premennej s_t ani na rozhodnutí a_t. Aj keď tento predpoklad je vo veľkej

skupine modelových úloh DP splnený, nie je to vždy tak. Sú k dispozícii príklady, kde rozdelenie W_{t+1} závisí len na s_t, len na a_t, alebo na oboch. V týchto prípadoch je samozrejme nutné počítať strednú hodnotu podmienenú tou správnou premennou (prípadne obomi), podľa zápisu 1.3.4. V poslednom prípade by sme tak chápali zápis strednej hodnoty v 1.3.4 ako podmienenú strednú hodnotu veličiny W_{t+1} za podmienky, že sa nachádzame v stave s_t a za predpokladu, že vykonáme rozhodnutie a_t.

Teória vyložená vyššie bola zatiaľ plne obecná. Pripomeňme ale, že algoritmy na riešenia sa spravidla fundamentálne líšia pre dynamické programy modelované v konečnom čase (tzv. finite horizon problems), a tie modelované v čase nekonečnom (infinite horizon). Spomeňme obe triedy problémov v krátkych podkapitolách.

1.4 Problémy s konečným horizontom

Modelové úlohy tohoto typu sa dajú spravidla zaradiť do troch kategórii:

- problémy, v ktorých sa snažíme riešiť istý problém vo fixnom vopred určenom časovom úseku. Takto by sme modelovali napríklad prípad nášho obchodníka na trhu, ktorý obchoduje len istý vopred známy počet počet $N \in \mathbb{N}$ časových úsekov.

- problémy, v ktorých je vopred známy presne určený cieľ, ktorý je nutné dosiahnuť. V aplikáciach tohto typu tak nemusí byť známy konečný časový úsek, v ktorom sa vývoj systému odohráva, ale je k dispozícii istý cieľ, ktorého dosiahnutím v (dopredu neznámom) konečnom čase vývoj systému skončí. Týmto spôsobom môžeme modelovať napríklad problémy gamblingu, pri ktorých hráč stávkuje peniaze až dovtedy, kým všetky neprehrá.

- problémy nekonečného horizontu modelované technikami konečného horizontu. Pri týchto problémoch sa sústredime sa na len hľadanie optimálnej politiky pre rozhodovanie v terajšom okamihu, prípadne v blízkej budúcnosti. Takto by sme dokázali modelovať problém nášho obchodníka, ktorý na trhu obchoduje ľubovoľne dlho. Jeho rozhodovanie pri nákupoch akcií či surovín by tak bolo modelované vždy len vo fixnom období dĺžky $T < \infty$, ktorého riešením dostaneme optimálne rozhodnutia, ktoré je potrebné robiť dnes, zajtra, či v inej blízkej budúcnosti.

Princíp riešenia týchto problémov je vo svojej podstate jednoduchý. Ako sme už v kapitole 1.3 spomínali, hodnotová funkcia sa počíta rekurzívne odzadu. Potrebujeme teda predpokladať, že hodnoty v poslednom úseku máme na začiatku riešenia k dispozícii ako dáta (terminal valuation). Obecný postup by tak mohol vyzerať nasledovne:

Algoritmus 1.2 Obecný princíp riešenia spätného dynamického programu v konečnom horizonte

1. Inicializujem hodnoty v poslednom kroku (terminal valuation):

$$\forall s_N \in S \;\; \text{uložím } V_N(s_N);$$

uložím pre začiatok $t = N - 1$.

2. Pre všetky možné stavy $s_t \in S$ riešim rovnice optimality v príslušnom tvare (pre názornosť voľba 1.3.2)

$$V_t(s_t) = \max_{a_t \in A_t(s_t)} \big(C_t(s_t, a_t) + V_{t+1}(s_{t+1})\big)$$

a volím optimálne rozhodnutie ako argument maxima.

3. Ak $t > 0$, uložím $t \leftarrow t - 1$ a pokračujem krokom 2. Inak končím.

1.5 Problémy s nekonečným horizontom

Úlohy modelujeme s nekonečným horizontom prevažne vtedy, ak kľúčové parametre úlohy (napr. zisková funkcia, funkcia prechodu, či exogénne informácie) sa nevyvíjajú v časových úsekoch. Môžeme tak predpokladať nezávislosť parametrov systému na časových krokoch, čím definujeme tzv. stacionárny proces. Ak označíme

$$\begin{aligned} V(s) &= \lim_{t\to\infty} V(s_t) = \max_{a\in A} \big(C(s,a) + \gamma \sum_{s'\in S} \mathbb{P}[s'|s,a] V(s')\big) \qquad (1.5.1)\\ &= \max_{a\in A} \big(C(s,a) + \gamma \mathbb{E}[V(s')]\big), \end{aligned}$$

kde predpokladáme, že limita existuje (zariadiť tento predpoklad v praxi nie je vždy jednoduché, technické detaily ale ponecháme), dostávame tzv. **stacionárnu rovnicu optimality**. Je možné ukázať, napr. [Powell], Kap. 3.10., že takto definovaná funkcia rieši už nám známy optimalizačný problém s nekonečným horizontom

$$\max_{P\in\mathbb{P}} \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t C_t\big(s_t, P_t(s_t)\big)\Big],$$

kde značíme rozhodovaciu politiku $P \in \mathbb{P}$ (priestor všetkých možných politík).

Na tejto formulácii stoja hneď niekoľké algoritmy riešenia problémov s nekonečným horizontom. Za všetky vymenujme aspoň dva hojne používané postupy: iteráciu hodnôt (value iteration) a iteráciu politík (policy iteration). Tieto algoritmy patria medzi najpoužívanejšie postupy dynamického programovania vôbec. Pre analógiu s Algoritmom 1.2. pre problémy s konečným horizontom, opíšme v krátkosti princíp fungovanie iterácie hodnôt. Algoritmus je skutočne veľmi podobný spätnej rekurzii z Kap. 1.4., miesto parametrizácie priestotu krokmi t však využíva postupnú iteráciu pre $n = 0, 1, ...$ až pokým nie je splnené

isté kritérium konvergencie. Taktiež v prípade nekonečného horizontu zrejme nemôže byť reč o terminal valuation, ako sme ju spomínali v 1.3.6. Zhrnutie poskytne Algoritmus 1.3.:

Algoritmus 1.3 Obecný algoritmus iterácie hodnôt

1. Pre všetky $s \in S$ inicializujem počiatočné hodnoty $v_0(s) = 0$.

   Zvolím parameter tolerancie $\epsilon > 0$; uložím $n = 1$.

2. Pre všetky $s \in S$ počítam rovnice optimality v tvare (1.5.1.), teda

$$v_n(s) = \max_{a \in A} \left(C(s,a) + \gamma \sum_{s' \in S} \mathbb{P}[s'|s,a] v(s') \right) \tag{1.5.2}$$

3. Ak pre nejaké $s \in S$ platí $|v_n(s) - v_{n-1}(s)| > c\epsilon$, kde $c = c(\gamma)$ je vhodne zvolená* konštanta, kladiem $n \leftarrow n + 1$ a pokračujem krokom 2; inak uložím do výslednej politiky rozhodnutia a_s^*, ktoré v daných stavoch riešia 1.5.2, a výpočet končím.

   *[Powell], str. 69, uvádza napíklad $c = \frac{\epsilon(1-\gamma)}{2\gamma}$.

Nakoľko je algoritmus iteratívny a opiera sa o postupný výpočet hodnôt pre rastúce n, je na mieste diskusia o existencii limity a teda konvergencii postupných aproximácii riešenia $(v_n)_{n \in \mathbb{N}}$. Táto diskusia vo svojej podstate nie je technicky náročná. Po ozrejmení faktu, že operátory $\max(\cdot)$ resp $\min(\cdot)$ z 1.5.2 sú na (úplnom) priestore hodnotových funkcií kontrahujúce zobrazenia, sa jedná o aplikáciu Banachovej vety o pevnom bode, ktorá zaručuje konvergenciu a existenciu jednoznačného riešenia. Nakoľko však cieľom práce nie je dôkladný matematický rozbor týchto metód, ponechávame technické detaily k nahliadnutiu napríklad v [Powell], Kap. 3.10.

1.6 Približné dynamické programovanie

V mnohých prípadoch, medzi ktoré spravidla patria aj finančné problémy, nie je vždy možné exaktne počítať hodnotú funkciu pre všetky možné stavy a rozhodnutia. Už spomínané prekliatia dimenzionality, ktoré tomu bránia, sa obecne dajú zaradiť do troch skupín týkajúcich sa rôznych parametrov úlohy.

1. **Stavový priestor**. Samotná stavová premenná môže byť výpočetne neúnosná. Pre predstavu modelujme problém pomocou stavovej premennej $s_t = (s_{t1}, \ldots, s_{tN}\}$, kde každá hodnota s_{t1} môže mať hodnoty z množiny M (predpokladajme spočítateľnú). Celý stavový priestor tak bude rozmerov $|M|^N$, čo je veľmi rýchlo rastúca hodnota (už napr. $2^{100} \approx 10^{30}$ je výpočtovo neúnosné číslo).

2. **Náhodný vývoj.** Možnosti náhodných exogénnych vplyvov sú v dynamickom programovaní všadeprítomné. Nakoľko ich priebeh vopred nepoznáme, v modelovaní bežnými postupmi DP je potrebné rátať so všetkými možnosťami. Podobne ako v

prvom prípade, obecne tieto vplyvy modelujeme vektormi $W_t = (W_{t1}, \ldots, W_{tL})$, kde taktiež rozmer priestoru všetkých možností exponenciálne rastie.

3. **Priestor rozhodnutí.** Analogicky k prvým dvom možnostiam, rozhodnutia sú obecne vektory $a_t = (a_{t1}, \ldots, a_{tI})$. Pri modelovaní napr. priradzovania vodičov autobusu k linkám, by sme mohli rozhodnutia modelovať ako priradenie I vodičov k rôznym linkám $1, ..., I$. Priestor možných rozhodnutí $A = \cup_t A_t$ už pri napr. I =141 linkách v Prahe a rádovo niekoľko tisíc vodičoch DPP je nepredstaviteľné číslo.

Pripomeňme si situáciu DP pre diskrétny prípad, v ktorom sme riešili rekurzívne odzadu rovnice optimality

$$\begin{aligned} V_t(s_t) &= \max_{a_t \in A_t} \Big(C_t(s_t, a_t) + \gamma \mathbb{E}\big[V_{t+1}(s_{t+1}) | s_t\big] \Big) \\ &= \max_{a_t \in A_t} \Big(C_t(s_t, a_t) + \gamma \sum_{s \in S} \mathbb{P}\big[s | s_t, a_t\big] V_{t+1}(s) \Big) \end{aligned}$$

pre všetky možné stavy $s_t \in S$. Tu vidíme prvý problém s prekliatím dimenzionality - tento postup nie je aplikovateľný na rozsiahle stavové priestory, kde výpočetné možnosti nedovoľujú prechádzať v každom kroku všetky stavy. Nebudeme tak schopní presne vypočítať hodnoty $V_t(s_t)$ pre všetky stavy a pre robenie rozhodnutí si budeme musieť vystačiť s aproximáciou hodnotovej funkcie.

