Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra aplikované fyziky a techniky

Moderní trendy v oboru počítačová fyzika

Diplomová práce

Vedoucí práce: RNDr. Petr Bartoš, Ph.D.

Autor: Bc. Radek Surynek

Anotace

Diplomová práce nabízí přehled základních moderních metod, které mohou být využity v oblasti počítačové fyziky. Jedná se konkrétně o paralelní výpočty, neuronové sítě, genetické algoritmy a fuzzy logiku. V každé kapitole je uveden teoretický popis metody, zjednodušené matematické vyjádření, návrhy technické realizace a stručně jsou zmíněny i konkrétní aplikace. Text je doplněn řadou jednoduchých příkladů. Závěr práce shrnuje získané poznatky a nastiňuje budoucí vývoj.

Klíčová slova

Paralelní výpočty, neuronové sítě, neuron, perceptron, genetické algoritmy, evoluční algoritmy, fuzzy logika, fuzzy množiny.

Abstract

The theme of the thesis is to make a list few fundamental modern methods which can be used in computerized physics. The thesis describes parallel computing, neural networks, genetic algorithms, fuzzy logic. Every chapter include theoretical description, simplified mathematical expression, proposals of technical solution. Applications are briefly mentioned here too. The printed matter is completed with a few simple examples. The closing part of the thesis acquired information about these methods and outlines their future development.

Key words

Parallel computing, neural networks, neural networks, neurone, perceptron, genetic algorithms, evolutionary algorithms, fuzzy logic, fuzzy set theory

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně, pouze s použitím pramenů a literatury, uvedených v seznamu použitých pramenů a literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě archivované Pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 3. 1. 2013 Podpis:

Obsah

Úvod

Fyzika využívá v současnosti ke studiu přírodních jevů tři základní přístupy – teoretický, experimentální a poměrně novou oblastí je využití postupů počítačové fyziky. Experimentální přístup zkoumá fyzikální jevy pomocí experimentů. Získává tak cenná data, která jsou následně využívána pro formulování teoretických hypotéz, které jsou následně znovu ověřovány pomocí navržených experimentů. Symbióza teoretického a experimentálního přístupu vedla k vybudování systému poznatků, na kterých stojí stávající fyzika. S prudkým rozvojem počítačů, ke kterému dochází v posledních letech, hraje však stále významnější roli také třetí přístup – počítačové modelování. Pomocí sofistikovaných algoritmů je možné na počítači simulovat průběh fyzikálních jevů, které není možné studovat experimentálně, pomocí počítače je možné řešit složité soustavy rovnic, na jejichž řešení nestačí stávající analytické metody, pomocí počítače je možné řídit a vyhodnocovat komplikované experimenty.

Důvodů, proč má dnes počítačová fyzika tak významné postavení, je několik. Jedná se například o možnost snadné změny podmínek systému a možnost studia jejich vlivu na studovaný jev, dále o možnost snadného opakování téhož experimentu nebo o možnost studovat systémy, které bychom za normálních podmínek studovat nemohli. Nezanedbatelným důvodem, především v oblasti aplikovaného výzkumu, je výrazné snížení nákladů na výzkum a vývoj. Zařídit specializované pracoviště se všemi přístoji a pomůckami dle příslušných norem může být drahé. Samotný počítač je levnější, navíc více univerzální přístroj.

Z důvodu snadnějšího porozumění problematice je v této práci matematický aparát redukován na minimální možnou míru, definice a věty jsou mnohdy upraveny tak, že jsou méně obsáhlé (tudíž tedy i méně přesné), ale vystihují klíčové poznatky matematického popisu. Vzhledem k bouřlivému rozvoji a rychlosti vydávaných prací není tato práce zaměřena na prezentaci nejnovějších výsledků popisovaných metod, ale spíše na vysvětlení základních pojmů, pochopení různých operací s nimi a aplikací v pestrých oblastech lidské činnosti.

1. Rozvoj hardwarových a softwarových prostředků pro potřeby počítačové fyziky

Pro řešení úloh pomocí postupů počítačové fyziky musíme mít k dispozici odpovídající vybavení – jak výpočetní techniku (hardware), tak i prostředky softwarové (vhodný programovací jazyk, operační systém, atd.). Zcela nezbytné jsou také znalosti, jak tyto prostředky použít k vyřešení našeho problému [1].

Hardware

Rozvoj hardwaru tolik nutného k počítačové fyzice nastal již v průběhu 2. světové války, kdy vývoj počítačů tzv. nulté generace probíhal souběžně s vývojem šifrovacích a dešifrovacích strojů. Tyto počítače fungovaly na principu relé. V roce 1944 nastal ve vývoji určitý zlom, kdy tým vědců, do něhož patřil i významný matematik John von Neumann, sestavil ENIAC – počítač obsahující elektronky. Výkony těchto prvních počítačů byly přes jejich velikost poměrně malé, měly velký odběr el. energie a musely být chlazeny leteckými motory. Přesto znamenaly pro fyziku a vědu obecně ohromný pokrok. Roku 1946 představil John von Neumann vlastní koncepci počítače vyznačující se společnou pamětí pro data i instrukce. Tato koncepce se používá dodnes a byla jedním s milníků počítačové fyziky a informatiky vůbec. Dalším milníkem byl výzkum polovodičů a objev tranzistorového jevu, jež dotáhl roku 1951 do úspěšných aplikací známý americký fyzik Wiliam Shockley. Výhody využití těchto objevů ke konstrukci počítačů byly nasnadě, menší příkon zároveň s miniaturizací obvodů přinesl větší výpočetní výkon a nižší náklady. Poslední důležitý milník se udál v roce 1958 sestavením prvního integrovaného obvodu Jackem Kilbym z firmy Texas Instrument. Tento objev umožnil výraznou miniaturizaci obvodů a vedl ke vzniku osobních počítačů, které se následně rozšířily i mezi veřejnost. Od té doby se miniaturizace stala ústředním motivem počítačů. Součástky se vyrábějí stále menší a menší, což vede ke snížení jejich příkonu a také je možno integrovat na obvod mnohem více součástek (stále platí Moorův zákon). Narůstá jejich výkon a díky masovému rozšíření počítačů do většiny oblastí lidské činnosti, tedy zvětšení výrobních kapacit, dochází i ke snižování cen počítačů.

Softwarové prostředky

Přes stále lepší konstrukci hardwaru počítače bylo neméně významné nalézt způsob, jak počítač jednoduše obsluhovat. Vždy se mluví o dvou typech efektivity, které spolu vzájemně souvisí. První typ je efektivita obsluhy počítače, aneb jak náročné je zadat počítači

daný úkol. Druhý typ efektivity souvisí se samotným řešením úkolu počítačem, jak dlouho mu vyřešení úkolu bude trvat. Čím více se bude blížit zadávání úkolů jazyku počítače (strojovému kódu), tím bude efektivita samotného řešení (např. výpočtu) vyšší. Naopak čím více se bude blížit zadávání úkolů jazyku člověka, tím bude efektivita samotného řešení nižší. Jelikož vytváření programu ve strojovém kódu je mimořádně náročné, od počátku rozvoje výpočetní techniky byla snaha tuto činnost programátorům ulehčit – uživatelům ponechat pouze tvůrčí práci a mechanickou činnost přenechat počítači. To vedlo k vytvoření programovacích jazyků [1]. Mezistavem bylo zavedení tzv. assemblerů, tj. jazyků symbolických adres či jazyků symbolických instrukcí. Není to ještě programovací jazyk, ale zrychluje psaní ve strojovém kódu. Dnes se stále ještě používá např. u mikropočítačů (např. Intel 8051). O programovací jazyk se jedná tehdy, když jedné instrukci programu obecně odpovídá více instrukcí ve strojovém kódu [1]. Programovacích jazyků se vyvinula celá řada, jazyky používané v počítačové fyzice (ale nejenom tam) jsou např. C, Scilab nebo Fortran. Pro technické aplikace je velice populární například program MATLAB.

2. Paralelní výpočty

2.1 Úvod

Druhá kapitola je věnována paralelním výpočtům a paralelnímu zpracování dat obecně. Jedná se o jednu z nejstarších myšlenek, jak zvyšovat výpočetní výkon, tudíž i dobře popsanou a zpracovanou metodu. Existuje celá řada aplikací, např. i náš mozek zpracovává informace paralelně. Struktura této kapitoly vychází z teoretických poznatků o paralelizaci, které jsou zde shrnuty v zákonech Amdahla a Gustafsona. Na tuto část jsou navázány poznatky z architektury systémů, programování a základní metody konstrukce paralelních výpočetních úloh. Závěr kapitoly se věnuje aplikacím, kde jsou stručně popsány projekty využívající paralelizaci a paralelních systémů, které by měl každý fyzik znát.

2.2 Teorie paralelního zpracování

Počítačová fyzika, stejně jako ostatní obory používající pro svou práci počítač, řešila již od prvopočátku problém nedostatečného výpočetního výkonu počítačů. I dnes je tento problém stále aktuální. Využití paralelizace byla jednou z prvních myšlenek, jak zvýšit výkon počítačů. Vývoj teorie i realizace provázely určité překážky, které se však podařilo překonat a dnes již hovoříme o velmi rozšířené technice, jenž udává směr ve výpočetním zpracování dat.

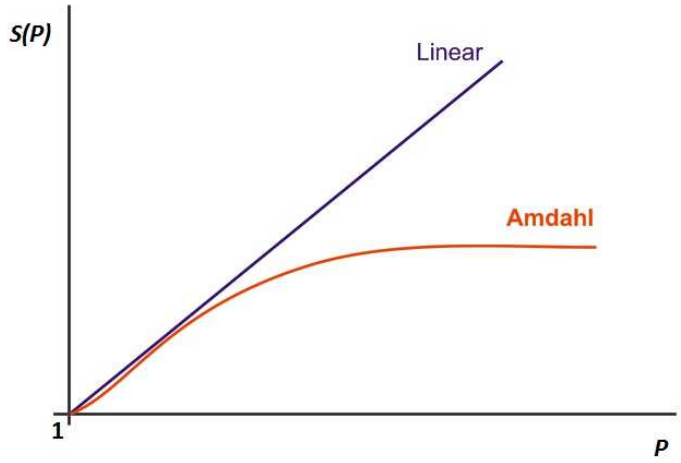
Amdahlův zákon

Gene Amdahl publikuje v roce 1967 zákon, podle kterého zrychlení výpočtů, kterého lze dosáhnout využitím paralelizace kódu na více výpočetních jednotkách, má jisté omezení. Uvádí, že vždy bude existovat část kódu, která bude prováděna sekvenčně, a v tu chvíli bude pracovat pouze jedna výpočetní jednotka a ostatní na ni budou čekat. I přes zvyšující se počet výpočetních jednotek bude toto omezení stále platit. Dosažené zrychlení *S(P)* lze definovat takto [2]:

$$S(P) = \frac{T_S}{T_P} = \frac{W/R}{W\left(\frac{\alpha}{R} + \frac{1-\alpha}{PR}\right)} = \frac{P}{1+\alpha(P-1)}. \qquad (1)$$

T_S a T_P jsou doby sekvenčního a paralelního řešení, W značí výpočetní práci, R je výkon procesoru, αW je část úlohy, kterou nelze paralelizovat, $(1-\alpha)W$ je pak část úlohy, kterou paralelizovat lze. P označuje počet procesorů. S rostoucím počtem procesorů

je přírůstek zrychlení stále menší a zrychlení se limitně blíží k hodnotě $\frac{1}{\alpha}$ [2]. Průběhy zrychlení zobrazuje obr. 1, upravený na základě [2].



Obr. 1: Průběhy lineárního a Amdahlova zrychlení

Toto omezení, které se promítlo do vývoje i realizací paralelních systémů narušil až roku 1988 John Gustafson, který přišel s myšlenkou, že část úlohy, která se zpracovává paralelně nemusí být nutně konstatntní, jako tomu je u Amdahlova zákona, ale může lineárně růst s počtem procesorů [2]:

$$S(P) = P - \alpha(P-1) \,. \qquad (2)$$

Zrychlení v případě Gustafsonova modelu nevykazuje nasycení jako v případě Amdahlova modelu [2]. Gustafson ve své práci zamýšlí, že zrychlení výpočtu nedosáhne pomocí časové úspory, nýbrž rozšířením problému, kdy se navýší právě ta část kódu (či dat), která má být zpracována paralelně na více procesorech [3]. Gustafsonovým zákonem se podařilo obejít hlavní překážku v podobě Amdahlova zákona a paralelní systémy se rozvinuly až do dnešní podoby.

2.3 Typy paralelních systémů

V současnosti dochází ke značnému rozšíření paralelních systémů a rychlému vývoji různých systémových konstrukcí. Tento trend přináší problémy škálovatelnosti různých typů paralelních systémů, přesto jej lze přibližně zařadit do určitých tříd, které definoval Flynn již roku 1966 [4]:

Paralelní systémy lze třídit dle počtu toků instrukcí a dle počtu toků dat na:

SI (Single Instruction stream) – systém s jedním tokem instrukcí

MI (Multiple Instruction stream) – systém s několika toky instrukcí

SD (Single Data stream) – systém s jedním tokem dat

MD (Multiple Data stream) – systém s několikanásobným tokem dat

Kombinací různých systémů toků instukcí a dat lze definovat 3 základní systémy:

SISD – jednu instrukci provádí jedna výpočetní jednotka,

SIMD – jednu instrukci provádí více výpočetních jednotek,

MIMD – více různých instrukcí provádí více výpočetních jednotek.

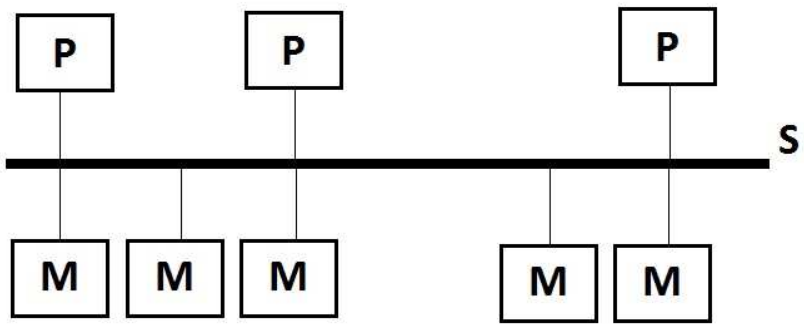
Systému SISD odpovídá klasická architektura (Von Neumann), systém SIMD představuje představuje vektorové a maticové procesory, u systému MIMD se jedná o tzv. **multiprocesor** [5]. Multiprocesorů je více druhů, zde zmíníme pouze dva základní:

a) Symetrický multiprocesor se sdílenou pamětí

b) Multiprocesor s distribuovanou pamětí

Symetrický multiprocesor se sdílenou pamětí

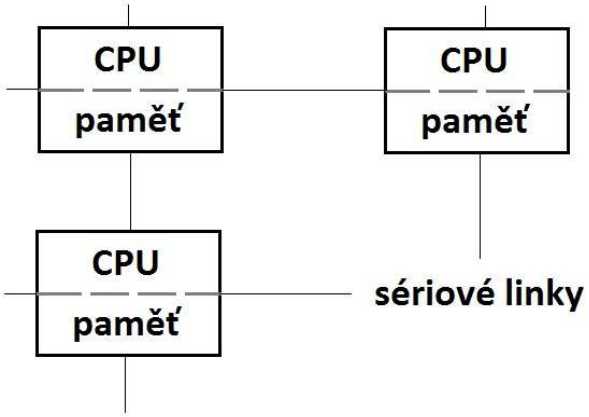
Jedná se o skupinu stejných typů procesorů, žádný není zvýhodňován, mají stejné možnosti přístupu ke zdrojům. Procesory jsou vzájemně redundantní. Systém váže data i kód v paměťových modulech. Procesory přistupují k datům skrze sběrnici. Paměťových modulů bývá více než procesorů, což zvyšuje celkovou propustnost [6]. Výhody tohoto systému jsou spolehlivost, univerzálnost a jednoduchost adresace. Za nevýhodu je možné považovat slabý výpočetní výkon, neboť všechny procesory přistupují k datům přes jednu sběrnici, ta je tímto námáhána (až přetěžována) a představuje i fyzické omezení při přidávání dalších procesorů do systému. Schéma této architektury znázorňuje obr. 2, upravený na základě [6]. **P** znamená procesor (obecněji výpočetní jednotka), **M** vyjadřuje paměťový modul a **S** označuje sběrnici.



Obr. 2: Schéma multiprocesoru se sdílenou pamětí

Multiprocesor s distribuovanou pamětí

Jedná se o skupinu výpočetních jednotek, kdy každá výpočetní jednotka (např. procesor) obsahuje i svoji lokální pamět. Neexistuje nějaká sdílená paměť. Výpočetní jednotky mezi sebou komunikují po sériových linkách. Je možno vytvořit různé typy takovýchto sítí, kdy se daný typ sítě hodí k řešení určitého typu úloh. Dle typu sítě je třeba vstupní – jednotná data výpočtu rozdělit do jednotlivých lokálních pamětí výpočetních jednotek; obdobně to platí i u procesů. Výhodou je poměrně velký výpočetní výkon (pro zcela určitý typ úlohy a typ sítě). Nevýhodu představuje potřeba jednoznačného zaměření systému na daný typ úloh, tudíž nízká univerzálnost. Schéma této architektury znázorňuje obr. 3, upravený na základě [5].



Obr. 3: Schéma multiprocesoru s distribuovanou pamětí

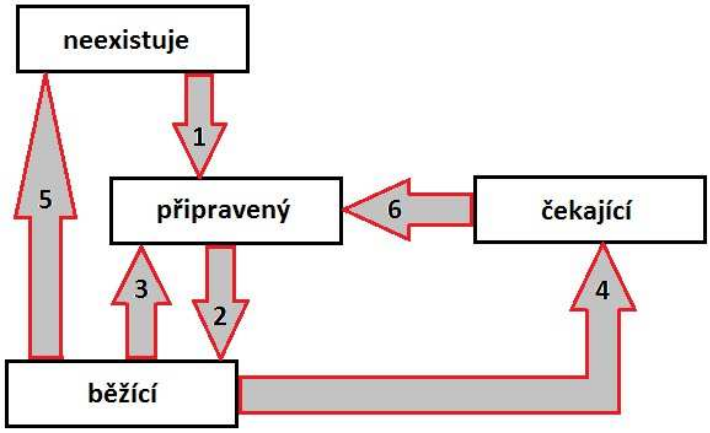
2.4 Paralelní procesy

Pojem „proces“ lze popsat jako blok instrukcí, jinými slovy sekvenční posloupnost příkazů, jež realizuje samotný procesor. Počítač s tímto blokem pracuje jako s dále nedělitelným prvkem. Je to instance programu zavedená do operační paměti počítače. Pojem „paralelní procesy“ bude v této práci úzce určen pro procesy, které v daný čas běží souběžně na více procesorech (výpočetních jednotkách).

Každý proces obsahuje dvě entity, **program** dle kterého je proces realizován a **data procesu**. Data procesu obsahují jednak konkrétní (lokální) data, se kterými daný proces pracuje, dále informace o stavu procesu. S procesy pracuje operační systém, který pomocí určitého řídícího algoritmu spravuje práci s procesy. Tedy dodává procesoru (či procesorům) procesy dle pořadí, které vytváří a upravuje s ohledem na priority řídícího algoritmu a zdroje,

které dané procesy ke své realizaci potřebují. **Process control block** je datová oblast řídícího algoritmu, která obsahuje informace o stavech procesů, slouží zejména k samotné řídící funkci. Další využití se týká **přepnutí kontextu (contex switch)**. Jestliže řídící algoritmus vyhodnotí, že běžící proces bude přepnut, tak se zde uloží průběžná data procesu, neboť při opětovném přepnutí na původní proces, může původní proces pokračovat v práci bez přerušení.

Z hlediska řídícího algoritmu paralelního výpočtu se rozlišují pouze makrostavy procesů. Typické makrostavy a přechody mezi nimi ukazuje obr. 4. , převzatý z práce [2].



Obr. 4: Diagram stavů procesu a přechodů mezi nimi

Popis přechodů mezi stavy:

1 – vytvoření procesu

2 – spuštění procesu

3 – přerušení procesu

4 – proces nemá k dispozici požadovaný zdroj (obecně není splněna nějaká podmínka)

5 – proces došel na konec svého programu

6 – proces dostal k dispozici požadovaný zdroj (obecně je splněna nějaká podmínka).

2.4.1 Deadlock

V určitých úlohách dochází k tomu, že paralelní procesy společně sdílejí zdroje, může se jednat například o proměnnou, registr či vstupní / výstupní zařízení. Většinou si řídící algoritmus s těmito nároky poradí a procesy se zdárně ukončí. Nejen v paralelních systémech ale může nastat stav, kdy dva i více procesů na sebe vzájemně čekají a tím dojde k zablokování celého výpočtu, anglicky deadlock. Tento stav nastává za určitých podmínek:

1) K určitému zdroji může mít přístup v určitý čas pouze jeden proces

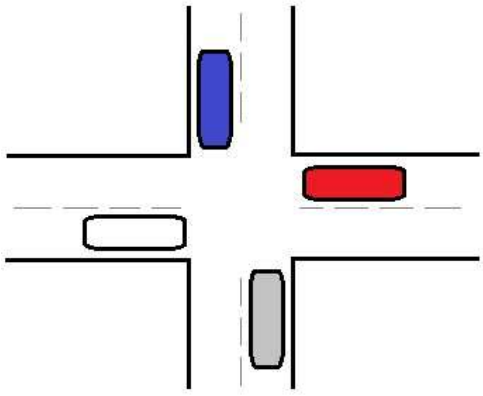
2) Proces, který již má nějaké zdroje přiděleny, může žádat o další

3) Řídící algoritmus procesů (obvykle jádro OS) nemá možnost ukončit běžící proces, proces sám vrací používané zdroje

4) Procesy musí čekat a vzájemně držet své zdroje do kruhu

Symbolicky si lze deadlock představit na obrázku obr. 5, kde je zobrazena křižovatka bez značení. Každý automobil (proces) se řídí pravidlem pravé ruky (vyžaduje splnění podmínky nebo čeká na přidělení zdroje). Všechny automobily (procesy) vzájemně čekají dle pravidla pravé ruky na uvolnění některé z cest (splnění podmínky), nastává tedy deadlock.

Deadlock lze řešit buď na úrovni řídícího algoritmu (OS), kdy se schválně narušují výše zmíněné podmínky deadlocku nebo jej lze řešit na programové úrovni, kdy je daný program napsán tak, aby deadlock nemohl nastat – vhodnost přidělování zdrojů a čekání na podmínky. V určitých případech se deadlock neřeší, např. kdyby vyřešení deadlocku zabralo více času než přerušení a restartování procesů.



Obr. 5: Deadlock - křižovatka

2.4.2 Send, Take

Velmi často bývá vhodné, aby paralelní procesy, které řeší určité části úloh, mohli vzájemně spolupracovat. Například si mohou předávat informace o svém stavu, sdílet, porovnávat či předávat výsledky, nebo na ně určitým způsobem reagovat, apod. Je tedy důležité, aby procesy mohly spolu nějakým způsobem komunikovat. Komunikaci lze buď řešit na úrovni operačního systému – nízkoúrovňová komunikace nebo na úrovni programovacího jazyka – vysokoúrovňová komunikace.

