VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

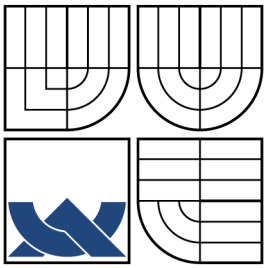
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SIMULACE ŘÍZENÍ SÍŤOVÉHO PRVKU S NEURONOVOU SÍTÍ

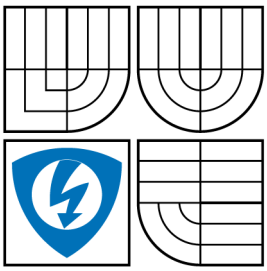
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. PETR ŠILHAN
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SIMULACE ŘÍZENÍ SÍŤOVÉHO PRVKU S NEURONOVOU SÍTÍ

SIMULATING THE CONTROL OF A NETWORK ELEMENT WITH NEURAL NETWORK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. PETR ŠILHAN
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. VLADISLAV ŠKORPIL, CSC.
SUPERVISOR

BRNO 2008

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

ZDE VLOŽIT PRVNÍ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ZDE VLOŽIT DRUHÝ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ABSTRAKT

Práce se zabývá problematikou užití neuronových sítí pro řízení telekomunikačních síťových prvků. Cílem této práce je vytvoření simulačního modelu síťového prvku se spojovacím polem s centrální pamětí, u kterého je optimalizace řízení spojovacího pole řešena pomocí Hopfieldovy neuronové sítě. Veškerý programový kód je vytvořen v integrovaném prostředí MATLAB s využitím Neural Network Toolboxu.

KLÍČOVÁ SLOVA

Neuronová síť, Hopfield, síťový prvek, spojovací pole, řízení, optimalizace, prioritní přepínání.

ABSTRACT

The thesis deals with the use of neuronal networks for the control of telecommunication network elements. The aim of the thesis is to create a simulation model of network element with switching array with central memory, in which the optimization control switching array is solved by means of the Hopfield neural network. All source code is created in integrated environment MATLAB with the use of Neural Network Toolbox.

KEYWORDS

Neural Network, Hopfield, Network Element, Switching Array, Control, Optimalization, Priority Switching.

ŠILHAN P. *Simulace řízení síťového prvku s neuronovou sítí*. Brno: Vysoké Učení Technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2008. Počet stran 76, Počet stran příloh 1. Diplomová práce. Vedoucí práce byl doc. Ing. Vladislav Škorpil, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Simulace řízení síťového prvku s neuronovou sítí“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Tímto bych rád poděkoval panu doc. Ing. Vladislavovi Škorpilovi, CSc. za metodickou pomoc, cenné rady a čas, který věnoval mé diplomové práci.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Dnešní moderní širokopásmové integrované sítě slučují klasické telekomunikační sítě se sítěmi datovými a dovolují tak provozovat různé typy uživatelských služeb ve společné síti. Umožňují společný přenos hovorových, datových i obrazových signálů například formou videokonferenčních aplikací a různých multimediálních služeb. Tento druh provozu ovšem klade stále větší nároky na aktivní síťové prvky, které musí být schopné nejen poskytnout dostatečnou šířku pásma, ale zajistit i požadované parametry spojení. Z tohoto důvodu se v oblasti vývoje telekomunikačních prvků stále hledají nové možnosti jak zvýšit jejich propustnost a efektivitu. Hlavním úkolem aktivních síťových prvků je propojit jednotlivá zařízení v síti a umožnit jejich vzájemnou komunikaci, tj. předat datové jednotky směrem od odesilatele k příjemci na základě výsledků zpracování v řídicí části síťového prvku. Celý proces by byl vcelku jednoduchý, kdyby všechny přicházející datové jednotky měly stejnou hodnotu **priority**. Jednotlivé služby mají ovšem odlišné požadavky např. na zpoždění datových jednotek v síti. Z tohoto důvodu je nutné **prioritní zpracování** datových jednotek v řídicí části síťového prvku, jehož smyslem je určit, ve kterém časovém okamžiku a jaká datová jednotka má být přenesena přes spojovací pole síťového prvku na jeho výstup. Tento způsob zpracování klade velké nároky na výpočetní výkon řídicí části aktivního síťového prvku.

Prioritní zpracování (optimalizace) datových jednotek může být realizována například jedním z následujících způsobů:

- **Klasickým sekvenčím zpracování dat** - které je ovšem omezeno rychlostí centrální procesorové jednotky. V dnešní době se sice stále zvyšuje frekvence procesorů, tento postup ale již dnes naráží na technologické možnosti výroby.
- **Paralelním řazením více komplexních funkčních prvků** - má za následek pouze malé zvýšení efektivity celého systému, protože klade velké nároky na řízení jejich vzájemné spolupráce.
- **Paralelní propojení velkého počtu elementárních funkčních bloků** (s centrální nebo rozprostřenou pamětí) - vzájemná kooperace elementárních funkčních bloků je mnohem jednodušší, což má za následek, že výkonnost prvku jako celku je vyšší nežli v případě použití složitějších prvků.

Analogii paralelního propojení velkého počtu elementárních funkčních bloků lze nalézt v oblasti **umělých neuronových sítí**, které vycházejí z jejich biologického vzoru. Umělá neuronová síť je paralelní distribuovaný systém elementárních výkonných prvků (většinou velkého počtu), které jsou účelně uspořádány tak, aby byla

schopena požadovaného zpracování informací. V dnešní době existuje mnoho struktur těchto sítí, které v mnoha případech nabízí velmi perspektivní řešení při zpracování dat. Diplomová práce je zaměřena na jeden speciální typ struktury umělé neuronové sítě, na tzv. Hopfieldovy sítě, které je možné použít k řešení optimalizačních problémů.

Pokud vezmeme v úvahu skutečnost, že aktivní síťové prvky musí řešit při svém provozu optimalizační problém datových jednotek s možností využít paralelní architekturu prvků, tak lze konstatovat, že pro jejich řízení by se daly velmi efektivně využít právě umělé neuronové sítě.

Cílem diplomové práce bude vytvořit simulační model síťového prvku se spojovacím polem s centrální pamětí, u kterého bude optimalizace řízení spojovacího pole řešena pomocí Hopfieldovy neuronové sítě. Veškerý programový kód bude vytvořen v integrovaném prostředí MATLAB verze 7.5 s využitím Neural Network Toolbox verze 5.1.

1 AKTIVNÍ SÍŤOVÉ PRVKY

1.1 Opakovače a rozbočovače

Opakovače (repeaters) a **rozbočovače** (HUBs) jsou aktivní síťové prvky pracující na **fyzické vrstvě** modelu vrstvové síťové architektury OSI (dále jen fyzická vrstva). Provádějí obnovu (regeneraci) příchozích signálů, tedy obnovení jejich tvaru, časové polohy pulzů a doplnění bitové informace přidané na fyzické vrstvě tak, aby mohl být signál správně přijat cílovou stanicí. Opakovač je zařízení pouze se dvěma porty a rozbočovač je zařízení s více porty. Rozbočovač je prakticky multiportový opakovač. Oba tyto prvky vkládají do přenosové cesty určité zpoždění. Pokud je tedy daná síť citlivá na zpoždění, tak je počet opakovačů a rozbočovačů v síti omezen. Všechny části sítě, které jsou těmito prvky přímo propojeny, tvoří jeden fyzický kanál (jednu **kolizní doménu** v případě sítí Ethernet) a všechny porty pracují se stejnou rychlostí. Opakovače a rozbočovače se používají u **LAN** (Local Area Network) sítí v případě Ethernetu, protože u kruhové topologie je regenerace prováděna v přijímací části u každé stanice (nabízí se zde použití pouze v případě, že by od sebe byly jednotlivé stanice vzájemně dál, než povoluje limit pro daný typ sítě). Pokud to síťová technologie dovoluje, je možné tyto prvky řadit do **kaskády**[1], případně je **stohovat**[2]. Přímé propojení dvou rozbočovačů přímým kabelem se musí provádět přes přepínač. Opakovače a rozbočovače jsou pro ostatní zařízení v síti **transparentní**, to je způsobeno tím, že nemají fyzickou ani síťovou adresu (neobsahují-li dohledový modul) a ostatní zařízení v síti o jejich přítomnosti tedy nic neví. Ovšem vyšší třídy rozbočovačů jsou vybaveny **SNMP modulem** (Simple Network Management Protocol Module), případně možností rozpoznání a odpojení vadného portu, či dokonce možností zálohovat port jiným portem, který se v případě selhání zálohovaného portu uvede v činnost. [10]

1.1.1 Princip funkce rozbočovačů

Princip funkce rozbočovačů: [8]

- Nepřetržitě sleduje signál na přijímači každého portu,
- pokud se objeví signál na přijímači i-tého portu, tak jej ihned vysílá do všech ostatních portů,

[1] Pro spojení do kaskády se používá buď překřížený kabel (cross over cable), nebo je vyhrazen speciální port s možností přepnutí přijímacího a vysílacího páru.

[2] Při stohování se rozbočovače propojí krátkým rychlým spojem, takže se více rozbočovačů chová jako rozbočovač jedniný.

- pokud se objeví signál na přijímači dvou a více portů (tzv. kolize), tak okamžitě do všech portů vyšle speciální sekvenci bitů tzv. **JAM signál**, na základě které stanice tuto kolizi detekují pomocí **CSMA/CD** (Carrier Sense Multiple Access with Collision Detection).

V dnešní době se již rozbočovače moc nepoužívají a nahrazují je inteligentnější zařízení - **přepínače**.

1.2 Mosty a přepínače

Mosty (bridges) a **přepínače** (switches) jsou aktivní spojovací síťové prvky, které svoji činnost rozšiřují oproti opakovačům a rozbočovačům o **linkovou vrstvu** modelu vrstvové síťové architektury OSI (dále jen linková vrstva). Používají se k propojení/oddělení částí sítě mající vlastní přenosový kanál. Jedná se například o sítě: [10]

- **Ethernet** - propojují/oddělují tzv. kolizní domény.
- **Token Ring** - propojují/oddělují nezávislé okruhy.
- **ATM** - přepínají buňky podle tzv. **virtuálních kanálů** (VCI - Virtual Channel Identifier) a **virtuálních cest** (VPI - Virtual Path Identifier), dále implementují **kvalitu služeb** (QoS - Quality of Service).
- **Frame Relay** - přepínají rámce podle **identifikátorů virtuálních okruhů** (DLCI - Data Link Connection Identifier).

V dalším textu se ovšem budeme věnovat především mostům a přepínačům v sítích LAN (Ethernet, Token Ring).

1.2.1 Vlastnosti přepínačů

Přepínače se liší od rozbočovačů hlavně tím, že provádějí **komutaci rámců** na základě informací uložených v **přepínací tabulce** (tabulce MAC adres). V ní je uložena vazba mezi hardwarovou (fyzickou, MAC) adresou a odpovídajícím fyzickým portem, kam je stanice připojena buď přímo nebo přes další aktivní síťové prvky. Naplňování přepínací tabulky je automatický proces, tedy bez zásahu lidské obsluhy. Tabulku si přepínač postupně naplňuje informacemi z příchozích rámců formou dvojice (zdrojová MAC adresa rámce - číslo portu odkud byl rámec přijat). K tomuto procesu dojde pouze v případě, že tento záznam ještě není v tabulce zapsán a pokud je ještě volná paměťová kapacita.

Požadavkem pro správnou funkci sítě s přepínači je **stromová struktura sítě uzlů** (v případě sítě Ethernet) nebo **stromová struktura kruhů** (v případě sítě Token Ring). Přepínače lze propojovat do stromové struktury pomocí překříženého kabelu (cross over cable) nebo využitím speciálního portu, který umožňuje přepnutí vstupního a výstupního páru vodičů. Moderní přepínače již disponují funkcí automatického rozpoznání protějšího prvku a přepnutí úlohy páru v konektoru. Pokud by požadavek stromové struktury sítě uzlů (kruhů) nebyl splněn, tak by došlo ke **zhroucení sítě** (frame storming). Pro zvýšení spolehlivosti a bezpečnosti sítě se ovšem redundantní cesty budují, ale během normálního provozu jsou blokovány. O to se stará protokol **STP** (Spanning Tree Protokol) řešící problematiku stromové struktury pomocí algoritmu **STA** (Spanning Tree Algorithm). Hlavním úkolem algoritmu STA je v případě výpadku aktivního spoje obnovit konektivitu stromu aktivací náhradního spoje (řádově v jednotkách až desítkách vteřin). Výhodnou vlastností přepínačů je možnost propojení segmentů sítě pracujících s různou přenosovou rychlostí, případně možnost několika rychlostí na jednom fyzickém portu a jejich automatické rozpoznání. Za tímto účelem je přepínač vybaven **vyrovnávacími pamětmi**, které vyrovnávají krátkodobé špičky v zatížení při provozu. Při úplném zahlcení vstupních vyrovnávacích pamětí by však mohlo dojít k tomu, že následující rámce určené k přepnutí budou ztraceny. Z tohoto důvodu umí většina přepínačů řídit provoz tak, aby k tomuto stavu nedocházelo. Řeší se to vytvářením kolizních stavů na zahlceném portu do té doby, než se přijímací paměti opět neuvolní. [10]

Přepínací režimy přepínačů

K nejčastějším přepínacím režimům přepínačů patří: [8]

- **Ulož a pošli** (store and forward) - rámec z daného portu se nejprve celý uloží do vyrovnávací paměti, zkontroluje se z hlediska chyb a následně, pokud není detekován jako chybný, tak dojde k jeho přepnutí na příslušný výstupní port. Tento režim přepínání klade velké paměťové nároky na vyrovnávací paměti a vkládá velké zpoždění do přenosové cesty. Jeho výhodou je podpora různých přenosových rychlostí jednotlivých portů a zamezení předání neúplných a chybných rámců.

- **Přiřaď a posílej** (cut through) - ze záhlaví rámce se zjistí cílová fyzická adresa příjemce a okamžitě se zahájí vyhledání cílového portu a následně se rámec přepíná na příslušný cílový port. Podstatnou nevýhodou tohoto režimu je šíření chyb (např. kolizí, přepínání neúplných a chybných rámců) a nutnost stejných přenosových rychlostí všech portů. Mezi výhody můžeme počítat rychlost přepínání, malé paměťové nároky na vyrovnávací paměti a malé

zpoždění vkládané do přenosové cesty.

- **Kontrola minimální délky rámce** (fragment free) - jedná se o zdokonalení předešlé techniky. Přepínač v tomto režimu začne přeposílat rámec až po přijetí 64 B (v případě Ethernetu 10 Mbit/s a 100 Mbit/s) nebo 512 B (v případě Ethernetu 1 Gbit/s), kdy je jasné, že na daném segmentu nevznikla kolize. Tato minimální délka rámce je specifikována v přístupové metodě CSMA/CD. Hlavní výhodou je omezení šíření chyb (např. zamezení přepínání rámců, které byly poškozeny kolizemi).

Některé přepínače umožňují průběžné vyhodnocování kvality připojených spojů a četnosti kolizí a na základě získaných údajů přepínají mezi přepínacími režimy tak, aby efektivní propustnost přepínače byla co nejlepší a síť se příliš nezatěžovala neúplnými či chybnými rámci.

1.2.2 Princip funkce přepínačů

Princip funkce přepínačů při příchodu nového rámce můžeme popsat následovně: [10]

- Rámec je zahozen:
  - je-li rámec neúplný či chybný a pracuje-li přepínač v režimu s kontrolou minimální délky či s úplnou kontrolou,
  - je-li určen stanici, která je připojena ke stejnému segmentu sítě, odkud rámec přišel (v tomto případě není potřeba nic přepínat),
  - dojde-li k vyčerpání kapacity vstupní vyrovnávací paměti.
- Rámec je přepnut na odpovídající cílový port, byl-li v přepínací tabulce nalezen odpovídající záznam.
- Rámec je rozeslán do všech ostatních portů (formou **záplavy**):
  - není-li nalezen v přepínací tabulce odpovídající záznam (umístění cílové stanice je neznámé),
  - pokud je cílová adresa rámce všesměrová.