Postup je vo svojej podstate podobný už uvedeným postupom klasického DP. Jedinou zmenou je tentokrát fakt, že postupovať budeme odpredu - spätný postup by vzhľadom na charakter úlohy (neriešiteľnosť Bellmanovej rovnice vo vyššie uvedenom tvare) pochopiteľne nedával zmysel. Hodnotovú funkciu budeme v jednotlivých stavoch iteratívne aproximovať hodnotami $\overline{V}_t^n(s_t)$, kde $n \in \mathbb{N}$ značí index iterácie (nezamienať s mocninou). Na začiatku žiadny odhad ešte nemáme, a teda volíme počiatočné odhady $\overline{V}_t^0(s_t)$ podľa nejakých informácii z charakteru úlohy, prípadne ak nemáme nič k dispozícii, volíme $\overline{V}_t^0(s_t) = 0$ pre všetky stavy a časové kroky.

Ďalej postupujeme analogicky. V iterácii n máme k dispozícii odhady $\overline{V}_t^{n-1}$, a to pre všetky $t \in T$. Vyberme ďalej ľubovoľne počiatočný stav s_0 (v čase 0). Nakoľko poznáme odhady $\overline{V}_t^{n-1}(s)$ pre všetky s, môžeme ich využiť k nájdeniu rozhodnutia

$$a_0 = \arg\max_{a \in A_0} \Big(C_0(s_0, a) + \gamma \sum_{s \in S} \mathbb{P}[s | s_0, a] \overline{V}_1(s) \Big),$$

kde predpokladáme, že jednokroková matica prechodu $\mathbf{P}_k = \big(\mathbb{P}[s|s_k, a]\big)_{s \in S, a \in A}$ je nám vždy v danom čase k známa (alebo ju dokážeme aproximovať).

Po vykonaní rozhodnutia a_0 predpokladáme príchod exogénnej informácie W_1, ktorú vopred nepoznáme. Túto náhodnú veličinu budeme teda aproximovať. Je množstvo spôsobov, ktoré k tejto aproximácii môžeme zvoliť (napr. generovanie hodnôt pri známom rozdelení, Monte Carlo sampling, atď.). Pre ADP je kľúčové, že tieto aproximácie exogénnych informácii (tzv. sample paths) dokážeme v jednotlivých iteráciách resp. časových krokoch spoľahlivo generovať. Nakoľko pri ADP neprechádzame všetky možné stavy, je

potrebné zariadiť, aby sa postupným iterovaním modelovaný systém dostal do čo najviac z nich, aby bolo možné aproximovať ich hodnoty. To je jednou z úloh sample paths. Opäť existuje niekoľko možností, ako dosiahnuť čo najlepšie aproximácie optimalizáciou postupov už pri samotnom generovaní náhodných informácii.

Pomocou už spomenutého náhodne generovaného pozorovania nájdeme stav, do ktorého sa vykonaním optimálneho rozhodnutia dostaneme. Klasickým značením teda dostaneme nasledujúci stav

$$s_1 = S^M(s_0, a_0, W_1).$$

Máme už tak k dispozícii všetky informácie na hľadanie ďalšieho rozhodnutia: v stave s_1 pomocou známych hodnôt $\overline{V}_2(s)$ z predošlej iterácie (tie z predpokladu poznáme $\forall t \in T$ a $\forall s \in S$) môžeme nájsť rozhodnutie

$$a_1 = \arg\max_{a \in A_1} \left(C_1(s_1, a) + \gamma \sum_{s \in S} \mathbb{P}_1[s|s_1, a] \overline{V}_2(s) \right).$$

Následne opäť generujeme náhodnú vzorku exogénnych informácii W_2, a pokračujeme ďalej.

Jedným chodom takéhoto algoritmu využitím jednej vzorky náhodných informácii pochopiteľne nedostaneme žiadne rozumné výsledky. Preto je tento postup iteratívny: opiera sa o postupnú aproximáciu po veľkom množstve iterácii, vždy s rôznymi vzorkami náhodných javov. Približné dynamické programovanie tak môžeme považovať za istú kombináciu optimalizácie a náhodného samplingu (napr. metódou Monte Carlo). Možné aplikácie sú teda všade tam, kde máme možnosť rozumne generovať vzorky náhodných informácii, a zároveň charakter problému povoľuje opis pomocou metodiky dynamického programovania (aj keď presné riešenie napr. spätným iterovaním nie je výpočtovo možné). Obecne postup zhrnie Algoritmus 1.4.

Algoritmus 1.4 Obecný dopredný ADP algoritmus

1. Inicializujem hodnoty $\overline{V}_t^0(s_t)$ pre všetky $s_t \in S$, $t \in T$ (napríklad nulami). Zvolím počiatočný stav s_0^1; uložím $n = 1$ (iteračný index).

2. Vygenerujem vzorku náhodných informácii $\omega^n = (\omega_1^n, \ldots, \omega_T^n)$.

3. Pre všetky $t = 0, ..., T$ aproximujem hodnotu v práve dosiahnutom stave

$$\overline{V}_t^n(s_t^n) = \max_{a \in A_t^n} \left(C_t(s_t^n, a) + \gamma \sum_{s \in S} \mathbb{P}[s|s_t^n, a] \overline{V}_{t+1}^{n-1}(s) \right);$$

   a v ostatných stavoch ponechávam $\overline{V}_t^n = \overline{V}_t^{n-1}$. Volím približné optimálne rozhodnutie a_t^n ako argument maxima vyššie.
   Pomocou zvoleného rozhodnutia a vzorky náhodnej premennej určím $s_{t+1}^n = S^M(s_t^n, a_t^n, \omega_t^n)$.

4. Ak $n = N$, výpočet končím; inak pokračujem v iterovaní krokom 2 pre $n \leftarrow n + 1$.

Všimnime si, že v algoritme prechádzame stavy (a teda aproximujeme hodnoty v nich) istým znáhodneným spôsobom pomocou generovania vzoriek zo známeho rozdelenia. Tento fakt je pochopiteľne pre ADP kľúčový, nakoľko sa používa na riešenie problémov, v ktorých nie je možné prechádzať všetky prvky stavového priestoru. Zároveň tento fakt ale spôsobuje, že vopred nemáme istotu, že stavový priestor prechádzame „rozumným spôsobom". Bežne sa totiž môže stať, že sa do veľkého množstva (často veľmi výhodných) stavov vôbec nedostaneme a ich potenciál tak algoritmus vôbec nevyužije. Taktiež stavy kvôli aproximatívnemu postupu môže algoritmus vyhodnotiť ako neoptimálne, aj keď v skutočnosti sú veľmi výhodné. Tento problém sa z podstaty postupov ADP nedá vyriešiť v plnej obecnosti. Prvý krok k jeho eliminácii sa v praxi robí použitím tzv. exploration postupov - napríklad ϵ-greedy politiky, v ktorej v každom časovom úseku s istou malou pravdepodobnosťou ϵ systém prejde do náhodne zvoleného stavu, namiesto stavu, do ktorého by ho zaviedlo algoritmom zvolené optimálne rozhodnutie. Takto sa docieli postupné dosiahnutie čo najväčšieho množstva stavov (v limite všetkých stavov), aby sa minimalizoval vplyv napríklad spomínaných problémov, kedy sa hodnota potenciálne veľmi výhodných stavov vôbec neaktualizuje, pretože systém sa do daných stavov nedostáva. Rozsiahlu diskusiu o týchto postupoch poskytuje [Powell] v Kap. 11 a 12.

Týmto oboznámením sa so základnými algoritmami ukončíme úvod do teórie (približného) dynamického programovania. Podotknime, že tento len veľmi stručný úvod nie je zďaleka vyčerpávajúci a čitateľa usmerňujeme k literatúre [Bellman, Powell, Puterman], ale aj mnohým ďalším publikáciam tohto oboru, kde je možné sa o technikách dynamického programovania dozvedieť mnohonásobne viac.

V ďalších kapitolách (A)DP aplikujeme na finančné problémy.

Kapitola 2

Rozhodovacie stratégie na trhu s 1 akciou

Hľadanie optimálnej politiky na finančnom trhu môže mať rôzne podoby. V tejto kapitole si predstavíme a implementujeme niekoľko investičných stratégii na zjednodušenom modeli trhu a zanalyzujeme ich profitabilitu na vopred známych historických dátach. V prvých dvoch sekciách pre bližšiu ilustráciu princípov využijeme framework dynamického programovania, ako bol opísaný v Kapitole 1, na riešenie dvoch toy príkladov na hľadanie optimálnej stratégie pre investovanie pri známych resp. znáhodnených cenách akcie; v tretej sekcii k obchodovaniu so známymi cenami predstavíme tri indikátory technickej analýzy.

2.1 Obchodovanie s fixnými cenami

Majme model trhu, na ktorom je možné obchodovať fixný počet $N \in \mathbb{N}$ časových úsekov (napr. ticks na burze). Na tomto trhu je k dispozícii k nákupu len 1 akcia, ktorú môžeme z trhu kupovať a predávať naspäť na trh. Každý časový úsek $t \in \{1, ..., N\}$ je možné vykonať len 1 transakciu, t.j. nákup alebo predaj, pričom celkovo máme k dispozícii fixný počet $T \in \mathbb{N}$ transakcií (reálna situácia; napr. obmedzenie pochádzajúce od brokera). Ďalej predpokladajme, že 1 transakcia znamená nákup a následný predaj, teda pri nákupe sa transakcia len „otvorí" a spotrebovaná je až pri následnom predaji akcie (uzavretí pozície). Transakcie teda spotrebúva iba predaj akcie.

Problém. Predpokladajme, že časový rad cien akcie v jednotlivých úsekoch je vopred známy, ozn. $c_1, ..., c_N$. Tento predpoklad je zrejme nereálny, príklad ale využijeme na ilustráciu už predstavených konceptov. Úlohou bude nájsť optimálnu stratégiu pre obchodovanie.

Riešenie. Neprekvapí, že k riešeniu použijeme spätný dynamický program. Pripomeňme, že optimálnou stratégiou rozumieme tú, ktorá maximalizuje profit. Než začneme, potrebujeme preložiť zadanie do reči dynamického programovania. Modelujme situáciu na trhu nasledovným spôsobom:

1. **Stavový priestor.** V každom časovom úseku budem zrejme potrebovať informáciu, čo práve mám na účte. Inými slovami: vlastním akciu, alebo nie. To ale nie je postačujúce; pre robenie rozhodnutí musím totiž vedieť aj to, či ešte môžem v ďalších úsekoch obchodovať, alebo nie. Ak totiž mám už vyčerpané transakcie, ďalšie obchodovanie nie je možné. Stavová premenná tak musí obsahovať aj informáciu o počte vykonaných transakcií (prípadne o počte voľných - doposiaľ nevyužitých - transakcií). Dostávame tak stavový priestor so stavovými premennými

$$S = \{0, 1\} \times \{0, ..., T\}$$

$$\forall t \in \{1, ..., N\} \; s_t = (m_t, k_t) \in S$$

kde hodnoty $m_t \in \{0, 1\}$ budú vyjadrovať počet akcií, ktoré aktuálne vlastním, a hodnoty $k_t \in \{0, ..., T\}$ počet už využitých (uzavretých) transakcií.