Nízkoúrovňová komunikace se provádí buď asynchronně (procesy komunikují skrze vyrovnávací paměť, kde si nechávají zprávy) nebo synchronně (procesy se komunikace přímo účastní) [5]. Dále lze komunikaci procesů dělit dle způsobu adresace na [5]:

a) **Symetrickou adresaci** – obsahuje jména obou komunikujících procesů

b) **Asymetrickou adresaci** – v odeslané zprávě se úvádí pouze jméno příjemce

c) **Nepřímou adresaci** – neuvádí se žádná jména procesů, komunikace se provádí přes speciální kanál, známý oběma procesům

Základní operace pro nepřímou adresaci jsou SEND a TAKE, mají tento tvar [5]:

SEND (kanál, zpráva) - zpráva obsahuje adresu v lokální paměti odesílajícího procesu
TAKE (kanál, zpráva) - zpráva obsahuje adresu v lokální paměti přijímacího procesu

2.4.3 Monitor, Rendezvous

Jsou operace vysokoúrovňové komunikace, tedy na úrovni programovacího jazyka. Důvod jejich zavedení spočívá v určitém nebezpečí použití operací nízkoúrovňové komunikace, kdy překladač nemusí objevit logické chyby programu. Samotný kód komunikace může být syntakticky správně napsán, ale vykonané příkazy mohou vést k nechtěným stavům, například deadlocku či různým typům chyb.

Monitory jsou vyšší jazykovou konstrukcí pro implementaci vzájemně výlučného přístupu ke sdíleným zdrojům, obsahují sdílená data a procedury zajišťující přístup k nim [5]. Požadavků procesů může být v proceduře obecně více. Procedury mohou aktivovat požadavky procesů (frontou typu FIFO: *first in, first out*) nebo mohou požadavky procesů uspávat. Monitory se používají v paralelním multiprocesorovém systému se sdílenou pamětí [5].

Podstatou **Rendezvous** je společné provedení určitého úseku programu dvěma procesy, kdy nosné téma vychází z myšlenky, že předávání dat mazi procesy a jejich synchronizace jsou neoddělitelné aktivity [5]. Procesy na sebe vzájemně čekají, poté synchronizovaně provedou komunikační akce a dále pokračují samostatně ve své práci [5].

2.5 Paralelní výpočty

Abychom mohli paralelně zpracovávat data, je potřeba o tuto myšlenku rozšířit všechny tři předpoklady počítačové fyziky: harware, software a znalosti. Až vzájemným propojením

těchto tří hledisek lze úspěšně aplikovat paralelní výpočty na konkrétní výpočetní úlohy. Přizpůsobení hardwarové architektury bylo popsáno v části 2.3. (Typy paralelních systémů), několik zásadních pojmů z hlediska programování bylo uvedeno v části 2.4. (Paralelní procesy). Knihovny pro paralelní zpracování dat obsahuje řada programovacích jazyků, např. C, C++, Java, Fortran, MATLAB a další. Nyní bude následovat rozšíření myšlenky paralelizace na úrovni znalostí. Budou zde popsány určité cesty, jak lze myšlenkově provést paralelizaci úloh, případně výpočtů.

Výpočty lze paralelizovat na několika úrovních [7]:

1. na úrovni bitového kódu
2. na úrovni instrukcí
3. na úrovni dat
4. na úrovni úloh

Paralelizace na úrovni bitového kódu je založena na možnosti zvýšení počtu bitů, se kterými počítač pracuje najednou. Tím lze dosáhnout sníženého počtu instrukcí, jež musí počítač vykonat.

Paralelizace na úrovni instrukcí spočívá v tom, že ty instrukce, které jsou na sobě v danou chvíli nezávislé, lze vykonat najednou. Např. máme tři proměnné ***x***, ***y***, ***z***, se kterými se provádí danné instrukce:

$x = x + 1$	*První dvě instrukce jsou navzájem nezávislé, lze je počítat*
$y = y - 5$	*najednou - paralelně. Třetí instrukce musí čekat na před-*
$z = x + y$	*chozí dvě, neboť je závislá na prvních dvou instrukcích.*

$x = x + 1$	
$x = x + 3$	*Žádné z instrukcí nelze počítat paralelně, neboť všechny*
$y = 2x$	*tři jsou vzájemně na sobě závislé.*

O tomto typu paralelizace se také nechá mluvit jako o modelu MPMD (Multiple Program Multiple Data). Tuto paralelizaci je vhodné použít pro úlohy s komplikovanějším kódem a menším množstvím dat.

Paralelizace na úrovni dat se realizuje tak, že každá výpočetní jednotka (jádro, procesor nebo i celý počítač) provádí stejnou úlohu se stejným kódem nad různými částmi dat [7]. Často však má jedna jednotka řídící roli a koordinuje celý průběh výpočtu,

nazýváme ji **MASTER**. Ostatní jednotky se ji podřizují, říkáme jim SLAVE, ty obvykle vykonávají stejnou úlohu, mají tedy obdobný zdrojový kód. Například máme vzestupně seřadit čísla [5, 2, 7, 3, 1, 9, 8, 4, 6] libovolným řadícím algoritmem. Můžeme použít např. jednoduchou řadící metodu bubble sort, kdy se budou porovnávat dva sousední prvky a za podmínky, že nejsou ve správném pořadí (tedy $x_1 > x_2$), si prohodí místo. Úlohu můžeme rozdělit do více jednoduších úloh, které se budou řešit najednou – paralelně např. na třech výpočetních jednotkách.

Příkaz 1: MASTER dělí 1 pole devíti prvků na 3 pole po třech prvcích;

Příkaz 2: MASTER přiřadí každému ze tří SLAVŮ 1 pole o třech prvcích;

Příkaz 3: Každý SLAVE seřadí pomocí buble sort 1 pole o třech prvcích;

Příkaz 4: MASTER složí 3 pole o třech prvcích v 1 pole o devíti prvcích;

(může např. z důvodu lepší uspořádanosti seřadit 3-prvková pole dle součtu jejich prvků)

Příkaz 5: MASTER seřadí 1 pole o devíti prvcích pomocí bubble sort;

[5, 2, 7, 3, 1, 9, 8, 4, 6] => [5, 2, 7], [3, 1, 9], [8, 4, 6]

[5, 2, 7], [3, 1, 9], [8, 4, 6] => [2, 5, 7], [1, 3, 9], [4, 6, 8]

součet[2, 5, 7] = 14, *součet*[1, 3, 9] = 13, *součet*[4, 6, 8] = 18

[2, 5, 7], [1, 3, 9], [4, 6, 8] => [[1, 3, 9], [2, 5, 7], [4, 6, 8]]

[[1, 3, 9], [2, 5, 7], [4, 6, 8]] => [1, 3, 9, 2, 5, 7, 4, 6, 8]

[1, 3, 9, 2, 5, 7, 4, 6, 8] => [1, 2, 3, 4, 5, 6, 7, 8, 9]

Ze sledu příkazů výše si lze všimnout, že příkazy, které vykonává MASTER, mají sekvenční charakter a příkazy, které vykonává SLAVE (**Příkaz 3**) mají charakter paralelní. Jedná se o typický rys tohoto typu paralelizace. V ideálním případě, kdy můžeme zanedbat časové nároky komunikace mezi výpočetnímy jednotkami, je výpočetní čas přímo úměrný velikosti dat a nepřímo úměrný počtu výpočetních jednotek [7]. Velmi úspěšné použití tohoto typu paralelizace je dosaženo ve výpočetních úlohách s maticemi, kdy jednotlivé řádky či sloupce matic lze počítat paralelně (SLAVE) a výslednou matici na konci složit z těchto mezivýpočtů (MASTER).

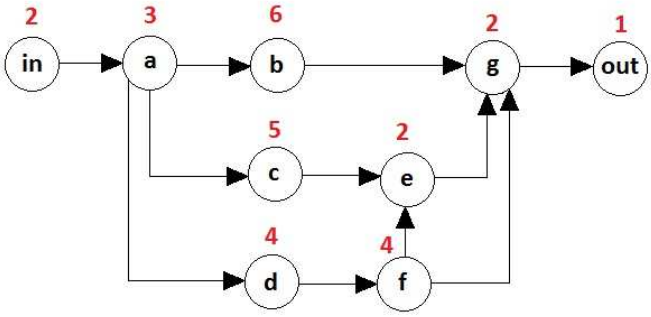
O tomto typu paralelizace se také nechá mluvit jako o modelu SPMD (Single Program Multiple Data). Tuto paralelizaci je vhodné použít pro úlohy s jednoduším kódem a větším množstvím dat.

Paralelizace na úrovni úloh vychází z předchozího typu paralelizace, přičemž ale nemusí být přesně stanovena role MASTER/SLAVE či může být i rolí více, stejně tak každá výpočetní jednotka může vykonávat jiné úlohy s odlišným kódem. Z toho také vyplývá, že čas výpočtu nemusí být přímo úměrný velikosti problému a nepřímo úměrný počtu výpočetních jednotek [7].

Precedenční graf

Jedná se o model paralelních výpočtů, který je zaměřen na zrychlení výpočtů použitím paralelizace. Uzly grafu reprezentují jednotlivé procesy a hrany představují komunikaci mezi procesy [5]. U každého uzlu je číslo, které vyjadřuje dobu, jak dlouho bude trvat vykonání určité uzlu (procesu). Dobu sekvenčního výpočtu lze z grafu zjistit sečtením všech čísel uzlů (tedy sečtením všech dob jednotlivých procesů) [5]. Dobu paralelního výpočtu lze z grafu zjistit sečtením čísel uzlů při nejdelší možné cestě, (za podmínky, zda-li je k dispozici potřebný počet procesorů.

Příklad:



Obr. 6: Precedenční graf

Výsledná doba sekvenčního výpočtu = 2 + 3 + 6 + 2 + 1 + 5 + 2 + 4 + 4 = 29

Výsledná doba paralelního výpočtu = 2 +3 + 4 + 4 + 2 + 2 + 1 = 18

Obecně můžeme vyjádřit maximální možné urychlení výpočtu při paralelním zpracování takto [1]:

$$\frac{\sum časů\ všech\ procesů}{\sum časů\ nejdelší\ cesty}. \qquad (3)$$

2.6 Aplikace

Aplikací paralelních výpočtů v ruzných technických oborech je velmi mnoho. Zde se zaměříme na paralelní řešení výpočtů, které se týkají počítačové fyziky. Z hlediska výkonu, snadnosti přístupu i finanční stránky výpočtů je nejzajímavější použití výpočetních clusterů a gridů. To bude právě obsahem této podkapitoly.

Výpočetní cluster

Výpočetní cluster je síťové propojení výpočetních jednotek (obvykle počítačů), které slouží především k paralelním výpočtům typu SPMD, tedy např. práce s maticemi, faktorizace na prvočísla, simulace vývoje počasí, numerické řešení parciálních diferenciálních rovnic, SETI program. Výhody jsou: velký výpočetní výkon za cenu nižší než u obdobně výkoného superpočítače, poměrně snadný přístup k velkému výpočetnímu výkonu. Nevýhody jsou: pomalá komunikace mezi procesy (proto vyšší efektivita výpočtů typu SPMD, kdy se využívá lokálních dat), dále nutnost předem promýšlet priority a pořadí výpočtů – nutnost fronty, kde se čeká na přiřazení zdrojů; dále sdílení dat.

Grid

Nejjednodušeji se lze dívat na Grid jako na dynamický, virtuální, výpočetní, informační nebo znalostní systém, tj. soustavu především výkonných počítačů propojených vysoko-rychlostní sítí, určených pro řešení nejnáročnějších výpočetních a datových problémů [8]. Grid je jakási nadstavba clusteru, je to výpočetní řešení obecně na vyšší úrovni. Obvykle se skládá z více různých clusterů. Do gridu jsou připojeny i superpočítače, každý obvykle provádí úzce zaměřené výpočty. Dle typu výpočtu lze na gridu přiřadit právě vhodné typy superpočítačů či clusterů, které daný typ úlohy budou řešit co nejefektivněji. Pro gridy je typické, na rozdíl od clusterů, že se jeho výpočetní struktura často mění. V poslední době dochází za podpory našeho státu (MetaCentrum) i EU (VOCE) k masivnímu rozšíření gridů.

MetaCentrum

Projekt MetaCentrum se roku 1999 začlenil do skupiny CESNET, výzkumné skupiny založené vysokými školy a AV ČR. Heslo „***síť pro výzkum & výzkum pro síť***“- na jejich webových stránkách jasně popisuje práci této skupiny. Samotné MetaCentrum má řadu cílů, které poměrně úspěšně plní. Dlouhodobým cílem je provoz a koordinace distribuované výpočetní infrastruktury a datových uložišť a odpovídajícího podpůrného prostředí umožňující využití dostupných výpočetních zdrojů pro řešení velmi náročných výpočetních úloh [9].

3. Neuronové sítě

3.1 Úvod

Třetí kapitola je věnována neuronovým sítím, respektive realizací umělých neuronových sítí. Myšlenka napodobování přírody s konstrukcí informačních systémů obdobných u organismů je poměrně stará, první práce se začaly vytvářet již v průběhu 2. světové války. Toto téma je poměrně složité, neboť k úspěšnému pochopení požaduje značné kvantum znalostí především z oborů biologie, psychologie, matematiky a informatiky. V případě určitých nedostatků z již zmíněných oborů dochází k funkčnímu omezení tématu neuronových sítí jako celku, kdy dané omezení nezpůsobuje pouze horší pochopení práce umělých neuronových sítí, ale omezuje i pestrost funkcionality těchto informačních systémů, což se jistě musí projevit také v aplikačním použití. Tento fakt se promýtl i do struktury této kapitoly. Obsahuje tedy základní biologický popis neuronu, neuronových sítí a informačních systémů organismů, část práce je věnována matematickému popisu, respektive určité aproximaci těchto biologických poznatků. Mimo jiné jsou zde stručně popsány i metody učení jak na bázi biologicko-psychologické, tak na bázi technické. Jsou zde uvedeny základní metody realizací umělých neuronových sítí a aplikační možnosti. Aplikace jsou popsány spíše na úrovni určitých typů aplikací.

3.2 Informační systémy organismů

Z hlediska biologického popisu neuronových sítí je důležité znát alespoň základy informačních systémů organismů. Nějaký informační systém si dnes (v informační době) umí asi každý z nás lehce představit. Tato představa by pravděpodobně obsahovala nějakou skupinu počítačů propojených sítí, které by určitými algoritmy zpracovávaly nějaké vstupní informace a určitým způsobem se poté zachovaly – např. by vytiskly výsledek, uložily data do paměti, daly příkaz k vydání horké kávy či nahuštění pneumatiky, apod.

Lze si ale takto jednoduše představit nějaký informační systém organismů, např. u člověka? A má vůbec člověk nějaký informační systém? Pokusíme se tedy představu klasického informačního systému aplikovat na člověka. Nejprve informační systém dostal vstupní data pravděpodobně na nějakém médiu (např. CD; flash, hard disk) či je zaslalo nějaké čidlo. Poté jej poslal formou elektrických impulsů ke zpracování. Lze tyto úkony nalézt u člověka? Zcela jistě, člověk má také čidla (zraková, sluchová, …), také má paměť a také tyto

informace zasílá formou elektrických impulsů. Dále u klasického informačního systému by procesor (či počítač) dle nějakého algoritmu informace zpracoval a systém by se poté určitým způsobem zachoval. Člověk nemá žádný procesor, leč příroda vyvinula jinou konstrukci, poměrně složitý procesor nahradila obrovským množstvím jednodušších prvků a tyto prvky vzájemně propojila. Výsledná síť těchto prvků zajišťuje nejen zpracování, ale i uchování dat. Výstupní reakce obou systémů už může být podobná. Vzhledem k množství vstupních informací (velké množství čidel člověka, požadavků orgánů, buněk, atd.) lze usuzovat, že výpočetní výkon informačních systémů člověka je obrovský. Rozdíl ve výkonu lze hledat i v charakteristickém rysu těchto systémů, tedy v metodě zpracování informací. Klasický informační systém potřebuje znát přesný algoritmus, u organismů nikoliv, zpracování se řeší jiným způsobem. Informační systémy organismů tedy skutečně existují, dokonce ty lidské jsou i z hlediska technického využití opravdu zajímavé. Obecně je možno konstatovat, že existence a adekvátní funkce informačního systému je základní podmínkou existence, přežití a rozvoje všech systémů vůbec, biologických, technických, ekonomických i společenských [10]. Proto tedy popis jejich částí, jejich struktur a vztahů mezi nimi představuje náplň této podkapitoly.

3.2.1 Centrální nervový systém

Informační systém člověka se skládá z více částí, které lze rozdělit do dvou skupin:

a) Centrální nervový systém
b) Periferní nervový systém

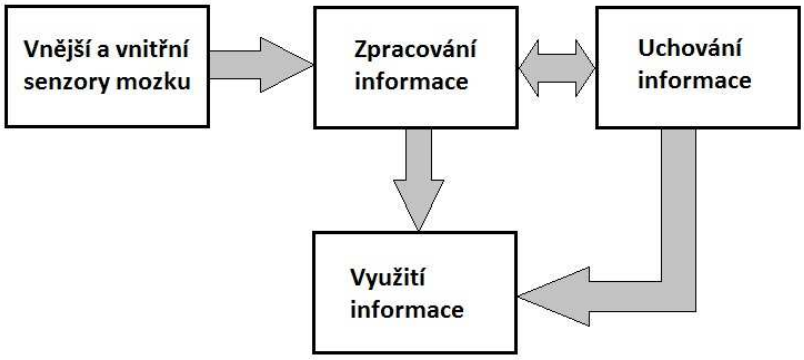
Centrální nervový systém (CNS) se skládá z mozku a míchy, periferní nervový systém představují nervy zajišťující činnost senzorickou (cit, bolest) nebo motorickou (pohyb svalů). Z hlediska informačního je cennější centrální nervový systém, který zajišťuje řízení celého těla, zpracovávání a uchovávání informací.

Mozek

Mozek je nejdůležitější částí celého nervového systému. Jeho struktura a funkce nejsou dosud přesně známy a stále jsou předmětem horlivého výzkumu, v němž mohou mimo jiné pomoci i umělé neuronové sítě. Ví se o něm, že obsahuje desítky miliard neuronů – základních nervových buněk, dále další desítky a stovky miliard dalších specializovaných buněk. Vše je vzájemně, různorodě propojeno, buňky a sítě buněk na sebe vzájemně působí a celý systém se v průběhu času značně vyvíjí. Samotný neuron není také dosud přesně

popsán, ale jeho struktura se jeví také jako velmi složitá, někdy je dokonce hovořeno o stejné složitosti v poměru k samotnému mozku [11]. Celková složitost takového systému je skoro nepředstavitelná.

Na obr. 7, upraveno na základě [10], si lze všimnout základního uspořádání mozku s příslušnými informačními procesy.



Obr. 7: Blokové schéma uspořádání mozku

Mozek plní řadu méně či více složitých funkcí, jejichž řízení zajišťují právě neuronové sítě. Tyto sítě jsou dále vzájemně propojeny a tvoří různé bloky a podbloky. Bloky si nelze představovat jako soustředěné, na jisté lokality mozku vázané biologické neuronové struktury, ale bližší je představa vzájemných velmi složitým způsobem propojených podbloků téhož bloku i bloků cizích [10]. Mezi všemi podbloky pak existují složité dopředné i zpětné vazby [10]. V řadě studií, popsaných v [10] či v [11], lze vyčíst, že výkonost mozku čili jeho funkcionální úroveň je závislá nejen na počtu základních prvků - neuronů (jak by se dalo očekávat), ale i na způsobu a složitosti propojení mezi těmito prvky. Tedy i velikostně menší mozek obsahující méně neuronů může být výkonější než mozek větší s více neurony. Např. ale z vývoje lidského druhu, kdy se mozek člověka postupně zvětšoval, lze usuzovat, že tato (ne)závislost má i určitá omezení.

3.2.2 Faktory ovlivňující CNS

Faktorů, jenž ovlivňují CNS je více. Můžeme je rozdělit na vnější a vnitřní. Vnější by byla např. dostatečné podráždění smyslů, receptorů, apod. Při dostatečně silné úrovni či frekvenci lze i trvale skrze změny v propojení neuronových sítí měnit CNS (např. posttraumatické stresové poruchy). Faktory vnitřní jsou z hlediska neuronových sítí zajímavější. Jedná se především o endokrinní a imunitní systém. Endokrinní systém spolu se systémem nervovým zajišťují řídící systém člověka. Systém nervový je rychlý (elektrické pulsy) a jeho řídící výstupy

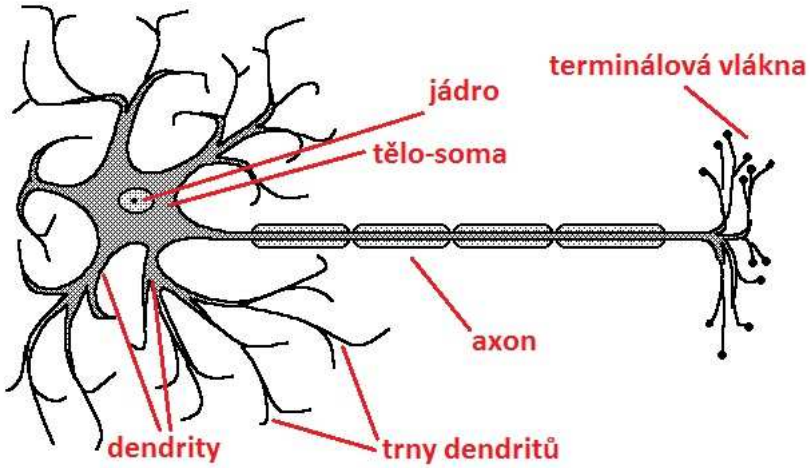
mají krátkou dobu trvání, kdežto systém endokrinní je pomalejší (pohyb látek - hormonů) a jeho řídící výstupy mají delší dobu trvání. Jako takové spolu určitými způsoby spolupracují a vzájemně se ovlivňují tak, aby tělo mohlo fungovat v ideálních vnitřních podmínkách. Každá buňka přijímá ze svého okolí informace, na které reaguje a mění své chování. Např. v práci [12] se hovoří i o tzv. neuroendokrinním systému, kdy provázanost systému nervového a endokrinního je prováděna na dvojí úrovni – lokalizovaně a difuzně. Lokalizovaný neuroendokrinní systém funguje na tom principu, že určité části mozku (např. hypotalamus) sekreují hormony do svého okolí a okolní buňky na něj reagují pozitivní nebo negativní zpětnou vazbou (lokálním zvýšením či snížením citlivosti určitých nervových spojů). Difůzní neuroendokrinní systém pak funguje na klasickém prinicipu vylučování hormonů ve více orgánech, kdy se krevním přenosem těchto látek ovlivňuje stav buněk CNS. Vzájemnou ovlivnitelnost nervového a imunitního systému lze pozorovat např. u leukocytů (bílých krvinek), o kterých je známo, že ovlivňují činnost některých neurotransmiterů (látek zajišťujících činnost nervových pulsů).