Při návrhu sítě je nutné vzít v úvahu, že všechny její segmenty propojené přepínači (rozbočovači) tvoří jedinou síť, což má za následek, že se celou sítí šíří rámce se všesměrovou adresou. Tato vlastnost může v případě rozlehlejší sítě s množstvím poskytovaných služeb a větším provozem způsobovat velkou zátěž. Z tohoto důvodu (ovšem i kvůli jiným faktorům jako je například vyšší bezpečnost) se začaly

používat virtuální LAN neboli **VLAN** (Virtual Local Area Network, specifikované standardem IEEE 802.1Q). V těchto sítích jsou stanice připojené k přepínači přidělené k určité síti VLAN a všesměrové rámce se šíří pouze uvnitř dané VLAN. Pro každou VLAN existuje separátní instance protokolu STP. [10]

Přepínače vyšší třídy mohou obsahovat dohledový modul **SNMP** (Simple Network Management Protocol) a také mohou umožňovat agregaci více portů do jediného směru pro zvýšení přenosové rychlosti a bezpečnosti spoje. Dále existují přepínače, které pracují až na **podvrstvě LLC** (Logical Link Control) a jednotlivé porty mohou tedy mít i odlišnou **MAC podvrstvu** (Media Access Control). Tento druh přepínačů pak umožňuje propojit sítě se shodnou LLC podvrstvou (např. Token Ring a Ethernet). [10]

1.2.3 Vlastnosti mostů

Mosty jsou na rozdíl od mnohaportových přepínačů vybaveny pouze dvěma porty. V pravidelných intervalech (každé 1 - 4 vteřiny) si vzájemně vyměňují zprávy pomocí **BPDU** (Bridge Protocol Data Unit) o konfiguraci.

Rozlišujeme dva typy mostů: [10]

- **Lokální** - používají se k propojení dvou segmentů sítí LAN nacházejících se v jedné lokalitě.
- **Vzdálené** - slouží k propojení vzdálených segmentů LAN přes transportní síť pevným či komutovaným spojem (např. ISDN). Komutované propojení pomocí mostů je však značně nevýhodné, poněvadž mosty nefiltrují rámce se všesměrovou adresou a spojení je tudíž často udržováno (a placeno) zbytečně jen kvůli přenosu často nepotřebných dat.

Většinu ostatních vlastností mají mosty a přepínače společnou.

1.2.4 STP - Spanning Tree Protocol

V případě moderních sítí se kvůli spolehlivosti a bezpečnosti budují **redundantní spoje**. Ovšem tyto redundantní spoje mohou být příčinou vzniku smyček v síti Ethernet, které vyžadují přísně stromovou strukturu. Vzniku smyček zabraňuje právě protokol **STP**, který dále řeší přesměrování toku dat v případě výpadku jednoho spoje, existuje-li spoj záložní. Přenos rámce více cestami může také způsobit jeho zacyklení a případně zahlcení sítě. Z tohoto důvodu se buduje tzv. **přenosová kostra sítě**, což je v podstatě topologická struktura, ve které existuje mezi libovolnou dvojicí komunikujících pouze jediná přenosová cesta. Přenosová kostra se

v redundantní síti přepínačů určuje pomocí distribuovaného algoritmu **STA** (Spanning Tree Algorithm). [8]

Existují dvě verze protokolu, a to **STP** a **RSTP** (Rapid STP specifikovaný standardem IEEE 802.1w). Verze RSTP zajišťuje v Ethernetových metropolitních sítích zotavení v případě výpadku páteřní linky (2 - 3 vteřiny), což představuje řádový rozdíl oproti standardnímu STP (30 - 40 vteřin). [10]

Princip funkce STP

Přepínače mají určeno své **identifikační číslo IDP** a jednotlivé porty jsou odlišeny pomocí **identifikačního čísla IDport**. Kostra je konstruována pomocí postupného výpočtu z tzv. **kořenového přepínače** (root switch) na principu upřednostňování přepínačů a portů s nejmenšími čísly. Kořenovým přepínačem se stává přepínač s minimální hodnotou IDP. Ostatní přepínače přiřadí v průběhu algoritmu každému ze svých portů jeden ze stavů: [8]

- **R (Root)** - takto je označen port vedoucí ke kořenovému přepínači (takto označený port je vždy jediný).
- **D (Designated)** - takto je označen port, kterým je ke kostře připojen jiný přepínač nebo segment.
- **B (Blocked)** - do tohoto stavu přejde port, který je pro běžný provoz zablokován (aby se zabránilo vzniku smyčky).

Během provozu vysílají přepínače do všech svých portů speciální rámce, ve kterých uvádějí:

- IDP kořenového přepínače (na počátku algoritmu uvádějí vlastní IDP),
- délku cesty v počtech spojů ke kořenovému přepínači (na počátku je nastavena na hodnotu 0),
- vlastní číslo IDP,
- číslo portu, ze kterého je daný rámec odeslán.

Tyto speciální rámce jsou v přepínači analyzovány a jsou na jejich základě průběžně aktualizovány stavy portů.

1.2.5 Zdrojové přepínání - Source Route Bridging

Pomocí zdrojového přepínání se propojují kruhy Token Ring na linkové úrovni. Zdrojový uzel si sám zjišťuje nejvhodnější cestu k cíli a vkládá ji do záhlaví rámců (konkrétně do pole **RIF** - Routing Information Field), které obsahují data pro cílovou stanici. Díky této technice je umožněna existence více cest na linkové úrovni. Průběh hledání nejvhodnější cesty lze popsat následovně: [10]

- Zdrojová stanice odešle průzkumný rámec, který prochází všemi mosty a kruhy směrem k cílové stanici.
- Mosty do průzkumného rámce postupně zaznamenávají identifikátor mostu, identifikátor kruhu, adresu odeslání rámce a velikost **MTU** (Maximum Transfer Unit). Tímto se vybuduje cesta a zamezí se vzniku smyček.
- Cílový uzel následně vrátí rámec po stejné trase zpět.
- Nakonec zdrojová stanice vyhodnotí vrácené rámce podle určitých kritérií (počet mezilehlých mostů, rychlost návratu, velikost MTU) a vybere optimální cestu.

1.3 Směrovače

Směrovače (routers) jsou síťové prvky, které svoji činnost rozšiřují oproti přepínačům a mostům o **síťovou vrstvu** modelu vrstvové síťové architektury OSI (dále jen síťová vrstva). Používají se pro propojení/oddělení jednotlivých sítí LAN, dále k připojení sítí LAN k sítím WAN nebo k propojení částí sítí WAN. Jejich hlavním úkolem je **směrování paketů** mezi jednotlivými sítěmi, které leží na přenosové cestě mezi zdrojovou a cílovou sítí. Díky tomu, že oddělují dílčí sítě, tak dochází k filtrování všesměrových paketů určených pro danou síť. Existují ovšem i nesměrovatelné protokoly (např. NetBios), které směrovače distribuují všesměrově (pokud jejich šíření podporují). Vzdálenost, na kterou se daný nesměrovatelný protokol šíří, bývá ovšem většinou omezena (počtem směrovačů v přenosové cestě). Díky tomu, že směrovače znají strukturu paketů, tak jsou v nich často implementovány bezpečnostní mechanizmy (tzv. firewall), které zvyšují bezpečnost v sítích. [10]

Princip funkce směrovačů je založen na **směrovacím mechanizmu** (nejčastěji se jedná o distribuovaný způsob směrování), kdy si jednotlivé směrovače díky vzájemné komunikaci podle určitého směrovacího protokolu budují vlastní **směrovací tabulku** (routing table). V ní jsou uloženy záznamy, které určují kam mají být pakety s danou cílovou sítí předány. Pakety mimo jiné nesou informace o síťové adrese zdroje a cíle. Síťová adresa bývá většinou rozdělena na dvě až tři části (adresu sítě,

adresu síťového rozhraní, případně ještě adresu podsítě). Ve směrovací tabulce bývají uloženy záznamy pouze o cílové síti (případně podsíti) a také běžně obsahuje záznam o implicitním směru pro směrování do sítí, pro které v tabulce neexistuje žádný záznam.

1.3.1 Směrovací tabulky

Směrovací tabulky (routing tables) jsou základní datovou strukturou pro směrování paketů a dělíme je na **statické** a **dynamické**.

Statické směrovací tabulky

Do tohoto typu tabulek směrovací informace vkládá administrátor **manuálně** při konfiguraci směrovače. Jakékoliv změny musí být prováděny také ručně nebo pomocí řídicího protokolu síťové vrstvy (např. protokolu ICMP protokolové sady TCP/IP). Tento způsob zadávání směrovacích informací je vhodný pouze pro jednoduché a stálé sítě. Statická směrovací tabulka obsahuje nejčastěji následující základní údaje: [10]

Cílová síť	Maska podsítě	Adresa následujícího směrovače	Síťové rozhraní	Stav rozhraní	Četnost zprac. paketů

Obr. 1.1: Základní údaje ve statické směrovací tabulce

- **Cílová síť** - zde je uložen záznam o IP adrese cílové stanice.
- **Maska podsítě** - udává, jaká část IP adresy je adresa podsítě. Slouží při prohledávání směrovací tabulky k určení rozhraní, kterým se bude daný paket posílat. Nejprve se cílová IP adresa vynásobí s maskou a výsledek operace se následně porovnává s hodnotou cílové sítě. V případě, že je více pozitivních výsledků, tak se vybere ten směr, pro který byla maska podsítě nejdelší (nejdelší sled jedniček).
- **Adresa následujícího směrovače** - udává adresu následujícího směrovače v přenosové cestě.
- **Síťové rozhraní** - určuje, přes které rozhraní se bude paket posílat.

- **Stav rozhraní** - nese informaci o stavu rozhraní (aktivní/neaktivní).
- **Četnost zpracovaných paketů** - podává informaci o množství paketů poslaných za určitou dobu daným směrem.

Dynamické směrovací tabulky

Tento druh směrování je v mnoha aspektech odlišný od statického směrování. Budování směrovací tabulky je zde **automatický proces**, který je založený na pravidelném vzájemném vyměňování informací o struktuře a stavu sítě mezi směrovacími uzly. Cílem je nalezení optimální cesty pro danou cílovou síť. Kritéria optimální cesty jsou: minimální počet spojů v cestě, minimální střední doba přenosu paketu v síti, maximální přenosová kapacita spoje dané cesty, maximální spolehlivost cesty [8]. Výhodou dynamických směrovacích tabulek je možnost rychlé adaptace na změny síťové topologie a možnost efektivně řešit směrování u rozsáhlých sítí. Za nevýhody můžeme považovat zatížení procesorů směrovačů a zatížení spojů režijním provozem. Výměny informací mezi směrovači jsou zajišťovány pomocí směrovacích protokolů. V sítích TCP/IP to jsou nejčastěji protokoly **RIP** (Routing Information Protocol) a **OSPF** (Open Shortest Path First). Dynamická směrovací tabulka obsahuje nejčastěji následující základní údaje: [10]

Cílová síť	Maska podsítě	Adresa následujícího směrovače	Síťové rozhraní	Cena/vzdálenost spoje	Stáří směrové informace	Stav rozhraní	Četnost zprac. paketů

Obr. 1.2: Základní údaje v dynamické směrovací tabulce

- **Cena/vzdálenost spoje** - udává, jak „výhodné“ je poslat paket danou cestou. Vhodnost spoje přitom může být určena na základě „vzdálenosti“ (počtu mezilehlých směrovačů) či jeho cenou, což je hodnota vypočtená z mnoha parametrů dané cesty (počtu mezilehlých směrovačů, přenosové rychlosti mezi jednotlivými úseky, hodnotou MTU pro jednotlivé úseky, momentálního stavu úseku, atd.).
- **Stáří směrové informace** - při dynamickém směrování je nutné pravidelně aktualizovat informace uložené ve směrovacích tabulkách. Pokud tedy není informace obnovena do určité doby, tak je považována za starou, a tedy neplatnou.

1.3.2 Směrovací protokoly

Směrovací protokoly a algoritmy slouží především k vytváření a udržování aktuálnosti směrovacích tabulek.

Podle místa použití dělíme směrovací protokoly na: [10]

- **Vnitřní** (IGP - Interior Gateway Protocol) - protokoly používané uvnitř autonomního systému (oblast s jednotnou směrovací politikou). Klasickými představiteli jsou například protokoly **RIP** a **OSPF**.
- **Vnější** (EGP - Exterior Gateway Protocol) - protokoly používané mezi hraničními směrovači odlišných autonomních systémů. Jedná se například o protokol **BGP** (Border Gateway Protocol).

Mezi další úkoly směrovačů patří překlad logických síťových adres na konkrétní fyzické adresy sítě. K tomuto překladu se používá například protokol ARP (Address Resolution Protocol) v sadě TCP/IP. Mezi další úkoly směrovačů patří rozdělení směrovaných paketů do několika menších tak, aby byla splněna podmínka MTU (Maximum Transmission Unit). K tomuto dělení dochází z toho důvodu, že směrovače bývají často připojeny k různým typům rozhraní s různou přenosovou rychlostí a odlišnou hodnotou MTU, která udává maximální velikost datové jednotky přenášené daným spojem. [10]

V případě, že nastane situace, že jsou informace ve směrovací tabulce chybné, tak může dojít k tomu, že paket nemůže být odeslán a bloudí sítí. Tento druh problému řeší směrovače kontrolou tzv. „doby života“ paketu (**TTL** - Time To Live). Hodnota TTL je uložena v záhlaví paketu a každým průchodem směrovačem se snižuje, dokud není paket doručen nebo dokud tato hodnota není nulová. V tom případě je paket zahozen, aby zbytečně nezatěžoval provoz v sítí. S touto činností souvisí přenos různých typů zpráv (nejčastěji chybových), které se týkají stavu síťové vrstvy. Mezi tyto zprávy patří: zprávy o nedosažitelnosti cíle, zprávy o zahození paketu, zprávy o existenci vhodnější cesty přes jiný směrovač, časové informace udávající zpoždění mezi zdrojovou a cílovou stanicí, apod. [10]

Dále můžeme směrovací protokoly dělit podle tzv. „znalosti topologie“ na protokoly: [8]

- **S úplnou znalostí topologie** (link state) - každý směrovač rozesílá informace o topologii svého okolí všem ostatním směrovačům sítě, a směrovače tak postupně získávají informace o topologii celé sítě. Na základě těchto informací vypočítávají nejlepší cesty sítí a odvozují směrovací tabulky. Takto pracují například protokoly **OSPF** a **IS-IS** (Intermediate System to Intermediate System).

- **Se zprostředkovanou znalostí topologie** (distance vector) - každý směrovač rozesílá informace o topologii sítě pouze svým sousedům (směrovačům připojeným přímo k jeho rozhraním), a směrovače tak postupně získávají znalosti o vzdálenostech mezi adresáty a sousedními směrovači. Na základě těchto informací a na základě údajů o vzdálenostech k sousedním směrovačům si budují směrovací tabulky. Takto pracují například protokoly **RIP** a **BGP**.

1.4 Brány

Používají se v případě, kdy je potřeba vzájemně propojit sítě zcela odlišných koncepcí, které používají zcela jiné soustavy protokolů. Jejich hlavní funkce tedy spočívá ve vzájemné konverzi protokolů. **Brány** (gateways) pracují na takové úrovni, na které je možné příslušnou konverzi realizovat - tedy například až na úrovni aplikační vrstvy.

1.5 Kombinované propojovací prvky

Vývoj techniky umožnil vznik **kombinovaných propojovacích prvků**, které umožňují pružné vytváření různých topologií při možnosti zachování starší a pomalejší techniky. Mezi nejčastější kombinované prvky patří: [10]

- **Dvourychlostní rozbočovače** - kombinují vlastnosti fyzické a linkové vrstvy. Tento druh rozbočovače existuje ve dvou podobách:
  - **Oddělené skupiny portů s různými rychlostmi** (většinou 10 nebo 100Mb/s) - tyto porty jsou uvnitř vzájemně propojeny nízkoportovým přepínačem.
  - **Libovolný z portů může pracovat s různými rychlostmi** (většinou 10 i 100 Mb/s) - funkce je založena na automatickém rozpoznání rychlosti komunikujících stran a připojení portu přes transceiver do patřičné skupiny portů sdílející danou přenosovou kapacitu. Tyto skupiny jsou potom propojeny a navenek připojeny přes přepínač.
- **Integrované přepínače/směrovače** - směrovač je integrován do přepínače, což umožňuje vzájemné propojení sítí VLAN a dále vytvoření hraničního prvku sítě LAN, který se směrem do sítě WAN (či rozdílné LAN) pracuje jako směrovač a směrem do sítě LAN jako přepínač.
- **Přepínače pracující na 3. vrstvě** (tzv. Layer 3 Switching) - v tomto případě se vlastně jedná o směrování realizované hardwarově. Jedná se v podstatě

o obdobu přepínání na linkové vrstvě (na základě MAC adres), na síťové vrtsvě je přepínání rovněž řešeno hardwarově a rozhodovací algoritmy jsou rozšířeny o další tabulku - tabulku logických adres (převážně IP).