2. **Rozhodnutia a priestor rozhodnutí.** Rozhodnutia budú z podstaty problému riešiacou stratégiou. Budú zrejme označovať, čo v danom úseku mám urobiť - kúpiť alebo predať akciu, prípadne nerobiť nič. Je ale potrebné určiť priestory rozhodnutí, nakoľko očividne nie všetky rozhodnutia sú prístupné v každom stave. Ak akciu vlastním, môžem ju predať ($a_t = -1$), ak naopak nevlastním, môžem ju kúpiť ($a_t = 1$); v každom prípade však môžem nerobiť nič ($a_t = 0$). Označíme tak

$$a_t \in \{1, -1, 0\} \text{ ... akciu kúpim/akciu predám/nerobím nič}$$

$$A_t(s_t) = \begin{cases} \{0, 1\} & m_t = 0 \;\&\; k_t < T \\ \{0, -1\} & m_t = 1 \;\&\; k_t < T \\ \{0\} & k_t = T \end{cases}$$

3. **Zisková funkcia.** Zisk resp. prípadná strata bude určená tým, aké rozhodnutie v danom úseku vykonám. Podľa toho buď zinkasujem cenu akcie, ktorú predám, alebo zaplatím cenu akcie, ktorú kupujem. Bude teda

$$C_t(s_t, a_t) = -a_t c_t$$

všímame si, že v tomto prípade stavová premenná nie je explicitne nutná k vyjadreniu C_t. Vidíme ale implicitnú závislosť v $a_t = a_t(s_t)$.

4. **Funkcia prechodu.** Pri prechode zrejme zaváži len to, aké rozhodnutie v danom stave vykonám. Exogénne informácie totiž v probléme sú prítomné len v podobe cien, tie ale podľa zadania poznáme vopred a teda nemáme náhodné prvky. Nezabúdajme len na to, že pri predaji sa stavová premenná musí aktualizovať o fakt, že sme spotrebovali transakciu. Spolu teda máme

$$s_{t+1}(s_t, a_t) = \left(m_t + a_t, k_t + \mathbb{I}_{\{a_t = -1\}}\right)$$

kde pre zjednodušenie zápisu značíme charakteristickú funkciu javu

$$\mathbb{I}_{\{M\}} = \begin{cases} 1 & \text{ak logický výraz } M \text{ platí} \\ 0 & \text{inak} \end{cases}.$$

Jej tvar vystihuje fakt, že pri nákupe sa transakcia nespotrebuje, zatiaľ čo pri predaji už áno.

5. **Hodnotová funkcia.** Pri tomto probléme nám zrejme stačia deterministické rovnice optimality. Ich tvar 1.3.2 je nám známy, zamyslime sa teda len nad finálnym hodnotením. Tu si musíme pripomenúť, že hľadáme optimálnu stratégiu. Zrejme sa v konečnom stave v úseku n môžeme stretnúť len s dvomi možnosťami - buď na konci akciu vlastníme, alebo nie. V prvom prípade nemôže byť optimálnym rozhodnutím si ju nechať - akcia pre nás po skončení obchodovania už nemá hodnotu, pretože ju už viac nedokážeme predať. Optimálne teda musí byť túto akciu predať v poslednom kroku. Podobne v druhom prípade: ak akciu v poslednom kroku nevlastníme, nie je racionálne ju kupovať - už totiž nedostaneme príležitosť ju predať. Máme tak finálne hodnotenie
$$V_N(s_N) = \begin{cases} 0 & m_N = 0 \\ c_n & m_N = 1 \end{cases}$$
Fakt, že sme v tomto výpočte nikde nepoužili počet voľných transakcií, môže vyzerať podozrivo. Uvedomme si ale, že vyplýva z podstaty našich predpokladov o tom, kedy transakciu otvárame a uzatvárame: ak v poslednom dni akciu vlastním ($m_N = 1$), mám danú transakciu otvorenú a teda ju stále môžem zatvoriť (nutne tak $k_N < T$,a akciu môžem predať za cenu c_N). Každú transakciu totiž považujeme za vyčerpanú až po predaji.

Dá sa povedať, že týmto rozborom je problém vyriešený. Jeho riešenie - optimálna politika, to jest tabuľka optimálnych rozhodnutí $A = \big(a_t(s_t)\big)_{s_t \in S}$, ktoré vykonávať v jednotlivých časových úsekoch - vyplynie rekurzívnym riešením rovníc optimality v spomínanom tvare 1.3.2:
$$V_t(s_t) = \max_{a_t \in A_t} \big(C_t(s_t, a_t) + V_{t+1}(s_{t+1})\big)$$
resp.
$$a_t^*(s_t) = \arg\max_{a_t \in A_t} \big(C_t(s_t, a_t) + V_{t+1}(s_{t+1})\big),$$
kde už všetky objekty máme detailne definované vyššie, a tak môžeme dosadiť do finálnych tvarov
$$V_t(s_t) = \max_{a_t \in A_t} \Big(- a_t c_t + V_{t+1}\big((s_t + a_t, k_t + \mathbb{I}_{\{a_t = -1\}})\big)\Big)$$
$$a_t^*(s_t) = \arg\max_{a_t \in A_t} \Big(- a_t c_t + V_{t+1}\big((s_t + a_t, k_t + \mathbb{I}_{\{a_t = -1\}})\big)\Big)$$

Dosadením hodnôt cien $c_1, ..., c_N$ tak môžeme získať numerické riešenia explicitných prípadov. Tieto predvedieme v sekcii 2.4.

2.2 Obchodovanie so znáhodnenými cenami

V prvom toy príklade sme si situáciu použitím vopred známych cien skutočne uľahčili. Na obecnom trhu ceny samozrejme vopred nepoznáme, a pri modelovaní, sledujúc trh ako

dynamický proces ovplyvňovaný množstvom exogénnych faktorov, vývoj ceny akcií vo väčšine prípadov považujeme za náhodný proces. Urobme prvý krok k modelovaniu trhu pomocou znáhodnených cien so známym pravdepodobnostným rozdelením. Ponechajme si model trhu s jednou akciou a transakciami presne z predošlej sekcie, a pozmeníme len zadanie samotného problému.

Problém. Predpokladajme, že ceny nie sú vopred známe. Poznáme ale fixnú počiatočnú cenu c_0 a fakt, že v každom kroku sa táto cena vyvinie istým spôsobom - bude k nej so známou pravdepodobnosťou pripočítaná/odpočítaná istá známa hodnota. Môžeme teda v každom kroku modelovať cenu ako

$$c_{t+1} = c_t + D_{t+1}$$

kde D_{t+1} je náhodná veličina so známym rozdelením

$$\mathbb{P}[D_{t+1} = \alpha_t] = p_t^+;\ \mathbb{P}[D_{t+1} = \beta_t] = p_t^-.$$

Zopakujme, že pre všetky $t \in \{1, ..., n\}$ sú hodnoty α_t, β_t známe konštanty, podobne ako pravdepodobnosti $p_t^-, p_t^+ = 1 - p_t^- \in (0, 1)$. Tiež budú vždy $\alpha_t > 0, \beta_t < 0$. Inými slovami, hovoríme, že cena sa v každom kroku s istými pravdepodobnosťami posunie o istú známu hodnotu nahor alebo nadol.

Našou úlohou je opäť nájsť optimálnu stratégiu pre obchodovanie.

Riešenie. Neprekvapí, že niektoré parametre tohto programu budú rovnaké ako v predošlom príklade - stratégia pre obchodovanie bude stále len výber istých rozhodnutí, ktoré predstavujú nákup a predaj. Uveďme parametre modelu:

1. **Stavový priestor.** Nepochybne budeme potrebovať minimálne stavové premenné z predošlého príkladu. Teraz ale vopred nepoznáme ceny akcie, a tak kvôli výpočtom ziskovej funkcie bude potrebné si jednotlivé ceny postupne pamätať. K usporiadanej dvojici (m_t, k_t) musíme pridať aktuálnu cenu c_t, a teda máme stavový priestor

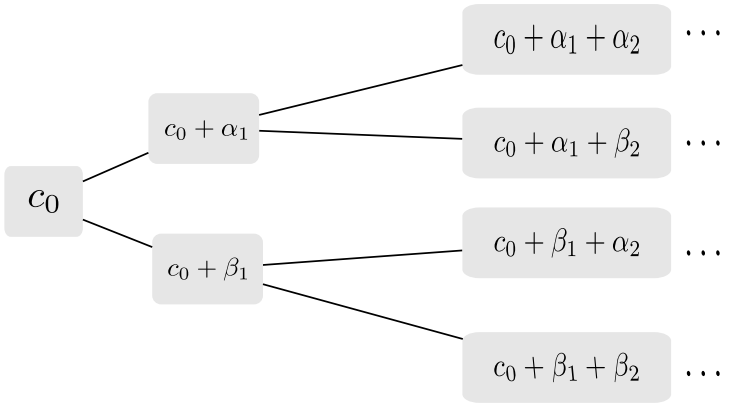
$$S = \{0, 1\} \times \{0, ..., T\} \times P$$

$$\forall t\ s_t = (m_t, k_t, c_t)$$

kde P je množina všetkých možných cien. Z obrázku 2.2.1 je viditeľné, že v každom úseku pribudnú vždy dve možnosti novej ceny. Stavový priestor obsahujúci všetky možnosti musí obsiahnuť ľubovoľnú kombináciu, takže množina P spomínaná vyššie by mohla byť reprezentovaná napríklad tvarom

$$P = c_0 + \sum_{i=1}^{N} (r_i\alpha_i + s_i\beta_i) \text{ kde } r_i, s_i \in \{0, 1\} \text{ pre } i \in \{1, ..., N\},$$

kde ceny v úsekoch $t < N$ získame voľbou $r_i = s_i = 0\ \forall i > t$. Kardinalita takejto množiny je pri neprekrývaní cien (nie nutne reálny prípad) rovná 2^N, kde N je počet časových úsekov, v ktorých sa cena vyvíja. V tomto počte časových úsekov môžeme



Obr. 2.2.1: Možný vývoj ceny akcie v prvých dvoch časových úsekoch

vidieť napríklad ticky pri vývoji cien na burze, kde sa bežne stretávame napríklad s frekvenciou jedného ticku za 5-10 sekúnd. Ak by sme teda chceli modelovať 1 osemhodinový pracovný deň obchodovania na burze, naša P by bola kardinality minimálne 2^{2880}. To je samozrejme číslo nepredstaviteľnej veľkosti. Stretávame sa tak prvýkrát s praktickým príkladom Bellmanovho „prekliatia dimenzionality" - takýto stavový priestor nikdy nedokážeme prejsť celý. Už teraz ale vidíme, že implementácia algoritmu si bude musieť s týmto problémom poradiť. V tomto prípade to nevyzerá až na tak veľký problém - pri vývoji ceny môžeme totiž v každom kroku zahadzovať veľké časti stavového priestoru - ak napríklad cena v prvom dni porastie, teda spadne do vetvy s $c_0 + \alpha_1$, vetvu s $c_0 + \beta_1$ už nikdy nedosiahneme, a žiadne stavy z nej tak vyčísľovať nebude potrebné. Takto dokážeme stavový priestor skresať na tvar, ktorý už je implementačne zvládnuteľný.