3.3 Neuron

Základním prvkem veškeré nervové struktury je nervová buňka, neboli neuron. Jedná se o buňku, která byla v průběhu evoluce „vyšlechtěna“ k rychlému přenosu, zpracování a uložení informací. Kromě neuronů obsahují nervové tkáně spoustu jiných buněk, které s neurony funkčně spolupracují, např. se jedná o buňky gliové, které zajišťují výživu, ochranu a spolehlivost celého nervového systému. Samotný neuron, jak již bylo řečeno, představuje velmi komplikovanou buňku, jejíž činnost je pro nás stále velká neznámá. Neuron je základním kamenem biologických neuronových sítí, stejně tak je tomu i u sítí umělých. To je také důvodem ke studiu této buňky, neboť čím lépe budeme znát popis, strukturu a činnost biologického neuronu, tím lépe budeme moci jeho funkce matematicky aproximovat a výsledná technická realizace bude přesnější k přírodě. Lze očekávat, že evoluční vývoj těchto informačních systémů, který trvá miliony let, bude mnohem efektivnější než několikadenní práce člověka-technika, proto se jeví inspirace přírodou jako vhodný postup i pro technické aplikace. Tato podkapitola je strukturovaná právě za tímto účelem, tedy napřed je neuron a činnosti s ním spojené popsán biologicky, aby poté došlo k matematické aproximace těchto poznatků. To nám umožní efektivní realizaci umělého neuronu.

3.3.1 Biologický popis neuronu

Neuron se skládá z těla (*soma* či *neurocyt*) a různých výběžků, viz obr. 8, upraveno na základě [13]. Velikost těla kolísá v širokých mezí od obrovských po nejmenší buňky v těle, tvar je ovlivněn počtem výběžků [25]. Tělo obsahuje jádro (zde je uloženo DNA), jadérko (zde je uloženo RNA), membrány a obsahuje např. i centrioly (části buňky potřebné k buněčnému dělení), přesto se neurony nerozmnožují a jejich zánik je trvalý – nenahrazují se novými neurony. Neuronů existuje mnoho druhů a dají se dělit do různých skupin např. dle tvaru, či počtu výběžků, typů spojení, apod.



Obr. 8: Popis neuronu

Výběžky lze dělit na dva typy a to dendrity a axon. **Axon** představuje výstup neuronu, je pouze jeden a jeho délka se pohybuje v rozmezí několika mikrometrů až asi 60 cm [10]. Axon bývá na svém konci silně rozvětven do tzv. terminálových vláken. **Dendrity** představují vstupy buňky, je jich více (řádově až tisíce) a jejich délka je maximálně 2-3 mm [11]. Neurony se spojují do neuronových sítí. Toto spojení ale není prováděno přímo (jako axon-dendrit), ale přes určitá biologická rozhraní (interface) nazývající se **synapse**. Na konci dendritů se během celého života organismu vytvářejí výstupky-trny, děje se tak v souvislosti se vznikem nových či aktivací starých, nefunkčních synaptických spojů. Synapse hrají v celém systému velmi důležitou roli a je jim věnována další podkapitola.

Funkce neuronu lze rozdělit do dvou skupin. První skupina, nazvěme ji např. životní, obsahuje funkce, které musí buňka zajišťovat k samotnému životu a ideálnímu pracovnímu stavu buňky. Jednalo by se především o příjmové a vylučovací funkce, kdy podstatný není pouze příjem živin a vylučování odpadů, ale i příjem a vylučování látek podílejících se na dalších funkcích. Druhá skupina, nazvěme ji např. transmisní, obsahuje funkce, které zprostředkovávají

detekci vzruchů, jejich přenosy, zpracování a uložení. Přenos v samotném neuronu je založen na bázi elektrického náboje. Buňka je od okolí oddělena polopropustnou membránou, kdy touto membránou přijímá a vylučuje různé látky, mimo jiné přijímá do svého těla ionty draslíku. Mimobuněčná hmota (z venku membrány) ale převážně obsahuje ionty sodíku a chlóru. Existují tak různé el. náboje uvnitř a vně buňky, tudíž se nechá mluvit o rozdílném elektrickém potenciálu. Při podráždění v okolí buňky se tento potenciál mění a pokud dosáhne určité prahové hodnoty, dochází k polarizaci a vzruch se ve formě potenciálové vlny šíří buňkou v maximální amplitudě. Když se vlna dostane na výstup (konce dendritů), dochází poté k depolarizaci – potenciál vlny se sníží pod prahovou úroveň. Existuje tedy i jistá relaxační doba neuronu, kdy neuron nepřijímá další podněty. Prahová hodnota není obecně konstantou, nýbrž funkcí - závisí na velikosti signálu (vzruchu), času, atd. Pro přenos vzruchů dál mezi neurony je důležitá práce synapsí.

3.3.2 Synapse

Synapse by se daly technicky popsat jako biologický interface, který zajišťuje více druhů funkcí, z nichž mnohé jsou dnes ještě neznámé. Asi nejdůležitější budou jistě funkce transmisní, přenosové. Počet synapsí silně převyšuje počet neuronů a závisí bezprostředně na složitosti propojení (tj. na topologii) příslušné neuronové sítě [10]. Přenos vzruchu je prováděn mezi dvěma neurony, synapse má tedy vstupní a výstupní část, mezi nimi se nachází synaptická štěrbina. Evolucí se vyvinulo více typů přenosů, tomu odpovídá i existence více typů synapsí. Bude řeč o přenosu elektrickém (tudíž elektrické synapsi) a přenosu chemickém (tudíž chemické synapsi).

Přenos elektrický je vývojově starší, jednodušší, ale i méně efektivní. Je typický pro nervový systém bezobratlých a nižších druhů obratlovců, v malé míře se vyskytuje i u savců. Přenos je zajišťen úzkými kanálky vedoucími mezi neurony, v nichž na principu el. náboje (potenciálu) putují mezi neurony jednotlivé ionty. Přenos může být veden obousměrně. Kanálky jsou nebo nejsou umístěny v synaptické štěrbině, podle toho potom rozeznáváme různé typy elektrických synapsí. Elektrická synapse má strukturálně i funkčně stejnou vstupní i výstupní část, synaptická štěrbina (pokud existuje u daného typu synapse) je velmi tenká. Obecně jsou neurony u tohoto typu přenosu velmi blízko sebe.

Přenos chemický je vývojově mladší, poměrně dost komplikovaný, ale velmi efektivní systém, který umožňuje na rozdíl od předchozího typu i určitou regulaci přenosu. Je typický pro vyšší řadu živočichů, především obratlovců. Přenos je zajištěn pomocí

přesunů a reakcí složitých, různě velkých chemických látek - transmiterů, které se vytvářejí v těle neuronu. Výstupní část je tvořena zakončením axonu, resp. jednoho z mnoha terminálů [10]. Ta, oddělena synaptickou štěrbinou, přiléhá ke vstupní části synapse spolupracujícího neuronu. Různé druhy vstupních a výstupních synaptických částí pak způsobují unikátní reakce a obecně unikátní typy přenosů pro jednotlivé kombinace. Tento typ přenosu umožňuje excitační i inhibiční reakci – tedy zesílení či utlumení vzruchu, navíc je možno transmitery kvantovat, tudíž přenos regulovat. Existuje tedy i více typů chemických synapsí. Synapse chemického typu mají také různá propojení s okolními buňkami (např. gliovými), složitost je tedy opravdu značná. Ukazuje se, že doba života a tedy i působení transmiterů je velmi podstatnou okolností [10]. Dále je zřejmé, že neuronové sítě našeho mozku mají k dispozici celé kontinuum dob působení transmiterů [10].

3.3.3 Paměť

Paměť je základní předpoklad pro existenci inteligentního života. Spolu s procesy učení zprostředkovávají dynamický vývoj neuronových sítí. Biologická funkce paměti je zajišťována pomocí určitých mechanismů, všechny se týkají obecně řečeno nervových tkání. Existuje tedy více druhů pamětí. Paměť v lidském mozku není lokálně centralizovaná (na jednom konkrétním místě), ale je rozložena ve více střediscích, celý systém paměti se značně vyvíjí a mění. Dále jsou některé neuronové sítě zálohovány, funguje zde pravděpodobně tedy i známy vztah z teorie obvodů, kdy redundance je nepřímo úměrná citlivosti. Lidskou paměť ze dělit na krátkodobou, střednědobou a dlouhodobou.

Paměť krátkodobá pracuje na principu cyklicky uzavřených oběhů vzruchů v určitých dílčích neuronových obvodech a sítích; obvykle se jedná o 80 až 90 neuronů. Doba udržení informace je u této paměti cca 30 sekund [10].

Paměť střednědobá vzniká díky působení paměti krátkodobé a to tak, že oběhem vzruchů určitými neurony dochází v chemických synapsích těchto neuronů k zesílení či utlumení vzruchů. Při opakovaném oběhu dochází díky této regulaci přenosu k trvalejšímu nastavení těchto synaptických spojů; tento proces je dosažen vznikem nových ribonukleových kyselin (RNA) [11]. Doba udržení informace je u této paměti v řádu hodin a dnů [11]. Tento mechanismus paměti hraje důležitou roli i v umělých neuronových sítí.

Paměť dlouhodobá vzniká díky působení paměti střednědobé, zprostředkovaně i krátkodobé. Podle současných představ vzniká určitým obtiskem předešlých mechanismů

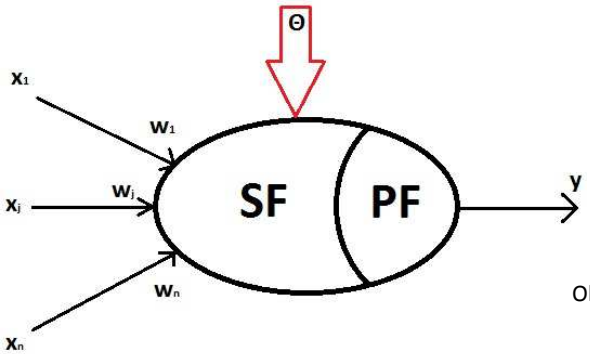
do bílkovinových struktur v nervových buňkách, především v jádře [11]. Tento děj probíhá v době spánku. Doba udržení informace je u této paměti až v řádech desítek let.

3.3.4 Matematický popis neuronu

Matematický popis neuronu vychází z popisu biologického. Již víme, že neuron má více vstupů – dendritů, pouze jeden výstup – axon, dále obsahuje tělo – soma a synapse – rozhraní. Dále z biologického popisu víme, že neuron (jeho soma) se aktivuje pouze pro vzruchy o určité síle (velikosti) – tomu matematicky odpovídá prahová funkce; také víme, že při dostatečně silném vzruchu se daný vzruch dostává skrze soma na axon – tomu matematicky odpovídá somatická funkce a funkce přenosová. Také již máme představu o komplikovanosti biologické skutečnosti, přesná matematická aproximace by byla prakticky nemožná. Z tohoto důvodu existuje více druhů matematických modelů neuronu, kdy jednotlivé modely zanedbávají určité skutečnosti. Dochází pak k značnému zjednodušení, lepšímu pochopení základních principů i snadnější realizaci.

Základní matematický model navrhl již roku 1943 Warren McCulloch a Walter Pitts, tento neuron nese tedy jejich jméno, také se používá název **formální neuron**. Tento neuron má více vstupů – matematický popis vstupu představuje obecně vstupní vektorovou funkci, dále má jeden výstup – matematický popis výstupu představuje obecně výstupní skalární funkci. Vlastnosti vstupů a výstupu jsou považovány za dokonalé, průchodem signálu se signál žádným způsobem nemění. Obecně lze pak činnost neuronu popsat jako činnost matematického procesoru, který vstupní vektorovou funkci zobrazuje na výstupní skalární funkci.

Synapse si lze představit v podstatě jako paměťové elementy, které se učí na základě informací přicházejících z okolí neuronu [10]. V tomto zjednodušeném přiblížení mají synapse především paměťovou funkci a spojitě adaptují své hodnoty (váhy), které vyjadřují průchodnost pro přicházející vzruchy, čímž v podstatě ukládají získané zkušenosti [10]. Model neuronu tohoto typu si lze představit na obr. 9, upraveném na základě [14]:



Obr. 9: Model formálního neuronu

x_1, x_j, x_n jsou vstupy; w_1, w_j, w_n jsou váhy synapsí, y je výstup. SF a PF znamená somatická a přenosová funkce. Θ představuje práh, prahovou hodnotu (obecně také funkci).

Somatická funkce ve formálním neuronu vyjadřuje vazbu vstupů a vah synapsí, což představuje reakci přijímaných vzruchů s uloženou zkušeností v synapsích. Matematicky se nechá popsat tato funkce jako [14]:

$$s = \sum_{i=1}^{n} x_i \cdot w_i + \Theta . \qquad (4)$$

V této funkci hraje určitou roli i prahová hodnota Θ, která určuje, při jaké velikosti příchozího signálu (vzruchu) se daný neutron aktivuje. Často je práh považován za nultou synaptickou váhu, pak vstup odpovídající této váze má hodnotu +1 [14].

Přenosová funkce v každém modelu neuronu vyjadřuje, jakým způsobem se bude přenášet výsledný stav somatické funkce (tedy výsledky zpracování soma neuronu) na výstup. Přenosových funkcí existuje mnoho druhů. Používá se např. funkce nelineární-skoková, nebo funkce lineární, hyperbolická, sigmoida, gaussova, apod. Výběr této funkce by měl být přizpůsoben konkrétním účelům. Odvíjí se od ní vlastnosti celé neuronové sítě, např. její učení, výpočetní čas, atd.

3.4 Neuronové sítě

Samotný neuron, přestože se jedná o komplexní buňku vlastnící dokonce i svůj informační systém, by z informačního hlediska byl funkčně značně omezen. Propojením neuronů vzniká velmi sofistikovaný a výkoný informační systém. Tato podkapitola bude členěna podobně jako podkapitola předešlá, věnující se neuronu. Bude tedy obsahovat základní biologický popis neuronových sítí a činností s nimi spojenými. Závěrem budou tyto poznatky využity k matematickému popisu neuronových sítí a jejich chování.

3.4.1 Biologický popis neuronových sítí

Problémy studia neuronových sítí tkví nejen ve složitosti struktury a funkcí samotného neuronu, ale i v metodách popisu sítí samotných. Sice dokážeme poměrně dobře pomocí mikroskopie zjišťovat struktury nervových tkání, horší výsledky se ale dosahují u činností a funkcí těchto struktur. Dnes dokážeme reálně studovat pouze činnosti a funkce určitých nervových celků, což nám dává pouze makroskopické hledisko, navíc značně omezené. Diskutabilní je i otázka ovlivnitelnosti měřením. Tyto problémy ale také zapříčinily vznik

myšlenky umělých neuronových sítí, kdy se výsledky výzkumu technického řešení dají aplikovat i do oblastí biologických. V současnosti tento výzkum probíhá a je velmi významnou aplikací umělých neuronových sítí.

Biologické neuronové sítě se skládají z mnoha neuronů, nejedná se pouze o neurony stejného typu, ale biologická síť obsahuje většinou více typů různých neuronů. Tímto je zajištěna i jistá funkční pestrost. Neuronové sítě nejsou statické, ale dynamicky se mění, vyvíjí. Sítě jsou vzájemně propojeny do různých funkční celků – bloků a podbloků. Tyto celky tvoří vzájemně propojené neuronové vrstvy, které jsou typické převážně pro vyšší třídu živočichů, kteří je potřebují pro obecně vyšší mozkové funkce. Další problém studia biologických neuronových sítí je jejich unikátnost. Každý jedinec vlastní jinou genetickou informaci, navíc se sítě různě vyvíjí – děje se tak v procesu učení a zapomínání. A protože každý z nás získává v průběhu života různé, pro danou osobu jedinečné zkušenosti, tak i vývoj struktury a funkcí neuronových sítí jsou do značné míry unikátní. Přesto lze pozorovat určité souvislosti mezi neuronovými sítěmi jedinců. Evolučním vývojem docházelo k určitým změnám ve struktuře a funkci mozku, tyto změny byly prováděny postupným navyšováním počtu neuronů a vznikem nových částí mozku (nových neuronových struktur). Tyto nové části mozku nebyly izolovány, nýbrž byly funkčně propojeny s již existujícími částmi. V mozku existuje velké množství různých struktur neuronových sítí, z nichž každá je specializována pro určitou činnost. Nabízí se analogie s technickými činnostmi, které je třeba řešit v daném výpočetním úkolu. Proto studiem těchto struktur se lze inspirovat i v reálné aplikaci. Nyní bude následovat popis jedné z mnoha takovýchto struktur.

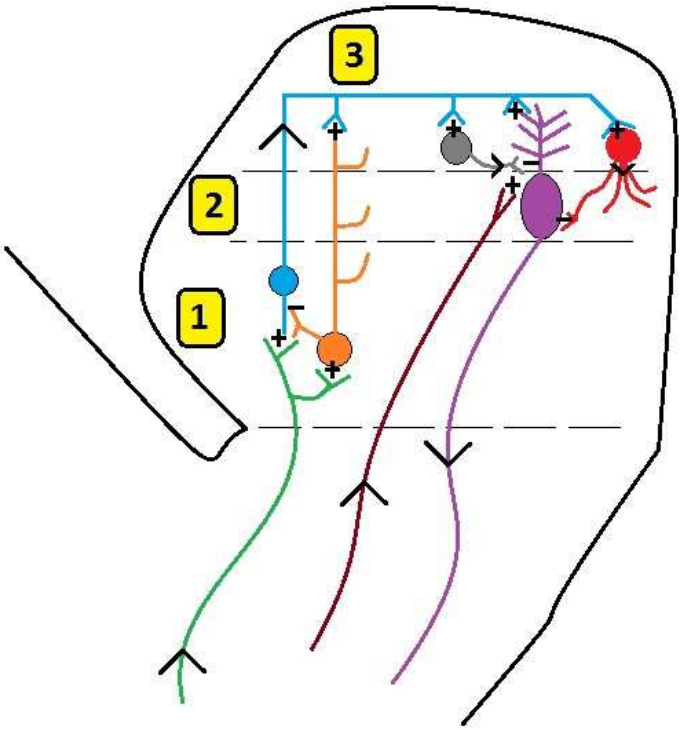
Základní neuronové obvody mozečku

Mozeček (*Cerebellum*) je jednou z vývojově starších částí mozku, je uložen pod týlními laloky, je propojen jak se středním mozkem, tak prodlouženou míchou. Řídí činnosti koordinace pohybů a rovnováhy, ovlivňuje ale i další funkce např. kognitivní. Mozeček má třívrstvou strukturu: ve spodní vrstvě (granulární) se nalézají tzv. granulární interneurony, ve vyšší vrstvě (ganglionální) pak jsou jednak výkonné Purkyňovy neurony, košíčkové neurony, hvězdicové interneurony a tzv. Golgiho buňky [11]. Ve třetí vrstvě (molekulární) jsou dendrity Purkyňových buňek a paralelní vlákna [11].

Granulární interneurony jsou velikostně poměrně malé neurony, jejichž vstupy (axony) se ve vyšší vrstvě terminálově rozvětvují (vzniká paralelní struktura) a tyto terminály pak doléhají na dendrity Purkyňových neuronů, které tímto způsobem aktivují. Purkyněho

buňky jsou poměrně velké neurony, dendrity se větví ve tvaru stromečku, axon míří většinou do jádra mozečku. Hvězdicovité a košíčkové neurony mají tlumící účinky na Purkyňovy buňky.

Golgiho buňky mají tlumící účinky na granulární interneurony a synapse vláken směřujících z jádra mozečku na povrch. Na obr. 10 je zjednodušeně naznačena tato struktura, obrázek upraven na základě [15].



3 – Molekulární vrstva
2 – Ganglionální vrstva
1 – Granulární vrstva

Granulární buňka – modrá
Golgiho buňka – oranžová
Hvězdicová buňka – šedá
Purkyňova buňka – fialová
Košíčkovitá buňka – červená

Mechová vlákna – zelená
Šplhavá vlákna – hnědá

Obr. 10: Základní neuronové obvody mozečku

Vstupní signály do mozečku jsou přiváděny axonovými vlákny z poměrně velmi vzdálených specifických oblastí mozku [11]. **Šplhavá vlákna** (climbing fibre) mají průměr 1-3 mikrometry a vedou přímo k dendritům Purkyňových neuronů; vlákna jsou u Purkyňových neuronů příčně rozvětveny [11]. **Mechová vlákna** (mossy fibre) jsou četnější a mají průměr poněkud větší, končí na dendritech granulárních neuronů; vlákna jsou u granulárních neuronů prostorově rozvětveny [11]. Šplhavá vlákna tedy mohou aktivovat Purkyňovy buňky přímo, kdežto mechová vlákna je mohou aktivovat pouze skrz granulové neurony, které pak tvoří biologickou obdobu relé.

3.4.2 Obecný popis neuronových sítí

Tato část práce je zaměřena na základní vlastnosti neuronových sítí. Jako taková vychází z popisu biologického a dalo by se říci, že směřuje k popisu technickému, který by se dal využít

v samotné realizaci a případných aplikacích. Ještě zde nebude hovořeno o jednotlivých druzích neuronových sítí, ale získané poznatky se i tak budou týkat všech druhů neuronových sítí.

Každou neuronovou síť lze popsat z hlediska její velikosti a struktury (stavby) [10]. Velikost je daná počtem neuronů či výkonných prvků neurony simulujících. Tyto prvky jsou buďto uloženy v jedné vrstvě, častější případ je ale vícevrstvý. Představu o více vrstvách si lze udělat v předešlé podkapitole popisující tří-vrstvou neuronovou síť mozečku. Strukturu sítě je možno charakterizovat jejím typem a geometrií; typ udává zásadní způsob uspořádání a propojení prvků, kdežto geometrie charakterizuje detailní uspořádání (např. 14-10-4 pro tří-vrstvou síť) [10].

K popsání neuronové sítě je třeba také vědět, které neurony jsou v ní aktivovány, to nám může říci tzv. aktivizační neboli stavový vektor. Zda jsou jednolivé neurony aktivovány či ne, závisí obecně na vstupech a prahové funkci (viz 3.3.4. Matematický popis neuronu). Dále je důležité znát také stav synaptických vah (jež uchovávají zkušenost), to nám může říci váhový vektor, možný je i popis pomocí matice.

Matematický popis neuronové sítě je podobný matematickému popisu neuronu, její práci lze charakterizovat tak, že zobrazuje vstupní vektor $\boldsymbol{X}$ na výstupní vektor $\boldsymbol{Y}$.

Každá síť má tedy funkční charakter: $\boldsymbol{Y} = f(\boldsymbol{X})$.

Tvar funkce je dán strukturou sítě i konkrétním modelem neuronu (např. matematickým). V každé síti mohou být buď všechny neurony stejného druhu – pak mluvíme o síti homogenní, nebo mohou být druhy různé – pak se jedná o síť heterogenní. Totéž dělení lze použít i na jednotlivé vrstvy sítě, pak se jedná buď o sítě s homogenními vrstvami nebo o sítě s heterogenními vrstvami [10].

3.4.3 Učení

Učení je naprosto zásadním procesem v činnosti neuronové sítě. Nutný předpoklad k procesu učení je existence paměti, bez paměti nelze učení provozovat. Učení samotné patří do skupiny vyšších mozkových funkcí, které vlastní pouze vyspělé organismy (vývojově vyšší živočichové). Pro tuto vyšší skupinu mozkových funkcí je typické vícevrstvé uspořádání neuronových sítí.

Učení v neuronových sítí můžeme definovat jako modifikaci synaptických vah a prahů podle zvoleného algoritmu učení tak, aby byla do sítě uložena informace [14]. Při učení se obecně snažíme o to, aby odchylka výsledků od požadovaného stavu byla co nejnižší.