2 FYZIOLOGICKÝ PŘEDOBRAZ UMĚLÝCH NEURONOVÝCH SÍTÍ

V této kapitole budou ve stručnosti popsána základní fakta o biologických neuronových sítích, neboť jejich poznání je základem pro hlubší porozumění sítím umělým.

2.1 Mozek

Jednou ze základních vlastností živých organismů je schopnost vnímat podněty z okolního světa i o svém vlastním stavu a vhodně na ně reagovat. Tuto činnost spolu s mnoha dalšími funkcemi zajišťuje **informační systém** daného organismu. 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. U člověka i u všech obratlovců funkci informačního systému organismu plní jeho **nervový systém** a **mozek**. Lidský mozek je jedním z nejdůležitějších systémů jak pro život jedince, tak pro lidstvo jako celek. Současně je to jedna z nejsložitějších známých soustav vůbec. [1]

Obsahuje 40 až 100 miliard **neuronů**, které jsou navzájem velmi složitě propojeny, vzájemně interagují a jejich struktura i vzájemné vztahy se v čase neustále vyvíjejí. Každý z neuronů má až 10 000 vstupů, které jej spojují prostřednictvím **synapsí** (specializované stykové jednotky - rozhraní) s výstupy jiných neuronů. Na informačních funkcích mozku se podílí i mnoho jiných specializovaných buněk (např. buňky gliové), díky tomu se odhaduje strukturální složitost mozku do řádu 10^{15} až 10^{16}. [1]

V centrálním nervovém systému člověka hrají neurony velmi důležitou roli při realizaci mnohých složitých řídicích, senzorických a kognitivních procesů. Neurony ve vzájemné součinnosti působí v paralelních řetězových strukturách vytvářejících vrstevnaté pyramediální útvary[1]. Informace se těmito útvary šíří většinou vpřed za současného a trvalého působení složitých zpětných vazeb. [1]

Z tohoto důvodu je možné modelovat chování a vlastnosti biologických neuronových sítí působících v mozku pomocí dynamických systémů.

2.2 Biologický neuron

Neurony jsou buňky specializované na přenos, zpracování, uchování a využívání informací. Těmto činnostem je přizpůsobena i jejich vnitřní struktura, skladba, funkce

[1] „Pyramediální“ je zde myšleno ve smyslu postupného snižování počtu - konvergence - signálu ve směru šíření.

a vnější vzhled. K nejnápadnější části neuronu patří **soma** (tělo) s buněčným jádrem uvnitř, které je vyplněno tzv. plazmou obklopenou velice tenkou membránou. Z těla typického neuronu obvykle vyrůstají dva druhy nervových výběžků, a to **dendrity** a **axony**. Dendrity vedou vzruch směrem k buňce a představují tedy vstupy do těla neuronu. Jedná se o krátké a tenké stromečkovitě rozvětvené výběžky. Výstup z neuronu je realizován pomocí axonu, který je mnohem silnější než dendrity a jeho délka se u člověka pohybuje od několika mikrometrů až asi po 60 cm. Obsahuje tzv. axoplazmu, která je obalena tenkou membránou. Většinu axonů navíc obklopuje myelinová pochva přerušovaná v určitých vzdálenostech tzv. **Ranvierovými zářezy**. V místech zářezů je axon v přímém kontaktu s okolím. Hlavní funkcí Rainvierových zářezů při přenosu informací je přispívat k rychlému a nezkreslenému přenosu dlouhými axony. Můžeme je přirovnat k opakovačům na elektrických vedeních. Na koncích bývají axony bohatě rozvětveny do tzv. kolaterálních či terminálových vláken. Na rozdíl od dendritů, kterých je u běžného neuronu velké množství, je axon u každého obvyklého druhu neuronu pouze jeden. [1]

Spojení neuronů navzájem (ovšem ne pouze laterálně, tedy na úrovni jednotlivých vrstev, ale i mezi jednotlivými vrstvami[2] či ob několik vrstev) zajišťují velmi specializované složité útvary - **synapse**. Jedná se tedy o **informační rozhraní** (interface) mezi jednotlivými neurony. Jejich počet až několikanásobně převyšuje počet neuronů samých a přímo závisí na složitosti propojení (tj. na topologii) příslušné neuronové sítě.

V závislosti na tom, v jakém místě se synaptický přenos uskutečňuje, mluvíme o synapsích **axosomatických** - axon dosedá na tělo jiné buňky, **axodendritických** - axon dosedá na dendrit jiné buňky (tento druh převažuje) a **axoaxoální** - axon končí na presynaptické části jiného axonu. [1]

Ovšem synapse nepůsobí pouze jako mezineuronová rozhraní uzpůsobená pro funkci přenosu signálu mezi neurony, ale také pro vznik **paměťových stop**.

V biologických neuronových soustavách existuje několik druhů synapsí, které se navzájem odlišují mechanismem přenosu vzruchů. U nižších živočichů, jako jsou například ryby, dominují synapse elektrické. U vyšších živočichů, především u primátů a u člověka, se nejčastěji vyskytují synapse chemické. Existují u nich však také synapse elektrické, mechanické i jiné kontakty mezi neurony, založené na přímém styku jejich membrán. [1]

Chemické synapse umožňují přenos informací po větších kvantech. Elektrické synapse naproti tomu představují elementární přenosy informací prostřednictvím iontů. Můžeme o nich říci, že pracují na úrovni přenosu jednotlivých bitů. [1]

[2]Lidský mozek má celkem šest „vrstev“, ovšem tvorové stojící níže na žebříčku vývoje svého informačního systému mají počet vrstev neocortexu nižší.

Vstupní a výstupní část synapse jsou navyzájem odděleny **synaptickou štěrbinou**. Přes tuto štěrbinu přecházejí jednotlivé nosiče informace (transmitéry) z výstupní části synapse do části vstupní. [1]

Průchodnost signálu synapsemi však není konstantní, ale adaptivně se mění dle konkrétních reálných situací. Z toho důvodu hovoříme o tom, že mají velkou (synaptickou) **plasticitu**. Díky tomu se může průchodnost synapsí měnit při každém průchodu signálu. Tento děj je také základem při procesu učení neuronových sítí, který má pro činnost neuronových sítí a pro život živých bytostí zcela mimořádnou a zásadní důležitost. [1]

Rychlost šíření signálu v axonovém vlákně se pohybuje zhruba od $0,5$ až do 2 m/s. Tyto rychlosti šíření sice umožňují přenos informací mezi dvěma spolupracujícími neurony za dobu 20 až 40 ms (tato doba je považována za jakési časové kvantum pro činnost mozku), ovšem pro přenos velkých objemů informací, které se očividně mezi různými částmi mozku navzájem přenášejí jsou, velmi malé. Z toho vyplývá, že přenos, zpracovávání a uchovávání informací v mozku se v mnoha situacích musí dít nikoliv sériově (tj. bit za bitem po jedné signálové cestě), ale **paralelně** po velkém počtu současně, i když ne nutně vždy synchronně pracujících cest. [1]

3 STRUČNÝ PŘEHLED HISTORIE UMĚLÝCH NEURONOVÝCH SÍTÍ

V roce 1921 předpověděl americký vědec McCulloch, že v budoucnu dojde k umělému napodobování biologických neuronových sítí, pomocí kterých se budou zpracovávat informace. Tento vědec se spolu se svým studentem W. Pittsem stali významnými průkopníky v tomto oboru, když se jim v roce 1943 povedlo navrhnout a popsat reálně použitelný **matematický model biologického neuronu**. McCullochův a Pittsův model neuronu, který se také nazývá **Perceptron**, našel po druhé světové válce velmi širokého uplatnění a je základem totální většiny umělých neuronových sítí (dále jen neuronových sítí) i v dnešní době, i když často v mírně modifikované, doplněné a zlepšené podobě. [6]

O šest let později v roce 1949 publikoval D. Hebb tzv. **Hebbův zákon učení** neuronových sítí. Pomocí tohoto zákona bylo možno popsat postup učení neuronových sítí a také na jeho základě přikročit ke konstrukci učících algoritmů, prakticky použitelných pro řešení konkrétních úloh jednoduchými **vrstevnatými neuronovými sítěmi**.

V následujících letech pracovala na problematice neuronových sítí celá řada významných vědců a bylo zveřejněno mnoho významných prací. K nejznámnějším z nich patří práce M. Minského (z roku 1954) a F. Rosenblatta (z roku 1958). Rosenblatt zveřejnil učící algoritmus vrstevnatých neuronových sítí s **dopředným přenosem** signálu mezi vstupem a výstupem. Následně pak prof. B. Windrow ze Standfordovy univerzity v Kalifornii a Hoff odvodili jeho zajímavou modifikaci, která je dnes známá jako **Windrowův a Hoffův zákon**.

V roce 1960 pak prof. B. Windrow a D. Hebb představili strukturu s jedním výkonným prvkem (neuronem), který pracuje v **bipolárním režimu**, a který přijímá kromě vstupů z několika jiných jednotek také doplňkový "jednotkový" signál, který je trvale roven +1. Tuto strukturu nazvali **ADALINE** a později uvedli její vylepšení **MADALINE**. Ve své době se používaly tyto struktury v řadě praktických aplikací v civilní i vojenské sféře a v některých aplikacích se používají doposud. [6]

Díky těmto pokrokům se jevila budoucnost neuronových sítí velmi slibně a mnoho vědců předpovídalo velmi brzké vytvoření umělého lidského mozku a lidského myšlení, ačkoli tato tvrzení nebyla nijak vědecky podložena a byla velmi optimistická. Ovšem druhá polovina šedesátých let přinesla vědeckému výzkumu v oboru neuronových sítí velkou ránu. Přičinili se o to v roce 1969 Minsky a Papert, autoři knihy Perceptron, ve které ukázali teoretické meze možností jednovrstvé (lineární) neuronové sítě - Perceptronu. Konkrétně se jednalo o jejich matematický důkaz, že touto sítí nelze realizovat ani funkci elementárního logického funkčního bloku **XOR**.

Tento důkaz byl sice správný, ovšem jeho sugestivní podání a následná kampaň, kterou oba autoři provedli, měla za následek (ovšem spolu s jinými faktory, jako bylo například uvedení mikroprocesorů), že zájem veřejnosti a s tím též podpora sponzorů se odklonila od výzkumu neuronových sítí k jiným projektům a tím byl výzkum tohoto oboru na celém světě velmi výrazně utlumen. Tento útlum trval následujících 20 let a dokonce se i ustoupilo od nasazení neuronových sítí v praktických aplikacích, a to i přesto, že v předchozích letech vedlo jejich používání v některých aplikacích k prokazatelným úspěchům. Někteří vědci ovšem pracovali na jejich výzkumu i nadále, ovšem bez potřebné bohaté finanční podpory jako doposud. I přesto se těmto vědcům podařilo získat během těchto 20 let velmi cenné, převážně ovšem teoretické poznatky. Mezi významné práce z této doby patří publikace T. Kohenena z roku 1979 o **modelech asociativních pamětí**. [1]

Výrazný zlom ve výzkumu neuronových sítí nastal až v roce 1983 díky americké výzkumné instituci **DARPA** (Defence Advance Research Project Association), která do jejich výzkumu investovala několik desítek milionů dolarů a tím vytvořila potřebné podmínky pro další výzkum. [6] Vytvořily se tím příznivé podmínky pro nasazení neuronových sítí v praktických aplikacích. DARPA byla poté následována jinými významnými sponzory výzkumu po celém světě a tento vysoký zájem o neuronové sítě trvá v podstatě dodnes.

4 UMĚLÉ NEURONOVÉ SÍTĚ

Vzhledem ke složitosti lidského mozku a biologických neuronových sítí, o nichž bylo stručně napsáno v kapitole 2 je jasné, že se lidstvu v dohledné době nepodaří vytvořit umělé systémy, které by ve větším rozsahu napodobili nebo dokonce nahradily lidský mozek. Ovšem již v dnešní době máme dostatek poznatků a výpočetního výkonu k tomu, abychom technickými prostředky napodobily některé dílčí jednodušší funkce mozku, i když nesrovnatelně jednoduššími umělými neuronovými sítěmi a řešili pomocí nich praktické problémy, které se konvenčními prostředky buď nedají řešit vůbec a nebo jen s velkými obtížemi a omezeními. [1]

Umělé neuronové sítě mohou naše biologické neuronové sítě dokonce v některých velmi specifických případech a z některých specifických hledisek (jako je např. jejich odolnost vůči vnějším vlivům, provozní spolehlivost, operační rychlost apod.) překonat. Při jejich navrhování v zásadě nemusíme vždy vycházet z principu napodobování přírodních systémů, tj. informačních systémů živých organizmů. Modely používané v některých aplikacích se proto dosti podstatně odchylují od principů, na nichž pracují biologické neurony. Ve většině případů je ovšem přírodní vzor alespoň částečně napodobován. [1]

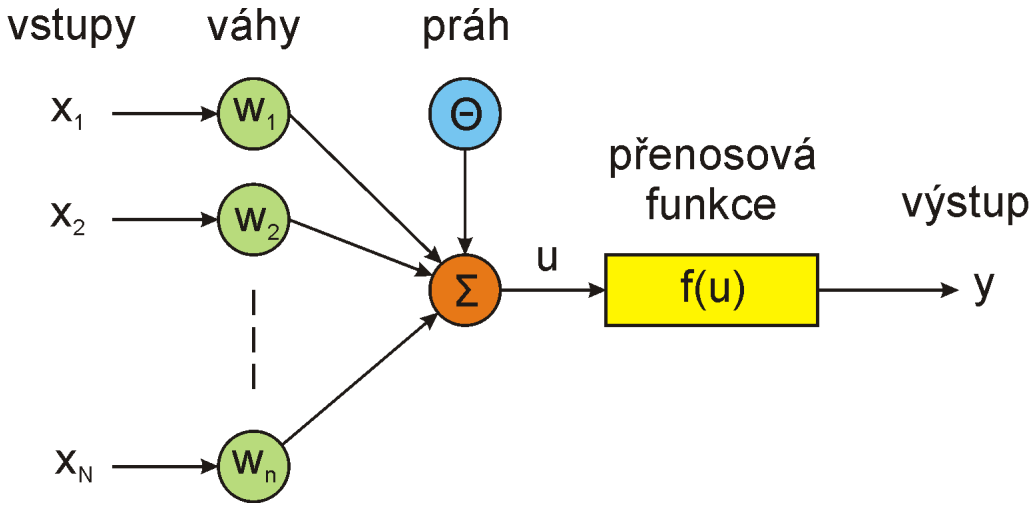
V této kapitole bude stručně popsáno z čeho se umělé neuronové sítě skládají a jakými parametry je popisujeme. Následně se podíváme na jejich základní topologie a k řešení jakých problémů se dají některé z nich s výhodou použít. Následující kapitola se bude věnovat umělým neuronovým sítím s **reflexním nastavením vah**. Především pak bude pojednáno o modelu **Hopfieldovy sítě**, která bude použita k optimalizaci datových jednotek v síťovém prvku.

4.1 Matematický model neuronu

Matematické modely neuronu jsou základními stavebními jednotkami umělých neuronových sítí. Rozdíly mezi jednotlivými modely jsou vytvářeny jak používanými **matematickými funkcemi**, tak různou **topologií** vlastního modelu. [6] Obecně se jedná o jistý matematický procesor, do kterého vstupuje n-rozměrný vektor vstupních signálů $\boldsymbol{x}(t) \in R^n$ a na jeho výstupu se objeví jediný výstupní signál $y(t) \in R^1$. Vstupní vektor přitom představuje signály přicházející z n sousedních neuronů (případně vlastních rekurentních spojů). Obecně lze činnost takto zavedeného matematického procesoru popsat jako zobrazení $\boldsymbol{N}$ ze vstupního prostoru R^n do výstupního prostoru R^1 [2]

$$\boldsymbol{N} : \boldsymbol{x}(t) \in R^n \rightarrow y(t) \in R^1. \tag{4.1}$$

Nejrozšířenějším matematickým modelem neuronu je tzv. **formální neuron**, který je podle svých autorů také často nazýván **McCulloch-Pittsův model neuronu** (obr.4.1). [1]



Obr. 4.1: Model McCulloch - Pittsova neuronu.