2. **Rozhodnutia a priestor rozhodnutí.** Aj v tomto prípade budeme robiť rozhodnutia pri nákupe/predaji 1 akcie. Nedôjde k žiadnej zmene oproti problému z predošlej sekcie, a tak len spomeňme finálne tvary:

$$\forall t \; a_t \in A_t(s_t) = \begin{cases} \{0, 1\} & m_t = 0 \;\&\; k_t < T \\ \{0, -1\} & m_t = 1 \;\&\; k_t < T \\ \{0\} & k_t = T \end{cases}$$

3. **Zisková funkcia.** Ani tu nedôjde k zmene. Zisk či cena vyplývajúca z daného rozhodnutia bude opäť len cena akcie s príslušným znamienkom. Pripomeňme len, že práve tento výpočet je dôvod, prečo si potrebujeme pamätať cenu ako stavovú premennú.

$$C_t(s_t, a_t) = -a_t c_t \text{ kde } c_t \in P$$

4. **Funkcia prechodu.** Prechod medzi stavmi bude tentokrát náhodnou veličinou. Zrejme už nebude závislý len na stave, v ktorom sa nachádzame a na rozhodnutí, ktoré vykonáme, ale oproti problému z predošlej sekcie už bude ovplyvňovaný aj

náhodnými faktormi (exogennými informáciami) v podobe zmeny cien. Podľa už známeho značenia budeme mať

$$s_{t+1} = S^M(s_t, a_t, D_{t+1}) = \big(m_t + a_t, k_t + \mathbb{I}_{\{a_t=-1\}}, c_t + D_{t+1}\big)$$

kde D_{t+1} je náhodná veličina s rozdelením ako vyššie. Pre cenovú zložku stavovej premmenej môžeme ekvivalentne písať aj

$$\mathbb{P}[c_{t+1} = c|c_t] = \begin{cases} p_t^+ & c = c_t + \alpha_t \\ p_t^- & c = c_t + \beta_t \\ 0 & \text{inak} \end{cases}$$

Viditeľne v tomto prípade nie je rozdelenie s_{t+1} závislé na rozhodnutí a_t.

5. **Hodnotová funkcia.** Zrejme budeme potrebovať už stochastickú rovnicu optimality. Spomeňme predtým ešte finálne hodnotenie - to sa oproti predošlému príkladu nezmení. Uveďme teda len pre úplnosť

$$V_N(s_N) = \begin{cases} 0 & m_N = 0 \\ c_n & m_N = 1. \end{cases}$$

Spomínaný tvar rovnice optimality v tomto znáhodnenom tvare bude

$$V_t(s_t) = \max_{a_t \in A_t} \big(C_t(s_t, a_t) + \mathbb{E}\big[V_{t+1}(s_{t+1})\big|s_t\big]\big).$$

V našom prípade máme len dve možnosti, ako sa stav s_{t+1} môže vyvinúť, a tak si môžeme dovoliť rozpísať a dosadiť do tvaru 1.3.3

$$V_t(s_t) = \max_{a_t \in A_t} \big(C_t(s_t, a_t) + \sum_{s \in S} \mathbb{P}[s_{t+1} = s|s_t] V_{t+1}(s)\big)$$

čím získavame

$$V_t(s_t) = \max_{a_t \in A_t} \Big(- a_t c_t + p_t^+ V_{t+1}\big((m_t + a_t, k_t + \mathbb{I}_{\{a_t=-1\}}, c_t + \alpha_t)\big) \\ + p_t^- V_{t+1}\big((m_t + a_t, k_t + \mathbb{I}_{\{a_t=-1\}}, c_t + \beta_t)\big)\Big).$$

Optimálne rozhodnutia v stavoch $s_t \in S'$ určíme spätným rekurzívnym riešením rovníc

$$a_t^*(s_t) = \arg\max_{a_t \in A_t} \Big(- a_t c_t + p_t^+ V_{t+1}\big((m_t + a_t, k_t + \mathbb{I}_{\{a_t=-1\}}, c_t + \alpha_t)\big) \\ + p_t^- V_{t+1}\big((m_t + a_t, k_t + \mathbb{I}_{\{a_t=-1\}}, c_t + \beta_t)\big)\Big),$$

kde S' značí už „okresaný" stavový priestor obsahujúci len stavy prípustné v daných trajektóriách. S tým musí počítať aj následná implementácia.

2.3 Stratégie technickej analýzy

Metódy technickej analýzy, ktorý otestujeme, budú jednoduché aplikácie zopár základných indikátorov. Ešte predtým, než ich implementujeme, uveďme základné informácie o týchto metódach.

Technická analýza (TA) je súhrnný názov pre množstvo heuristických metód používaných pre obchodovanie na finančnom trhu. Cieľom TA je pomocou historického vývoja trhu analýzou dát predpovedať budúci vývoj - prevažne predpovede cien či obchodovaného objemu. Jej účinnosť sa spolieha na to, že trhové trendy, a s nimi tzv. vzory (patterns) vo vývoji cien majú tendenciu sa opakovať, často v predpovedateľných intervaloch. Aj keď sa prevažne jedná o matematicky, finančne ani ekonomicky nepodložené techniky, TA je vďaka svojej dostupnosti a jednoduchosti veľmi využívaná a v spolupráci s fundamentálnou a kvantitatívnou analýzou patrí medzi najpoužívanejšie prostriedky malých denných traderov, dlhodobých investorov, ale aj veľkých investičných bánk či hedge fondov.

TA bola v minulosti a dodnes zostáva vo vedeckých kruhoch kontroverznou témou. Má množstvo podporovateľov a minimálne toľko kritikov (mimo iného napr. hypotéza eficientného trhu, podľa ktorej by ceny na historickom trende vôbec závisieť nemali), z čoho vzniknutá diskusia dala vznik taktiež niekoľkým výskumným projektom z oblasti technickej analýzy. Za všetky spomeňme aspoň články [Caginalp-Balenovich, Park-Irwin, Lo-Mamaysky-Wang, Lui-Chong] do ktorých autor pri tvorbe práce zahliadol. Po nahliadnutí do týchto článkov je zrejmé, že matematicky zrelý rozbor technickej analýzy je nepochybne veľmi zaujímavý problém. Venovať sa mu na úrovni porovnateľnej s týmito referenciami ale nemá byť cieľom práce. Technické detaily a ďalšie zaujímavosti tak ponechávame k nahliadnutiu vo vyššie uvedenej literatúre, zatiaľ čo sa budeme venovať zopár praktickým ukážkam.

V nasledujúcich odstavcoch postavíme a otestujeme jednoduché stratégie založené na troch indikátoroch technickej analýzy - MACD (moving average convergence-divergence), RSI (relative strength index) a EMV (ease of movement indicator). Len v krátkosti si predstavme každý z nich, predtým ale pripomeňme už spomínanú terminológiu kĺzavých priemerov, ktorá bude v nasledujúcich sekciách hojne používaná.

Definícia. Majme súbor (ďalej časový rad cien) $(p_t)_{t\in\{1,\dots,n\}}$. Jednoduchým kĺzavým k-priemerom (ďalej aritmetický $k-$dňový priemer) v časovom úseku n rozumieme nevážený aritmetický priemer posledných k hodnôt súboru, t.j.

$$SMA_k = \frac{1}{k}\sum_{i=n-k+1}^{n} p_i$$

Exponenciálnym kĺzavým priemerom (ďalej len exponenciálny priemer) rozumieme súbor $(E_t)_{t\in\{1,\dots,n\}}$

$$EMA_t = \begin{cases} p_1 & t = 1 \\ \gamma p_t + (1-\gamma)E_{t-1} & 1 < t \leq n. \end{cases}$$

Parameter $\gamma \in (0,1)$ ďalej budeme podľa konštantných časových úsekov dĺžky T ($T-$dňový exponenciálny priemer) voliť vzťahom $\gamma = \frac{2}{T+1}$.

2.3.1 Moving Average Convergence-Divergence

Indikátor MACD (moving average convergence-divergence) je indikátor spočívajúci v predpovedaní trendu pomocou vzťahov medzi dvomi kĺzavými priemermi ceny inštrumentu. Postup aj použitie je veľmi jednoduché (detailnejší opis je možné nájsť napr. v [ip-MACD]):

- Zaznamenajú sa exponenciálne priemery ceny dvoch rôznych dĺžok. Časové úseky sa prispôsobujú dobe investovania, tradične sa však využíva jeden kratší (napr. 12-dňový) „fast EMA" a jeden dlhší (napr. 26-dňový) „slow EMA". Následne sa vypočíta rozdiel, $MACD = EMA_{fast} - EMA_{slow}$.

- Vytvorí sa tzv. signálna čiara sgn, čo je časovo kratší exponenciálny priemer súboru $MACD$. Pri výbere $12 - 26$ $MACD$ sa zvyčajne používa 9-dňový priemer.

- Porovnajú sa hodnoty signálnej čiary s $MACD$: signály sú generované priesečníkmi signálnej čiary s $MACD$. Ak dôjde k priesečníku „zhora", t.j.

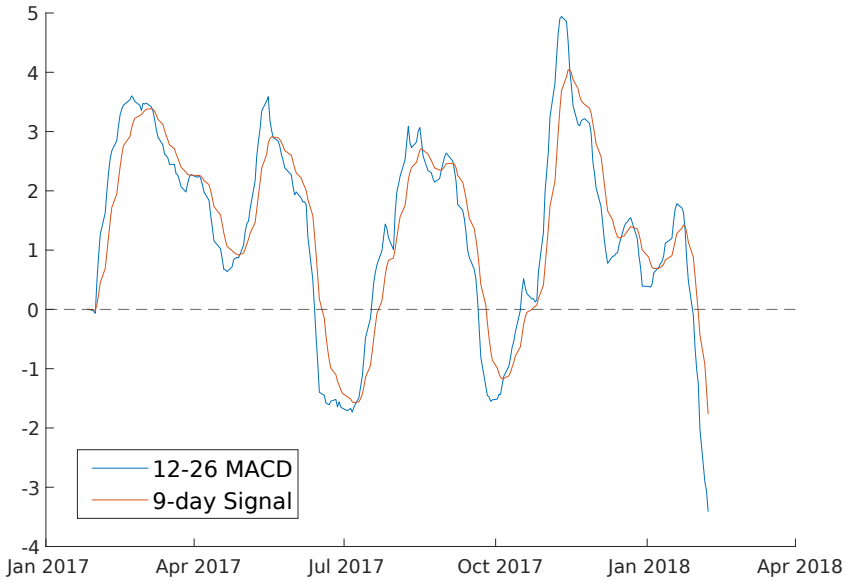
  $$MACD_t > sgn_t \ \& \ MACD_{t+1} < sgn_{t+1}$$

  jedná sa o bearish signál (očakávaný je pokles ceny; teda predávať akciu), a naopak v priesečníkoch „zdola", t.j.

  $$MACD_t < sgn_t \ \& \ MACD_{t+1} > sgn_{t+1}$$

  sa jedná o bullish signál (očakávaný je rast ceny; teda nakupovať akciu). Je zaužívané na signál (nakupovanie/predávanie) počkať aspoň nejaký čas, aby nedošlo k neodôvodnenému predaju či nákupu pri fluktuáciách. Predávať resp. nakupovať je tak odporúčané až po príchode ďalšieho údaja do $MACD$, čo je v našom prípade v nasledujúci burzový deň.