Učení můžeme dělit podle toho, co je úkolem učení. Jestliže je úkolem pouhé pamatování a vybavování si informací, pak se jedná o učení **neasociativní**. Jestliže je úkolem nejen pamatování a vybavování si informací, ale především hledáním vnitřních vztahů a vazeb mezi těmito informacemi, pak se jedná o učení **asociativní**. Tento typ učení umožňuje i vznik nové informace.

Učení lze dále dělit na **učení s učitelem** a **bez učitele**. V případě neuronových sítí znamená učení s učitelem to, že známe předem výsledky nějaké tréninkové množiny úloh a víme tedy, jak moc se výsledky neuronové sítě přibližují správným výsledkům. To nám umožňuje už i v průběhu učení neuronovou síť korigovat. Učení bez učitele, někdy nazýváno také jako samoorganizační, hledá ve vstupní množině informací podobnost těchto dat (např. dle vlastností, stuktury) a podle toho tyto informace třídí do skupin (vektorů), které dává na výstup.

Existuje mnoho různých algoritmů učení, některé jsou inspirovány přírodou, některé ne. Jedná se opravdu o základní funkci v činnosti neuronových sítí, proto je výzkum nových typů učení poměrně rozšířen a přináší bohaté výsledky, které lze poté i zpětně aplikovat k pochopení biologického učení organismů. Více poznatků o učení lze nalézt v následující podkapitole.

3.5 Realizace neuronových sítí

Tato část práce je věnována popisu základních modelů neuronových sítí a jejich technické reprezentaci. K jednotlivým modelům jsou připsány také konkrétní poznatky o jejich učení a trénování. Výběr konkrétních modelů byl motivován jistými milníky v realizaci umělých neuronových sítí.

3.5.1 Jednoduchý perceptron

První pokusy s umělými neuronovými sítěmi dělali samotní autoři formálního neuronu McCulloch a Pitts. Došli k závěru, že lze pomocí sítí s jejich modelem neuronu řešit kteroukoliv binární logickou funkci. Tyto sítě ale měli dva nedostatky: neumožňnovaly modelovat proces učení a nebyly tolerantní k poruchám či chybné funkci jednotlivých prvků [16].

Perceptron se stal prvním široce rozšířeným modelem umělé neuronové sítě, roku 1958 jej navrhl Frank Rosenblatt. Rosenblatt se nechal inspirovat McCullochem a Pittsem v použití jejich formálních neuronů ale s tím, že na rozdíl od nich svoji konstrukci doplnil myšlenkou trénovanosti své sítě. Tato myšlenka se realizovala nalezením takových algoritmů, kterými by se daly účelně měnit parametry neuronů dané sítě a tím by se tato síť učila. Máme-li v *n*-rozměrném prostoru lineárně separabilní třídy objektů, pak lze v konečném počtu kroků učení (iterací algoritmu) nalézt vektor vah **w** perceptronu, který oddělí jednotlivé třídy bez ohledu na počáteční hodnoty těchto vah [10].

Vektor vstupních signálů **x** bude obsahovat všechny vstupy x_1 až x_n. Lineární kombinace těchto vektorů dá výstup y = $\mathbf{w}^T\mathbf{x}$ + b, kde b označuje hodnotu prahu. Vektor vah **w** je vždy kolmý na vektor vstupů **x**. Všechny vstupy se budou dělit do dvou podprostorů (**x** se rozdělí na dvě třídy např. C_1 a C_2). Jak přesně se budou dělit závisí na vstupním vektoru, pro který je výstup nulový. Tedy 0 = $\mathbf{w}^T\mathbf{x}$+ b. Dále je tvar této hranice závislý i na konkrétní přenosové funkci neuronů i funkci prahové. V případě, že mají vstupy binární charakter, tak se používá jako přenosová funkce nelineární-skoková. V případě, že mají vstupy např. reálný charakter, pak se používá např. funkce sigmoidální nebo hyperbolická (je vhodné, aby funkce byla spojitá).

Příklad:

Zde si předvedeme velmi jednoduchý příklad, velmi pěkně popsaný v [17], jak pomocí perceptronu realizovat logickou funkci AND. Vstupy budou mít binární charakter, takže jako přenosovou funkci použijeme nelineární-skokovou. Výsledná hodnota této funkce *f*(*y*) bude rovna 1 za předpokladu, že výsledek somatické funkce perceptronu bude větší nebo roven 0, tedy: $f(y)=1 \quad if \ y \geq 0$. Výsledná hodnota této funkce *f*(*y*) bude rovna 0 za předpokladu, že výsledek somatické funkce perceptronu bude menší než 0, tedy: $f(y)=0 \quad if \ y \prec 0$.

Na začátku si zvolíme váhový a prahový vektor, tedy: $\mathbf{w} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, $\mathbf{\Theta} = \begin{bmatrix} 1{,}5 \\ 0 \end{bmatrix}$

Vstupní vektory jsou binární kombinace prováděné na 2-bitech, tedy:

$\mathbf{x}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $\mathbf{x}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\mathbf{x}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

1) Nejdříve vypočteme konkrétní prahovou hodnotu:

$\mathbf{w}^T\mathbf{x} + \Theta = 0$ $\quad [2 \quad 2] \cdot \begin{bmatrix} 1{,}5 \\ 0 \end{bmatrix} + b = 0$ $\quad 3 + b = 0$ $\quad b = -3$

2) Poté vypočteme hodnotu somatické funkce perceptronu např. pro vstupní vektor $\mathbf{x}_2$:

$y = \mathbf{w}^T\mathbf{x} + b$ $\quad y = [2 \quad 2] \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 3$ $\quad y = 2 - 3$ $\quad y = -1$

3) Z přesnosové funkce perceptronu určíme výslednou hodnotu na výstupu:

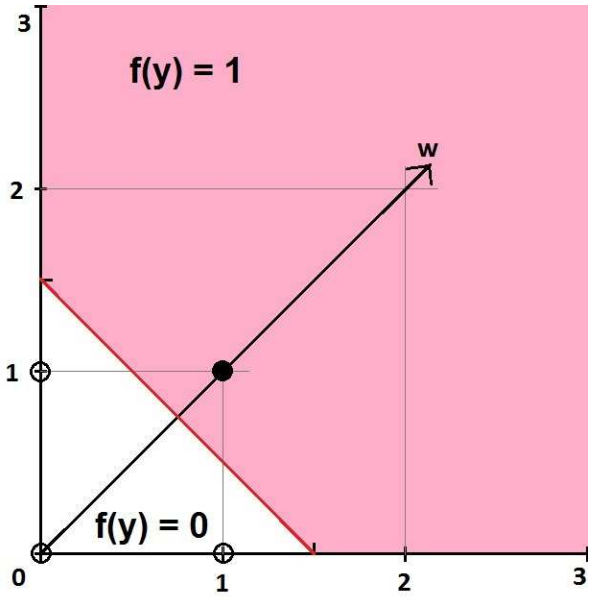
$y < 0$, tudíž $f(y) = 0$

Kroky 2 a 3 můžeme zopakovat např. pro vstupní vektor $\mathbf{x}_4$:

$y = \mathbf{w}^T\mathbf{x} + b$ $\quad y = [2 \quad 2] \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 3$ $\quad y = 2 + 2 - 3$ $\quad y = 1$

$y \geq 0$, tudíž $f(y) = 1$

Pokud by jsme provedli tyto výpočty pro všechny vstupní vektory **x**, viděli by jsme, že takto nastavený perceptron skutečně realizuje logický součin AND. Na obr. 11 je znázorněno grafické řešení tohoto příkladu. Lze si všimnout, že hraniční čára představující konkrétní přenosovou funkci je kolmá na vektor vah **w**.



Obr. 11: Grafické řešení logické funkce AND

Problém tohoto řešení tkví v tom, že musíme znát předem konkrétní nastavení sítě, tedy vah a práhu. To lze ale řešit algoritmy učení, kdy se v procesu učení budou měnit tyto hodnoty tak, aby vyhovovaly správným výsledkům realizované úlohy.

Nyní se tedy zaměříme na učení této sítě. U perceptronové sítě se používá učení s učitelem – známe tedy množinu úloh (vstupů) a jejich správné výsledky (výstupů). Nejdříve se libovolně nastaví parametry sítě (váhy a práh), poté se síti dají vstupní hodnoty a vypočte se dle již známého vzorce $s = \sum_{i=1}^{n} x_i \cdot w_i + \Theta$ hodnota výstupní (stejně jako v předchozím případě). Poté dochází k porovnání výstupní hodnoty s předpokládanou (výsledek, o kterém víme, že je správný = učitel). V případě, že chybová odchylka je nulová, může se přikročit k dalšímu výpočtu. To nastává, pokud je daný vstup zařazen do správné třídy C_1 nebo C_2. V případě, že chybová odchylka nulová není, pak dochází k úpravě parametrů sítě ještě před dalším výpočtem. Zavádí se pojem parametr učení, označíme ho jako *u*, který právě v případě chyby adaptuje synaptické váhy či práh. Způsobů, jak to provádí je více. Obvykle se jedná o nějaký algebraický tvar, např. dle [10]:

$$\boldsymbol{w}(j+1) = \boldsymbol{w}(j) - u(j)\,\boldsymbol{x}(j), \qquad (5)$$

kde parametr učení *u* může mít obecně funkční charakter, kdy by se jeho hodnota v opakování algoritmu učení měnila nebo se může jednat o konstantu.

Příklad

Vyjdeme z předchozího příkladu, tedy realizaci logické funkce AND, nyní ale neznáme přesné nastavení sítě, takže budeme volit náhodné hodnoty a síť v procesu učení bude tyto hodnoty měnit, až se dostane do stavu, kdy všechny výstupní hodnoty budou správné,
v tu chvíli se učící algoritmus ukončí.

Opět tedy máme již známé vstupní vektory $\mathbf{x}_1 - \mathbf{x}_4$, nyní náhodně nastavíme váhy,

tedy např. $\mathbf{w} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, $\boldsymbol{\Theta} = \begin{bmatrix} 1{,}5 \\ 0 \end{bmatrix}$

1) Nejdříve vypočteme konkrétní prahovou hodnotu:

$\mathbf{w}^T\mathbf{x} + \boldsymbol{\Theta} = 0$ $\qquad \begin{bmatrix} 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1{,}5 \\ 0 \end{bmatrix} + b = 0$ $\qquad 3 + b = 0$ $\qquad b = -3$

2) Poté vypočteme hodnotu somatické funkce perceptronu např. pro vstupní vektor $\mathbf{x}_2$:

$y = \mathbf{w}^T\mathbf{x} + b$ $\quad y = \begin{bmatrix}2 & 3\end{bmatrix} \cdot \begin{bmatrix}0\\1\end{bmatrix} - 3$ $\quad y = 3 - 3$ $\quad y = 0$

3) Z přesnosové funkce perceptronu určíme výslednou hodnotu na výstupu:

$y \geq 0$, tudíž $f(y) = 1$ -> **CHYBA!** *(Funkční hodnota log. součinu pro vstupy 0,1 je 0)*

4) Dojde tedy nyní k samotnému učení, změní se nastavení vah, parametr učení zvolíme jako konstantu 0,5 ($u = 0{,}5$):

$\boldsymbol{w}(j+1) = \boldsymbol{w}(j) - u(j)\,\boldsymbol{x}(j)$ $\quad \boldsymbol{w}(j+1) = \begin{bmatrix}2 & 3\end{bmatrix} - 0{,}5 \cdot \begin{bmatrix}0\\1\end{bmatrix}$ $\quad \boldsymbol{w}(j+1) = \begin{bmatrix}2 & 2{,}5\end{bmatrix}$

5) Provede se přepočítání prahové hodnoty b (v tomto případě se nijak nezmění):

$\mathbf{w}^T\mathbf{x} + \Theta = 0$ $\quad \begin{bmatrix}2 & 2{,}5\end{bmatrix} \cdot \begin{bmatrix}1{,}5\\0\end{bmatrix} + b = 0$ $\quad 3 + b = 0$ $\quad b = -3$

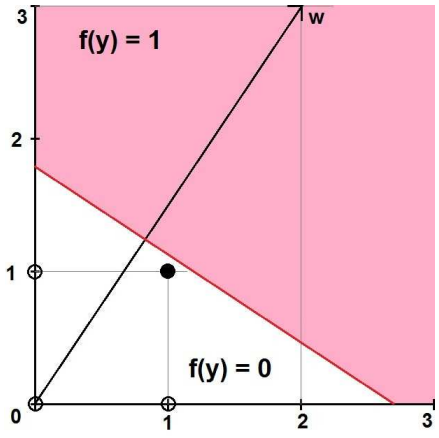
6) Nyní provedeme další výpočet hodnoty somatické funkce opět např. pro $\mathbf{x}_2$:

$y = \mathbf{w}^T\mathbf{x} + b$ $\quad y = \begin{bmatrix}2 & 2{,}5\end{bmatrix} \cdot \begin{bmatrix}0\\1\end{bmatrix} - 3$ $\quad y = 2{,}5 - 3$ $\quad y = -0{,}5$

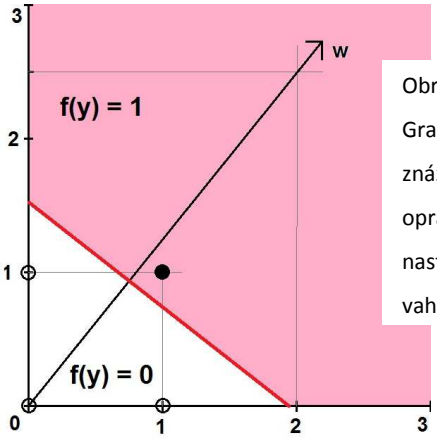
7) Z přesnosové funkce perceptronu určíme výslednou hodnotu na výstupu:

$y < 0$, tudíž $f(y) = 0$ -> **SPRÁVNĚ!**

Pokud by se spočítaly i ostatní vstupní vektory **x**, ukázalo by se, že takto „poučená“ síť bezchybně realizuje logickou funkci AND. Na obr. 12 a 13 je znázorněno grafické řešení tohoto příkladu, kde si lze všimnout skutečnosti, že změnou nastavení synaptických vah dochází ke změně orientace přenosové funkce, neboť ta je kolmá k vektoru vah **w**. V tomto případě se jednalo o učení založené na změně synaptických vah. Je i určitá možnost učit síť změnou prahového vektoru – tím dochází k rozšíření nebo zúžení dělených podprostorů (C_1 a C_2).



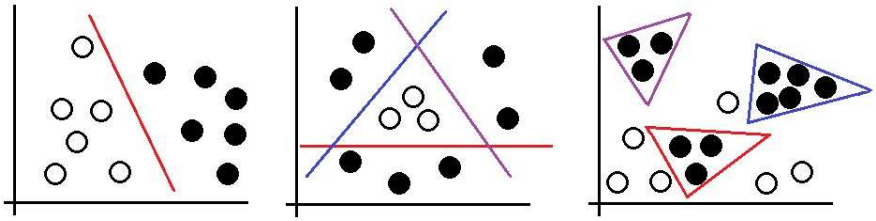
Obr. 12: Grafické znázornění chybného nastavení vah



Obr. 13: Grafické znázornění opraveného nastavení vah

3.5.2 Vícevrstvé perceptrony

Základní perceptron obvykle obsahuje pouze jednu vrstvu neuronů – výkonných prvků. To umožňuje dělení pouze do dvou tříd, podprostorů. Jeho užití je tedy značně omezené. V praxi se více uplatnili vícevrstvé perceptrony, které funkčně obohatili možnosti řešení a umožnili realizaci i více komplikovaných úloh. Struktura těchto neuronových sítí se více blíží biologické podobě. Použití vícevrstvých neuronových sítí není omezeno podmínkou lineární separovatelnosti vstupních dat [14]. Jednotlivé vrstvy dokáží oddělit i více členité vstupní informace, dokonce mohou i skrze různé operace (např. průnik, sjednocení,…) vytvořit více tříd (více podprostorů). Ukázka tohoto řešení je na následujícím obrázku 14, který byl upraven na základě [14]:



Obr. 14: Separace podmnožin:
jednoduchý perceptron – dvouvrstvý perceptron – třívrstvý perc. s back-propagation

3.5.3 Dopředné sítě se zpětným šířením při učení (back propagation)

Tento typ sítě byl vyvinut v 80. letech, čímž došlo ke značnému oživení neuronových sítí obecně. Je to v podstatě gradientní metoda minimalizující součet čtverců chyby výstupních hodnot uvažované sítě [10]. Často je použit biologický vzor popsaný také v této práci; tzn. vícevrstvá síť, kdy neurony jedné sítě nekomunikují mezi sebou, ale pouze s neurony odlišné vrstvy. Metoda spočívá v tom, že informace se šíří ze vstupní vrstvy do vrstev prostředních (skrytých) a poté na vrstvu výstupní (foward propagation). Na výstupu se určí odchylka výstupní hodnoty od požadované (opět učení s učitelem) a v závislosti na velikosti této odchylky se zpětně adaptují parametry sítě. Zpětné šíření (back propagation) tedy znamená šíření adaptace parametrů sítě směrem od výstupní vrstvy zpět přes vrstvy prostřední až na vrstvu vstupní.

V jednodušším pojetí učení je uvažován pouze jeden parametr, obvykle označovaný jako délka kroku, např. tedy u [10]. Obecně jich může být více (parametr změny vah, prahu, strmosti přenosové funkce,…). Proces obvykle probíhá tak, že po určení hodnot korekčního parametru u jsou všechny parametry (např. váhy) modifikovány současně [10].

Tyto poznatky lze aplikovat do mnoha typů sítí, vykazují ale také určité problémy. První problém se týká náhodného nastavení synaptických vah na počátku výpočtu. Obecně se doporučuje spíše vybírat menší náhodná čísla. V publikaci [10] se doporučuje pro nastavení vah využít vzorce: $-\beta \leq w \leq \beta$, $\beta = 0{,}7\sqrt[n]{p}$, (6)

kde *n* je počet výkonných prvků (neuronů) vstupní vrstvy, *p* označuje počet výkoných prvků prostřední (skryté) vrstvy.

Další problém se týká trénovanosti této sítě. Pro danou chybovou odchylku, (např. *e*) je potřeba použít tréninkovou množinu o určitém množství vzorů.

V publikaci [14] se doporučuje použít vztahu: $N_p = \frac{N_w}{e}$, (7)

kde N_p znamená počet vzorů, N_w znamená počet vah.

Přiklad:

3-vrstvá neuronová síť s geometrií 10-5-2 obsahuje celkem 64 vah. Urči počet vzorů tréninkové množiny tak, aby chybová odchylka byla maximálně 0,5.

$$N_p = \frac{N_w}{e} \qquad N_p = \frac{64}{0{,}5} \qquad N_p = 128$$

Tréninková množina potřebuje mít alespoň 128 kvalitních vzorů řešení.

3.5.4 Technická realizace neuronových sítí

V této části práce bude stručně hovořeno o tom, jaké možnosti máme pro vytváření umělých neuronových sítí, tedy jejich technickou realizaci. V průběhu vývoje realizací se vypreferovaly dva hlavní proudy: hardwarové řešení a počítačová simulace.

Do proudu hardwarového by patřily všechny realizace využívající elektroniku, opto-elektroniku, speciální počítačové a čipové konstrukce. Analogové řešení je vývojově nejstarší, lze jej realizovat např. pomocí operačních zesilovačů v sumačním zapojení, kdy vstupní odpory představují hodnoty synaptických vah. Výhodou je rychlost a paralelní zpracování signálů, nevýhodou je nízká přesnost [18]. Další nevýhody vyplývají z analogového pojetí obvodů, tedy vyšší citlivost na parazitní signály, změny napájecího napětí, atd [18]. Další pokročilejší možnosti hardwarového řešení jsou neuropočítače a neuročipy. Neuropočítače se buď konstruují záměrně dle kontrétních realizací neuronové sítě (struktury a učících algoritmů),

ty pak mají pevnou a často i neměnnou strukturu, nebo se standartní počítače doplní o neurokoprocesory a speciální instrukce = neuročipy [14]. Neuročipy moho pracovat jak v analogovém, tak digitálním režimu. Vyvíjí se i optická realizace pomocí fotodiod a detektorů, mají však své omezení, zajímavěji se jeví kombinace s elektronikou, tedy opto-elektronické řešení. Velmi úspěšně se využívá možnosti paralelizace, které se dosahuje připojením speciálních obvodů, tzv. akcelerátorů k počítači. Tyto obvody jsou realizovány hradlovými poli, kterými lze realizovat různá zařízení, ty pak lze různě propojovat; obsahují tedy mimo jiné spínací matici (ta slouží právě k propojování mezi obvody) a konfigurační paměť (zde jsou daná nastavení konkrétních propojení uložena) [14]. Hardwarové řešení je obecně vhodné u těch úloh, kde je potřeba pracovat v reálném čase, takže např. rozpoznávání hlasu, obrazu, apod.

Do proudu simulací na počítači lze zařadit taková řešení, která se snaží strukturu neuronů a neuronových sítí nasimulovat v podobě programu. Využívá se např. programovacího jazyku MATLAB, ve kterém je používán Neural Network Toolbox, lze využít i PHP nebo Javu. Tato softwarová řešení jsou obecně vhodná u těch typů úloh, které jsou strukturou či složitostí velmi náročné. Dále umožňují časově oddělit proces učení od procesu vybavování [14].

3.6 Aplikace

Umělé neuronové sítě nejsou univerzálním prostředkem k úspěšnému a efektivnímu řešení všech problémů, někdy dosahují lepších výsledků než standartní metody, někdy stejných, někdy i horších. Je důležité si před samotnou realizací uvědomit, zda-li jsou umělé neuronové sítě vhodnou metodou k vyřešení daného úkolu. Které typy úkolů a problému jsou tedy vhodné pro neuronové sítě? Dalo by se říci, že největší přínos lze vidět u problémů, které nelze algoritmicky popsat dostatečně přesně, nebo dokonce vůbec. Další přednosti lze spatřit v řešení úloh, kde se vyskytují nějaké nelineární jevy. Dále se nechá hovořit také o někdy dost podstatné výhodě ve zpracování dat, kdy vstupní množina dat nemusí být úplná - část může chybět nebo tam mohou být určité nepřesnosti. Přesto umělé neuronové sítě mohou zpracovat i takto nedokonalá data. Také při výpadku či chybné funkci jednotlivých neuronů (výkonových prvků) nedochází k okamžitému ukončení výpočtu, ale logika neuronové sítě si poradí a výpočet, byť s určitou odchylkou, dokončí. První typ aplikací vyplývá z analytické části výzkumu neuronových sítí. Tedy modelování biologického neuronu a sítí, nervových struktur a obvodů; také analýza metod učení je velmi významný cíl tohoto výzkumu. Výsledky se pak využívají v oborech biologie, psychologie,

medicíny, apod. Nyní ale budou stručně popsány dva typy aplikací neuronových sítí ze syntetické části výzkumu umělých neuronových sítí.

Analýza a syntéza zvukového signálu

Tato zajímavá a dnes již úspěšně používaná aplikace se zaobírá zpracováním či realizací zvukového signálu, především řeči. Velmi podobně lze ale pracovat i s jinými typy signálů. Analýza se snaží zvukový signál (dále jen řeč) rozkódovat. Tzn. určit jednotlivá slova, písmena, věty, může se jednat i o význam daných slov či celých vět. Syntéza se naopak snaží řeč vytvářet, reprodukovat. Situace není úplně jednoduchá, neboť řeč konkrétních lidí ovlivňuje mnoho faktorů ať už psychologických (stres, radost, smích), fylogenetických (řeč každého člověka je unikátní) i jiných (např. situace, kdy člověk žvýká žvýkačků a současně mluví). To vše se promítá do analogového spektra řeči, které je spojité, dynamické a obecně nelineární. Dále se vyskytuje problém trénovací množiny, neboť řeč se neustále vyvíjí, vznikají nová slova, nová spojení, nové významy. Řešení této aplikace je realizováno pomocí třívrstvé, dopředné sítě se zpětným šířením chyby, tedy back-propagation [14].