Mezi základní části tohoto modelu neuronu patří: [5]

- **Vstupní vektor** (input vector) - kde x_i (pro $1 \leq i \leq n$) představují elementy vstupního vektoru.
- **Vektor synaptických vah** (synaptic weights vector) - kde prvky w_{ij} (pro $1 \leq i \leq n$) představují elementy vektoru synaptických vah j-tého neuronu, který vyjadřuje uložení zkušeností do neuronu. Pomocí nich je neuron schopen adaptovat se na nově získané zkušenosti během učení. Při učení jsou tedy hledány optimální hodnoty w_{ij} tak, abychom pro vstupy z trénovací množiny dostali odpovídající odezvu na výstupu neuronu.
- **Práh - prahová hodnota** (bias, threshold) - hodnota prahu Θ_j j-tého neuronu určuje, kdy je neuron **aktivní** (vážená suma vstupů musí být větší než práh), resp. **neaktivní** (v tomto případě musí být vážená hodnota vstupů menší než práh).
- **Přenosová (aktivační) funkce** - může mít různý tvar, obecně bývá nelineární, spojitá nebo diskrétní.
- **Vnitřní potenciál neuronu** - reprezentuje stav j-tého emitovaného neuronu (vztah 4.4).

- **Výstup z neuronu** - výstup y se u McCulloch-Pittsova modelu neuronu vypočítá podle vztahu (4.2) [7].

$$y = \sum_{1}^{n} x_i w_i + \Theta \tag{4.2}$$

Někdy se také používá ve tvaru (4.3) [7]. V tomto případě představuje nultá váha w_0 hodnotu prahu Θ a nultý vstupní parametr x_0 je roven 1.

$$y = \sum_{0}^{n} x_i w_i, \quad w_0 = \Theta, \; x_0 = 1 \tag{4.3}$$

Následující oddíl se bude blíže věnovat obvodovým a přenosovým funkcím.

4.2 Obvodové a přenosové funkce

Obvodové a **přenosové (aktivační) funkce** mívají různý tvar. Obecně jsou nelineární, spojité nebo diskrétní. Jako obvodová funkce se nejčastěji používá **vážená lineární kombinace vstupů** (používá se u McCulloch-Pittsova neuronu) [6].

$$u_j = (\sum_{i=1}^{n} w_{ij} x_i + \Theta_j) \tag{4.4}$$

Takto definovaná přenosová funkce dělí vstupní prostor R^n lineárně prostřednictvím tzv. n-rozměrných nadrovin. Rovněž je možné použít vztah (4.5) [6]

$$u_j = \prod_{i=1}^{n} w_{ij} x_i. \tag{4.5}$$

Obě tyto funkce tj. (4.4) a (4.5) jsou tzv. **lineární basické funkce (LBF)**. Aktivační potenciál u_j definovaný následující rovnicí (4.6) patří do tzv. **radiálních basických funkcí (RBF)** [4], které umožňují rozdělit vstupní prostor prostřednictvím určitého počtu hyperkoulí.

$$u_j = \sqrt{\sum_{i=1}^{n} (x_i - w_{ij})} \tag{4.6}$$

Při použití některé z těchto funkcí bude pro výstupní funkci neuronu y modelu McCulloch-Pittsova neuronu (obr.4.1) platit

$$y_j = f(u). \tag{4.7}$$

Přenosové funkce převádí hodnotu výstupu obvodové funkce u do hodnoty výstupu neuronu y. Většinou se používají následující typy přenosových funkcí (kde e je základ přirozeného logaritmu a T je **strmost funkce**): [11]

- **sigmoida**

$$f(u) = \frac{1}{1 + \mathrm{e}^{u/T}}, \tag{4.8}$$

- **hyperbolická tangenta** (často používaná pro LBF)

$$f(u) = \tanh\left(\frac{u}{T}\right), \tag{4.9}$$

- **inverzní tangenta**

$$f(u) = \frac{2}{\pi} \tan^{-1}\left(\frac{u}{T}\right), \tag{4.10}$$

- **skoková funkce**

$$f(u) = \begin{cases} 1 & \text{pro} \quad u \geq 0 \\ -1 \quad \text{nebo} \quad 0 & \text{pro} \quad u < 0, \end{cases} \tag{4.11}$$

- **gaussova funkce** (často používaná pro RBF)

$$f(u) = \exp\left[-(u^2)\right], \tag{4.12}$$

- **lineární funkce**

$$f(u) = au + b. \tag{4.13}$$

Výběr konkrétní přenosové funkce pro danou UNS je závislý na typu úlohy, která má být řešena. Výběr funkce má také vliv na výpočetní náročnost a čas potřebný k trénování UNS. Lineární funkce se používá např. pro lineární aproximaci (filtraci) a skoková funkce je vhodná ke klasifikaci vstupních vzorů do dvou tříd. Pokud je potřeba modelovat nějakou funkční závislost, tak je vhodné použít nelineární funkce typu hyperbolické tangenty nebo sigmoidy (obě funkce jsou diferencovatelné a používají se tedy často při učení vrstevnatých sítí se zpětným šířením gradientu chyby). K řešení klasifikačních problému lze zase s výhodou použít některé z přenosových funkcí typu RBF. Při některých metodách analýzy signálů se nově také jako přenosové funkce využívají wavelet funkce. Jedná se například o analýzy signálů typu seismické chvění, řeč, obrazová data v medicíně, hudba, finanční data apod. [6]

4.3 Klasifikace UNS

Neuronovou síť charakterizuje její struktura[1], použité modely výkonných prvků (neuronů) a způsob její funkce (tj. učení a vybavování). Všechny tyto dílčí charakteristické vlastnosti nazýváme **paradigma neuronové sítě**. Paradigmat neuronových sítí existuje v dnešní době několik desítek a s postupujícím výzkumem jejich počet stále roste. Pro úspěšné vyřešení určité problematiky v praktických aplikacích je výběr vhodného paradigmatu sítě zcela zásadní. Některá hlavní paradigmata a jejich klasifikace z hlediska informačního toku a metod učení shrnuje tabulka 4.1 [5].

Tab. 4.1: Klasifikace UNS z hlediska informačního toku a metod učení

Způsob učení	Rekurentní sítě	Dopředné sítě
S učitelem	Brain-state-in-a-box	Boltzmannův stroj Neocognitron Backpropagation Perceptron Adaline/Madaline
Bez učitele	Hopfieldova síť ART1 i ART2 Obousměrná asociativní paměť	Lineární asociativní paměť LVQ

Problematika a možnosti použití jednotlivý paradigmat je velmi rozsáhlá a omezený prostor této práce neumožňuje se jí dopodrobna zabývat. Proto se bude následující kapitola snažit popsat pouze možnosti a princip sítí s reflexním nastavením vah, a to především **Hopfieldovy sítě**, která bude použita pro řízení spojovacího pole aktivního síťového prvku.

[1] „struktura“ - způsob uspořádání výkonných prvků sítě a jejich vzájemné propojení

5 SÍTĚ S REFLEXNÍM NASTAVENÍM VAH

Jedná se o skupinu sítí, které nemají vrstevnatý charakter, jako například vrstevnaté sítě typu Perceptron, ADALINE a MADALINE. Parametry jsou u těchto sítí nastaveny buď explicitně nebo jsou předem nastaveny aplikováním vhodného jednonárazového nastavovacího učebního procesu. Hlavními představiteli tohoto typu sítí jsou **Asociativní paměti**, **Hopfieldovy sítě** a **Grossbergovo učení**. [1]

5.1 Asociativní neuronové sítě

Funkce těchto sítí je založena na **asociativním učení**, při němž jsou parametry sítě explicitně **jednonárazově nastaveny**. Princip tohoto asociativního učení spočívá v přiřazování vstupního vektoru k sobě samému (vstupní vektor je tedy zároveň i vektorem výstupním).

Asociativní sítě se používají k rekonstrukci neúplného nebo částečně zkresleného podnětu přivedeného na vstup sítě do podoby původního nezkresleného vzoru (např. odstraňování šumu, k optickému rozpoznávání znaků - OCR, atd.).

Podle jejich uspořádání a funkce mohou být rozděleny na: [6]

- **Přímovazební** - jednonárazové vybavování.
- **Zpětnovazební** - iterační proces vybavování (spojité Hopfieldovy sítě, ART, Grossberg).
- **Lineární** (LAM).
- **Nelineární**.

5.1.1 Přímovazební a zpětnovazební asociativní sítě

Hlavní rozdíl mezi těmito dvěma druhy sítí je v charakteru procesu vybavování daného vzoru. U přímovazebních sítí se jedná o **jednorázový proces**, zatímco u zpětnovazebních sítí se vzor vybavuje **iteračně**.

5.1.2 Lineární asociativní síť

Jedná se o jednovrstvou přímovazební neuronovou síť, jejímž účelem je obnovení naučeného souboru výstupních vzorů na základě úplné nebo jen částečné informace ze vstupních vzorů. Vstupní vzory jsou tvořeny vektorem

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T, \tag{5.1}$$

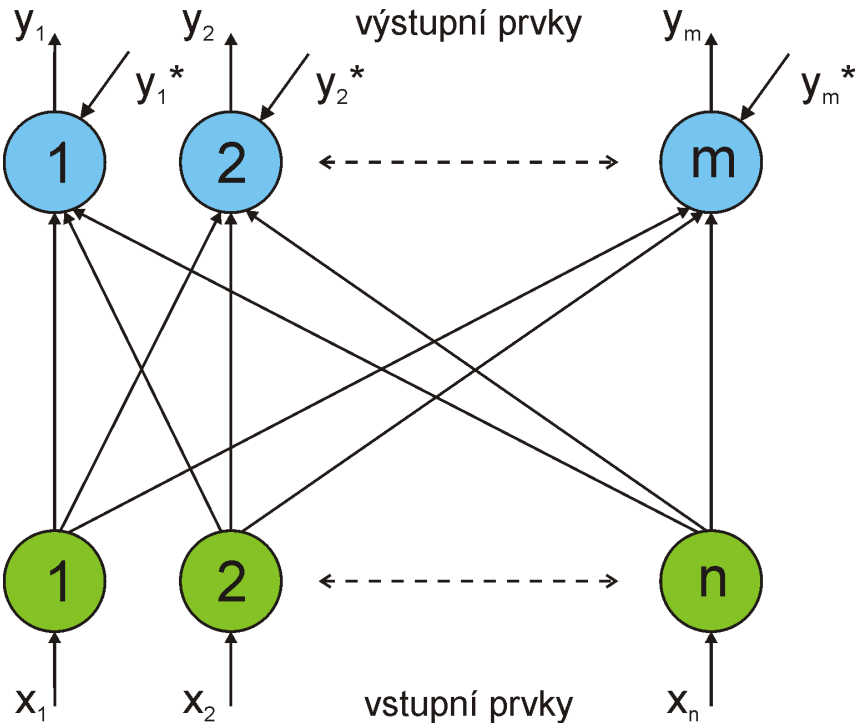
typu $n \times 1$.

Výstupní vzory jsou tvořeny vektorem

$$\boldsymbol{y} = [y_1, y_2, \ldots, y_m]^T, \tag{5.2}$$

typu $m \times 1$.

Síť může pracovat v jednom ze dvou režimů, buď **heteroasociativním** nebo **autoasociativním**. Ve většině případů pracuje v heteroasociativním režimu, kdy jsou dimenze vstupních a výstupních vzorů různé (tj. $m \neq n$). Ve speciálním případě, kdy $m = n$ pracuje v autoasociativním režimu. Struktura lineární asociativní sítě (paměti) je znázorněna na obrázku (5.1). [1]



Obr. 5.1: Model lineární asociativní sítě

Pro matici vah $\boldsymbol{W}$ platí vztah [6]

$$\boldsymbol{W} = \sum_{k=1}^{p} x_k y_k, \tag{5.3}$$

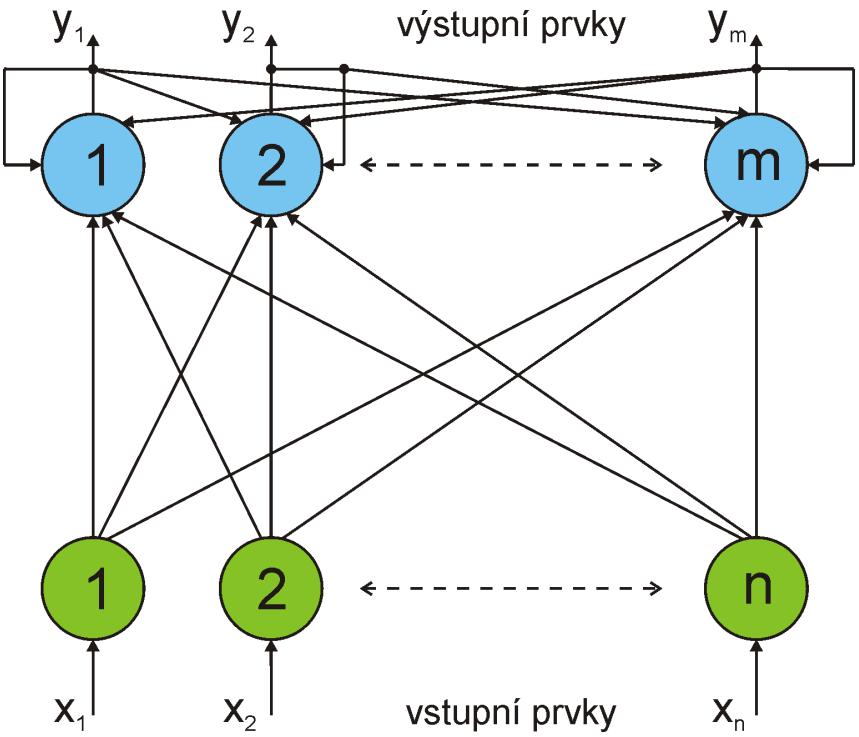
kde $k = 1, \ldots, p$ je počet vzorů.

Při tomto označení lze **aktivní dynamiku** lineární asociativní sítě formálně zapsat jako maticový součin [3]

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}. \tag{5.4}$$

5.1.3 Nelineární asociativní síť

Využití nelineárních technik umožňuje úplnou či alespoň částečnou eliminaci vlivu poruch, které mohou vznikat u lineárních asociativních sítí. Struktura umělé neuronové sítě, která pracuje jako nelineární asociativní síť, je znázorněna na obrázku 5.2. [1]



Obr. 5.2: Model nelineární asociativní sítě

U tohoto typu sítě je pomocí skalárního součinu testovacího vzoru $\boldsymbol{t}$ s jednotlivými naučenými vzory $\boldsymbol{a}^{(i)}$ definován tzv. **hodnotící vektor** [1]

$$\boldsymbol{s} = [< \boldsymbol{a}^{(1)}, \boldsymbol{t} >, < \boldsymbol{a}^{(2)}, \boldsymbol{t} >, \ldots, < \boldsymbol{a}^{(m)}, \boldsymbol{t} >], \tag{5.5}$$

kde $< \boldsymbol{a}^{(1)}, \boldsymbol{t} >$ představuje skalární součin vektorů $\boldsymbol{a}^{(1)}$ a $\boldsymbol{t}$ [1]

$$< \boldsymbol{a}^{(i)}, \boldsymbol{t} > \equiv \boldsymbol{a}^{(i)^T} \boldsymbol{t} = \sum_{j=1}^{k} \boldsymbol{a}_j^{(i)} \boldsymbol{t}_j. \tag{5.6}$$

Následně je hodnotící vektor $\boldsymbol{s}$ podroben nelineárnímu zpracování, jehož výsledkem je **binární rozhodovací vektor** $\boldsymbol{v} = N\{\boldsymbol{s}\}$, který obsahuje právě jeden nenulový prvek. Pokud je tento nenulový prvek ve vektoru $\boldsymbol{v}$ správně umístěn, získáme

na výstupu nelineární asociativní sítě správnou odezvu jako výsledek operace

$$\boldsymbol{Bv} = \begin{pmatrix} b_1^{(1)} & b_2^{(1)} & \dots & b_k^{(1)} \\ b_1^{(2)} & b_2^{(2)} & \dots & b_k^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{(n)} & b_2^{(n)} & \dots & b_k^{(n)} \end{pmatrix} \cdot \boldsymbol{v}, \tag{5.7}$$

kde $\boldsymbol{B}$ je matice tvořená řádkovými vektory naučených vstupů.