Nakoľko v tejto kapitole stále obchodujeme na trhu s jedinou akciou, jednoduchá stratégia obchodujúca podľa tohto indikátoru by mohla vypočítať $MACD$ a signálnu čiaru a pomocou nich vygenerovať signály (tzv. triggers), kedy nakupovať či predávať túto 1 akciu. Na príklade historickej ceny akcie spoločnosti Apple, Inc. (NASDAQ: AAPL) [Data src] ilustrujme tento indikátor. V Obrázku 2.3.1 by sme teda nakupovali resp. predávali akciu v nasledujúci deň po priesečníkoch dvoch grafov (podľa typu priesečníka).



Obr. 2.3.1: Indikátor 12-26 MACD a jeho 9-dňová signálna čiara.

2.3.2 Relative Strength Index

Indikátor RSI (relative strength index) je používaný na predpovede prepredaného (oversold) a prekúpeného (overbought) stavu trhu. Prepredaný trh je stav, ku ktorému dochádza v prípadoch, kedy cena inštrumentu poklesne - zvyčajne z dôvodu vysokého množstva predajných objednávok - a v najbližšej dobe sa tak očakáva sa jej korekcia nahor (rast ceny). Analogicky, v prípade prekúpeného stavu sa z dôvodu prevládajúcich nákupných objednávok očakáva korekcia nadol (pokles ceny).

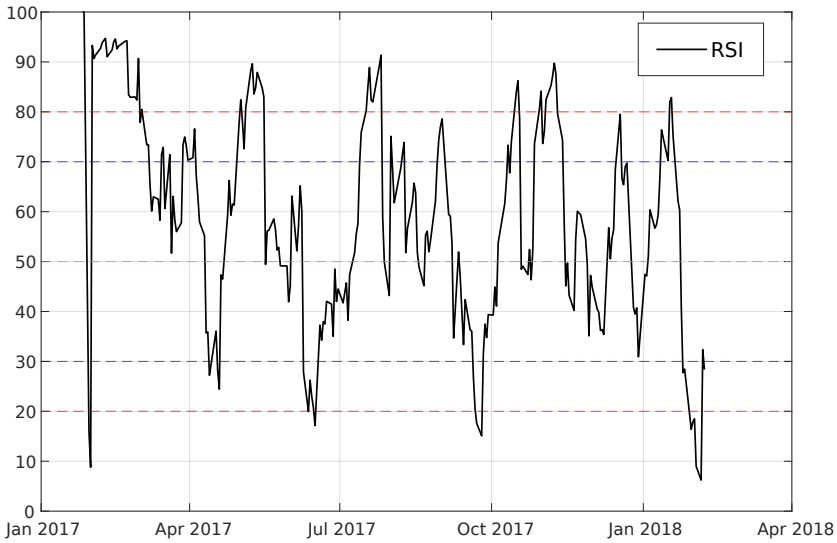
Výpočet RSI je nasledovný (detailnejšie pozri [ip-RSI]):

1. Vypočíta sa priemerný pokles a priemerný nárast ceny v istom časovom úseku (bežne 14 dňový úsek). Ak napríklad v siedmich dňoch cena za burzový deň (cena pri otvorení burzy v porovnaní s cenou pri zatváraní burzy - close) narástla, z týchto 7 dní sa spočíta priemerný percentuálny nárast $avggain$. Vo zvyšných siedmich dňoch cena za deň poklesla, a z týchto dní sa opäť vypočíta priemerný percentuálny pokles $avgloss$.

2. Vypočíta sa hodnota
$$RSI = 100 - \frac{100}{1 + \frac{avggain}{avgloss}}$$

3. Podľa hodnôt RSI sa odčíta, či je v danej situácii trh prepredaný resp. prekúpený. Signály pre nákup sú opäť indukované podľa týchto hodnôt, typicky v úsekoch, kde sa RSI drží pod 30, resp. signály pre predaj nad 70%.

Stratégia by tak mohla nakupovať akciu, ak RSI pretne a klesne pod hranicu 30. Naopak, predajný signál analogicky pri hodnote nad 70%. Príklad RSI na rovnakom datasete AAPL ilustruje obr. 2.3.2.



Obr. 2.3.2: Indikátor RSI s vyznačenými hranicami 20,30,70 a 80 % pre buy/sell signály.

2.3.3 Ease of Movement

Ease of movement indikátor je jediný zo spomínaných indikátorov, ktorý sa okrem cien venuje aj analýze obchodovaného množstva (volume). Hodnotí tzv. ease of movement, a pomocou neho pohyblivosť (momentum) cien v sledovanom období. Inými slovami, snaží sa odhadnúť, ako ľahko resp. ako ťažko sa cena pohybuje nahor/nadol medzi jednotlivými úsekmi. Výpočet je nasledovný (opäť bližšie v [ip-EMV]):

1. Vypočítajú sa priemerné rozdiely v denných cenových minimách a maximách za posledný časový úsek (tzv. distance moved)

$$D = \frac{1}{2}\left(max_{today} + min_{today} - (max_{yesterday} + min_{yesterday})\right)$$

2. Vypočíta sa tzv. box ratio

$$BR = \frac{\text{scaled volume}}{max_{today} - min_{today}}$$

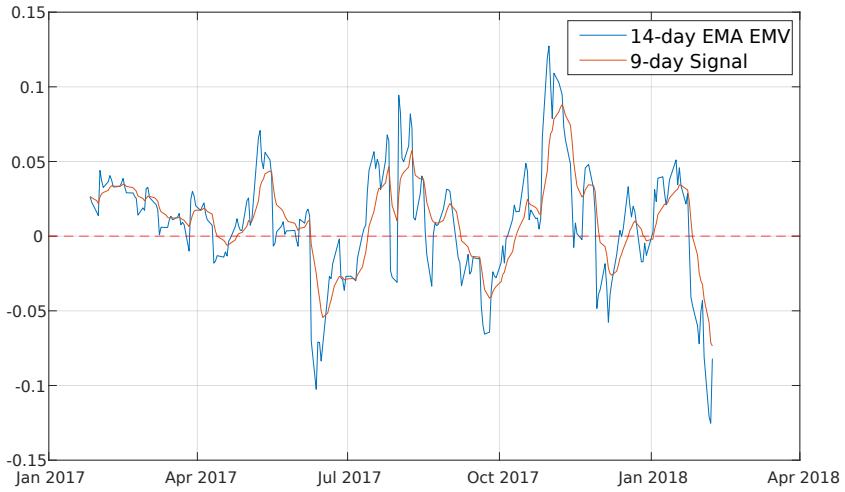
kde scaled volume je hodnota obchodovaného objemu za posledný deň vydelená konštantou 10^n; $n \in \mathbb{N}$ je volené tak, aby výsledná hodnota EMV bolo číslo z intervalu $[-1, 1]$. Táto normalizácia ale nie je nevyhnutná.

3. Za každý časový úsek sa uloží hodnota tzv. 1-period EMV

$$EMV_1 = \frac{D}{BR}$$

a vytvorí sa EMV ako kĺzavý priemer týchto 1-period hodnôt podľa dĺžky časového úseku, ktorý sledujeme. Tradičný výber je 14-dňový aritmetický priemer, v našej ilustrácii ale použijeme 14-dňový exponenciálny.

EMV sa potom spravidla používa ako oscilátor, t.j. zakreslí sa jeho priebeh a porovnáva sa s 0. Podľa priesečníkov môže tak jednoduchá stratégia generovať nákupné signály (nakupovať akciu), ak EMV pretne 0 zdola, analogicky naopak predajné signály (predávať akciu), ak EMV pretne nulu zhora. Ku generovaniu signálov je možné použiť EMV, ale aj jeho signálnu čiaru (t.j. 9-dňový exponenciálny priemer súboru EMV), ktorej prechody nulou su spravidla posunuté o časový úsek vopred, čo je už zo spomínaných dôvodov vhodné na odlíšenie falošných signálov vyplývajúcich z fluktuácii. Ilustráciu už na známom datasete poskytne obrázok 2.3.3.



Obr. 2.3.3: 14-dňový indikátor EMV s 9-dňovou signálnou čiarou

2.4 Implementácie

Súčasťou práce sú implementácie všetkých vyššie uvedených stratégii. Využime tieto implementácie a na historických dátach porovnajme stratégie metód technickej analýzy, a taktiež ilustrujme jeden z DP algoritmov. Využijeme spomínaný dataset AAPL, na ktorom aplikujeme simulácie Monte Carlo pre porovnanie profitability rôznych stratégii. Nakoľko sa jedná o vopred dané ceny, budeme aplikovať stratégiu zo sekcie 2.1. s neobmedzeným počtom možných transakcií (pre naše účely T = 450; čo vo vybraných úsekoch nejde dosiahnuť), zatiaľ čo z technickej analýzy otestujeme všetky 3 stratégie.

Postup bude teda nasledovný:

- Z dostupných dát cien akcie (obdobie 02/2013 - 02/2018; pre jednoduchosť volíme v jednotlivých dňoch closing ceny) sa vyberie časový úsek, na ktorom budú porovnávané stratégie obchodovať. Pre naše testy volíme úseky dĺžky 60 - 520 burzových dní (t.j. cca 3 mesiace až cca 2 roky pracovných dní), ktoré sú generované náhodne resp. psuedonáhodne v MATLABe funkciou `randi` (generovaný je počiatočný dátum obchodovania aj dĺžka intervalu). Výber intervalov je zvolený prihliadajúc k účinnosti používaných metód technickej analýzy, ktoré sú založené na sledovaní trendu istej dĺžky.