Predikce časových řad

Pojem „časová řada“ představuje vývoj určité veličiny, který je v čase kvantován. Predikce časových řad se týká toho, jak bude tento vývoj v čase dále pokračovat. Množství oblastí, kde lze typově podobné aplikace využívat, je velmi mnoho. Může se jednat samozřejmě o oblasti technické (veličina je fyzikální), společenské vědy (např. v publikaci [19] lze najít konkrétní aplikace z oborů sociologie – závilost kriminality na různých proměnných či faktory ovlivňující demografický vývoj, politologie – závislosti chování voličů, a další), ekonomika (vývoj kurzu měn, inflace, HDP,…). Většinou se jedná o počítačové simulace, kde daný specializovaný program umožňuje modelování různých typů neuronových sítí, výběr z metod učení a naplnění tréninkové množiny. Zvláštní význam mají umělé neuronové sítě v metodách analýzy, kdy představují další analytický prostředek, tudíž umožňují porovnání s dalšími analytickými metodami např. statistickými.

4. Genetické algoritmy

4.1 Úvod

Genetické algoritmy představují velmi snadno realizovatelné řešení pro poměrně složité typy úloh. Jsou založeny na odvozování přírodních entit a procesů, které se týkají genetiky a evoluce obecně. Přesto, že tyto biolgické principy jsou poměrně složité, tak základní techniky práce jsou poměrně jednoduché a lehce pochopitelné. To vše činní tyto algoritmy poměrně zajímavými a v aplikacích dnes již velmi rozšířenými, kdy se především ve spojení s klasickými metodami vytváří mocný nástroj pro řešení obtížných úloh. V této kapitole budou popsány základní biologické principy a pojmy, dále jednoduchý genetický algoritmus s naznačením možností, jak jeho práci zefektivnit. Na závěr jsou uvedeny dvě typické aplikace genetických algoritmů a jedna zajímavá aplikace evolučího algoritmu.

4.2 Evoluce

Evoluce má zásadní význam nejen pro samotnou genetiku a práci s dědičností z biologického hlediska. Velký význam má i pro technické obory, neboť obecně zahrnuje všechny evoluční a velkou část genetických procesů. Tudíž evoluce tvoří určitý přírodou inspirovaný základní aparát pro evoluční i genetické algoritmy. V mnoha případěch nastává jev obdobný u neuronových sítí; je možné konstruovat nové operace a techniky, které biologický vzor neumožňuje nebo je nepoužívá. Může se jednat například o nové metody vzniku potomků či výběru jedinců, které v přírodě běžně nenastávají. Obvykle jsou ale méně efektivní. Další paralelu s neuronovými sítěmi lze nalézt ve využití biologických poznatků, které neslouží pouze k základnímu pochopení, ale i k jisté inspiraci v řešení praktických úloh, navíc podstatně rozšiřují funkční pestrost jednotlivých kroků algoritmů. Náplní celé této podkapitoly bude tedy stručný nástin problematiky evoluce z biologického hlediska, kdy v závěru dochází k užšímu vymezení genetických algoritmů.

4.2.1 Evoluce v přírodě

Evoluční procesy probíhají v celé přírodě neustále. U všech organismů lze najít určitý společný základ těchto procesů, týkají se hlavně genetiky a přírodního výběru. Především u člověka dostává evoluce ale nový, rozšířený význam - jedná se o tzv. integrovanou evoluci. Jak je popsáno v publikaci [20], americký psycholog James Baldwin vysvětlil, že inteligentní

chování, imitace a učení může mít vliv na selekční tlak, tj. význam učení pro evoluci. Děje se tak tedy v procesech učení, kdy předchozí generace předává té současné informace (slovně nebo písemně), které ji zajišťují výhodu pro adaptaci v současném světě. Dědičná informace se přenáší ve formě genů, tato řekněme kulturní informace se přenáší ve formě memů. Geny jsou uloženy v samotných buňkách organismu, kdežto memy mohou být uloženy buď v mozku jedince či hromadné paměti lidstva (např. knihy, internet). Evoluce memů probíhá v současné době mnohem rychleji než evoluce genů [20]. Nyní bude stručně popsán vývoj evoluce na úrovni genů.

V práci Charlese Darwina *O společném původu* je zaznamenáno, že všechny buňky využívají pro dědičné informace stejný kód [21]. Dále se zjistilo, že tyto informace jsou zprostředkovány hmotou na molekulární úrovni. Přenos těchto informací se tedy děje změnou (např. pohybem) této hmoty. Později tyto myšlenky vedly k nalezení genu, jehož popis bude nyní následovat. Gen na molekulární úrovni představuje určitý úsek DNA, kdy při množení buňky (dělením) se tento úsek přenáší na potomky (další generaci). Geny tedy zprostředkovávají dedičné vlastnosti organismu. DNA obsahuje obecně popis tvorby všech proteinů v těle organismu.

Gregor Mendel ve své práci věnované křížení uvedl, že každý gen (dědičný znak) určují dvě alely (konkrétní formy genu), při křížení se alely rozdělí a nahodile rekombinují [20]. Z toho lze usoudit, že každý jedinec (vyjádřený soustavou genů) je naprosto jedinečný, neboť rekombinace jeho genů je náhodná. Z uvedeného faktu vyplývá výrazná pestrost populace i existence možnosti zpětné rekonstrukce.

Z dalších prací popisujících rekombinaci genů vyplývá, že výsledek rekombinace genů v celé populaci se změní pouze v případě, kdy je populace dostatečně velká a pokud existují určité faktory. Definovaly se 3 faktory, které mohou způsobit evoluční změnu: mutace, přírodní výběr a genetický drift. Mutace náhodně mění genetickou informaci v každém kroku evoluce. Přírodní výběr způsobuje, že především jedinci s nejlepší možnou kombinací genů se budou účastnit dalšího přenosu genů (v procesu rozmnožování). Genetický drift zkoumán Sewallevem Wrightem se týka nahodilých změn v genech populací různých velikostí [20]. Jestliže je velikost populace dostatečně velká, pak se změny v genech vzájemně kompenzují a v průměru celé populace se genetický fond nemění. Jestliže je ale velikost populace malá, ke kompenzaci nedochází a může dojít i k výrazným změnám vztahujícím se k průměru genů

v celé populaci. Pak tedy rozlišujeme dva typy evoluce: Darwinovu (obsahující mutaci + přirozený výběr) a Wrightovu (obsahující mutaci + genetický drift).

V současnosti se došlo k poznání, že na úrovni organismů jako celku převažuje Darwinova evoluce, kdežto na molekulární úrovni evoluce Wrightova [20]. Existuje nyní tedy určitý rozpor mezi genovými informacemi buněk a chováním organismů. Nyní se hledá tedy určitý most mezi těmito dvěma světy, který by tento rozpor vyřešil. Lze jej hledat například i v informacích vyskytujících se mimo geny, v tzv. epigenetických informacích. Další problém se vyskytl v Mendelově křížení, kdy Mendel předpokládal (a širá veřejnost až do konce 20. stol. s ním), že geny mají ve struktuře své stálé místo, tudíž je jejich pozice v procesech křížení neměnná. Zjistil se naprostý opak. Vývoj pochopení celé evoluce a jednotlivých účástnících se procesů tedy neustává a lze v budoucnu očekávat velké změny, které se jistě velmi rychle promítnou i do technického použití genetických a evolučních algoritmů.

4.2.2 Základní pojmy

Pro práci s genetickými algoritmy je potřeba si zapamatovat několik biologických pojmů, jejichž znalost je potřebná pro ujasnění biologických procesů i přímo pro výpočetní řešení.

Gen

- část DNA, která nese dědičnou informaci
- může přímo ovlivňovat určitou vlastnost jedince

Alely

- jsou konkrétní formy geny (různé variace lišící se strukturou na molekulární úrovni)
- jinak řečeno jsou to stavy, které mohou geny nabývat

Chromozom

- je tvořen lineárně uspořádanými geny, je uložen v buněčném jádře
- každý organismus má unikátní, neměnný počet chromozomů, jejich tvar a velikost
- obvykle jsou chromozony v buňce v párech, tedy každý chromozon existuje v buňce dvakrát

Genotyp

- množina všech genetických informací jedince; tedy nejen genů ale i dalších informací
- v oblasti genetických algoritmů se zjednodušuje definice a to tak, že genotyp obsahuje pouze jeden chromozom, který naprosto dostatečně charakterizuje daného jedince [22]

Fenotyp

- genotyp + prostředí, výsledek působení prostředí na genotyp
- množina všech vnějších vlastností konkrétního jedince

Genofond

- množina všech genů celé populace konkrétního druhu organismu

Fitness/Ohodnocující funkce

- měřítko, které udává jaký chromozom (genotyp, jedinec) je z určitého hlediska preferovanější
- obvykle vyjadřuje určitou požadovanou vlastnost či porovnání s ideálem (výsledkem výpočtu)

Křížení

- základní technika pro vznik nových jedinců, obecně vznik celé nové populace
- probíhá na základě výměny genů (dědičné informace)

Mutace

- geny každé nově vytvořené populace jsou náhodně měněny z důvodu zábrany degenerací
- tyto záměny probíhají pouze u minimálního množství genů

4.2.3 Genetické algoritmy

Genetické algoritmy jsou založeny na principech biologických, snaží se využívat toho, jakým způsobem řeší různé úlohy sama příroda a tato řešení se snaží napodobit. Genetické algoritmy se nechají zařadit do skupiny algoritmů evolučních, kam patří i evoluční strategie, evoluční programování a genetické programování. V řadě případů však přesahují rámec pojmu genetický a stávají se z nich algoritmy evoluční [22]. Lze si také povšimnout, že v biologickém úvodu této kapitoly také nelze jednoduše oddělit evoluci od genetiky, přestože se jedná o dva různé pojmy. Obzvláště v technických aplikacích však význam těchto pojmů opravdu někdy splývá a často se hovoří pouze o jedné skupině algoritmů. Nyní bude naznačeno, jaké vlastnosti tyto algoritmy mají a kdy je vhodné je použít.

Evoluční (i genetické) algoritmy jsou stochastické optimalizační metody založené na darwinovském principu evoluce [22]. Stochastické znamená, že v daném algoritmu hraje určitou roli náhoda. Slovo optimalizační znamená, že na konci takového algoritmu nemusí být jeden řekněme ideální výsledek, ale může se jednat o řadu různých výsledků, které dostatečně splňují zadání úlohy. Může to znít prapodivně, ale někdy skutečně nelze určit jeden přesný výsledek. Také se může stát, že tento přesný výsledek je ideální pouze pro určitý okamžik, v jiném čase je naprosto nepoužitelný. Z tohoto hlediska je potom lepší nalézt takové řešení,

které není vždy stoprocentní, ale v časovém vývoji je poměrně kvalitní a především dostatečně splňuje zadání úlohy. Darwinovský princip evoluce, jak se lze dočíst v části věnované historii evoluce, je zprostředkován dvěmi procesy – mutací a přirozeným výběrem. Lze si všimnout, že samotná evoluce i biologické entity, které v ní hrají roli, byly na technicko-výpočetní úrovni značně zjednodušeny. Z tohoto faktu vyplývají dva důsledky, strukturálně-funkční omezení a značné zjednodušení samotných výpočetních procesů. Jejich programování je poměrně snadné, patří též do skupiny metod nazvané „soft computing“, kde se vyskytují metody umožňující snadné zvládnutí obtížných úloh. Další vlastností těchto algoritmů je, že jsou vnitřně paralelní, vždy se současně pracuje s celou populací, tedy všemi realizacemi dané úlohy najednou.

Z uvedeného popisu genetických alogritmů vyplývají i možnosti jejich použití. Jsou poměrně univerzální, přesto se samozřejmě nejvíce hodí pouze pro určité typy úloh. Jestliže se zná dostatečně spolehlivý algoritmus speciálně vytvořený pro konkrétní úlohu, tak ten bude pravděpodobně vždy efektivnější. Pokud jej nelze vytvořit (často např. jako náhrada člověka či pro dlouhý čas výpočtu), pak jsou genetické algoritmy jednou z vhodných metod řešení. Hodí se zvláště pro úlohy, jejichž struktura či procesy jsou značně složité a nejdou přesně popsat, ale přesto se zná alespoň návrh nějakého dostatečně dobrého řešení. Jiné požadavky tyto algoritmy nemají. Vhodnost použití genetických algoritmů naznačují tyto indicie [22]:

- Prohledávaný prostor je velký
- Prohledávaný prostor není vyhlazený a unimodální
- Struktura prohledávaného prostoru je komplikovaná
- K řešení úlohy není třeba nutně najít globální optimum,
  ale přijatelné řešení v přijatelném čase

Vhodnost použití těchto algoritmů pro určité typy úloh je první předpoklad úspěchu. Ten druhý představuje kvalita samotné tvorby takového algoritmu, to bude právě popsáno v následující podkapitole.

4.3 Jednoduchý genetický algoritmus

Vznik genetických algoritmů je poměrně nedávný, zasloužil se o něj John Holland, který se věnoval výzkumu buněčných automatů. Holland vytvořil základní vizi jednoduchého

genetického algoritmu [23]. Tato vize se v průběhu dalšího vývoje různě upravovala, ale většina genetických algoritmů probíhá v následujících krocích, které budou dále popsány:

1) Vytvoření populace
2) Ohodnocení jedinců
3) Reprodukce, vznik potomků
4) Ohodnocení potomků
5) Vznik nové populace
6) Splnění či nesplnění zadané úlohy

1) Vytvoření populace

Vytvoření všech původních jedinců probíhá obvykle náhodným přiřazením čísel (které představují alely genů) do chromozomu (řetězec určité délky s lineárně řazeným, neměnným počtem pozic). Každý jedinec pak představuje určité řešení dané úlohy. Proto tedy naprosto zásadním pojmem je kódování jedinců. Je otázka, jak dosáhnout toho, aby chromozom daného jedince přesně vystihoval konkrétní řešení. V původním jednoduchém algoritmu se používal binární kód, který je jednoduchý a poměrně rozšířený, přesto jeho použití přináší určité problémy.

2) Ohodnocení jedinců

Ohodnocení jedinců neboli fitness se provádí porovnáváním jejich chromozomů s nějakým ideálním chromozomem – vzorem, který představuje uspokojivé řešení. Jsou to např. vědomosti člověka-experta potřebné k řízení určitého procesu. Ohodnocení je vždy závislé na konkrétní úloze a konkrétní představě řešení.

3) Reprodukce, vznik potomků

V tomto kroku by mělo dojít k vzniku potomků, které probíhá rozmnožováním jedinců. První úskalí představuje určení metody, jak vybrat konkrétní dva jedince do páru. Tento výběr by měl simulovat přírodní výběr, takže ti nejlépe ohodnocení jedinci by měli mít největší šanci. Přesto se i v přírodě někdy stává, že se občas reprodukce účastní i méně ohodnocený jedinec. Zřejmě nejrozšířenější formou implementace přirozeného výběru je použití tzv. ruletového mechanismu selekce [22]. Na rozdíl od klasické rulety nejsou u tohoto mechanismu obsahy výsečí čísel stejné (tudíž pravděpodobnost výběru také), ale odpovídají hodnotě fitness, tedy konkrétních ohodnocujících funkcí. Pravděpodobnost výběru je tedy přizpůsobena fitness

jedinců. Obvykle je předpokládáno, že plný kruh znamená pravděpodobnost 100 %, neboli souhrně 1. Aby se dosáhlo jisté náhodnosti výběru, tudíž možnosti reprodukce i pro slabší jedince, je výběr ovlivněn generací náhodného čísla od nuly do jedné, které se pak porovná s pravděpodobnostmi na ruletě a vybere se daný jedinec.

Darwinovský typ evoluce obsahuje dva procesy, křížení a mutace. Ty se musí při vzniku nového jedince realizovat. Nejjednodušší verzí křížení je jednobodové křížení, kdy se vybere jeden konkrétní gen (jedno konkrétní číslo v řetězci) a od tohoto genu si rodiče vymění geny; tedy vzájemně se prohodí všechna čísla v řetězci od daného konkrétního čísla včetně. Křížením dvou jedinců tedy vzniknou dva potomci, každý z nich má určité geny společné s rodiči. Pravděpodobnost, že k operaci křížení dojde, je poměrně vysoká (konkrétně 0,75-0,95) [22]. Nyní popíšeme operaci mutace. Jak je uvedeno v biologickém popisu – slouží jako prostředek proti degeneraci populace. Kdyby se populace pouze křížila, mohlo by dojít k vzniku velmi stejnorodých potomků a populace by předčasně konvergovala k nějakému meziřešení. Mutace se provádí tím, že se hodnoty náhodně vybraných genů chromozomu změní. Tím dochází k vytváření určité genové pestrosti populace a nedochází ke zmenšení genofondu.

4) Ohodnocení potomků

Probíhá obdobně jako u kroku 2.

5) Vznik nové populace

Dále se musí vybrat nová populace jedinců, neboť nyní je populace příliš velká (rodiče + potomci), tudíž existuje konkurenční boj. V přírodě je obvyklé, že rodiče a potomci jistou dobu spolu koexistují. Také je ale zřejmé, že za jistých podmínek tato koexistence probíhá na základě přírodního výběru a na tomto principu se poté populace opět sníží. Existují různé způsoby výběru nové populace, některé se nechávají inspirovat přirodou s tím, že z potomků a rodičů vybírají ty nejlepší jedince. Nejjednodušší možnost je nechat původní populaci vymřít a pracovat dále už jen s populací potomků.

6) Splnění či nesplnění zadané úlohy

V případě, že se cyklem algoritmu dosáhne vytvoření potomka nebo třídy potomků, které vyhovují danému vzoru řešení, pak se genetický algoritmus ukončí. V případě, že cyklus takového potomka nevytvoří, algoritmus dále pokračuje ve své práci v krocích 3-6. Tedy vytváří

stále nové potomky, které by díky evoluci měli konvergovat k uspokojivému řešení. Může se stát, že by daná konvergence probíhala příliš pomalu a teoreticky by algoritmus mohl vytvářet potomky do nekonečna. V tom případě se ukončení algoritmu může zajistit určitým počtem jeho iterací.

Nyní si můžeme jednotlivé kroky algoritmu ukázat na jednoduchém příkladě inspirovaném publikací [22]:

Příklad:

Generování původní populace:

jedinec	chromozom
1	1 0 1 1 0 1 0 1
2	0 0 0 1 1 1 0 1
3	1 1 0 1 0 1 1 1
4	0 1 0 0 1 0 1 0

Ohodnocení populace:

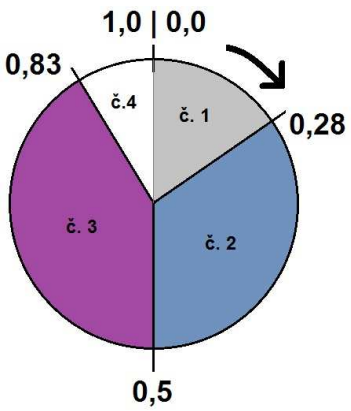
jedinec	chromozom	fitness
1	1 0 1 1 0 1 0 1	5
2	0 0 0 1 1 1 0 1	4
3	1 1 0 1 0 1 1 1	6
4	0 1 0 0 1 0 1 0	3

Populace obsahuje náhodně vygenerované jedince, jako ohodnocení bylo použito počet jedniček v chromozomu jedince.

Určení pravděpodonosti výběru:

jedinec	chromozom	fitness	pravděp. [%]	kumulovaná pravděp.
1	1 0 1 1 0 1 0 1	5	28 %	0-0,28
2	0 0 0 1 1 1 0 1	4	22 %	0,28-0,5
3	1 1 0 1 0 1 1 1	6	33 %	0,5-0,83
4	0 1 0 0 1 0 1 0	3	17 %	0,83-1

Pravděpodobnost výběru se určila z celkového součtu fitness (5+4+6+3=18) a poté se určily jednotlivé pravděpodobnosti jako podíl fitness jedinců ku celkovému součtu fitness. Např. tedy u třetího jedince: 6/18 = 0,33. Lépe si to lze představit na obrázku 15.



Obr. 15:
Ruleta pravděpodobnostího výběru
jedinců č. 1 - č. 4

Vybrání párů pro reprodukci a následné křížení jedinců:

1. pár:	jedinec č. 2:	0 0 0 1 1 1 0 1	potomek č. 1:	0 1 0 1 0 1 1 1
	jedinec č. 3:	1 1 0 1 0 1 1 1	potomek č. 2:	1 0 0 1 1 1 0 1
2. pár:	jedinec č. 3:	1 1 0 1 0 1 1 1	potomek č. 3:	1 1 0 1 0 1 0 1
	jedinec č. 1:	1 0 1 1 0 1 0 1	potomek č. 4:	1 0 1 1 0 1 1 1

Jedinci byli vybráni dle ruletového pravidla s generováním náhodného čísla. Např. pro 1. pár byly vygenerovány čísla 0,3 a 0,6 - tedy 2. a 3. jedinec. Můžete si všimnout, že křížení se nezúčastní jedinec č. 4, neboť měl pro vybrání nejmenší šanci.

Mutace:

potomek č. 1:	0 1 0 **1** 0 1 1 1	*po mutaci:* potomek č. 1:	0 1 0 0 0 1 1 1
potomek č. 2:	1 0 0 1 1 1 0 1	*po mutaci:* potomek č. 2:	1 0 0 1 1 1 0 1
potomek č. 3:	1 1 0 1 0 1 0 1	*po mutaci:* potomek č. 3:	1 1 0 1 0 1 0 1
potomek č. 4:	1 0 1 1 **0** 1 1 1	*po mutaci:* potomek č. 4:	1 0 1 1 1 1 1 1

Nižší pravděpodobnost způsobila, že mutace proběhla pouze u dvou potomků, kdy u potomka č. 1 se změnil 4. gen a u potomka č. 4 se změnil 5. gen.

Porovnání fitness původní a nově vytvořené populace:

jedinec	chromozom	fitness	potomek	chromozom	fitness
1	1 0 1 1 0 1 0 1	5	1	0 1 0 0 0 1 1 1	4
2	0 0 0 1 1 1 0 1	4	2	1 0 0 1 1 1 0 1	5
3	1 1 0 1 0 1 1 1	**6**	3	1 1 0 1 0 1 0 1	5
4	0 1 0 0 1 0 1 0	3	4	1 0 1 1 1 1 1 1	**7**
CELKEM:		**18**			**21**

Z porovnání kvality jedinců (řešení dané úlohy) se lze přesvědčit, že již při prvním cyklu tohoto genetického algoritmu se dosáhlo zlepšení fitness jak na úrovni nejlepšího jedince (modře vyznačeno), tak na úrovni celé populace (žlutě vyznačeno). Vzhledem k zadané úloze jsme tedy prací tohoto algoritmu dosáhli řady možných řešení, které jsou obecně kvalitnější než na počátku, dokonce jedno řešení skoro představuje globální optimum.