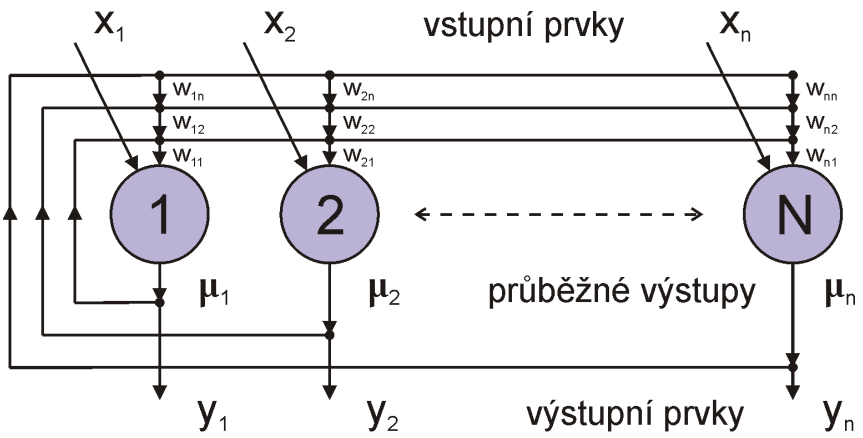
5.2 Hopfieldovy sítě

Autorem této neuronové sítě je John Hopfield, který ji uvedl počátkem osmdesátých let při studiu autoasociativních pamětí. Během vývoje podrobně rozpracoval použití **energetické funkce**, která má pro správnou funkci sítě fundamentální význam. Důležitost energetické funkce spočívá hlavně v tom, že jsou z ní odvozena pravidla pro učení a vybavování sítě. V současnosti existuje několik modifikací této sítě. Hopfieldova síť bývá nejčastěji používána jako **asociativní paměť**, **klasifikátor** a nebo k řešení **optimalizačních problémů**. Tato práce je zaměřena především na využití Hopfieldovy sítě k řešení optimalizačních problémů.

5.2.1 Struktura Hopfieldovy sítě

Hopfieldova síť obsahuje tolik neuronů, kolik má vstupů. Jedná se o úplně propojenou neuronovou síť, u které je výstup každého jejího výkonného prvku (neuronu) spojen se vstupem všech ostatních výkonných prvků (kromě sebe sama). Tento druh sítí řadíme do skupiny **rekurentních** (zpětnovazebních) sítí, které mají iterační aktualizaci stavů. Strukturální uspořádání dvourozměrné Hopfieldovy sítě je zobrazeno na obrázku 5.3 [12]. Poněvadž sítě Hopfieldova typu neobsahují skryté neurony, nejsou schopny kódovat data.

Zde $x_1, x_2, \dots, x_n$ označují vstupy sítě, $\mu_1, \mu_2, \dots, \mu_n$ jsou stavy v jednotlivých časových krocích, které se v následujícím kroku stávají opět vstupy a $y_1, y_2, \dots, y_n$ jsou výstupy sítě. Posledními použitými symboly na předchozím obrázku jsou $w_{11}, w_{12}, \dots, w_{nn}$, které značí váhy jednotlivých spojů mezi neurony. Váhy, které vedou z jednoho neuronu do druhého, jsou v obou směrech identické ($w_{ij} = w_{ji}$) a váhová matice je tedy **diagonálně symetrická**. Poněvadž se na vstupy neuronů nikdy nepřivádí jejich vlastní výstupy, tak jsou váhy na diagonále matice vah nulové ($w_{ii} = 0$). V konkrétním čase se mění hodnota synaptických vah pouze jediného neuronu, a to **McCulloch-Pittova modelu**. Jednotlivé neurony obsahují, stejně jako



Obr. 5.3: Struktura Hopfieldovy sítě

jiné druhy neuronů (např. perceptron), svůj vlastní práh a přenosovou funkci. Zvláštností Hopfieldových sítí je, že při učení sítě jednomu vzoru se jedná o **jednorázový proces**, ovšem při vybavování se jedná o opakovaný **iterační proces**. [12]

Hopfieldovy sítě dělíme na dvě hlavní skupiny:

- Binární.
- Spojité.

Nejdříve byla uvedena **binární Hopfieldova síť**, která bude popsána jako první.

5.2.2 Binární Hopfieldův model

Tento model může být buď **sekvenční** (pracuje asynchronně) nebo **paralelní** (pracuje synchronně).

Sekvenční model

Synaptické váhy w_{ij} a prahy Θ_i jednotlivých neuronů v síti jsou na začátku učení (inicializace) určeny podle vztahů (5.8) [6] a (5.9) [6].

- V případě binárních vzorů x_i (s hodnotami 0 a 1):

$$w_{ij} = \begin{cases} \sum_{k=1}^{m} 2x_i - 1 & \text{pro} \quad i \neq j \\ \\ 0 & \text{pro} \quad i = j \end{cases} \qquad \Theta = \frac{1}{2} \sum_{j=1}^{n} w_{ij}, \tag{5.8}$$

kde m je počet vzorů, n je počet vstupů.

- V případě bipolárních vzorů x_i (s hodnotami -1 a 1):

$$w_{ij} = \begin{cases} \sum_{k=1}^{m} 2x_i x_j & \text{pro} \quad i \neq j \\ \\ 0 & \text{pro} \quad i = j \end{cases} \qquad \Theta = \frac{1}{2} \sum_{j=1}^{n} w_{ij}, \tag{5.9}$$

  kde m je počet vzorů, n je počet vstupů.

Uveďme si nyní algoritmus funkce tohoto typu Hopfieldovy sítě:

1. Nastavení parametrů sítě dle předchozích vztahů.
2. Přivedení vstupního vektoru $\boldsymbol{x}$ na vstup sítě.
3. Počáteční stavový vektor sítě je nastaven na hodnotu $\boldsymbol{x}(0) = [x_1(0), x_2(0), \ldots, x_n(0)]^T$.
4. Iterativní proces nastavení a aktualizace jednotlivých složek stavového vektoru. Tato aktualizace probíhá vždy ve dvou krocích: [1]
   (a) Výpočet aktivity:

$$u_i(t+1) = \sum_{j=1}^{n} w_{ij} x_j(t) + \Theta_i, \tag{5.10}$$

   kde $j = 1, \ldots, n$.

   (b) Aktualizace stavu:

$$x_i(t+1) = \begin{cases} 0 & \text{pro} \quad u_i(t+1) < 0 \\ 1 & \text{pro} \quad u_i(t+1) > 0 \\ x_i(t) & \text{pro} \quad u_i(t+1) = 0 \end{cases}, \tag{5.11}$$

   kde x_i je složka vektoru vzoru.

Iterativní proces se opakuje i v dalších iteracích, dokud síť po konečném počtu kroků nedospěje do **stabilního stavu**. Kritériem přitom většinou bývá shoda stavů ve dvou po sobě následujících iteracích, kdy se síť dostane do **lokálního minima energetické funkce**. Toto lokální minimum se nazývá **atraktor**. Atraktor tedy představuje bod ve stavovém prostoru, který má energii menší nebo rovnu energii bodů ve svém okolí. Můžeme tedy říci, že pro sekvenční Hopfieldův model je každé lokální minum sítě atraktorem a naopak.

Princip hledání minima energické funkce bude nyní popsán podrobněji. Během každé iterace dochází u sekvenční Hopfieldovy sítě k aktualizaci jedné složky stavového vektoru, což má za následek změnu hodnoty **energické funkce** podle vztahu [1]

$$E = \frac{1}{2} \sum_i \sum_j w_{ij} x_i x_j + \sum_j \Theta_i x_i. \tag{5.12}$$

Jestliže označíme přírůstek prvku stavového vektoru $\Delta x_i(t) = x_i(t) - x_i(t-1)$ a přírůstek energetické funkce $\Delta E = E(t) - E(t-1)$, tak můžeme zapsat změnu energetické funkce během jedné aktualizace následovně [1]

$$\Delta E(t+1) = -u_i(t+1)\Delta x_i(t+1). \tag{5.13}$$

Ze vztahu (5.11) je patrné, že $\Delta x_i(t+1)$ je kladné (záporné) jen tehdy, pokud je $-u_i(t+1)$ záporné (kladné). Pokud tuto skutečnost uvážíme v souvislosti se vztahem (5.13), tak zjistíme, že ΔE může nabývat pouze nulových a záporných hodnot a energie sítě E tedy nemůže stoupat. Tato energie ovšem nemůže ani nekonečně klesat, a proto po určitém konečném počtu iterací musí skončit ve stabilním stavu (lokálním minimu). Existují však metody, pomocí kterých můžeme nalézt **globální minimum**. Jednou z nich je **Boltzmannova metoda** neboli **metoda simulovaného žíhání** (simulated annealing) [6]. Pomocí této metody lze řešit například **problém obchodního cestujícího**, který je popsán v kapitole 6.1.

Paralelní model

Hlavní výhodou paralelního Hopfieldova modelu oproti sekvenčnímu modelu je **paralelní aktualizace stavů sítě**, což se projevuje větší rychlostí procesu vybavení po předložení určitého vzoru. Tato paralelní aktualizace stavů se uplatňuje v průběhu **aktivační etapy** činnosti sítě. Synaptické váhy w_{ij} a prahy Θ_i jednotlivých neuronů v síti jsou na začátku učení (inicializace) určeny podle vztahů 5.14 [1] a 5.15 [1].

- V případě binárních vzorů x_i (s hodnotami 0 a 1):

$$w_{ij} = \sum_{k=1}^{m} (2x_i - 1)(2x_j - 1), \quad \Theta_i = \frac{1}{2} \sum_{j=1}^{n} w_{ij}, \tag{5.14}$$

  kde m je počet vzorů, n je počet vstupů.

- V případě bipolárních vzorů x_i (s hodnotami -1 a 1):

$$w_{ij} = \sum_{k=1}^{m} x_i x_j, \quad \Theta_i = \frac{1}{2} \sum_{j=1}^{n} w_{ij}, \tag{5.15}$$

  kde m je počet vzorů, n je počet vstupů.

Významný rozdíl oproti sekvenčnímu modelu je, že prvky na diagonále v matici vah jsou různé od nuly. Ještě si uveďme algoritmus funkce paralelního modelu Hopfieldovy sítě:

1. Nastavení parametrů sítě dle předchozích vztahů.
2. Přivedení vstupního vektoru $\boldsymbol{x}$ na vstup sítě.
3. Počáteční stavový vektor sítě je nastaven na hodnotu $\boldsymbol{x}(0) = [x_1(0), x_2(0), \ldots, x_n(0)]^T$.
4. Po předložení vstupního vektoru $\boldsymbol{x}$ je nejprve stanovena aktivita jednotlivých výkonných prvků, po níž následuje aktualizace jejich stavů: [1]
   (a) Výpočet aktivity:
   $$u_i(t+1) = \sum_{j=1}^{n} w_{ij} x_j(t) + \Theta_i, \tag{5.16}$$
   kde $j = 1, \ldots, n$.
   (b) Aktualizace stavu:
   $$x_i(t+1) = \begin{cases} 0 & \text{pro} \quad u_i(t+1) > 0 \\ 1 & \text{pro} \quad u_i(t+1) < 0 \\ x_i(t) & \text{pro} \quad u_i(t+1) = 0 \end{cases}, \tag{5.17}$$
   kde x_i je složka vektoru vzoru.

Iterativní proces se opakuje i v dalších iteracích, dokud síť po konečném počtu kroků nedospěje do **stabilního stavu**. Stabilním stavem je stejně jako u sekvenčního modelu shoda stavů dvou po sobě následujících iterací. Energetická funkce se v té chvíli nachází v lokálním minimu (atraktoru). V každém kroku dochází stejně jako u sekvenčního modelu k aktualizaci jedné složky stavového vektoru, což má za následek změnu hodnoty **energetické funkce** podle vztahu 5.12. Rovnice definující změnu energetické funkce během jedné aktualizace má však odlišný tvar [1]

$$\Delta E(t+1) = -u_i(t+1)\Delta x_i(t+1) - \frac{1}{2}\Delta \boldsymbol{x}^T(t+1)\boldsymbol{W}\Delta \boldsymbol{x}(t+1). \tag{5.18}$$

Důkaz konvergence vychází ze stejné úvahy jako u sekvenčního modelu. Je důležité si přitom uvědomit, že matice $\boldsymbol{W}$ je tvořena skalárními součiny bez nulových diagonálních prvků, jedná se tedy o **pozitivně semidefinitní matici**. Z tohoto důvodu může vztah (5.16) nabývat pouze záporných nebo nulových hodnot.

5.2.3 Spojitý Hopfieldův model

Jeho funkce byla odvozena Cohenenem a Grossbergem v roce 1983 z diskrétního autokorelátoru s Hebbovským učením. O rok později zlepšil Hopfield jeho stabilizační proceduru. Jedná se v podstatě o **rekurentní autoasociativní neuronovou síť**, která pracuje ve **spojitém čase** s **analogovými funkcemi**. Jeho struktura je jednovrstvá, zpětnovazební. Nejčastěji se používá pro zpracování obrazu, zpracování signálů (UNS Herault-Juttenova typu) nebo pro identifikaci zašuměných vstupních vzorů. [6]

6 NP - ÚPLNÉ PROBLÉMY

Velké množství úloh, které mají praktický i teoretický význam, je charakteristické svou **výpočetní náročností**. Do této skupiny spadají i tzv. **„NP - úplné problémy“** u nichž rostou nároky na výpočet se složitostí úlohy exponenciálně nebo rychleji, a pro které bylo dokázáno, že tuto náročnost nelze principiálně snížit. Často se jedná o úlohy, které mají charakter hledání nejlepšího řešení - **optimalizace**. Úspěšnost jejich řešení závisí na prohledání všech možných řešení, jejichž počet ovšem roste velmi prudce s velikostí problému. Prosté algoritmické hledání řešení u úloh tohoto typu je tedy velmi náročné na výpočetní výkon systémů a hlavně čas.

Mezi hlavní představitele těchto typů úloh patří: [5]

- **Úloha barvení mapy**. Vstupní znalostí při řešení této úlohy je fakt, že k obarvení mapy, tj. označení jednotlivých států různými barvami tak, aby žádné dva sousední státy neměly stejnou barvu, nám stačí pouze čtyři barvy. Úkolem je obarvení zadané mapy - resp. nalezení způsobu, jak ji obarvit.

- **Úloha o rozvrhování** (Job-Shop Scheduling). Máme n výrobků, a na každém z nich je potřeba provést jisté výrobní operace v určitém pořadí a s určitou dobou trvání. Ke splnění těchto úkonů je potřeba K strojů. Úkolem je zadat práci na jednotlivých strojích tak, aby čas potřebný ke zhotovení všech výrobků byl co nejkratší.

- **Úloha obchodního cestujícího** (TSP - Traveling Salesman Problem). Máme dán určitý počet měst a známe jejich vzájemné vzdálenosti. Úkolem je, aby obchodní cestující navštívil všechna města, přičemž jsou dány omezující podmínky. První z nich je, aby navštívil všechna města a nakonec se vrátil do výchozího bodu. Další podmínka říká, že každé město smí navštívit pouze jednou. A poslední, hlavní podmínkou je, aby byla vybrána nejkratší trasa ze všech možných.

Uvedené úlohy je často potřeba před jejich aplikací na praktické problémy upravit do dosti odlišné podoby. Mnohdy se aplikují intuitivní heuristická řešení, protože řešení úloh užitečné velikosti je velmi pracné a přitom **suboptimální řešení** je většinou zcela vyhovující. Jednou z možností, jak efektivně řešit „NP - úplné problémy“ je použití neuronových sítí (např. **Hopfieldovy sítě**). Nejdříve je ovšem zapotřebí zadanou úlohu převést na **optimalizační problém**, na který již lze aplikovat vhodnou neuronovou síť.

Problematice řízení síťového prvku, konkrétně problematice **optimalizace přepínání datových jednotek**, které se věnuje tato práce, je úloha obchodního cestujícího nejblíže. Následující kapitola se jí proto bude věnovat podrobněji.

Nejdříve bude popsána úloha obchodního cestujícího a následně bude tato úloha převedena na optimalizační problém řešitelný pomocí Hopfieldovy sítě.