- Prevedie sa $N = 1000$ takýchto simulácii a pre každú stratégiu sa zaznamená priemer zarobeného profitu spolu so štandardnou chybou

V tabuľke uveďme výsledky simulácii.

stratégia	DP (známe ceny)	12-26 MACD	RSI	EMV
priemerný profit	174.0147	2.3414	9.1558	13.4514
štandardná chyba	2.6851	0.5388	0.3288	0.5565

Tabuľka 2.1: Profitabilita testovaných stratégii zo sekcií 2.1 - 2.3

Priemerným profitom vyššie rozumieme priemer súčtov rozdielov predajných a nákupnych cien, za ktoré stratégie obchodovali, pričom všetky stratégie začínajú s profitom rovným 0 (môžu, resp. na začiatku musia nakupovať do mínusu). Štandardnú chybu sme definovali pre konečný súbor ($N = 1000$) pomocou výberovej smerodajnej odchýlky $\widehat{\sigma}_N$ ako

$$s_N = \frac{\widehat{\sigma}_N}{\sqrt{N}} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(X_i - \overline{X}_N)^2}$$

kde

$$\overline{X}_N = \frac{1}{N}\sum_{i=1}^{N} X_i$$

je klasický aritmetický priemer profitov v jednotlivých iteráciach.

Môžeme teda v rámci nášho príkladu konštatovať, že metódy technickej analýzy síce sú na našom datasete v priemere profitabilné, ale len veľmi málo. Toto zistenie ilustruje fakt, že jednotlivé indikátory technickej analýzy nie sú veľmi účinné samostatne. V každej investičnej stratégii založenej na TA je potrebné indikátory v správnom zložení kombinovať, a pre lepšiu funkcionalitu ich dopĺňať inými metódami, ako sú napríklad fundamentálna analýza alebo metódy kvantitatívnej analýzy.

Na mieste je tiež diskusia o samotnej simulácii: vykonávame síce náhodný výber úsekov cien, vyberáme ale vždy z celého rovnakého súboru dát. Aj pri náhodnom výbere úsekov je tak možné, že isté úseky alebo ich časti sú volené častejšie než iné. Jednotlivé pozorovania teda nie sú nutne nezávislé.

Naopak v prípade algoritmu DP vidíme, že je schopný dobre zarobiť. To je očakávané, nakoľko ceny pozná dopredu, čo je, ako už uvádzame v 2.1., nerealistický predpoklad slúžiaci len pre príklad na ozrejmenie princípov. Priame porovnanie tohto algoritmu s

metódami TA (ktoré ceny, pochopiteľne, dopredu nepoznajú) tak nie je na mieste, a tento výsledok slúži len pre ilustráciu. Porovnania samotné nechávame až na Kapitolu 3.

Jedno z miest, kde ale vidíme možnosti na zlepšenie samotnej TA, je pozrieť sa bližšie na parametre použité pri výpočtoch. Porovnajme preto napríklad indikátor MACD pre rôzne dĺžky použitých kĺzavých priemerov:

parametre MACD (fast-slow-signal)	12-26-9	21-44-18	18-38-14	35-62-28
priemerný profit	3.7451	8.5789	3.7221	9.0274
štandardná chyba	0.5496	0.6155	0.5520	0.5196

Tabuľka 2.2: Stratégie MACD pre rôzne parametre priemerov

V tomto príklade sme simulovali Monte Carlo ($N = 1000$) z náhodných výberov dĺžky 90 až 520 burzových dní (4.5 mesiaca až 2 roky). Je vidieť, že výber parametrov má na profitabilitu stratégie netriviálny vplyv. Kvantifikovať tento vplyv, ani opísať možnosti jeho využitia pre vytváranie lepších stratégii si v aktuálnom momente netrúfame, určite by ale mohlo byť zaujímavé skúsiť daný vplyv preskúmať bližšie a zaoberať sa napríklad optimalizovaným výberom týchto parametrov (na základe historického trendu, dĺžky obchodovaného úseku, a podobne). Z charakteru úlohy by k tomu veríme, mohol byť použiteľný práve framework (približného) dynamického programovania. Toto pojednanie nechávame na prípadné preskúmanie v budúcnosti.

Kapitola 3

Stratégie pre model vysokofrekvenčného trhu

Práca vzniká ako súčasť projektu, ktorý sa zaoberá racionálnym rozhodovaním na trhu s limitnými objednávkami (M. Šmíd, 2021)[1]. Výsledkom tohto projektu je softwarová implementácia simulátora finančného trhu, na ktorom obchodujú umelí agenti používajúci rôzne stratégie. Implementácia je voľne dostupná k nahliadnutiu ako online repozitár [github]. V tejto kapitole sa budeme venovať pohľadu vysokofrekvenčného obchodníka a implementujeme dve stratégie, ktorých ziskovosť na modelovom trhu porovnáme.

3.1 Model trhu a stratégie účastníkov

Na modelovom trhu sú na začiatku pridané stratégie obchodníkov, a následne je spustená simulácia. Mimo prirodzených parametrov každej jednotlivej stratégie má prirodzené parametre aj samotná simulácia: sú to dĺžka časového úseku, v ktorom sa obchoduje; prípadne tiež počet iterácii, ak chceme výsledky porovnávať ako priemery po niekoľkých chodoch simulácie.

Trh je formulovaný ako zero-sum game. To znamená, že jednotliví agenti na trhu si vymieňajú pomocou nákupu a predaja na trhu svoje zdroje tak, že konečný celkový súčet zdrojov obchodníkov je zhodný so súčtom zdrojov obchodníkov na začiatku. Inými slovami, zarobiť sa dá len tak, že pripravíme o zdroje iných účastníkov a analogicky prísť o zdroje je možné len na úkor obohatenia ostatných obchodníkov o to isté množstvo. Profitabilita obchodovacích stratégii tak v tomto prípade bude závisieť nielen na kvalite stratégie ako takej (napríklad ako tomu bolo v Kapitole 2 na trhu s jediným účastníkom), ale bude ovplyvnená aj tým, ako obchodujú ostatní účastníci.

V praxi je zoskupenie agentov na trhu, ako spomíname, tvorca trhu (ktorého úlohou je vytvárať na trhu likviditu; v praxi na základe dohody s organizátorom trhu, za čo je aj nejakým spôsobom odmeňovaný), príjemcovia likvidity (investori) a vysokofrekvenční obchodníci. Pre možnosti simulácií je vopred (práca školiaceho pracoviska) implementovaných zopár základných stratégii. Spomeňme dve, ktoré v tejto kapitole využijeme.

[1] bližšie na `http://www.utia.cz/cs/projects/3154` (prístup 5.7.2021)

- **marketmaker:** úlohou tejto stratégie je imitovať správanie tvorcu trhu ako je ilustrované vyššie. Implementácia je inšpirovaná článkom [Stoll], zatiaľ heuristická a parametre pre tvorbu trhu nie sú optimalizované (bližšie v sekcii 3.4.)

- **liquidity taker**: vydáva s určenou frekvenciou objednávky náhodného typu (nákup/predaj/nič).

Podotknime, že spomínané stratégie vytvorené na školiacom pracovisku, ako aj celý simulátor marketsim, sú zatiaľ v štádiu vývoja.

Každá stratégia má svoje parametre, ktoré nejakým spôsobom definujú ich správanie. Okrem názvu sú to obecne obchodované množstvo (volume), na ktoré sú vydávané objednávky, a tiež hodnotu časového „meškania", s ktorým vydáva objednávky (pre priblíženie reálnej situácie, kde reakčný čas obchodníkov predĺžujú potrebné výpočty, komunikácia, atď.). Naše stratégie budeme implementovať s reakčným časom 1 časového ticku (sekundy), stratégia tvorcu trhu ale napríklad vydáva objednávky bez omeškania (okamžite) a stratégia liquidity taker zasa v nepravidelných náhodne vyberaných intervaloch.

Celý simulátor trhu je obalený do menného priestoru `marketsim`, pomocou ktorého sú ďalej implementované ďalšie súčasti. Samotné implementácie jednotlivých stratégii sú napísané v obecnej základnej triede `marketsim::tstrategy`, z ktorej ďalej dedia tri triedy podľa typu stratégii. Jednou z nich je trieda `marketsim::tsimplestrategy`, ktorú využijeme na implementáciu našich dvoch stratégii.

3.2 Stratégia MACD

Indikátor MACD sme už spomínali v Kapitole 2. Tam sme otestovali jeho účinnosť na trhu, kde stratégia obchodovala samostatne na známych cenách. Tentokrát implementujeme stratégiu a otestujeme jej profitabilitu na trhu s viacerými agentmi. Algoritmus pre výpočet MACD a generovanie triggerov už bol opísaný v sekcii 2.3.1, opíšme teda len implementáciu.

Ako ilustrujeme vyššie, všetky stratégie sú podtriedami triedy `marketsim::tstrategy`. Pre naše potreby využijeme zdedenú triedu `marketsim::tsimplestrategy` a napíšeme triedu `marketsim::ta_macd`. Kostra implementácie je nasledovná:

```
class ta_macd : public tsimplestrategy
{
public:
  ta_macd(const std::string& name, tvolume delta, ttime timeinterval,
                  const int _s, const int _f, const int _sgn)
  : tsimplestrategy(name, timeinterval), fdelta(delta), s(_s), f(_f), sgn(_sgn)
  {}
private:
  virtual tsimpleorderprofile simpleevent(const tmarketinfo& mi, const ttradinghistory&);
  tvolume fdelta;
  const int f, s, sgn;
  const double alpha_f = 2.0 / (f + 1.0), beta_f = 1 - alpha_f,
               alpha_s = 2.0 / (s + 1.0), beta_s = 1 - alpha_s,
               alpha_sgn = 2.0 / (sgn + 1.0), beta_sgn = 1 - alpha_sgn;
  std::vector<double> ema_slow, ema_fast, macd, signal;
};
```

Samotná stratégia je implementovaná v metóde `ta_macd::simpleevent`, ktorá je volaná v jednotlivých časových úsekoch a rozhoduje o tom, akú objednávku naša stratégia v

danom úseku vydá na trh. Bude sa klasicky rozhodovať medzi nákupom, predajom resp. žiadnou objednávkou, a to podľa MACD triggerov ako sú uvedené v sekcii 2.3.1.

V konštruktore triedy `ta_macd` vidíme taktiež prirodzené parametre stratégie MACD. Jedná sa o konštanty použité vo výpočte exponenciálnych priemerov EMA fast, EMA slow a signálnej čiary, ktoré ukladáme ako súkromné premenné triedy. Hodnoty priemerov sú dynamicky ukladané do polí s odpovedajúcim názvom, ktoré následne využíva metóda `simpleevent` na vykonávanie objednávok.

Okrem spomínaných prirodzených parametrov stratégie poskytujeme konštruktoru aj prirodzené parametre triedy `tsimplestrategy:` názov stratégie, obchodovaný objem a dĺžka časového úseku, v ktorom je stratégia volaná.

3.3 Stratégia ADP

Ako už bolo v práci prednesené, stratégie technickej analýzy využívajú historický vývoj dát (trend) na predpoveď budúcich cien. Na porovnanie s TA teda navrhneme stratégiu, ktorá taktiež bude vykonávať rozhodnutia pomocou informácii o historickom trende, tentokrát ale pomocou približného dynamického programovania (ďalej ADP). Framework ADP sme detailnejšie opísali v sekcii 1.6., ktorú sme ukončili kostrou jedného z mnohých používaných ADP algoritmov. Tento známy algoritmus 1.4. ďalej použijeme k implementácii stratégie; najprv ale spomínanú stratégiu navrhnime. Prejdeme teda ku klasickému značeniu ADP.