Zda-li by se nyní algoritmus ukončil či ne, pak záleží na konkrétní úloze. Jestliže by pro danou úlohu postačovalo jako dostatečně dobré řešení chromozom potomka 4, pak by se algoritmus ukončil. Jestliže by toto řešení nepostačovalo, algoritmus by dále pokračoval reprodukcí nových potomků.

Protože genetický algoritmus pracuje pouze se syntaktickým popisem problému (chromozomy) a nemůže zachytit příslušnou sémantiku, je třeba mít na zřeteli, že vhodný způsob reprezentace, využití odpovídajících genetických operátorů a správná volba ohodnocující funkce jsou naprosto klíčovými faktory, které rozhodnou o úspěchu či nezdaru konkrétní aplikace genetického algoritmu [22].

4.4 Optimalizace genetických algoritmů

Z hlediska výpočetního řešení úloh pomocí genetických algoritmů je předešlý tzv. jednoduchý algoritmus poměrně nevýkonný a hodí se pouze pro úzkou část úloh. Tato podkapitola obsahuje různá vylepšení a rozšíření jednotlivých kroků algoritmu tak, aby ve výsledku byl algoritmus efektivnější, výkonnější a univerzálnější.

Kódování

Použití jednoduchého binárního kódování přináší jeden problém. Tento problém nastává v případě, kdy budeme např. binárně kódovat určitá čísla původně z desítkové soustavy a tato sousední desítková čísla budou vyjadřovat učitou souvislost – např. gradient. V tomto případě dochází k velké různorodosti mezi binárně kódovaným chromozomem a skutečností, kterou chromozom představuje – tedy číslem v desítkové soustavě.

Např. budeme binárně kódovat čísla 2 a 3, tedy 2 = 0 1 0, 3 = 0 1 1. Lze si všimnout, že v tomto případě je binární kód v pořádku, neboť jedna změna v desítkové reprezentaci způsobila jednu změnu v binárním chromozomu. Ale v případě čísel 3 a 4 už nastává odlišný jev: 3 = 0 1 1, 4 = 1 0 0. V tomto případě došlo také k jedné změně v desítkové reprezentaci, ale v binárním chromozómu proběhly hned změny tři.

Tento problém se řeší buď používáním jiných typů chromozomů než binárních, např. tedy reálných, komplexních, permutačních atd. To přináší určité komplikace, musí se použít např. jiné genetické operátory a jiný typ výběru. Jiné řešení nalezl Frank Grey, který použil sice binární soustavu, ale vymyslel odlišný typ kódování. Pro jeho kódování je charakteristické, že při jedné změně v desítkové reprezentaci dojde vždy pouze k jedné

změně v binárním chromozomu. Nechá se realizovat více způsoby, lze použít techniku zrcadlení [20]:

				000	->	0
		1		001	->	1
	0	1		011	->	2
0	1	0	…	010	->	3
1	1	0	…	110	->	4
	0	1		111	->	5
		1		101	->	6
				100	->	7

nebo využít logické funkce XOR (exkluzivní implikace neboli nonekvivalence).

Ohodnocující funkce

Vychází samozřejmě z konkrétní úlohy, ale v mnoha případech se daná úloha nechá pojmout jako optimalizační, kdy se jako ohodnocující funkce použije tzv. funkce účelová. Lze také využít mechanismy soutěže, kdy lze ohodnocení provádět na základě soubojů mezi jedinci (jejich chromozomy) [22]. Nechá se použít např. souboje každý s každým a výsledná tabulka vítezství určí ohodnocení nebo lze použít klasického turnajového pavouku, kdy vítěz postupuje do dalšího kola a získává určité body do hodnocení. Lze vymyslet i mnoho dalších způsobů.

Zcela zvláštním případem z hlediska ohodnocující funkce je tzv. interaktivní evoluce, kdy je do procesu hodnocení individuí zapojen přímo uživatel; např. algoritmus který slouží k identifikaci zločinců [22].

Výběr jedinců

Jednoduchý ruletový systém není z praktického hlediska moc efektivní, neboť dílem náhody se může stát, že se nejlepší jedinec nevybere (např. k reprodukci) a to způsobí ztrátu genetické informace. Sníží se genofond celé populace a může to véct k degeneraci.

Aby toto nenastalo, tak se např. využívá vícekuličková ruleta nebo tzv. setrvalý stav, kdy se nejhorší jedinci z dané generace nechají vymřít a jsou doplněni jedinci z další generace [22]. Nebo lze využít i tzv. elitismus, kdy se před samotným výběrem (např. ruletou)

automaticky vyberou nejlepší jedinci, toto řešení výrazně zlepšuje průběh algoritmu, neboť se zamezí ztrátě nejlepších řešení [24].

Genetické operace

Základní darwinovské evoluční operace jsou mutace a křížení. Obojí může být buď ve formě jednobodové, jak si lze všimnout u příkladu v předchozí podkapitole, nebo může být použita forma vícebodová, kdy se mění více částí chromozomů. Dále se využívá dalších modifikací těchto základních operací např. použitím různých aritmetických operací např. použití aritmetického průměru genů rodičů nebo rozdílu mezi geny rodičů, apod. [22]. Také se nechá použít tzv. operátor inverze, který obrací část řetězce nebo řetězec celý.

4.5 Aplikace

Problém batohu

Máme množinu věcí, které se mají umístit do batohu. Problém může být definován různě, např. „*Urči největší možný počet věcí, které lze do batohu umístit*“ nebo „*Umísti do baťohu co nejvíce věcí, které jsou světlé“*, apod. Obvykle je batoh omezen určitými faktory, tedy např. velikost, nosnost, tvar, atd. Jestliže budeme problém batohu řešit genetickými algoritmy, pak chromozomy můžeme kódovat binárními řetězci b_1, b_2, b_3 … b_n, kde $b_i \in \{0,1\}$, i = 1,2, … , n; kde hodnota 1, resp. 0 na i-té pozici znamená, že daná věc bude či nebude do batohu vložena; fitness funkce přímo odpovídá kriteriální funkci problému [20].

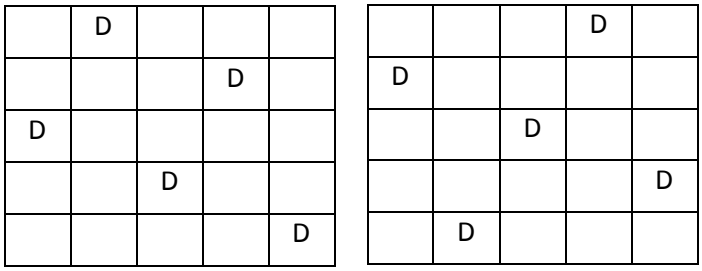
Praktickým příkladem vícerozměrného problému batohu je problém investování, kdy při omezených zdrojích máme z jisté množiny investičních akcií s předpokládanými výnosy volit takovou jejich podmnožinu, abychom maximalizovali celkový zisk [20].

N – dam

Jedná se o typický testovací problém, kdy se musí na šachovnicové pole o rozměru N x N umístit určitý počet dam, tedy N – dam. Problém je v tom, že se dámy musí umístit tak, aby se vzájemně neohrožovaly. Je to jednoduše definovaný problém, ale má velmi mnoho řešení. Pokud se nad daným problémem zamyslíme, možnosti umístění dam jsou omezené. Nelze např. umístit více než jednu dámu na jedno políčko a z logiky věci také plyne, že by bylo dobré předem vyloučit ty možnosti, kde by dámy stály v jedné řadě, sloupci či úhlopříčce.

V publikaci [22] je tento problém řešen využitím permutačního kódování, kdy u N = 5; číslo *i* na *j*-té pozici v každé permutaci pak reprezentuje dámu, která byla umístěna na *i*-tý řádek v *j*-tém sloupci. Např. chromozom 3 1 4 2 5 znamená, že první dáma bude umístěna na 3. řádku v 1. sloupci, druhá dáma bude umístěna na 1. řádku v 2. sloupci, atd.

Lépe si to lze představit na obr. 16 upraveném na základě [22].



Obr. 16: N-dam (3 1 4 2 5 ; 2 5 3 1 4)

Tím, že bylo užito permutační kódování, se zajistilo omezení možnosti umístění více dam na jedno políčko, neboť v permutacích se čísla neopakují. Dále bylo tímto způsobem vyřešeno i omezení některých nesmyslných možností, konkrétně umístění více dam do jednoho řádku či sloupce. K řešení tohoto problému je výhodnější využít kombinace genetického algoritmu s některými tradičními technikami, vzniká pak tzv. hybridní algoritmus [22]. Tento problém má řadu praktických aplikací, např. řízení letového provozu, testování komunikačních systémů, datová komprese a další [22].

Řízení plazmového reaktoru

K řešení tohoto problému nebylo využito genetických algoritmů, nýbrž algoritmů evolučních. Podrobně i s výsledky měření experimentu je tento problém popsán v publikaci [20], uvedu zde pouze krátký výtah. Jedná se o problém aktivní kompenzace šumu ze signálu generovaném Langmuirovou sondou. Plazma řízená radiovými frekvencemi je vnitřně nelineární. Okolo sondy se vytváří potenciálový val, který silně zkresluje užitečný signál snímaný sondou. Cílem bylo použití evolučních technik k syntéze sedmi harmonických signálů, přičemž u každého signálu se evolučně nastavovaly dva parametry a to frekvence a amplituda. Celkem tedy 14 neznámých parametrů. Na tento problém lze nahlížet jako na hledání extrému v 15-ti parametrovém prostoru, kdy 15. osa představuje vhodnost použití jedince, tedy příslušného řešení. Zmíněných sedm harmonických signálů bylo použito k syntéze výsledného signálu, který měl v optimálním případě kompenzovat rušení. Vzhledem k tomu,

že dynamika plazmy je silně nelineární, musí kompenzační signál vhodně reagovat nejen na signál budící, ale i na rušení plazmatu.

Součástí zařízení byl harmonický generátor se 14-ti D/A převodníky. Signál tohoto generátoru byl pouštěn na Langmuirovu sondu, plovoucí potenciál (jenž vyrovnává šum) interpretovaný jako vhodnost jedince byl převáděn A/D převodníkem do počítače a tam zpracován. Počet řešení byl konečný, leč obrovský (konkrétně v řádu 10^{50}, tedy 10^{41} let výpočetního času). Pomocí klasických metod neřešitelné. Součástí zařízení byl i digitální osciloskop pro měření kompenzačních signálů.

Délka jedince reprezentovala počet optimalizovaných parametrů. Maximální počet ohodnocení účelové funkce byl 12000, cca 4 minuty výpočetního času. Z poslední populace byl vybrán nejlepší jedinec, tedy nejlepší možné řešení.

Jednalo se o maximalizační úlohu, která byla stížena tím, že pozice a hodnota globálního maxima se měnila s časem. To bylo způsobeno právě dynamickým chováním plazmy, v tomto případě tzv. driftem. Přesto se při opakování experimentu vždy dosáhlo obdobných výsledků. Použití evolučních algoritmů se ukázalo jako velmi efektivní. Lze je úspěšně nasadit pro řešení velmi složitých úloh typu černá skříňka, včetně těch s proměnlivou dynamikou.

5. Fuzzy logika

5.1 Úvod

Čtvrtá kapitola je věnována fuzzy logice, teorii dnes již běžně rozšířené do praxe, kdy zejména fakt snadnosti tvorby a přiblížení techniky lidskému uvažování předurčuje úspěšné nasazení v mnoha technických, i mimotechnických oblastech. Kapitola popisuje vznik teorie fuzzy množin dle historického vývoje, základní matematický popis doplněný jednoduchými příklady. Snahou bylo poukázat na souvislosti mezi fuzzy množinami, fuzzy logikou a aplikacemi, kdy myšlenkovým propojením těchto částí lze získat určitý nadhled, tím i schopnost použití fuzzy přístupu v různých činnostech lidského bádání.

5.2 Rozdíl mezi klasickou a fuzzy množinou

Matematický popis teorie fuzzy množin vychází z rozšíření klasické teorie množin. Přestože se rozšíření provede úpravou pouze jediného parametru, tak důsledky, které vyplynou z této jediné úpravy, jsou dalekosáhlé. Otevírají nové obzory v práci s množinami, zavádějí nové operace, klasické pojmy získávají nový význam (případně mají významů vícero) a mnohé další. Tím vším obohacují klasickou teorii množin a práci s množinami více přibližují popisu běžného života s řešením běžných problémů, kde si klasická teorie množin občas neví rady.

Již od dob stoické logiky a Eukleidovy geometrie v době Antiky jasně vyplývalo, že výrok či důkaz je buď pravdivý, nebo nepravdivý. Buď je den, nebo noc; sklenice je prázdná, nebo plná; člověk je buď dobrý, nebo zlý; elektrický obvod je buď zapnutý, nebo vypnutý. Prvek do množiny patří, nebo nepatří. To je dvouhodnotová logika: 0 nebo 1, žádný jiný stav neexistuje, jsou jasně vymezené, ostré hranice mezi těmito dvěma stavy. Pokud ale budeme tuto klasickou teorii aplikovat do našeho běžného života, může nastat řada problémů. Když se rozednívá – to je den, nebo noc? A od kdy to je noc a od kdy to je den? Běžně řekneme: „Je to spíše ještě noc.“ nebo „To už je skoro den.“ Co taková sklenice, ve které je do poloviny jejího objemu nalita voda? Je plná nebo prázdná? Asi bychom řekli, že je poloprázdná nebo poloplná. Honza ukradl svačinu, je zlý. Ale před měsícem zachránil dítě z hořícího domu. Jaký je? Dále např. obvod je zapnutý nebo vypnutý, ale co přechodové jevy? Kdy je obvod ještě zapnutý a kdy je již vypnutý? Situace v reálném životě většinou nemají úplně ostré hranice, svět není černobílý, kolikrát nelze stav naprosto

přesně určit. Prvek do množiny buď patří, nebo nepatří; někdy patří jenom napůl (např. u té sklenice), někdy jenom trochu.

5.2.1 Charakteristická funkce

V této kapitole bude popsána již zmíněná změna parametru, kterak obohacuje klasickou teorii množin. Jedná se o základní pojem v práci s množinami a to, jestli prvek do množiny patří nebo nepatří. Tuto vlastnost prvku lze vyjádřit právě charakteristickou funkcí (jinak popisovanou také jako funkce příslušnosti).

V případě klasické teorie množin, je situace jednoduchá. Prvek buď do množiny patří, nebo nepatří. Prvky jednoznačně splňují či nesplňují kritéria, která rozhodují, zda prvek do dané množiny patří nebo nepatří. Vzniká tak klasická množina s ostrými hranicemi, tzv. crisp množina. Takovouto množinu (např. *A*) lze popsat díky charakteristické funkci μ_A. Definiční obor charakteristické funkce μ_A je universum *X* [25]. Obor hodnot μ_A u této ostré množiny A je buď 0, nebo 1 (patří, nepatří):

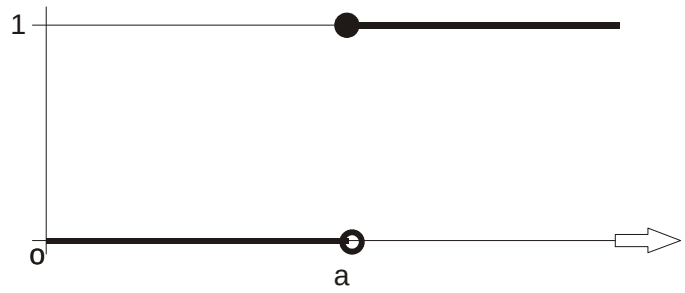
$$\mu_A(x) = \begin{cases} 1 & pro\, x \in A \\ 0 & pro\, x \notin A \end{cases} \tag{8}$$

Množina *A* je svou charakteristickou funkcí μ_A jednoznačně určena [25]:

$$A = \{x \in X : \mu_A(x) = 1\} = \{x \in X : \mu_A(x) > 0\} \tag{9}$$

(množina A je složena ze všech prvků x, které splňují tu podmínku, že obor hodnot charakteristické funkce těchto prvků je roven 1, tedy všech prvků, jejichž obor hodnot charakteristické funkce je větší než 0)

Charakteristickou funkci ostré množiny si lze lépe představit na následujícím obrázku (obr. 17) zobrazení intervalu $\langle a, \infty)$.



Obr. 17: Charakteristická funkce ostré množiny

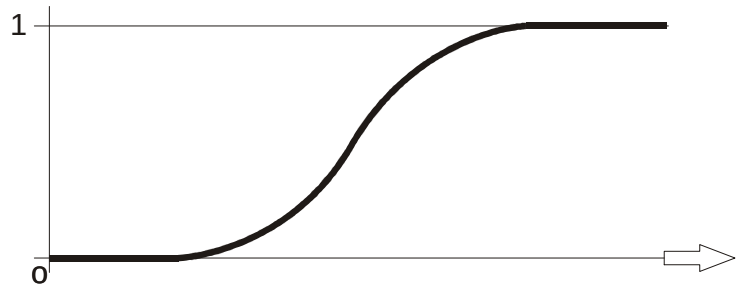
V případě fuzzy množin je situace odlišná. U některých prvků množiny se sice může stát, že prvek do dané množiny zcela patří nebo zcela nepatří, tím je situace obdobná jako u klasické teorie. U jiných prvků ale může nastat, že daný prvek do množiny patří pouze napůl, nebo jenom trochu. Tudíž obor hodnot charakteristické funkce μ_A může nabývat i jiných hodnot než 0 nebo 1 (např. 0,5; 0,001; 0,7; 0,99, …).

Obecně se nechá říci, že obor hodnot charakteristické funkce fuzzy množiny může nabývat libovolných hodnot (např. 1, 5, 7, 300, …). Pro jednoduchou ilustraci a pochopení zůstaneme v tomto případě pouze u oboru hodnot z intervalu $\langle 0,1 \rangle$. Definiční obor je již známé universum *X* z předchozího příkladu. Pro každý prvek $x \in X$ hodnota $\mu_A(x) \in \langle 0,1 \rangle$ říká, do jaké míry je *x* prvkem fuzzy množiny *A* [25].

Každá funkce z *X* do $\langle 0,1 \rangle$ určuje jednoznačně nějakou fuzzy množinu [25]:

$$\mu_A : X \rightarrow \langle 0,1 \rangle \tag{10}$$

Charakteristickou funkci fuzzy množiny si lze lépe představit na následujícím obrázku (obr. 18) zobrazení intervalu $\langle a, \infty)$:



Obr. 18: Charakteristická funkce fuzzy množiny

5.2.2 Lukasiewiczova logika

S myšlenkou vícehodnotové logiky se začali matematici zabývat již v 1. polovině 20. století, nejznámější je Lukasiewiczova a Gödelova logika. Dosud bylo v této práci hovořeno pouze o množinách, přestože název této kapitoly je fuzzy logika. Tabulka na následující straně, která byla převzata z publikace [26], poukazuje právě na souvislost mezi teorií množin, logikou (výrokovým počtem) i Boolovou algebrou, hojně se vyskytující v technické praxi.

Teorie množin	Boolova algebra	Výrokový počet
P(X)	B	F(V)
$\cup$	+	$\vee$
$\cap$	.	$\wedge$
-	-	-
X	1	1
O	0	0
$\subseteq$	$\leq$	$\Rightarrow$

Tab. 1 Interdisciplinární souvislosti

Nyní budeme sledovat, jak se mění význam klasických operací při použití vícehodnotových logik. Budou uvažovány základní operace: negace ($\neg$), konjunkce ($\wedge$), disjunkce ($\vee$), implikace ($\Rightarrow$), ekvivalence ($\Leftrightarrow$).

Nejdříve si ukážeme na následující pravdivostní tabulce, jak vypadají pravdivostní hodnoty klasické dvouhodnotové logiky, kde *a*, *b* jsou pravdivostní hodnoty libovolných dvou výroků :

a	*b*	$\neg a$	$\neg b$	$\wedge$	$\vee$	$\Rightarrow$	$\Leftrightarrow$
0	0	1	1	0	0	1	1
0	1	1	0	0	1	1	0
1	0	0	1	0	1	0	0
1	1	0	0	1	1	1	1

Tab. 2 Pravdivostní tabulka dvouhodnotové logiky

Lukasiewicz nejprve přišel s myšlenkou, že původní dvouhodnotovou logiku rozšíří o hodnotu třetí. Jeho logika uvažuje pouze nad intervalem $\langle 0,1 \rangle$. První dvě pravdivostní hodnoty zachovává, 0 a 1 (pravda, nepravda). Doplňuje třetí pravdivostní hodnotu, a to ½

(může nabývat hodnoty např. neznámo). Pro výpočet pravdivostních hodnot libovolné vícehodnotové logiky odvodil Lukasiewicz pět základních axiomů [26]:

$$\begin{aligned} &\neg a = 1 - a \\ &a \wedge b = \min(a, b) \\ &a \vee b = \max(a, b) \\ &a \Rightarrow b = \min(1, 1 + b - a) \\ &a \Leftrightarrow b = 1 - |a - b| . \end{aligned} \tag{11}$$

Jaké pravdivostní hodnoty dostaneme pro stejné operace jako v předchozím případě je zřejmé z následující tabulky převzaté z publikace [26]:

a	b	$\neg a$	$\neg b$	$\wedge$	$\vee$	$\Rightarrow$	$\Leftrightarrow$
0	0	1	1	0	0	1	1
0	½	1	½	0	½	1	½
0	1	1	0	0	1	1	0
½	0	½	1	0	½	½	½
½	½	½	½	½	½	1	1
½	1	½	0	½	1	1	½
1	0	0	1	0	1	0	0
1	½	0	½	½	1	½	½
1	1	0	0	1	1	1	1

Tab.3 Pravdivostní tabulka tříhodnotové logiky

Z pravdivostní tabulky tříhodnotové logiky vyplývá, že uvedené základní operace dostávají nový význam. Z porovnání obou tabulek lze například zjistit, že pravdivostní hodnota operací nabývá hodnoty 1 (pravda) v mnohem více případech než u logiky dvouhodnotové. Někdy dokonce operace nabývají i třetí hodnoty ½ (neznámo). Lze si tohoto jevu všimnout např. v případě negace, kdy je pravdivostní hodnota rovna ½. Negace dostává jiný význam. Může tedy existovat např. i více druhů negací. Sice k těmto jevům dochází v odlišných případech než u logiky dvouhodnotové, ale lze si právě povšimnout, že charakter uvedených operací se rozšířil (obohatil), navíc lze zavést i operace nové. To jsou také jedny z důvodů, proč používat vícehodnotové logiky. Samozřejmě toto rozšíření významu, charakteru operací se promítá nejen do logiky, ale také do souvisejících oborů jako je teorie množin a Boolova algebra.

Z porovnání obou tabulek lze vyčíst ještě jednu zajímavou souvislost. V tabulce tříhodnotové logiky mají všechny operace za podmínky, že pravdivostní hodnota výroků *a*, *b* jsou buď 0, nebo 1 naprosto totožný tvar jako v tabulce logiky dvouhodnotové. Z tohoto poznatku lze tedy usoudit, že logika dvouhodnotová je speciální případ logiky tříhodnotové.

Vzhledem k tomu, že Lukasiewicz odvodil své axiomy (11) pro libovolnou vícehodnotovou logiku, lze usuzovat, že pro n-hodnotovou logiku, kdy $n \to \infty$, budou pravdivostní hodnoty reálná čísla z intervalu $\langle 0,1 \rangle$ [26]. A pro všechny logiky, kdy $n < \infty$, bude platit, že to jsou speciální případy této nekonečně hodnotové logiky. Tento případ můžeme nazývat jako Lukasiewiczův hodnotový interval a značit ho jako ***Ł*** (dle anglického pojetí: „*Lukasiewicz interval-valued propositional logic system*“).