6.1 Problém obchodního cestujícího - TSP

Jedná se o typický **optimalizační problém**, který spadá do kategorie „NP - úplných problémů“. Vstupními informacemi jsou znalost jistého konečného počtu měst N a jejich vzájemných vzdáleností, příp. jejich souřadnic na mapě. Úkolem je navštívit všechna tato města, a to nejkratší možnou cestou. Mezi **omezující kritéria** této úlohy tedy patří: [9]

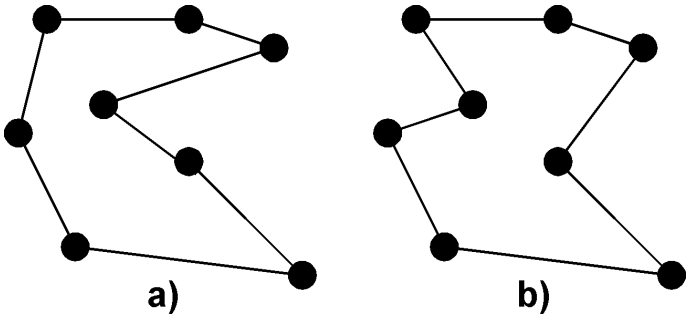
- Obchodní cestující musí navštívit všechna zadaná města.
- Každé z měst smí navštívit pouze jednou.
- V jeden časový okamžik smí být cestující pouze v jednom městě.
- Výsledná trasa musí být nejkratší.

Celkový počet možných cest mezi N městy lze vyjádřit vztahem [1]

$$\frac{N!}{2N}, \tag{6.1}$$

kde čitatel $N!$ udává celkový počet možných cest, který je ovšem výrazem $2N$ ve jmenovateli zredukován. Tato redukce je nutná z toho důvodu, že v celkovém počtu je každá cesta obsažena N-krát, podle zvolení počátečního města v uzavřené cestě a navíc 2-krát v závislosti na směru cesty.

Jednou z metod, jak problém obchodního cestujícího vyřešit, je prohledat všechny možné uzavřené cesty mezi zadanými městy a vybrat z nich tu nejkratší. Hlavní a podstatnou nevýhodou tohoto přístupu je fakt, že s rostoucím počtem měst nám velmi rychle narůstá počet všech možných cest, a tím se čas potřebný k výpočtu hrubou silou stává i u moderních výpočetních systémů zcela neúnosný již při několika málo desítkách měst. Klíčovou úlohou je tedy nalezení časově efektivního algoritmu hledání nejkratší trasy. V praktických aplikacích se úloha podobného typu obvykle řeší pouze přibližně - cílem je tedy nalézt **suboptimální řešení**. K tomu se nejčastěji využívají **heuristické algoritmy** (např. genetické algoritmy) nebo **neuronové sítě**. V roce 1985 pan Hopfield a Tank realizovali experiment úlohy TSP pro deset měst s využitím Hopfieldovy sítě. Výsledek experimentu byl fakt, že 16 případů z 20 konvergovalo ke správnému řešení. Přičemž 50% z nich patřilo k nejkratším trasám nalezených úplným řešením úlohy. [5] Úkázka optimálního vs. suboptimálního řešení je vidět na obrázku 6.1. [9]



Obr. 6.1: Problém obchodního cestujícího a) suboptimální řešení b) optimální řešení

Tímto byl popsán cíl optimalizační úlohy TSP a také omezující kritéria, která jsou nutná při řešení této úlohy vzít v úvahu. V dalším textu se budeme věnovat matematickému vyjádření tohoto problému pomocí vektorů a matic, abychom následně mohli tento problém jednodušeji transformovat na optimalizační problém přepínání v aktivním síťovém prvku s využitím Hopfieldovy neuronové sítě.

6.1.1 Vyjádření TSP pomocí binární logiky

Vstupní a výstupní veličiny spolu s parametry optimalizačního problému mohou být vyjádřeny pomocí vektorů a matic. Města, která mají být navštívena, označíme postupně celými kladnými čísly (1 až N). Například výsledek optimalizace pro 5 měst bychom mohli vyjádřit pomocí **vektoru** následovně

$$\boldsymbol{v} = \begin{pmatrix} 5 \\ 3 \\ 4 \\ 1 \\ 2 \end{pmatrix}. \tag{6.2}$$

Jednotlivé prvky vektoru **v** představují jednotlivá města. To, v jakém pořadí budou města navštívena, nám udává pořadí zápisu těchto prvků ve vektoru. Z vektoru **v** lze tedy vyčíst **optimální trasu**. Optimální trasa je v tomto případě *5.město*, *3.město*, *4.město*, *1.město* a nakonec *2.město*. Poněvadž řešení problému obchodního cestujícího je uzavřená cesta, tak nám nezáleží na tom, které město navštívíme jako první. Výsledný vektor **v** tedy můžeme zapsat například i takto $\boldsymbol{v}^T = (5, 3, 4, 1, 2)$, $\boldsymbol{v}^T = (3, 4, 1, 2, 5)$, $\boldsymbol{v}^T = (4, 1, 2, 5, 3)$ atd. Abychom mohli řešení tohoto problému vyjádřit pomocí binární logiky, kterou budeme potřebovat v další části této práce,

tak je vhodné rozvinout vektor $\boldsymbol{v}$ do **matice**

$$\boldsymbol{M} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}. \tag{6.3}$$

Díky tomuto zápisu je přehledně vidět omezující kritéria TSP. Podmínka, že cestující musí navštívit všechna zadaná města, je určena celkovým počtem jedniček v matici, kterých je v tomto případě 5. Další podmínka, že cestující může být v jeden časový okamžik pouze na jednom místě, se do matice promítla tak, že v každém sloupci je pouze jedna jednička. A skutečnost, že každý řádek obsahuje pouze jednu jedničku, je důsledkem podmínky, že každé město může být navštíveno pouze jednou.

6.1.2 Řešení TSP pomocí Hopfieldovy neuronové sítě

Využití neuronových sítí k řešení optimalizačních problémů jako je TSP urychluje výpočet tak, že doba řešení narůstá pouze **lineárně** s jejich rozměrem oproti **exponenciálnímu** nárůstu doby při použití klasických algoritmických metod. Tento fakt je hlavně důsledkem vysokého stupně propojení výkonných prvků v neuronové síti. Řešení optimalizačního problému TSP je v případě použití Hopfieldovy neuronové sítě hledáno jako **minimum energetické** (Ljapunovovy) **funkce** [2]. Výsledkem optimalizace pro 5 měst může být například matice $\boldsymbol{M}$ (6.3), která byla uvedena v předchozím textu. **Energetická funkce** Hopfieldovy sítě má tvar [5]

$$\begin{aligned} E \;=\; & \frac{A}{2}\sum_{i}\sum_{j,j\neq i} V_{X_i}V_{X_j} + \\ & + \frac{B}{2}\sum_{i}\sum_{X}\sum_{Y,Y\neq X} V_{X_i}V_{Y_i} + \\ & + \frac{C}{2}\Big(\sum_{X}\sum_{i} V_{X_i} - N\Big)^2 + \\ & + \frac{D}{2}\sum_{X}\sum_{Y,Y\neq X}\sum_{i} d_{XY}V_{X_i}(V_{Y_{i+1}} + V_{Y_{i-1}}), \end{aligned} \tag{6.4}$$

kde $V_{X_i} = g(u_{X_i})$ je výstup neuronu i pro místo X. Ve výrazu d_{XY} v rovnici (6.4) je vyjádřena vzdálenost mezi městy X a Y. Člen za vzáleností d je různý od nuly jen v případě, že se obě města vzájemně následují na trase.

Pohybové rovnice jsou [5]

$$\begin{aligned}\frac{du_{X_i}}{dt} &= -\frac{u_{X_i}}{\tau} - A\sum_{j,j\neq i} V_{X_j} - \\ &\quad -B\sum_{Y,Y\neq X} V_{Y_i} - \\ &\quad -C(\sum_{X}\sum_{j} V_{X_j} - N) - \\ &\quad -D\sum_{Y} d_{XY}(V_{Y_{i-1}} + V_{Y_{i+1}}).\end{aligned} \tag{6.5}$$

Porovnáním rovnice (6.4) se standardní definicí energie obdržíme vztah pro hodnotu váhy mezi dvěma neurony [9]

$$\begin{aligned}w_{M_i,N_j} &= -2A\cdot\delta_{MN}(1-\delta_{ij}) - \\ &\quad -2B\cdot\delta_{ij}(1-\delta_{MN}) - \\ &\quad -2C - \\ &\quad -D\cdot\delta_{MN}(\delta_{j,i+1} + \delta_{j,i-1}),\end{aligned} \tag{6.6}$$

kde $\delta_{ij} = 1$ pro $i = j$, v ostatních případech se $\delta_{ij} = 0$.

Nyní bude popsáno, co jednotlivé členy rovnice vyjadřují (6.4) [5]:

- První člen nabude hodnoty nula, pokud v každém řádku bude aktivní (excitován) pouze jediný výkonný prvek (neuron). Což nám koresponduje s omezující podmínkou TSP, která říká, že každé město smí být navštíveno pouze jedenkrát. V matici $\boldsymbol{M}$ tedy bude na místě aktivního neuronu jednička a ostatní prvky v řádku matice budou nulové.

- Podobně vymizí druhý člen, jestliže bude v každém sloupci aktivní pouze jediný neuron. Což nám koresponduje s další omezující podmínkou TSP, která říká, že v daném kroku může být navštíveno pouze jedno město. V matici $\boldsymbol{M}$ tedy bude na místě aktivního neuronu jednička a ostatní prvky v daném sloupci budou nulové.

- Také třetí člen bude nulový, bude-li počet aktivních neuronů právě roven počtu měst N. Tento fakt nám koresponduje s podmínkou TSP, že každé město musí být navštíveno právě jedenkrát. V matici $\boldsymbol{M}$ tedy musí být počet jedniček roven počtu měst N.

- Poslední čtvrtý člen vyjadřuje proces minimalizace délky trasy. Což odpovídá poslední podmínce, a to, že výsledná cesta má být co nejkratší.

Tímto byly položeny matematické základy pro řešení optimalizačního problému pomocí Hopfieldovy neuronové sítě, který bude s jistými úpravami aplikován na optimalizaci datových jednotek v aktivním síťovém prvku.

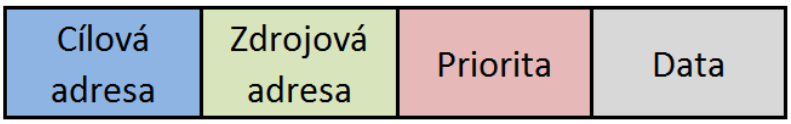
7 NÁVRH AKTIVNÍHO SÍŤOVÉHO PRVKU

Pro návrh modelu aktivního síťového prvku bylo použito integrovaného prostředí MATLAB verze 7.5 s využitím Neural Network Toolbox verze 5.1. a simulačního prostředí SIMULINK verze 7.0. Jedná se o přepínač s centrální pamětí a jednocestným spojovacím polem typu křížový přepínač. Navržený model umožňuje navolit požadovaný **počet portů přepínače**, **velikost okna vstupní vyrovnávací paměti** a **počet priorit**. Pro vysvětlení funkce modelu síťového prvku byly v této práci zvoleny následující parametry:

- Počet portů $\boldsymbol{p = 4}$.
- Velikost okna $\boldsymbol{w = 5}$.
- Rozsah priorit **1** až **8**, přičemž rámec s hodnotou 1 má **nejvyšší prioritu**.

7.1 Struktura datové jednotky - rámce

Pro potřeby simulace modelu aktivního síťového prvku byl vytvořen **rámec**, jehož struktura je vidět na brázku 7.1.



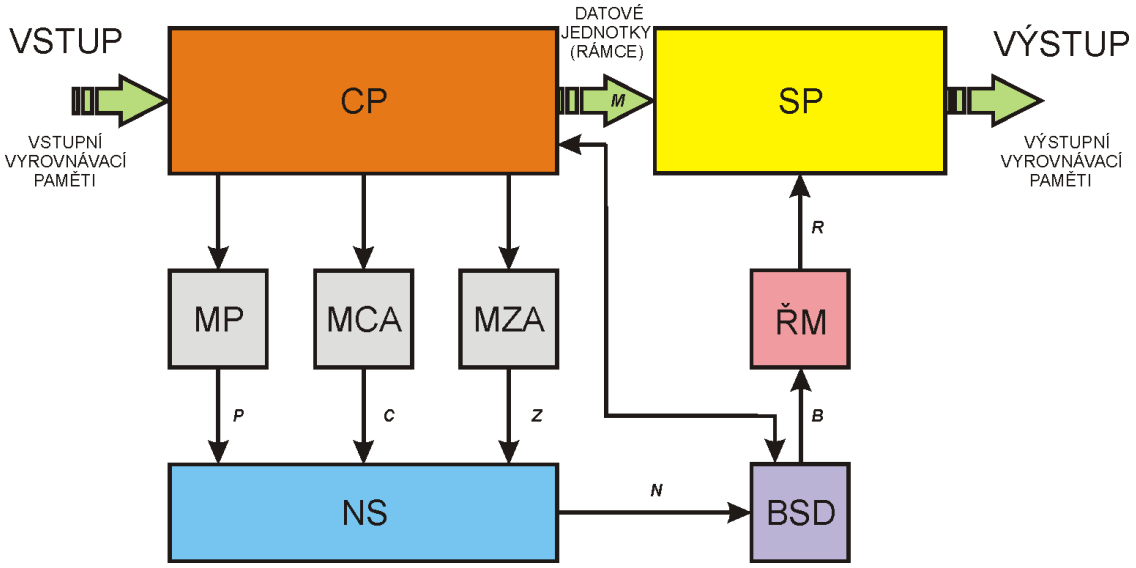
Obr. 7.1: Struktura datové jednotky - rámce

Velikost jednotlivých částí rámce závisí na počátečním nastavení modelu při spuštění simulace. Pouze velikost datové části byla omezena na maximální hodnotu 1500B, což lze ovšem změnit ve zdrojovém kódu v souboru `model_sitoveho_prvku.m` editací konstanty `max_vel_dat` .

7.2 Struktura síťového prvku

7.2.1 Celkové blokové schéma síťového prvku

Celkové blokové schéma navrženého modelu síťového prvku spolu se stručným popisem jednotlivých bloků je na obrázku 7.2 na následující stránce.



Obr. 7.2: Blokové schéma prvku

Stručný popis funkce a významu jednotlivých bloků:

- **CP (Centrální paměť)** - zde jsou uloženy rámce, které čekají na odeslání do spojovacího pole. Jejich celkový počet n závisí na nastavené velikosti okna w a počtu vstupních portů p, přičemž velikost okna udává maximální počet rámců, které jsou načteny ze vstupních vyrovnávacích pamětí jednotlivých vstupních portů a poté uloženy do centrální paměti. Pokud bychom tedy uvažovali prvek se 4 porty a nastavení velikosti okna na hodnotu 5, tak by byl celkový maximální počet uložených rámců v centrální paměti $n = p * w = 4 * 5 = \mathbf{20}$. Ze všech těchto rámců jsou v každém cyklu načteny informace (priorita, cílová adresa, zdrojová adresa), které jsou odeslány do neuronové sítě (NS) ke zpracování. Centrální paměť je řízena blokem selekce datových jednotek (BSD), který určuje, které rámce budou odeslány z centrální paměti přes spojovací pole na výstup síťového prvku. Po odeslání vybraných rámců z centrální paměti dojde k defragmentaci volného paměťového místa a načtení dalších rámců ze vstupních vyrovnávacích pamětí. Rámce jsou v centrální paměti uloženy do paměťových míst, které jsou indexovány čísly v rozsahu 11 až 45. Při popisu simulačního programu v oddílu 7.3.2 odpovídá obsah centrální paměti matici $\boldsymbol{M}$ s prvky m_{11} až m_{45}.
- **MP (Matice priorit)** - obsahuje priority rámců uložených v centrální paměti. Její rozměr je $p \times w$. Při popisu simulačního programu v oddílu 7.3.2 je tato matice označena písmenem $\boldsymbol{P}$ s prvky p_{11} až p_{45}. Priorita uložená na pozici p_{11} patří rámci uloženému na pozici m_{11} v centrální paměti. Obsah matice priorit je jedním ze vstupů do neuronové sítě.