Obchodovať budeme na trhu v časových úsekoch $t \in \{1, \dots, T\}$. Označme $W_t = x_t p_t + y_t$ hodnotu účtu v čase $t \leq T$, kde p_t je aktuálna cena akcie, x_t počet vlastnených akcií a y_t hotovosť (t.j. nejaké voľné prostriedky, ktoré máme na účte k dispozícii neviazané v akciách). Našim cieľom bude maximalizovať hodnotu účtu na konci obchodovacieho obdobia.

Rozhodnutia v jednotlivých úsekoch voľme pre jednoduchosť $a_t \in A_t = \{0, -1, 1\}$, t.j. v každom úseku predávame alebo kupujeme len 1 akciu. Je to postačujúce, nakoľko pri použití v simulátore bude v prípade nášho algoritmu rozhodnutie určovať trigger na nákup resp. predaj vopred určeného objemu. Nakoľko pri prechode z t do $t+1$ očakávame nejakú (vopred neznámu) zmenu ceny $\pi_{t+1} = p_{t+1} - p_t$, zmena prostriedkov na účte po vykonaní rozhodnutia a_t bude náhodná veličina v tvare

$$W_{t+1} = (x_t + a_t)(p_t + \pi_{t+1}) + y_t - p_t a_t = x_t p_t + y_t + (x_t + a_t)\pi_{t+1} = W_t + (x_t + a_t)\pi_{t+1}.$$

Našou úlohou je tak maximalizovať hodnotu

$$\mathbb{E}[W_T] = \mathbb{E}\left[\sum_{i=0}^{T-1} (x_t + a_t)\pi_{t+1}\right],$$

kde iniciálnu hodnotu W_0, ktorá je vždy rovnaká, už v maximalizácii vynechávame.

Pre náš príklad budeme predpokladať, že matica prechodu veličiny π_{t+1} závisí na trende za posledných m dní: teda na raste/poklese ceny v posledných m časových úsekoch. Označme maticu prechodu $\mathbf{P}[\pi|h]$, kde $h \in H = \{-1, 0, 1\}^m$ vyjadruje pokles/nemennosť/rast ceny v jednotlivých úsekoch. Tiež pre jednoduchosť predpokladáme, že π môže naberať celočíselné hodnoty $\{-q, \dots, 0, \dots, q\}$ pre pevne volené $q \in \mathbb{N}$.

Dostaneme tak rovnice optimality $\forall t \in \{0, ..., T\}$

$$\begin{aligned} V_t(h,x) &= \max_{a \in A_t} \mathbb{E}\big[(x+a)\pi_{t+1} + V_{t+1}\big(\eta(h,\pi_{t+1}), x+a\big)\big|h\big] \\ &= \max_{a \in A_t} \sum_{\pi=-q}^{\pi=q} \big[(x+a)\pi + V_{t+1}\big(\eta(h,\pi), x+a\big)\big]\mathbb{P}[\pi_{t+1} = \pi|h] \end{aligned} \tag{3.3.1}$$

kde značíme $\eta(h, \pi_{t+1})$ funkciu, ktorá prevedie „starú" históriu trendu h a posun ceny π_{t+1} na „novú" históriu trendu.

Pre zjednodušenie stavového priestoru založíme tiež obmedzenie množstva akcií, ktoré môžeme vlastniť, na pevné $\mu \in \mathbb{N}$. Stavový priestor príkladu tak je celkovo

$$S = H \times \{0, \dots, \mu\} = \{-1, 0, 1\}^m \times \{0, \dots, \mu\}$$

Nakoľko príklad pracuje s konečným horizontom $T < \infty$, je na mieste spomenúť terminal valuation. V poslednom úseku volíme hodnoty $V_T(s) = 0 \ \forall s \in S$.

Ako uvádzame už v prvej kapitole pri opise Algoritmu 1.4., pre potreby výpočtu rovníc optimality týmto spôsobom je nutné poznať jednokrokovú maticu prechodu $\mathbf{P}[\pi|h]$. Túto zrejme nepoznáme, a budeme musieť jej hodnoty aproximovať. To urobíme pomocou simulátora trhu, ktorý pre dostatočný počet iterácii spustíme (zatiaľ bez novej ADP stratégie), a z vývoja cien v závislosti na historickom vývoji odhadneme pravdepodobnosti prechodu. Odhadovať tak potrebujeme rozdelenia $(\mathbf{P}[\pi = p|h])_{p \in \{-q,\dots,q\}}$ pre všetky možné histórie $h \in H$. To je zrejme spolu 3^m rozdelení, kde každé má obor hodnôt $\{-q, \dots, q\}$. Postup aproximácie upresní Algoritmus 3.1.

Algoritmus 3.1 Generovanie pravdepodobností prechodu v závislosti na histórii

1. Zvolím požadovanú presnosť (počet iterácii) $N \in \mathbb{N}$, dĺžku obchodovaného obdobia T, dĺžku max. historického trendu m. Nulami inicializujem tabuľku $(\mathbf{P}[\pi = p|h])_p$ pre $p \in \{-q, \dots, q\}$, $h = \{0, \dots, 3^m - 1\}$ (index h už prevedený do desiatkovej sústavy; pôvodná história kódovaná v trojkovej sústave napr. $(-1, 1, 1) \leftrightarrow (0, 2, 2) \leftrightarrow 0 \cdot 3^2 + 2 \cdot 3^1 + 2 \cdot 3^0 = 8$).

2. Pre $n = 1, \dots, N$

   - Spustím simulátor trhu s danými účastníkmi
   - Pre $i = 0, \dots, T - m$
     - z historických cien $p_i, \dots, p_{i+m-1}$ zaznamenám trend h (určím, o akú históriu sa jedná)
     - podľa ceny p_{i+m} inkrementujem v príslušnej tabuľke $\mathbf{P}[\pi = p_{i+m}|h]$ hodnotu na danom mieste o $+1$

3. Tabuľku $(\mathbf{P}[\pi = p|h])_p$ znormujem, aby v jednotlivých históriach zostali rozdelenia pravdepodobnosti. T.j. aby $\forall h \in H$ platilo $\sum_{p=-q}^{q} \mathbf{P}[\pi = p|h] = 1$

S takto odhadnutou maticou prechodu už dokážeme riešiť rovnice optimality v tvare 3.3.1, a teda hľadať optimálne rozhodnutia.

Navrhnutý ADP algoritmus, a teda hľadanie optimálnej politiky v jednotlitých stavoch, implementujeme vo funkcii `adp_getPolicy`. Táto využije už vopred odhadnutú maticu rozdelení $(\mathbf{P}[\pi|h])_{h\in H}$ a prirodzené parametre príkladu k nájdeniu politiky ako súboru matíc $A = (A_t(s_t))_{t\leq T}$ kde stavy $s_t = (h_t, x_t)$ sú usporiadané dvojice indexov histórie h_t (už v desiatkovej sústave; bližšie algoritmus 3.1.) a počtu práve vlastnených akcií $0 \leq x_t \leq \mu$. Ďalším prirodzeným parametrom funkcie bude počet iterácii, ktoré chceme v ADP algoritme využiť. Návratová hodnota funkcie bude politika ako súbor polí obsahujucích optimálne rozhodnutia $(-1/0/1)$ v daných stavoch a v jednotlivých časových úsekoch.

```
TSpace_d adp_getPolicy(const unsigned iter, const unsigned time,
                       const unsigned max_stocks, const unsigned max_hist, TState P)
```

Takto získanú rozhodovaciu politiku následne využijeme pre obchodovanie na simulátore. Implementácia prebehne založením triedy `marketsim::adp_trend`, využitím ktorej je stratégia volaná pomocou metódy `adp_trend::simpleevent.` Tá si postupne sleduje históriu trhu podľa vývoja cien a počet vlastnených akcií, a následne prečíta optimálne rozhodnutie pre daný stav, pričom súbor rozhodnutí v jednotlivých stavoch a časových úsekoch je už vopred vypočítaný funkciou `adp_getPolicy` a predávaný ako parameter konštruktoru. Prirodzené parametre bližšie ilustruje kostra implementácie, ktorá je podľa očakávaní analogická k triede z 3.2..

```
class adp_trend : public tsimplestrategy
{
public:
  adp_trend(const std::string& name, tvolume delta, ttime timeinterval,
            const int _max_stocks, const int _max_hist, TSpace_d _P)
  : tsimplestrategy(name, timeinterval), max_stocks(_max_stocks),
    max_hist(_max_hist), fdelta(delta), P(_P)
  {}
private:
  virtual tsimpleorderprofile simpleevent(const tmarketinfo& mi, const ttradinghistory& th);
  tvolume fdelta;
  TSpace_d P;
  const int max_hist, max_stocks;
};
```

Implementácia teda prebehla v troch krokoch, ktoré pre ozrejmenie zopakujme v nasledujúcom zhrnutí:

1. Pomocou iterovanej simulácie trhu bez ADP algoritmu (pre náš príklad sme využili účastníkov so stratégiami marketmaker, liquidity taker a MACD ako ilustrovaná v 3.2.) sa odhadne matica prechodu

$$\mathbf{P} = \left(\mathbb{P}[\pi = p|h]\right)_{p\in\{-q,\dots,q\}}^{h\in H}$$

2. Využitím odhadnutej matice $\mathbf{P}$ je pomocou ADP (funkcia `adp_getPolicy`) vypočítaná rozhodovacia politika A, ktorá obsahuje rozhodnutia v daných stavoch

3. Na simulátor je pridaná stratégia `adp_trend`, ktorá sleduje vývoj cien a pomocou poskytnutej politiky A vykonáva rozhodnutia (vysiela objednávky na nákup/predaj/nič).

Samotné simulácie vyhodnotíme v nasledujúcej sekcii.

3.4 Simulácie

Opakovanou simuláciou ($N = 100$) najprv otestujeme metódu technickej analýzy - indikátor MACD. Na trh pridáme troch účastníkov: tvorcu trhu, obchodníka so stratégiou liquidity taker a obchodníka so stratégiou MACD. Pre názornosť každý z nich dostane rovnaký inventár: hotovosť vo výške 100000 a 500 vlastnených akcií na začiatok. Obchodovať sa bude 100 časových tickov (v našom prípade sekúnd). Stratégii MACD poskytneme parametre 26-12-9 ako v predošlej kapitole.

Výsledky simulácie poskytuje tabuľka 3.1. Viditeľne teda MACD dokáže na trhu v

stratégia	MACD	Market maker	Liquidity taker
priemerný profit	8757.84	-20449	11691.1
št. chyba	2033.2	6581.16	6324.97

Tabuľka 3.1: Výsledky simulácie 1

priemere zarobiť. V každom prípade ale stratégia prehráva už proti triviálnej stratégii liquidity takera.