5.2.3 Základní fuzzy pojmy

Přesto, že obor hodnot charakteristické funkce fuzzy množiny může nabývat i jiných hodnot než z ***Ł*** intervalu $\langle 0,1 \rangle$ a obecně charakteristická funkce může být funkcí vektorovou (může být i více měřítek příslušnosti k množině), se v reálných aplikacích nejvíce používá jako obor hodnot charakteristické funkce fuzzy množiny již několikrát zmíněný interval $\langle 0,1 \rangle$ a charakteristická funkce je vnímána jako funkce skalární.

Jádro (core) [25]**:** $core(A) = \{ x \in X : \mu_A(x) = 1 \}$ (12)

(jádro fuzzy množiny A je ostrá množina obsahující všechny prvky x dané fuzzy množiny, pro něž obor hodnot charakteristické funkce je rovný 1)

Nosič (support) [25]**:** $Supp(A) = \{ x \in X : \mu_A(x) > 0 \}$ (13)

(nosič fuzzy množiny A je ostrá množina obsahující všechny prvky x dané fuzzy množiny, pro něž obor hodnot charakteristické funkce je větší než 0)

Normální, subnormální fuzzy množina [27]**:** $core(A) \neq 0$ (14)

(fuzzy množina A se nazývá normální, jestliže existuje alespoň jeden prvek v jádře fuzzy množiny. V opačném případě se fuzzy množina A nazývá subnormální)

Obor pravdivostních hodnot (range, level) [25]**:**

$$Range(A) = \{ \alpha \in \langle 0,1 \rangle : (\exists x \in X : \mu_A(x) = \alpha) \} \quad (15)$$

(obor pravdivostních hodnot splívá s oborem hodnot charakteristické funkce neprázdné fuzzy množiny A)

Výška (height): $h(A) = \sup Range(A)$ **(16)**

(výška fuzzy množiny A je rovna suprému oboru pravdivostních hodnot Range(A))

α-hladina (α-level): $A^{\alpha} = \{ x \in X : \mu_A(x) = \alpha \}$ **(17)**

(např. 0,5 hladina obsahuje všechny prvky x, pro něž platí, že obor hodnot charakteristické funkce fuzzy množiny A je roven 0,5)

α-Řez (α-cut): $A_{\alpha} = \{ x \in X : \mu_A(x) \geq \alpha \}$ **(18)**

(např. 0,5 řez obsahuje všechny prvky x, pro něž platí, že obor hodnot charakteristické funkce fuzzy množiny A je větší nebo roven 0,5)

5.3 Popis pomocí řezů hladiny

Fuzzy množiny lze popisovat různými způsoby, kdy každý způsob popisu se lépe hodí pro určité práce s množinami. Jeden z druhů popisu fuzzy množiny je popis pomocí charakteristické funkce. Do zápisu fuzzy množiny se uvede výčet prvků dané fuzzy množiny s tím, že ke každému prvku bude zapsán obor hodnot charakteristické funkce daného prvku (tzn. míru, jak daný prvek do dané množiny patří). Do tohoto zápisu se nevkládají prvky, jejichž obor hodnot charakteristické funkce má hodnotu 0 (tzn. všechny ty, které do dané množiny zcela jistě nepatří).

Příklad zápisu fuzzy množiny *A*: $\mu_A = \left\{ \left(2, \frac{1}{2}\right), \left(3, \frac{1}{4}\right), \left(5, \frac{1}{4}\right), (4,1) \right\}$.

Z tohoto popisu fuzzy množiny *A* je možné získat základní kategorie, např. :

$core(A) = \{4\}, \qquad Supp(A) = \{2,3,4,5\},$

$A^{0,5} = \{2\}, \qquad A_{0,4} = \{2, 4\}.$

Další z možných popisů fuzzy množin je popis pomocí α – řezů. **Věta o systému řezů:** Nechť *M*: $\langle 0,1 \rangle \rightarrow P(x)$ je systém řezů fuzzy množiny $A \in F(X)$, tj. $M = A_{\alpha}$ [25]. Pak *M* splňuje podmínky [25]:

$$M(0) = X, \tag{19}$$

$$0 \le \alpha < \beta \le 1 \Rightarrow M(\alpha) \supseteq M(\beta), \tag{20}$$

$$0 < \beta \le 1 \Rightarrow M(\beta) = \bigcup_{\alpha<\beta} M(\alpha). \tag{21}$$

Každé zobrazení *M*: $\langle 0,1\rangle \to P(x)$ splňující podmínky (19), (20), (21) je systémem řezů nějaké fuzzy množiny $A \in F(X)$, tj. $M = A_\alpha$ [25].

Příklad: Na univerzu $X = \{u, v, t, z\}$ je dána fuzzy množina popsaná pomocí charakteristické funkce $\mu_A = \{(u; 0{,}25), (t; 0{,}5), (z; 1)\}$. Popište ji pomocí systému řezů.

Nejdříve rozdělíme interval na podintervaly, dáme tak vzniknout jednotlivým α – hladinám: $\langle 0,1\rangle \approx 0, (0; 0{,}25\rangle, (0{,}25; 0{,}5\rangle, (0{,}5; 1\rangle$. Nyní budeme přiřazovat prvky daným α – řezům (A_α) tak, aby splňovaly podmínky jednotlivých řezů:

Pro $\alpha = 0$**:** A_α **= *X* = *u, v, t, z***

(viz rovnice (19), která říká, že v případě, kdy je obor hodnot charakteristické funkce daných prvků větší nebo roven 0, základní α – řez obsahuje všechny prvky dané fuzzy množiny).

Pro $\alpha \in (0; 0{,}25\rangle$**:** A_α **= *u, t, z***

(viz rovnice (18), která objasňuje vztah mezi řezem A_α a char. funkcí μ_A : z výčtu prvků vypadl prvek **v**, neboť jeho obor charakteristické funkce neleží v intervalu $\alpha \in (0; 0{,}25\rangle$ *).*

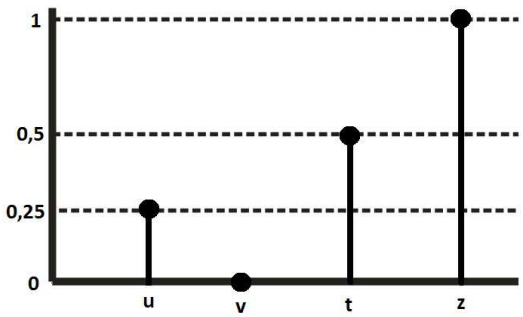
Pro $\alpha \in (0{,}25; 0{,}5\rangle$**:** A_α **= *t, z***

*(totéž jako v předchozím případě, vypadl prvek **u**).*

Pro $\alpha \in (0{,}5; 1\rangle$**:** A_α **= *z***

*(totéž jako v předchozím případě, vypadl prvek **t**).*

$$A_\alpha = \begin{cases} u, v, t, z & \text{pro } \alpha = 0 \\ u, v, t & \text{pro } \alpha \in (0; 0{,}25\rangle \\ t, z & \text{pro } \alpha \in (0{,}25; 0{,}5\rangle \\ z & \text{pro } \alpha \in (0{,}5; 1\rangle \end{cases}$$



Obr. 19: α - řezy fuzzy množiny *A*

5.4 Fuzzifikace

Řešení většiny praktických úloh pomocí fuzzy množin má velmi podobný průběh, musí se provést následující kroky [26]:

1. Změřit vstupní veličiny
2. Zobrazit změřené veličiny ve vhodném měřítku na použitá univerza
3. Převést vstupní data na fuzzy data
4. Provést vlastní fuzzy odvození výstupní fuzzy množiny
5. K takto získané fuzzy množině nalézt vhodnout ostrou hodnotu

První krok nás v tuto chvíli nezajímá. Druhý krok většinou nazýváme normalizace a spočívá v tom, že různé proměnné, jejichž hodnoty se pohybují v různých rozsazích transformujeme tak, aby všechny nabývaly hodnot z jistého normalizovaného univerza, např. intervalu [0, 1] [26]. Třetí krok se nazývá **fuzzifikace** a spočívá v tom, že každé ostré hodnotě z normalizovaného univerza přiřadíme stupeň příslušnosti (obor hodnot char. funkce) do fuzzy množiny [26]. Poslední krok bývá zpravidla převedení výstupní fuzzy množiny na ostrá data, čemuž se říká **defuzzifikace** (např. pomocí těžiště výstupní fuzzy množiny).

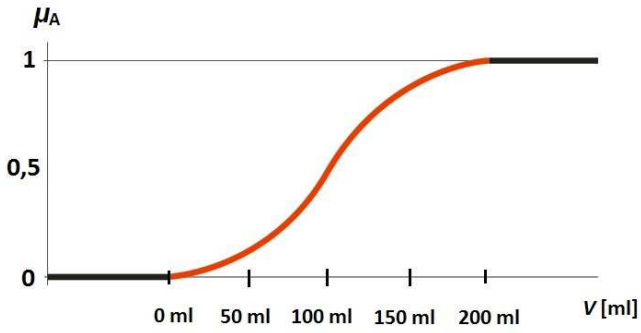
Fuzzifikace je proces, v jehož průběhu se převádí problém či úloha do fuzzy logiky. Tento proces lze rozdělit na dvě části. 1. část se týká právě přímo převodu daného problému či úlohy do fuzzy logiky. Nejdříve se určí měřítko, dle kterého se bude přiřazovat příslušnost prvků k dané množině, tzn. určí se, jak bude vypadat charakteristická funkce dané množiny. 2. část se věnuje řízení nebo řešení daného problému, kdy se převádí klasická logická funkce na fuzzy logickou funkci. Z důsledků Lukasiewiczovy logiky vyplývá, že v případě, kdy daný prvek má obor hodnot charakteristické funkce roven 0 nebo 1, musí fuzzy funkce dávat stejný výsledek řízení jako klasická logická funkce (neboť platí, že logika dvouhodnotová je pouze speciální případ logiky vícehodnotové). Tato podkapitola bude věnována pouze 1. části fuzzifikace, tedy převodu úlohy na fuzzy logiku. 2. část fuzzifikace je postupně popisována v dalších podkapitolách (5.5 až 5.7).

Ve většině praktických příkladů se jako obor hodnot charakteristické funkce fuzzy množiny považují všechna reálná čísla z intervalu $\langle 0,1 \rangle$, tedy Lukasiewiczův hodnotový interval ***L.*** V následujících příkladech bude vždy stručně popsána daná situace, dále následuje přiřazení prvků k oboru hodnot charakteristické funkce a následuje vizualizace charakteristické funkce.

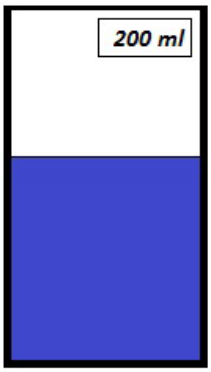
Příklad: Jsou sklenice o objemu 200 ml plné?

Máme skupinu sklenic o daném objemu, do každé z nich je nalita kapalina a ptáme se, zda-li je sklenice plná. Určíme tedy ke každému prvku (každé sklenici) obor hodnot char. funkce (příslušnost) k množině „plná sklenice“.

V = 0 ml	$\mu_A = 0$	sklenice je prázdná
V = 50 ml	$\mu_A = 0{,}25$	sklenice je trochu plná
V = 100 ml	$\mu_A = 0{,}5$	sklenice je poloplná
V = 150 ml	$\mu_A = 0{,}75$	sklenice je skoro plná
V = 200 ml	$\mu_A = 1$	sklenice je plná



Obr. 20: Char. funkce fuzzy množiny



Obr. 21: Sklenice

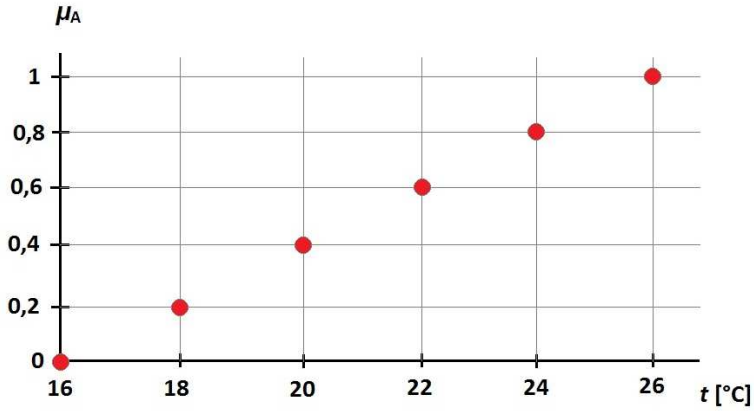
Příklad: Je v domě horko?

Pro jednoduchost uvedeme, že v domě je instalováno pouze jedno čidlo (či operátor), které měří teplotu na celé stupně Celsia.

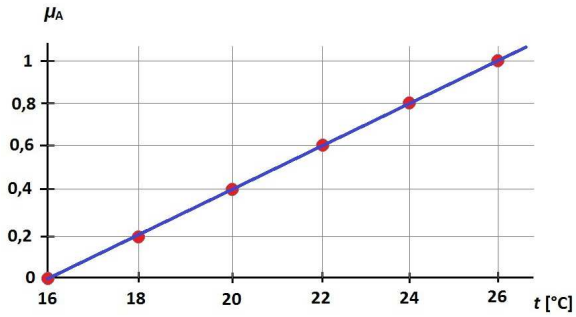
t = 16 °C	$\mu_A = 0$	zcela jistě ne
t = 18 °C	$\mu_A = 0{,}2$	moc ne
t = 20 °C	$\mu_A = 0{,}4$	spíše ne
t = 22 °C	$\mu_A = 0{,}6$	spíše ano
t = 24 °C	$\mu_A = 0{,}8$	skoro ano
t = 26 °C	$\mu_A = 1$	zcela jistě ano

Obrázek 22 znázorňuje pouze body, jimiž charakteristická funkce prochází. Charakteristická funkce je obecně nelineární a na jejím tvaru pak závisí chování celého systému.

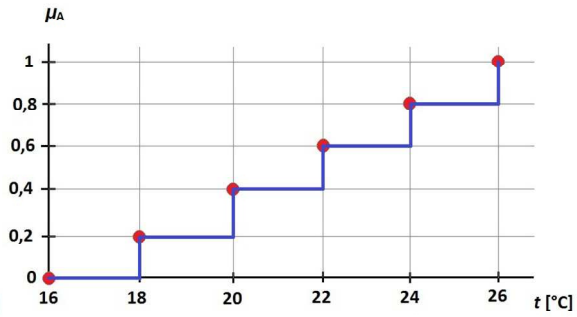
Na obrázcích 23 a 24 si lze všimnout, že charakteristická funkce může nabývat různých tvarů. Který z tvarů charakteristická funkce fuzzy množiny nabývá lze buď odhadnout ze vstupních dat, nebo lze subjektivně rozhodnout kvalifikovaným operátorem, který daný systém úspěšně řídí.



Obr. 22: Průsečíky char. funkce fuzzy množiny „horko“



Obr. 23: Lineární tvar char. funkce fuzzy množiny „horko“



Obr. 24: Diskrétní tvar char. funkce fuzzy množiny „horko“

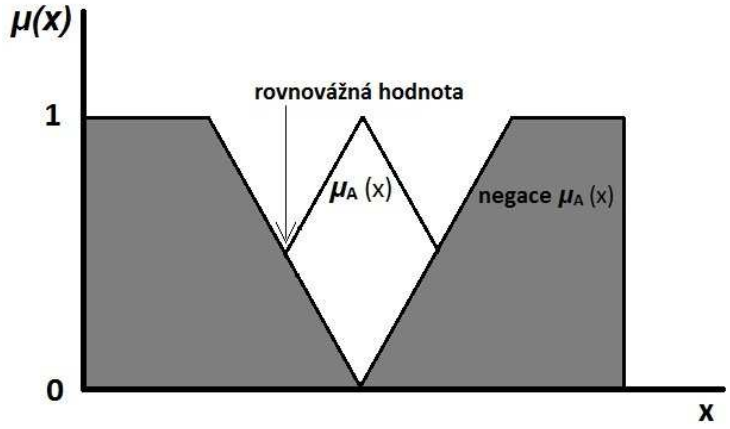
5.5 Operace

V této kapitole budou stručně popsány některé základní operace, které lze provádět s fuzzy množinami. Jak již bylo hovořeno v části týkající se Lukasiewiczovy logiky - tím, že dojde ke změně podoby charakteristické funkce se základní známé operace obohatí, nabývají nových významů a lze zavést i celé třídy nových operací. Jejich výčet by vyčerpal rozsah části této práce věnující se fuzzy logice, proto jich je zde uveveno pouze naprosté minimum. V různé literatuře lze najít různé typy značení fuzzy operací či tříd fuzzy operací, zde si vystačíme s tím, že abychom odlišily fuzzy operace od těch klasických, budou fuzzy operace popisovány stejně jako klasické, jen s tím rozdílem, že pod ně bude vyznačena tečka. Např. negace ($\dot{\neg}$), konjunkce ($\dot{\wedge}$), disjunkce ($\dot{\vee}$), implikace ($\dot{\rightarrow}$), atd.

Fuzzy negace

Fuzzy negace je stejně jako klasická negace unární operací. Charakteristická funkce fuzzy množiny nám říká míru, jak moc který prvek do dané fuzzy množiny patří. Tuto skutečnost lze také popsat slovy, že nám char. funkce říká míru určité vlastnosti, jež charakterizuje danou fuzzy množinu. Fuzzy negace $\neg_{\bullet}$ této hodnoty nám tedy bude říkat, do jaké míry prvek tuto vlastnost nemá.

Standardní negace $\neg_S$ je definována vztahem $\neg_S \alpha = 1 - \alpha$ [25]. (22)



Obr. 25: Fuzzy negace

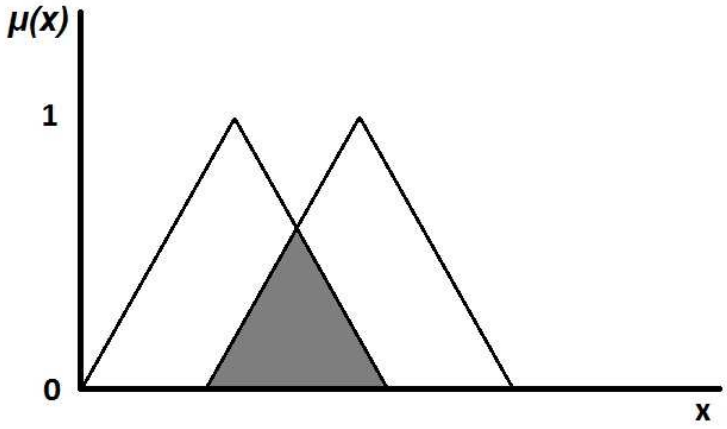
Standardní operace jsou velmi přirozeným fuzzy zobecněním operací. Mohlo by se zdát, že není třeba uvažovat žádné jiné fuzzy negace. Můžeme uvažovat již známý příklad plné a prázdné sklenice, kdy pojem prázdný je opakem plného. V případě, že je do poloviny sklenice nalita kapalina, má daná sklenice obor hodnot charakteristické funkce hodnotu ½ ($\mu_A = 0{,}5$). Negace této hodnoty bude dle rovnice (22) také ½ ($\neg_{\bullet}\mu_A = 0{,}5$). V tomto případě můžeme říci, že sklenice je buď poloprázdná, nebo poloplná. Fuzzy logika připouští zavedení nových operací. Některý hodnotitel je v takových soudech umírněný; tomu odpovídá fuzzy negace $\neg_1$, pro niž platí, že $\neg_1 \mu_A(x) \prec \frac{1}{2}$ [25]. Naopak někdo je velmi nespokojený, což vystihne fuzzy negace $\neg_2$, pro niž platí, že $\neg_1 \mu_A(x) \succ \frac{1}{2}$. Různé typy negací potom můžou nabývat různých významů, např. optimistický nebo pesimistický pohled, nebo méně či více přísný, apod. Pro každou fuzzy negaci $\neg_{\bullet}$ existuje právě jedna hodnota $\mu_A \in (0,1)$, pro kterou platí,

že $\underset{\bullet}{\neg}\mu_A = \mu_A$. Nazývá se rovnovážnou hodnotou (equilibrium), viz obr. 25. Např. v uváděném příkladě se sklenicí je rovnovážná hodnota rovna ½.

Fuzzy konjunkce

Fuzzy konjunkce je stejně jako klasická konjunkce operace binární, je tedy zapotřebí existence dvou fuzzy množin. Fuzzy konjunkce $\underset{\bullet}{\wedge}$ nám říká míru pravdivosti dvou vlastností (každé vlastnosti odpovídá jedna fuzzy množina).

Standardní konjunkce je definována vztahem $\alpha \underset{S}{\wedge} \beta = \min(\alpha, \beta)$ [25]. (23)



Obr. 26: Fuzzy konjunkce

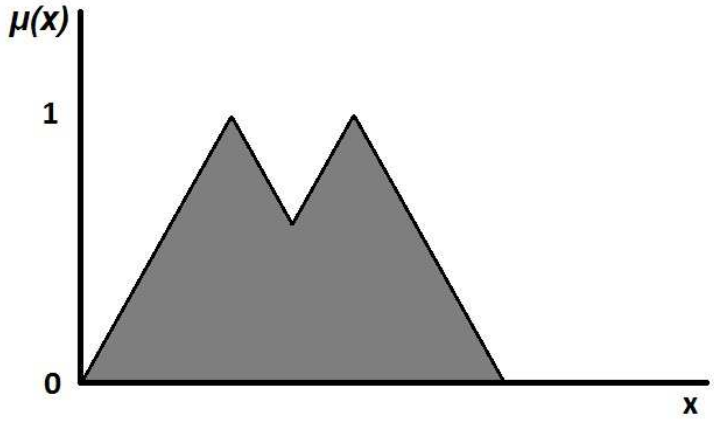
Standardní typ konjunkce, jinak též nazývaný jako Gödelova nebo Zadehova, vzniká nalezením minima dvou výroků (oborů hodnot char. funkcí $\mu_{A,B}$ dvou fuzzy množin). Můžeme opět uvažovat známý příklad se sklenicí, kdy jedna fuzzy množina bude popisovat, jak je daná sklenice naplněna, zatímco druhá fuzzy množina bude popisovat, do jaké míry je daná kapalina vhodná k pití. Opět krajní hodnoty 0, 1 budou v případě první fuzzy množiny znamenat prázdná, plná; v případě druhé fuzzy množiny nevhodná, vhodná k pití. Řekněme, že máme sklenici do poloviny plnou ($\mu_A = 0{,}5$) a vhodnost k pití je charakterizována také tak na půl ($\mu_B = 0{,}5$) -> např. teplé pivo. Standardní konjunkce těchto dvou množin v tomto případě má hodnotu 0,5. Jenže si lze všimnout i případu, kdy je sklenice také do poloviny plná ($\mu_A = 0{,}5$), ale vhodnost k pití je charakterizována hodnotou 1 ($\mu_B = 1$) -> např. studené pivo. V tomto případě má standardní konjunkce hodnotu také 0,5; přitom ale druhý případ má jasně lepší předpoklady. Dochází zde k podobnému problému jako u fuzzy negace, řešením je zavedení dalších typů fuzzy konjunkcí jako je např. součinová nebo Lukasiwieczova (určité typy jsou více či méně optimistické). Najít vhodné fuzzy konjunkce k různým typům úloh nebývá někdy

jednoduché. Často slouží fuzzy logika k vyjádření a algoritmizaci vědomostí lidského experta; pak série podobných otázek může naznačit, jaká fuzzy konjunkce se nejlépe hodí na konkrétním místě [25].