- **MCA (Matice cílových adres)** - obsahuje cílové adresy rámců uložených v centrální paměti. Její rozměr je $p \times w$. Při popisu simulačního programu v oddílu 7.3.2 je tato matice označena písmenem $\boldsymbol{C}$ s prvky c_{11} až c_{45}. Priorita uložená na pozici c_{11} patří rámci uloženému na pozici m_{11} v centrální paměti. Obsah matice cílových adres je dalším ze vstupů do neuronové sítě.

- **MZA (Matice zdrojových adres)** - obsahuje zdrojové adresy rámců uložených v centrální paměti. Její rozměr je $p \times w$. Při popisu simulačního programu v oddílu 7.3.2 je tato matice označena písmenem $\boldsymbol{Z}$ s prvky z_{11} až z_{45}. Priorita uložená na pozici z_{11} patří rámci uloženému na pozici m_{11} v centrální paměti. Obsah matice zdrojových adres je posledním ze vstupů do neuronové sítě.

- **NS (Neuronová síť)** - zpracovává informace zaslané z bloků MP, MCA a MZA. Výsledek zpracování je uložen do matice $\boldsymbol{N}$, která je rozměru $p \times w$. Hodnoty uložené v této matici jsou v rozsahu -1 až $+1$, což je dáno použitou přenosovou funkcí **satlins** (symmetric saturating linear transfer function) v neuronové síti. Velikost těchto hodnot odpovídá ohodnocení jednotlivých rámců v centrální paměti z hlediska optimalizace přepínání. Hodnota uložená na pozici n_{11} v matici $\boldsymbol{N}$ odpovídá pozici rámce m_{11} uloženého v centrální paměti. Matice $\boldsymbol{N}$ je zaslána do bloku BSD k dalšímu zpracování. Blíže je o neuronové sítě pojednáno v kapitole 7.2.3.

- **BSD (Blok selekce datových jednotek)** - provádí konečný výběr datových jednotek na základě hodnot zaslaných z neuronové sítě v podobě matice $\boldsymbol{N}$. Výsledkem zpracování jsou jednak řídicí informace pro centrální paměť (uložené v matici $\boldsymbol{B}$) obsahující indexy rámců, které mají být odeslány do spojovacího pole. A dále jsou zaslány informace do bloku ŘM, na jejichž základě se tam sestaví řídicí matice. Rozměr matice $\boldsymbol{B}$ je $p \times w$ a hodnota uložená na pozici b_{11} v této matici odpovídá rámci uloženému na pozici m_{11} v centrální paměti.

- **ŘM (Řídicí matice)** - tento blok na základě informací z bloku BSD sestaví řídicí matici $\boldsymbol{R}$ rozměru $p \times p$ obsahující pouze binární hodnoty, která slouží k ovládání spínačů ve spojovacím poli (SP). Hodnota uložená na pozici r_{11} v této matici odpovídá stejné pozici spínače ve spojovacím poli. Princip řídicí matice je podrobně vysvětlen v oddílu 7.2.2.

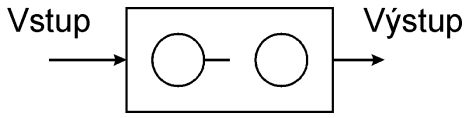
- **SP (Spojovací Pole)** - zajišťuje přepnutí rámců ze vstupních na výstupní porty přepínače. Jedná se o jednocestné spojovací pole typu křížový přepínač. Rozměr spojovacího pole je $p \times p$ a zavisí tedy na počtu portů p přepínače. V případě zde popisovaného přepínače se 4 porty je tedy rozměr spínacího

pole 4×4 a obsahuje 16 spínacích bodů. Pozici spínacích bodů, které mají být sepnuty, udává řídicí matice $\boldsymbol{R}$ z bloku ŘM.

Popisem jednotlivých částí síťového prvku byla obecně popsána i funkce celého modelu síťového prvku. V následujících kapitolách bude popsána funkce dílčích bloků podrobněji.

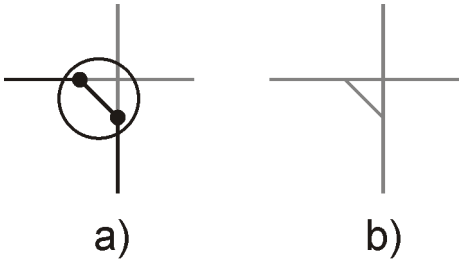
7.2.2 Spojovací pole

Jedná se o **jednocestné spojovací pole typu křížový přepínač**, u kterého může být libovolný vstupní port propojen s libovolným výstupním portem. Přepínač s p porty vyžaduje u tohoto typu spojovacího pole p^2 samostatně řízených spínacích bodů, z nichž každý může v jeden časový okamžik propojit jednu dvojici vstupní port - výstupní port. Protože pro popis funkce navrženého modelu síťového prvku byly zvoleny 4 porty, tak použité spojovací pole obsahuje celkem 16 spínacích bodů. Následující obrázek zobrazuje schematickou značku spínacího bodu.



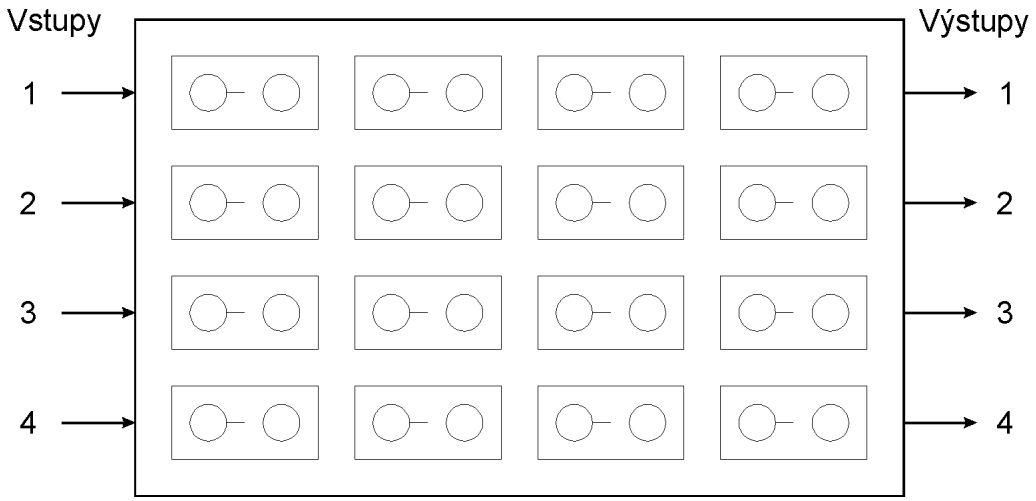
Obr. 7.3: Spínací bod - spínač

Každý spínací bod se může nacházet v jednom ze dvou stavů (**rozepnutý**/ **sepnutý**), jak je vidět z obrázku 7.4, přičemž ve výchozím stavu jsou všechny spínače ve spojovacím poli rozepnuté.



Obr. 7.4: Spínací bod a) v sepnutém stavu b) v rozepnutém stavu

Aby bylo možné přenést datovou jednotku přes spojovací pole, bylo zavedeno adresování jednotlivých vstupů a výstupů spojovacího pole. V případě přepínače se čtyřmi porty má použité spojovací pole celkem 4 adresy pro vstupy do spojovacího pole a 4 adresy pro výstupy. Způsob adresování je zřejmý z obrázku 7.5 na další stránce.



Obr. 7.5: Spojovací pole 4x4 s adresami vstupů a výstupů

Spojení mezi vstupním portem i a výstupním portem j se realizuje uvedením spínacího prvku (i, j) do sepnutého stavu. Uvedení konkrétních spínacích bodů do sepnutého stavu má na starosti blok ŘM, který řídí spínací pole. Nyní bude popsán princip řízení spojovacího pole podrobněji.

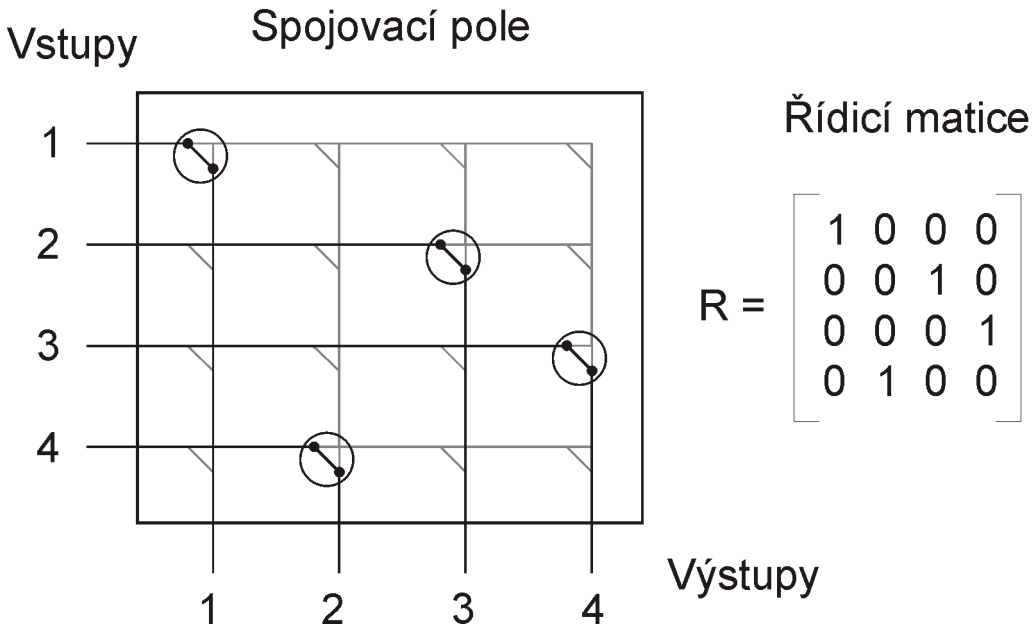
Princip řízení spojovacího pole

Řízení spínacích bodů ve spojovacím poli je realizováno pomocí binární řídicí matice $\boldsymbol{R}$, která má **stejný rozměr** jako spojovací pole a každý prvek matice r_{ij} řídí stav jednoho spínacího bodu. V případě 16 spínacích bodů je rozměr řídicí matice 4×4.

$$\boldsymbol{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{pmatrix}$$

Řádky matice odpovídají vstupům a sloupce výstupům spojovacího pole. Z toho je zřejmé, že spínací bod, který přepíná první vstup spojovacího pole (adresa 1) s prvním výstupem (adresa 1), odpovídá prvku matice r_{11}. Podobně spínací bod, který přepíná druhý vstup spojovacího pole (adresa 2) se třetím výstupem (adresa 3), odpovídá prvku matice r_{23}, atd. To, zda bude konkrétní spínací bod ve stavu sepnuto nebo rozepnuto, je dáno hodnotou příslušného prvku řídicí matice $\boldsymbol{R}$. Protože se jedná o **binární matici**, tak může nabývat pouze dvou hodnot, a to buď *log 1* nebo *log 0*. Hodnota prvku řídicí matice $\boldsymbol{R}$ *log 1* znamená, že dojde k sepnutí odpovídajícího spínacího prvku ve spojovacím poli a přepnutí rámce z příslušného vstupního portu na daný port výstupní. V případě, že daný prvek matice nabývá

hodnoty *log 0*, tak zůstane spínací bod v rozepnutém stavu a k žádnému přepnutí rámce v tomto bodě nedojde. Vztah mezi řídicí maticí a spojovacím polem je patrný z obrázku 7.6. V podstatě se jedná o stejné spojovací pole jako je na obrázku 7.5, pouze jsou v nákresu pro názornější představu přesunuty výstupy z pravé strany na stranu spodní.



Obr. 7.6: Spojovací pole - Řídicí matice **R**

Vodorovné linky odpovídají vstupům a svislé linky výstupům spojovacího pole. Z obrázku 7.6 je zřejmé, že prvek r_{11} obsahující hodnotu *log 1*, způsobí sepnutí odpovídajícího spínacího bodu ve spojovacím poli a rámec bude přenesen z prvního vstupu na první výstup. Také prvek matice r_{23} s hodnotou *log 1* způsobí, že dojde k přepnutí rámce z druhého vstupu na třetí výstup. Obdobně hodnoty prvků matice r_{34} a r_{42} způsobí přenos rámců z příslušných vstupních portů na porty výstupní.

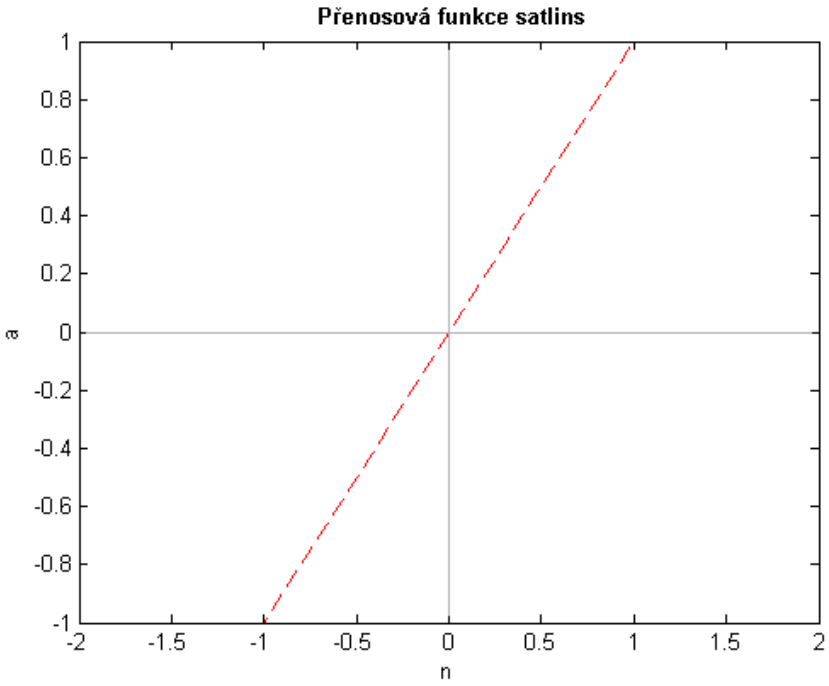
V ideálním případě jsou v každém cyklu přeneseny rámce ze všech vstupů na všechny výstupy spojovacího pole, přičemž v jednom okamžiku může být přenesen pouze jeden rámec z jednoho vstupu na jeden výstup. O efektivní využití spojovacího pole se stará neuronová síť, která provádí optimalizaci prioritního přepínání. V případě, že je v centrální paměti uloženo více rámců, které směřují na společný výstup (mají stejnou cílovou adresu), tak je na tento výstup přepnut ten rámec, který má vyšší prioritu. Pokud ovšem nastane situace, že tyto rámce mají stejnou prioritu, tak je přepnut ten z nich, který přišel do síťového prvku dříve. V tomto případě je tedy uplatňován **princip FIFO fronty**. Podrobně jsou tyto způsoby rozhodování popsány v oddílu 7.3.2.

7.2.3 Neuronová síť přepínače

Blok obsahující neuronovou síť je nejdůležitější částí navrženého modelu síťového prvku. Konkrétně se jedná o model Hopfieldovy sítě.