Pre iné parametre tvorcu trhu, a teda inú stratégiu cenotvorby (zmena parametru α pre exponenciálny priemer aproximácie „skutočnej" ceny z 0.1 na 0.01; bližšie dokumentácia [github] a [Stoll]) už v priemere MACD nezarobí takmer žiadny profit, a opäť prehráva aj proti triviálnej stratégii liquidity takera.

stratégia	MACD	Market maker	Liquidity taker
priemerný profit	31.3571	-26846.7	26815.3
št. chyba	60.544	10760.6	10734.3

Tabuľka 3.2: Výsledky simulácie 2

Tento výsledok je pravdepodobne ovplyvnený skutočnosťou, že metódy technickej analýzy historicky kalibrované na reálne kapitálové trhy nie sú až také účinné na simulovanom trhu, ktorého správanie nie je úplne realistické. Taktiež ale napovedá o fakte, ktorý sme už spomenuli v kapitole 2 a ktorý je každému investorovi na finančnom trhu veľmi dobre známy: jednotlivé indikátory technickej analýzy nie sú nutne spoľahlivé len samé o sebe. Preto sa v reálnych stratégiach pre investovanie používajú kombinácie rôznych indikátorov technickej analýzy, a vždy v spolupráci s ďalšími metódami (ani v tomto prípade ale, samozrejme, úspech nie je zaručený).

Ďalej otestujeme ADP algoritmus. Na trhu necháme obchodovať stratégie z predošlých simulácii, a pre porovnanie k nim pridáme stratégiu ADP, ktorej do začiatku poskytneme inventár s rovnakým množstvo hotovosti (100000) a 10 akciami (volíme nižšie číslo kvôli zníženiu rozmerov stavového priestoru). Táto stratégia, ako opisujeme vyššie, bude vysielať objednávky na nákup/predaj/nič podľa vopred vypočítanej rozhodovacej politiky. Parametre pre výpočet politiky budú $N = 10000$ iterácii pre učenie sa, dĺžka sledovaného

trendu $m = 4$ (porovnajme s 26, 12 a 9 dňovými priemermi v MACD), a maximálny počet akcií $\mu = 100$. Opäť využijeme 100 simulácii, každú pre 100 časových tickov. Výsledky poskytne Tabuľka 3.3.

stratégia	ADP	MACD	Market maker	Liquidity taker
priemerný profit	120820	3161.01	-139221	15239.7
št. chyba	36964.1	891.022	41760.3	6837.08

Tabuľka 3.3: Výsledky simulácie 3

Je teda vidieť, že ADP algoritmus je na simulovanom trhu profitabilný, a dokáže spoľahlivo poraziť triviálnu stratégiu liquidity takera aj indikátor MACD.

Na záver uvedieme ešte dve simulácie, tentokrát ADP algoritmu sledujúceho trendy iných dĺžok. Prvá využíva trend dĺžky 6 časových tickov.

stratégia	ADP	MACD	Market maker	Liquidity taker
priemerný profit	54851.3	1118.42	-64859.8	8890.12
št. chyba	22121	571.553	25314	3691.73

Tabuľka 3.4: Výsledky simulácie 4

Výsledky nie sú oproti simulácii 3 prekvapivé. ADP algoritmus je opäť vysoko profitabilný, a dokáže na trhu poraziť ostatných obchodníkov.

Posledná simulácia využíva trend dĺžky 8 časových tickov. Charakter ADP algoritmu, ale taktiež algoritmu pre aproximáciu matíc prechodu (práve v tom totiž nastáva výpočtový bottleneck) prirodzene vyžaduje veľké výpočtové kapacity. Táto simulácia je preto z dôvodov našich obmedzených výpočtových kapacít na hranici toho, čo v rámci práce dokážeme otestovať. Môže ale poslúžiť ako bližšie porovnanie stratégie ADP so stratégiou MACD (ktorá pre výsledné rozhodovanie využíva signálnu čiaru dĺžky 9 tickov). Ostatné parametre výpočtu politiky aj simulácie zostanú rovnaké.

stratégia	ADP	MACD	Market maker	Liquidity taker
priemerný profit	113032	2356.66	-140483	25094.4
št. chyba	30949.9	918.133	38872.4	10586.6

Tabuľka 3.5: Výsledky simulácie 5

Výsledky sú opäť analogické k predošlým simuláciam.

Kapitolu teda zakončíme konštatovaním, že navrhnutý ADP algoritmus je schopný samostatného obchodovania na simulátore trhu, kde porazí stratégie ostatných obchodníkov a spoľahlivo zarobí profit. Na simulovanom trhu si tak vie poradiť nad naše očakávania. Zarába prevažne na úkor tvorcu trhu, ktorý je takmer vždy v mínuse. Tento fakt je dôsledkom toho, že stratégia tvorcu trhu implementovaná školiacim pracoviskom je stále v rámci vývoja (ostatne, ako aj celý simulátor), ale tiež charakterom článku [Stoll], ktorým

je implementácia inšpirovaná. Očakávame, že stratégie (napr. tvorca trhu) sa budú v budúcnosti vylepšovať a ich následné porovnanie s našim ADP algoritmom tak bude o čosi zaujímavejšie.

Za spomenutie na záver určite stojí skutočnosť, že implementácia ADP algoritmu (pozri [github]) má stále priestor na zlepšenie. Prvky približného dynamického programovania by bolo napríklad možné obohatiť o exploration metódy, ktoré by dovolili lepšie prechádzanie veľmi veľkého stavového priestoru. Pri prechode väčšieho množstva stavov by totiž, veríme, došlo k vylepšeniu rozhodovacej politiky, a teda k lepšej profitabilite. Tento implementačný upgrade nechávame otvorený napríklad pre budúce skúmanie.

Záver

V práci sme sa snažili o otestovanie metód dynamického programovania pre investovanie na finančnom trhu a tiež o ich porovnanie so štandardnými metódami technickej analzýzy.

Ilustráciou dynamického programovania na dvoch toy príkladoch obchodovania s jednou akciou ukazujeme, že framework dynamického programovania si dokáže nájsť uplatnenie aj pri modelovaní investičných stratégii. Aj v jednoduchých modeloch ale narážame na problémy vyplývajúce z prekliatí dimenzionality, ktoré sa v tomto formalizme vyskytujú. Potrebu o ich prekonávanie vidíme ako jedno z hlavných obmedzení tohto postupu.

Ďalej sa venujeme trom indikátorom technickej analýzy (moving average convergence-divergence, relative strength index a ease of movement), ktorých implementáciou v programovacích prostrediach a simulovaným obchodovaním na reálnych historických dátach prichádzame k záveru, že samostatne použité nie sú veľmi profitabilné. Aj preto investičné stratégie väčšinou využívajú kombináciu viacerých indikátorov, taktiež v spolupráci s ďalšími metódami.

Ďalšími výsledkami práce sú návrhy a implementácie dvoch stratégii pre použitie na simulátore vysokofrekvenčného finančného trhu. Implementujeme stratégiu využívajúcu indikátor MACD a simuláciou zisťujeme, že na modelovom trhu je, podobne ako pri testoch na reálnych dátach, len málo profitabilná. Zarobí dokonca menej než triviálna stratégia, ktorá obchoduje náhodne. Mimo dôvodov už spomínaných pri testovaní na reálnych dátach si to tentokrát vysvetľujeme tiež skutočnosťou, že simulovaný trh nie je veľmi vhodným prostredím pre testovanie metód technickej analýzy, ktoré su historicky kalibrované pre reálne trhy.

Testovaním profitability navrhnutého ADP algoritmu zisťujeme, že si dokáže na simulovanom trhu poradiť veľmi dobre. Spoľahlivo profituje oveľa lepšie než triviálne stratégie aj než indikátor MACD, a to najmä na úkor stratégie tvorcu trhu, ktorá je ale stále v rámci vývoja na školiacom pracovisku. Do budúcnosti teda vidíme možnosti skúmanie obohatiť o simulácie ADP stratégie na trhu s už vylepšenými stratégiami ostatných účastníkov, pri ktorých by, veríme, bolo porovnanie zaujímavejšie. Taktiež vidíme priestor pre vylepšovanie ADP stratégie, napríklad dodaním exploration metód pre lepšie prechádzanie stavového priestoru. V tejto časti práce narážame na potenciálne obmedzenia - navhrnuté algoritmy sú implementačne aj výpočtovo veľmi náročné, čo je potrebné brať do úvahy pri ich prípadnom vylepšovaní.

Literatúra

[Powell] W. B. Powell: Approximate Dynamic Programming: Solving the Curses of Dimensionality. John Wiley & Sons, 2011.

[Puterman] M. L. Puterman: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, 1994.

[Bellman] R. E. Bellman: Dynamic programming. Princeton University Press, 1957.

[HoO] D. Z. Du; P. M. Pardalos; W. Wu: History of Optimization. Encyclopedia of Optimization. Boston: Springer, 2008. pp. 1538–1542. DOI

[Birds] D. D. Chin; D. Lentink: How birds direct impulse to minimize the energetic cost of foraging flight. American Association for the Advancement of Science, 2017. DOI

[Knapsack MBO] E. Ulker; V. Tongur: Migrating birds optimization (MBO) algorithm to solve knapsack problem. Procedia Computer Science, Volume 111, 2017, pp. 71-76. DOI

[HistofDP] R. Bellman; E. Lee: History and Development of Dynamic Programming. IEEE Control Systems Magazine, Vol. 4, 1984, pp. 24-28. DOI

[Caginalp-Balenovich] G. Caginalp; D. Balenovich: A theoretical foundation for technical analysis. Journal of Technical Analysis. 59, 2003, pp. 5–22.

[Lui-Chong] K.M. Lui; T. T. L. Chong: Do Technical Analysts Outperform Novice Traders: Experimental Evidence. Economics Bulletin, AccessEcon, vol. 33(4), 2013, pp. 3080-3087.

[Park-Irwin] C-H. Park; S. Irwin: The Profitability of Technical Analysis: A Review. AgMAS Project Research Report No. 04, 2004. DOI

[Lo-Mamaysky-Wang] A.W. Lo; H. Mamaysky; J. Wang: Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. The Journal of Finance, Vol. 55, 2000, pp.1705-1765. DOI

[ip-MACD] Moving Average Convergence-Divergence. Investopedia, Investopedia LLC. Dostupné na https://www.investopedia.com/terms/m/macd.asp (prístup 7.7.2021)

[ip-RSI] Relative Strength Index. Investopedia, Investopedia LLC. Dostupné na https://www.investopedia.com/terms/r/rsi.asp (prístup 7.7.2021)

[ip-EMV] Ease of Movement. Investopedia, Investopedia LLC. Dostupné na https://www.investopedia.com/terms/e/easeofmovement.asp (prístup 7.7.2021)

[Data src] Databáza Yahoo Finance. Dostupné na https://finance.yahoo.com/quote/AAPL/history/ (prístup 18.4.2021)

[Stoll] H. R. Stoll: The supply of dealer services in securities markets. The Journal of Finance 33.4, 1978, pp. 1133-1151.

[github] Dostupné na https://github.com/cyberklezmer/marketsim (prístup 7.7.2021)