Fuzzy disjunkce

Fuzzy disjunkce je stejně jako klasická konjunkce operace binární, je tedy zapotřebí existence dvou fuzzy množin. Fuzzy disjunkce $\underset{\bullet}{\vee}$ nám popisuje výsledný vztah těchto dvou fuzzy množin k nově získané fuzzy množině (výsledkem je funkce všech jevů vykazující příslušnost k jevu novému).

Standardní disjunkce je definována vztahem $\alpha \underset{S}{\vee} \beta = \max(\alpha, \beta)$ [25]. (24)



Obr. 27: Fuzzy disjunkce

Budeme uvažovat příklad závislosti sportovního výkonu na různých předpokladech. Nejdříve uvedeme, kterou vlastnost popisují jednotlivé fuzzy množiny. První fuzzy množina bude představovat fyzické předpoklady lidí, druhá fuzzy množina trénovanost. Výsledná disjunkce bude nová fuzzy množina, která bude představovat určitý sportovní výkon, např. běh na 200 metrů. Krajní hodnoty 0, 1 budou v případě první fuzzy množiny znamenat 0 – postižený člověk, 1 – anatomický ideál běžce. V případě druhé množiny bude 0 – nic nedělání, 1 – pravidelná profesionální příprava. Řekněme, že první běžec má průměrné fyzické předpoklady ($\mu_A = 0{,}5$), ale jeho příprava je naprosto profesionální ($\mu_B = 1$). V tomto případě má standardní disjunkce hodnotu 1. Dále druhý běžec má atletické předpoklady ($\mu_A = 0{,}8$), příprava je ale naprosto nulová, není žádná ($\mu_B = 0$). V prvním případě má standardní disjunkce hodnotu 1, v druhém případě hodnotu 0,8. Často vidíme v praxi, že člověk přírodou méně obdařený může pílí překonat i člověka nadaného, ale laxního. Obdobně jako u předchozích operací i u fuzzy disjunkce může dojít k určitým problémům, např. kdyby druhý

běžec byl typem ideálního běžce s profesionální přípravou, fuzzy disjunkce by dosáhla hodnoty 1, stejné jako u člověka průměrného, což by úplně neodpovídalo realitě. Tyto nesourodosti jsou opět řešeny zavedením více druhů fuzzy disjunkcí.

Fuzzy implikace

Fuzzy implikace $\underset{\bullet}{\rightarrow}$ je operace, jejíž popis není jednoznačně ustálený, přesto hraje důležitou roli hlavně ve fuzzy logice a jejich aplikací, proto je také dále popisována i v následujících kapitolách. Fuzzy implikace je stejně jako v klasické logice operací duální, lze ji vykonstuovat z již známých operací. Možností konstrukce je vícero, např. pomocí Boolovy algebry (včetně De Morganových zákonů). V klasické logice všechny konstrukce vedou ke stejnému typu implikace, zatímco ve fuzzy logice každá z konstrukcí vede k odlišným typům fuzzy implikací [25]. Zde je uveden standardní typ implikace.

Standardní implikace je definována vztahem $\alpha \underset{S}{\rightarrow} \beta = \underset{S}{\neg} \alpha \underset{\bullet}{\vee} \beta$ [25]. (25)

Tím, že existuje více typů fuzzy konjukcí, disjunkcí a negací, lze společně s různými druhy konstrukcí implikace vytvořit různé typy fuzzy implikací. Např. od Lukasiewieczovy konjukce lze odvodit Lukasiewiczovu implikaci, ale i jiné typy fuzzy implikací, apod.

5.6 Fuzzy logika

Studium většiny typů logik se věnuje objasnění dvou základních pojmů a vztahů mezi nimi. Je to syntax a sémantika. Syntax se zabývá popisem základních prvků každé logiky, kdežto sémantika se věnuje vztahům mezi těmito základními, ale i jinými prvky. V neposlední řadě se sémantika zabývá také významem – ať už základních prvků, vztahů či odvozených entit, snaží se slovně formulovat dosažené výsledky logické práce. Dále se nechá říci, že se syntax zabývá otázkou dokazatelnosti, kdežto sémantika zkoumá pravdivost.

Nyní zde budou stručně vysvětleny základní pojmy týkající se obecně mnoha typů logik včetně fuzzy logiky. **Výrok** je tvrzení, o němž má smysl zkoumat míru jeho pravdivosti. **Axiom** je určité tvrzení, o němž se automaticky předpokládá, že je pravdivé a nemusí se tudíž dokazovat. **Výroková formule** je složena z různých výroků, lze zkoumat míru její pravdivosti, ale i význam. Další důležitý pojem v logice je **tautologie**. V klasické logice se jedná o výrokovou formuli, jejíž míra pravdivosti je naprostá pravda, nic jiného. Je naprosto pravdivá za všech okolností. Fuzzy logika takto jednoznačný význam pro tautologii nemá. Ve fuzzy logice

se lze dívat na tautologii v pozměněném smyslu, např. že tautologie je každá formule, jejíž míra pravdivosti je různá od nuly. Tedy je alespoň částečně pravdivá za všech okolností.

Další pojem v logice je velmi rozšířená metoda odvozování, tzv. ***modus ponens***, pravidla odvozování platné jak v klasické, tak fuzzy logice. Má toto znění: „Z platnosti prvé formule vyplývá platnost formule druhé“.

Jedná se o implikaci: $\varphi \rightarrow \psi$.

Zjednodušeně vyjádřeno: „jestliže platí φ a z φ vyplývá ψ , pak platí i ψ “.

Pomocí duality implikace (možnost odvození např. z disjunkce) lze vyvodit fakt, že existuje více fuzzy logik. Jestliže rozeznáváme různé druhy implikací, disjunkcí, apod., tak totéž se promítne do fuzzy logiky skrze základní odvozovací pravidlo *modus ponens*, které tyto operace používá. Jedná se o speciální fuzzy logiky, jež obsahující axiomy základní fuzzy logiky, ale dle druhu operací dosazují axiomy nové, které ovlivňují chování dané logiky. Například použitím standardní konjukce vzniká speciální fuzzy logika nazývaná Gödelova, použitím Lukasiewiczovy konjunkce vzniká speciální Lukawiewiczova logika, která opět vykazuje odlišné chování, apod. [25].

Přirozený přístup k fuzzy logice je použití tzv. **ohodnocené syntaxe**, kdy se pracuje s tzv. **ohodnocenou formulí**, což je dvojice $\langle \varphi, a \rangle$, kde φ je formule, a je pravdivostní hodnota [28]. Pak teorie je množina ohodnocených formulí a to, že $\langle \varphi, a \rangle$ patří do teorie, říká, že formuli φ považujeme za pravdivou alespoň ve stupni a. Podobně i axiomy jsou ohodnocené formule.

Míru pravdivosti lze převést na charakteristickou funkci fuzzy množin a dále pracovat jako s fuzzy množinami. Problémy může dělat přesná slovní interpretace výsledků fuzzy logiky, což ale např. u technických aplikací nemusí být tak velký problém. Dále je podstatný i rozdíl s prací ve fuzzy logice oproti klasické logice. Ve fuzzy logice nelze používat pravdivostní tabulky, neboť pravdivostní hodnoty fuzzy logiky odpovídají reálným číslům z intervalu $\langle 0, 1 \rangle$, je jich tedy nekonečně mnoho, tudíž nelze pracovat s nekonečně mnoho řádky.

Příklad: Jedná se o starý antický příklad paradox hromady písku, kde si klasická logika láme zuby a dochází k mylným výsledkům. Jsou zde dva předpoklady:

Jedno zrnko písku netvoří hromadu písku.

Z něčeho, co netvoří hromadu písku, nevznikne hromada tím, že k tomu přidáme jedno zrnko písku.

Obě tvrzení jsou pravdivá, ale vedou k závěru, že postupným přidáváním jakéhokoliv počtu zrnek k jednomu zrnku písku nikdy nezískáme hromadu. Tedy v klasické logice vyplývá závěr:

n zrnek písku netvoří hromadu písku.

Z pravdivých předpokladů vznikl zřejmě nepravdivý závěr.

Ve fuzzy logice to lze elegantně řešit tak, že druhý předpoklad není naprosto pravdivý, ale můžeme zde užít míru pravdivosti (odpovídá charakteristické funkci fuzzy množiny). Přidáním jednoho zrnka získáme něco, co přece jen připomíná více hromadu. Pravdivostní hodnota 1 u druhého předpokladu bude tedy postupným přidáváním zrnek klesat, limitně může dosáhnout i nuly a z reálu věci chybný závěr, že miliarda zrnek netvoří hromadu vychází jako nepravdivý.

5.7 Aplikace

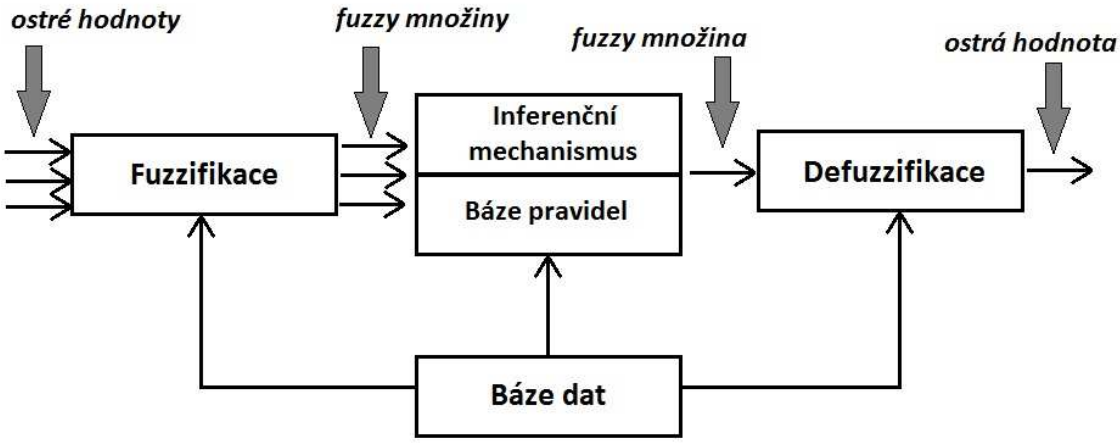
Fuzzy regulátory

Největší rozvoj fuzzy logiky bylo dosaženo především úspěchy v oblasti řízení, přičemž zejména v Japonsku se samotný pojem fuzzy vžil do mnoha oblastí lidské činnosti. Důvodů lze najít několik, ať již samotná japonská kultura lačná po novinkách trhu, relativní jednoduchost a příznivá cena, či snadnost implementace. Dnes se fuzzy řízení používá nejen v různé spotřební elektronice (myčky, pračky, kamery, atd.), ale prosazuje se i ve větších projektech, např. řízení metra, cementárenské pece, autopilot helikoptér nebo řízení Hubbleova teleskopu. Základním prvkem v řízení je regulátor. V následujících odstavcích bude popsán pouze ten základní – Mamdaniho regulátor. Ostatní typy regulátorů (např. Sugenovův) jsou v principu práce obdobné.

Mamdani v roce 1973 řešil problém řízení nelineární soustavy (současné řízení tlaku páry a otáček parního stroje), kdy jej napadla myšlenka aproximovat charakteristiky pomocí fuzzy logiky, konkrétně pomocé fuzzy implikací [26]. Tvůrce fuzzy množin íránský vědec Lotfi Zadeh popsal fuzzy implikaci ve tvaru IF-THEN, jež se ukázala k danému problému velmi vhodná.

Mamdani a spol. zjistili, že stačí jen velmi málo pravidel k úspěšnému řízení, dokonce takovýto regulátor vykazuje lepší dynamické vlastnosti než klasický PI nebo PD regulátor [26].

Na obrázku 28 , převzato z [29], je vykreslena základní struktura fuzzy regulátoru.



Obr. 28: Struktura fuzzy regulátoru

Proces fuzzifikace a defuzzifikace byl již v této práci (část 5.4.) stručně vysvětlen, takže bude nyní pozornost zaměřena na bázi dat, bázi pravidel a inferenční mechanismus. Báze dat komunikuje s ostatními částmi regulátoru a ovlivňuje jejich chování. Obsahuje charakteristické funkce všech fuzzy množin (vstupů a výstupů) včetně intervalů a měřítek převodu ostrých hodnot na fuzzy množiny a obráceně. Báze pravidel obsahuje všechna pravidla pro dostatečnou lingvistickou (slovní) aproximaci mezi vstupy a výstupem regulátoru [26]. Pravidla mají tvar IF <stav procesu> THEN <zásah do řízení>. Pravidla jsou mezi sebou spojena disjunkcí. Lze použít i jiné operátory, to pak vede ke vzniku nových typů regulátorů. Pravidla lze buď získat komplexních vyhodnocením řízení lidskou obsluhou - expertem (méně častý případ), nebo je lze získat z báze dat, jež charakterizují řízený proces. S úspěchem lze použít např. učící se neuronovou síť, která zpracuje bázi dat a nastaví bázi pravidel. Inferenční mechanismus vyhodnocuje fuzzy logikou (pomocí fuzzy operací) vstupní fuzzy množiny a výsledkem je jediná výstupní množina.

Podstatný rozdíl mezi klasickými metodami aproximace a metodou fuzzy aproximace je ten, že aproximovaná (obecně nelineární) funkce je slovně popsána pomocí fuzzy pravidel [29]. Tudíž je možno expertní znalosti (získané člověkem) reprezentovat fuzzy systémem, např. jako byla jedna z prvních aplikací fuzzy logiky - řízení cementárenské pece, jež klasickými metody nešla řídit. Mnoho aplikací řízení spojitě vstupních veličin by šlo řešit i klasickými

metody, ovšem fuzzy řízení nabízí lepší dynamické vlastnosti, cenově úsporný vývoj i realizaci a výhodu lokální citlivosti – možnosti změny pouze části aproximace.

Databáze využivající fuzzy logiku

V oblasti informací a informatiky obecně lze fuzzy logiku používat nejen pro řízení jevů (cyklů, větví,…), ale úspěšného použití lze dosáhnout i v případě databází, oblasti pro dnešní život poměrně významné. Jelikož klasický relační model a jeho rozšíření o fuzzy přístup nenabídl uspokojivé řešení, většina výzkumu v této oblasti se soustředila na fuzzy objektově orientované datové modely za účelem komplexnost dat a jejich neurčitost řešit zároveň [30]. Klasickou třídu objektů lze teoreticky vymezit dvěma odlišnými způsoby [30]:

pomocí extenze, tj. třída definovaná seznamem svých instancí,
pomocí intenze, tj. třída definovaná množinou atributů a jejich přípustných hodnot.

Klasický přístup říká, že instance buď do třídy patří nebo nepatří, fuzzy přístup přináší možnost objektu patřit do seznamu jenom trochu. Totéž lze říci u atributů i jejich přípustných hodnot. Atribut do třídy může také patřit jenom z části, např. můžeme mít třídu objektů „hezká auta“, kam by z části patřil atribut „světlá barva“, jehož doména (přípustné hodnoty) by byla všechna reálná čísla z intervalu $\langle 0, 1 \rangle$. Takže např. modrá barva určitého auta by byla ohodnocena reálným číslem (např. 0,5), to by se promítlo do obsahu třídy „hezká auta“, kam by toto konkrétní auto z části patřilo. Klasický přístup by tam dané auto buď zcela jistě zařadil, nebo zcela jistě nezařadil. Při práci s databázemi se používají také určité množinové operace, které mohou být obohaceny fuzzy přístupem. Např. i ve vyhledávání lze dosáhnout zajímavých výsledků, kdy výsledkem fuzzy přístupu může být stupeň pravdivosti výroku v souvislosti dotazu s vyhledávanými entitami.

Závěr

Cílem diplomové práce bylo vytvoření přehledu moderních trendů v oboru počítačové fyziky. Vznikl text, který popisuje sice pouze základní aspekty daných metod, ale podstatným rysům je věnována značná šíře.

Byly zjednodušeně vystvětleny základní mechanismy těchto metod. Jedná se tedy o určitý úvod do problematiky. Danému tématu se věnuje velmi mnoho autorů, v knihkupectvích je možné zakoupit nepřeberné množství publikací (například [2], [5], [6], [10], [11], [14], [18], [20], [22], [25], [26]), které se věnují jak samotnému úvodu metod, tak dané metody zkoumají do hloubky. Tyto práce může čtenář využít k dalšímu studiu.

Přečtením této práce by měl čtenář získat základní přehled o daných metodách. Měl by vnímat souvislosti mezi teorií, matematickou aproximací a technickým řešením. Ve výsledku by měl principiálně přemýšlet o možnosti vhodného použití popsaných metod v různých aplikacích včetně aplikací počítačové fyziky.

V budoucnu by bylo vhodné sestavit soubor typových úloh zaměřených na praktické využití získaných poznatků. Mělo by se jednat o řadu typických aplikací, které by se realizovaly běžně dostupnými programovacími prostředky, např. pomocí programovacího jazyka MATLAB.

Vývoj se momentálně ubírá možná konzervativním, leč z praktického hlediska velmi účinným směrem. Jako velmi efektivní se jeví možnosti kombinování popsaných metod vzájemně mezi sebou i s jinými moderními metodami. Dále dochází k prolínání kódu moderních metod s klasickými metodami výpočetní techniky, kdy vznikají tzv. hybridní metody, hybridní kódy. Stále existují určité úlohy, které aplikačně nelze dnes řešit jinak, než pomocí moderních výpočetních metod. Lze tedy očekávat, že význam počítačové fyziky při studiu přírodních jevů ještě poroste.

Seznam použitých pramenů a literatury

[1] HRACH, Rudolf. *Počítačová fyzika 1*. Ústí nad Labem: UJEP, 2003.

[2] RACEK, Stanislav; JEŽEK, Karel. *Paralelní programování*, Plzeň: ZČU Plzeň, 1994 ISBN 80-7082-128-0.

[3] GUSTAFSON, John L. *Reevaluating Amdahl's law.* New York: ACM, 1988. ISSN 0001-0782.

[4] *Charakteristika moderních pentií* [online]. [cit. 2012-08-05]. Dostupné z: http://www.fit.vutbr.cz/study/courses/PTP/public/texty/pentium03.pdf

[5] JEŽEK, Karel; MATĚJOVIC, Přemysl; RACEK, Stanislav. *Paralelní architektury a programy.* Plzeň: Vydavatelství ZČU, 1997. 294 s. ISBN 80-7082-322-4.

[6] DVOŘÁK, Václav. *Architektura a programování paralelních systémů*. Brno: VUTIUM, 2004. ISBN 80-214-2608-X.

[7] HAPLA, Václav: *Vytvoření rozhraní knihovny MatSol pro paralelní řešení kontaktních úloh*. Ostrava, 2010. Diplomová práce. Technická univerzita Ostava.

[8] KMUNÍČEK, Jan. *Gridy jako klíčový fenomén informačních technologií nového tisíciletí* [online]. [cit. 2012-08-09]. Dostupné z: http://www.ics.muni.cz/bulletin/articles/343.html

[9] *Projekt metacentrum* [online]. [cit. 2012-08-09]. Dostupné z: http://www.metacentrum.cz/cs/about/meta/index.html

[10] NOVÁK, Mirko; kolektiv. *Umělé neuronové sítě.* Teorie a aplikace. Český Těšín: C. H. Beck, 1998. 382 s. ISBN 80-7179-132-6.

[11] NOVÁK, Mirko; FABER Josef; KUFUDAKI, Olga. *Neuronové sítě a informační systémy živých organismů.* Praha: GRADA, 1993. 272 s. ISBN 80-85424-95-9.

[12] VYTOPILOVÁ, Simona. *Nové poznatky z molekulární patologie neuro-endokrinních tumorů a jejich využití v diagnostické a prediktivní onkologii* [online]. Bakalářská práce. [cit. 2012-08-17]. Dostupné z: http://is.muni.cz/th/211525/prif_b/Bakalarska_prace.pdf

[13] *Nervová tkáň* [online]. [cit. 2012-08-09]. Dostupné z: http://uhem.ray.cz/uploads/1063_5Nervovatkan.pdf

[14] TUČKOVÁ, Jana. *Vybrané aplikace umělých neuronových sítí při zpracování signálů.* Praha: Nakladatelství ČVUT, 2009. ISBN 978-80-01-04229-8.

[15] ÚSTAV NORMÁLNÍ ANATOMIE. *Struktury CNS* [online]. [cit. 2012-08-10]. Dostupné z: http://www.nan.upol.cz/neuro/cd411.html

[16] HAKL, František; HOLEŇA, Martin. *Úvod do teorie neuronových sítí.* Praha: Vydavatelství ČVUT, 1998. ISBN 80-01-01716-8.

[17] *Perceptron Learning Rule* [online]. [cit. 2012-08-12]. Dostupné z: http://hagan.ecen.ceat.okstate.edu/4_Perceptron.pdf

[18] ŠNOREK, Miroslav. *Neuronové sítě a neuropočítače.* Praha: Vydavatelství ČVUT, 2002. ISBN 80-01-02549-7.

[19] FANTA, Jiří. *Neuronové sítě ve společenských vědách.* Praha: Nakladatelství Karolinum, 2000. ISBN 80-246-0175-3.

[20] ZELINKA, Ivan; OPLATKOVÁ, Zuzana; ŠEDA, Miloš; OŠMERA, Pavel; VČELAŘ, František. *Evoluční výpočetní techniky.* Principy a aplikace. Praha: BEN, 2009. s. 534. ISBN 978-80-7300-218-3.

[21] BARBIERI, Marcello. *Organické kódy.* Úvod do sémantické biologie. Praha: Academia, 2006. ISBN 80-200-1403-9.

[22] HYNEK, Josef. *Genetické algoritmy a genetické programování.* Praha: GRADA, 2008. ISBN 978-80-247-2695-3.

[23] HOLLAND, John H. Adaptation in natural and artificial systems. Cambridge: MIT Press, 1992. ISBN 0-262-58111-6.

[24] DOSTÁL, Martin. *Evoluční výpočetní techniky.* Olomouc: Univerzita Palackého, 2007. učební text.

[25] NAVARA, Mirko; OLŠÁK, Petr. *Základy fuzzy množin*. Praha: Nakladatelství ČVUT, 2007. ISBN 978-80-01-03668-6.

[26] VYSOKÝ, Petr. *Fuzzy řízení.* Praha: Vydavatelství ČVUT, 1996. ISBN 80-01-01429-8.

[27] NOVÁK, Vilém. *Fuzzy množiny a jejich aplikace.* Praha: SNTL, 1990. ISBN 80-03-00325-3.

[28] BĚLOHLÁVEK, Radim. *Matematická logika* [online]. [cit. 2012-07-15]. Dostupné z: http://belohlavek.inf.upol.cz/vyuka/ML.pdf

[29] JURA, Pavel. *Fuzzy logika v modelování a řízení dynamických systémů.* Brno: VUTIUM, 2005. ISBN 80-214-3019-2.

[30] VOLRÁB, Ondřej. *Metody a nástroje pro tvorbu informačních systémů na internetu.* Praha. Disertační práce. Česká zemědělská univerzita v Praze.

Seznam obrázků