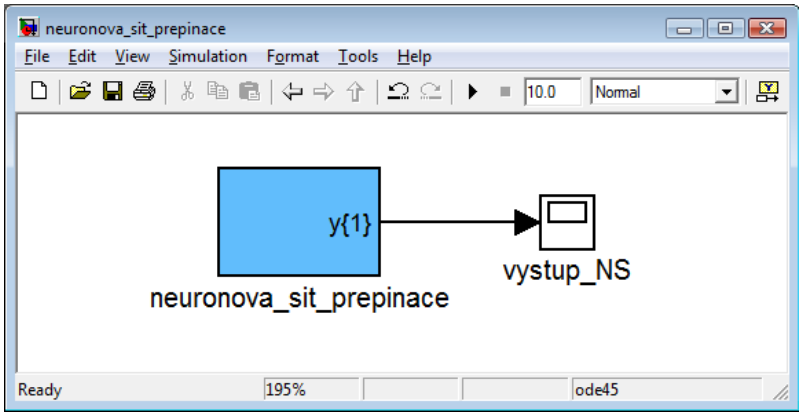
Z bloků MP, MCA a MZA jsou do ní zasílány informace (priority, cílové a zdrojové adresy rámců), které jsou v bloku s neuronovou sítí před jejich odesláním na její vstup nejdříve normovány na **incializační hodnoty** v rozsahu -1 až $+1$. V Hopfieldově neuronové síti jsou aplikovány tyto inicializační hodnoty na všechny neurony sítě. Poté následuje cyklus postupných změn excitací neuronů až do okamžiku dosažení **stabilního stavu - atraktoru**, který reprezentuje některé z lokálních minim energetické funkce neuronové sítě. Tento stabilní stav koresponduje s maximálním využitím přenosové kapacity spojovacího pole a přitom zohledňuje hodnoty priorit jednotlivých rámců (prioritní zpracování). Neuronová síť je tedy použita pro **optimalizaci přepínání** ve spojovacím poli přepínače tak, aby bylo toto pole efektivně využito a současně provádí **prioritní zpracování rámců**. Výstupem z neuronové sítě je matice výstupních hodnot $\boldsymbol{N}$, která představuje ohodnocení jednotlivých rámců hodnotami z intervalu -1 až $+1$ (což je dáno použitou přenosovou funkcí satlins, obr. 7.7). Pozice prvků v matici $\boldsymbol{N}$ odpovídá pozici rámců uložených v matici $\boldsymbol{M}$, jejíž obsah představuje centrální paměť. Rozměr matice $\boldsymbol{N}$ je tedy stejný jako rozměr spojovacího pole $(p \times p)$ a hodnota uložená na pozici n_{11} v matici $\boldsymbol{N}$ koresponduje s rámcem, který je uložený na pozici m_{11} v centrální paměti. Velikost jednotlivých ohodnocení vyjadřuje nutnost odeslání daného rámce do spojovacího pole v dané periodě tak, aby byla maximálně využita přenosová kapacita spojovacího pole při současném zohlednění priorit jednotlivých rámců. Příklad výstupu z neuronové sítě je vidět ve výpisu simulačního programu v oddílu 7.3.2.

Na následujících stránkách je zobrazena celá struktura neuronové sítě přepínače (obrázky 7.8 až 7.11) navržená v simulačním prostředí **Simulink** s využitím **Neural Network Toolboxu**. Při návrhu sítě byla použita váhová funkce **dotprod**, síťová funkce **netsum** a symetrická saturovaná lineární přenosová funkce **satlins** (obr. 7.7). Konkrétně na obrázku 7.11 je část neuronové sítě přepínače, která umožňuje nastavit váhy a prahy u jednotlivých neuronů podle potřeby. Protože byly u síťového modelu přepínače nastaveny 4 porty a velikost okna vstupních vyrovnávacích pamětí na hodnotu 5, tak je počet neuronů v neuronové sítí roven 20. Počet neuronů v neuronové síti, která byla použita při návrhu modelu síťového prvku, je tedy přímo úměrný počtu rámců, u nichž chceme provádět prioritní zpracování během každé rozhodovací periody.

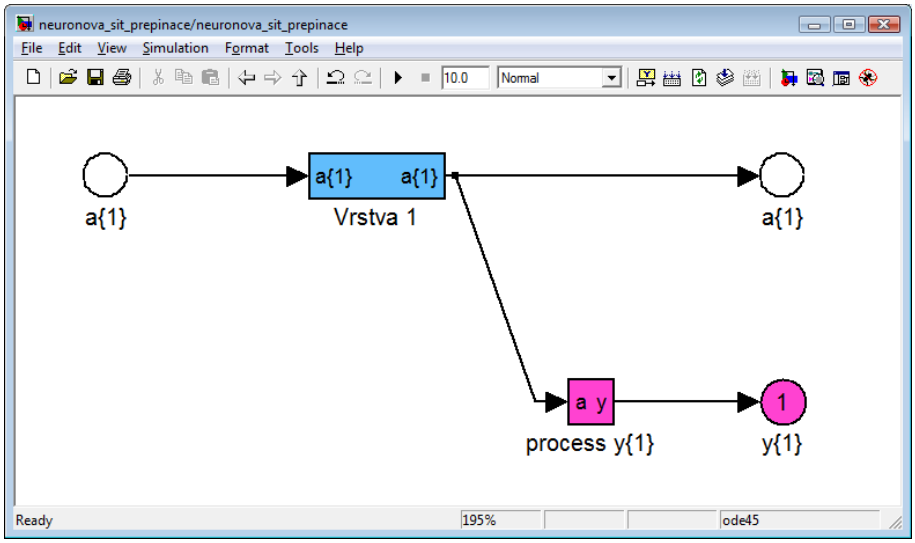


Obr. 7.7: Přenosová funkce satlins: $a = satlins(n)$

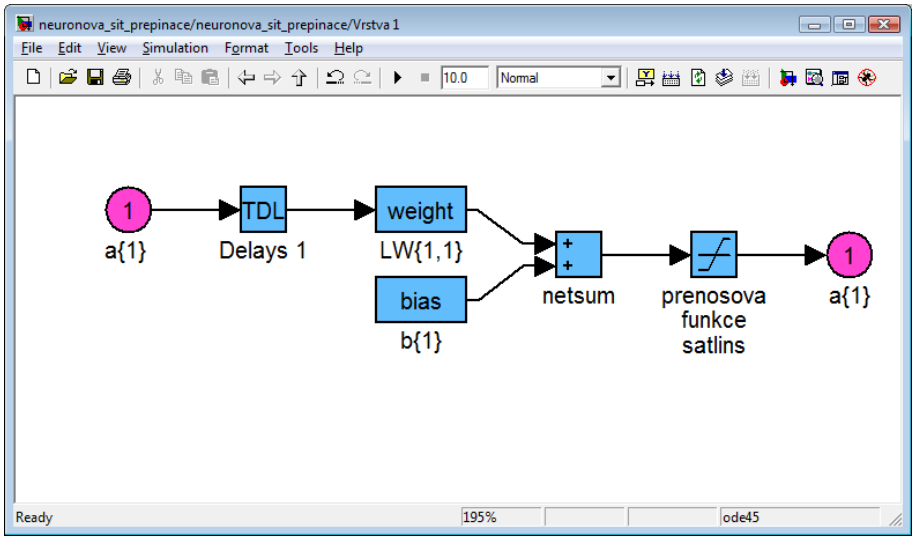
Hodnotám vstupních veličin, které jsou menší než -1, přiřadí funkce satlins výstupní hodnotu -1. Naopak pro vstupní hodnoty větší než $+1$ přiřadí výstupní hodnotu $+1$. Vstupním hodnotám, které se nacházají v rozmezí hodnot -1 až $+1$, přiřadí funkce satlins stejnou hodnotu jako byla na vstupu.



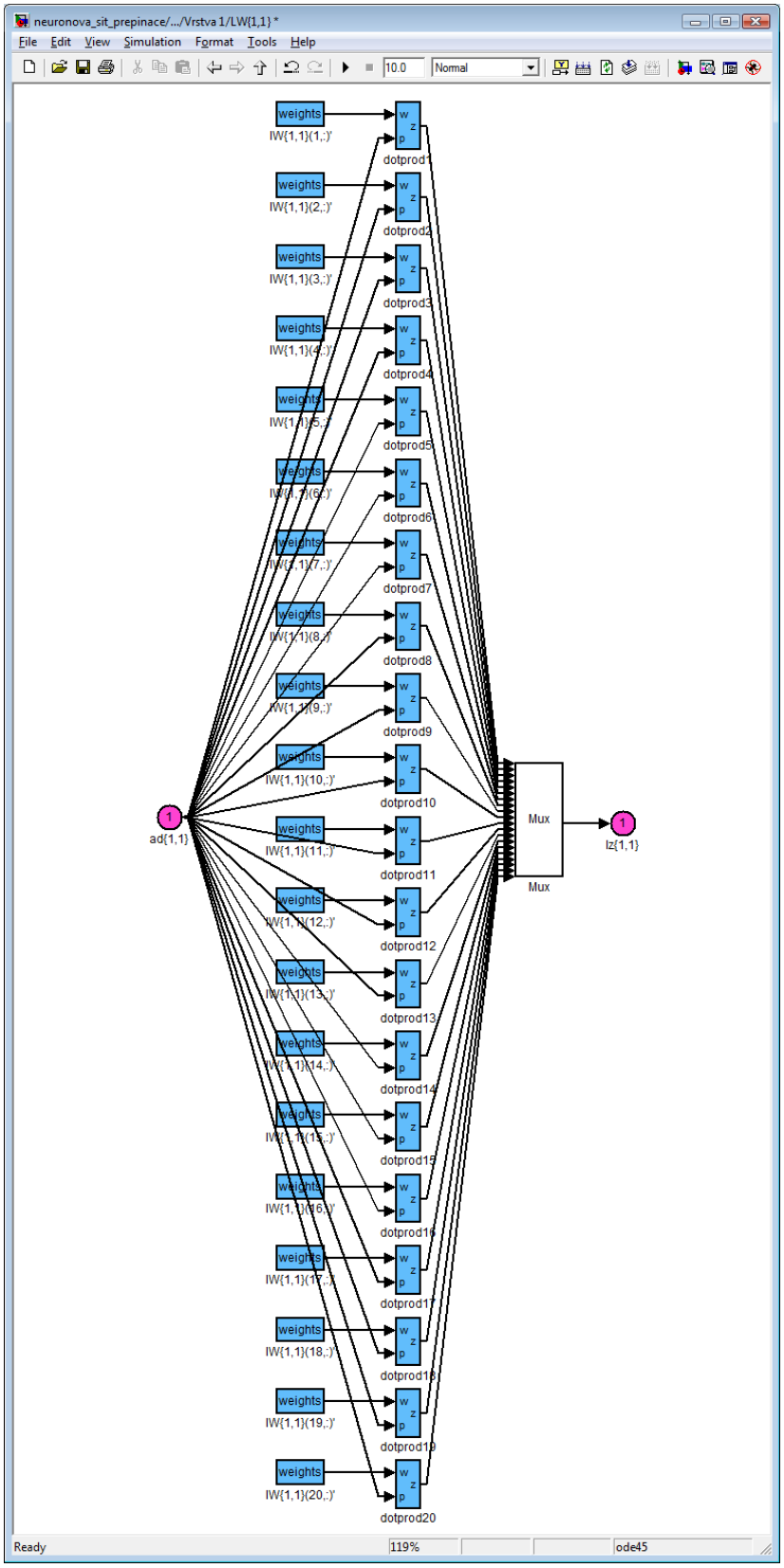
Obr. 7.8: Blok neuronové sítě přepínače v simulačním prostředí Simulink



Obr. 7.9: Vnitřní struktura neuronové sítě přepínače



Obr. 7.10: Struktura konkrétní vrstvy NS



Obr. 7.11: Část neuronové sítě přepínače, která umožňuje nastavit váhy a prahy u jednotlivých neuronů

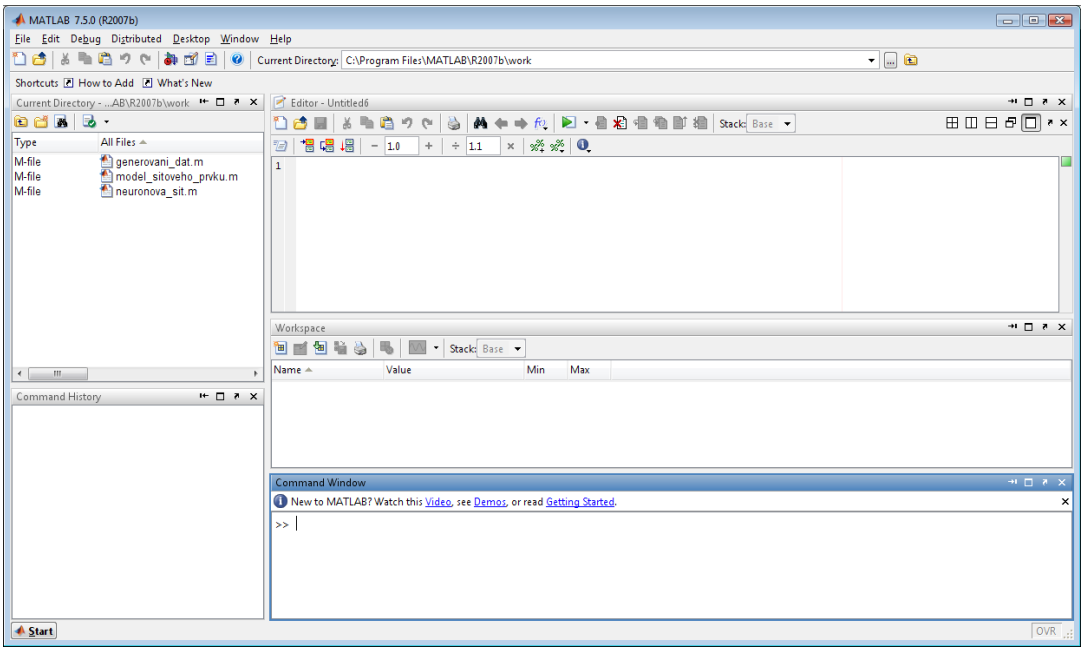
7.3 Popis simulačního programu

Program byl vytvořen v integrovaném prostředí MATLAB verze 7.5, které je nutné pro spuštění simulace navrženého modelu síťového prvku - přepínače. Nejdříve si proto prostředí, kde se bude vlastní simulace odehrávat, stručně popíšeme.

7.3.1 Prostředí simulace

MATLAB (MATrix LABoratory) je výkonné integrované interaktivní prostředí pro vědecké výpočty. Spojuje technické výpočty, vizualizaci dat, simulační nástroje a programovací jazyk v jednom prostředí. Společně s množstvím dostupných modulů tak vytváří ideální prostředí pro inženýry, vědce, matematiky a učitele při řešení problémů z mnoha oblastí. Při návrhu síťového prvku bylo využito rozšiřujícího modulu Neural Network Toolbox verze 5.1. a simulačního prostředí SIMULINK verze 7.0.

Současná verze Matlabu je k dispozici pro několik platforem a operačních systémů, jako jsou Windows, operační systémy založené na linuxovém jádře, unixové systémy nebo MacOS X. Pro procesory x86 je dostupná 32bitová i 64bitová verze. V operačním systému Windows můžeme spustit MATLAB z nabídky *Start menu* postupným rozbalením položek *Start* - *MATLAB*. V případě operačních systémů založených na linuxovém jádře spustíme Matlab z příkazové řádky příkazem `matlab &` . Za chvíli se objeví podobné prostředí Matlabu, jaké je vidět na obrázku 7.12.



Obr. 7.12: Prostředí Matlabu

Z obrázku 7.12 je vidět, že pracovní prostředí Matlabu se skládá z několika částí:

- **Current Directory** - z tohoto adresáře se přednostně spouští skripty a funkce. Standardně zde bývá nastavena cesta k adresáři *Work* v adresáři *Matlab*. Do tohoto adresáře nakopírujeme soubory, které jsou nezbytné pro vlastní simulaci síťového prvku. Jedná se o soubory:
  `generovani_dat.m`, `model_sitoveho_prvku.m` a `neuronova_sit.m`.

- **Command Window** - jedná se o příkazové okno, které slouží ke komunikaci s výpočetním jádrem Matlabu. V tomto okně budeme zadávat parametry síťového modelu a také se nám zde bude postupně vypisovat průběh samotné simulace.

- **Workspace** - v tomto pracovním okně můžeme sledovat, případně editovat, všechny proměnné použité během simulace.

- **Command History** - zobrazuje historii všech příkazů, které byly napsány do příkazového okna (*Command Window*).

- **Editor** - slouží k editaci zdrojového kódu. Zde můžeme zobrazit, případně editovat, všechny zdrojové soubory, které jsou součástí modelu síťového prvku. Před vlastním popisem programu bude vhodné, když si zde zobrazíme všechny tři zdrojové soubory síťového prvku (tedy soubory `generovani_dat.m`, `model_sitoveho_prvku.m` a `neuronova_sit.m`).

Tímto bylo stručně popsáno simulační prostředí Matlab a nutné kroky k tomu, abychom mohli popsat vlastní funkci a průběh simulace navrženého modelu síťového prvku - přepínače.

7.3.2 Analýza funkce programu

Nejdříve je potřeba definovat označení a rozměry matic, které jsou použity při vysvětlování funkce programu. Tento krok je učiněn pro zvýšení srozumitelnosti. Při popisu simulace obsahuje matice označená písmenem:

- $\boldsymbol{M}$ - obsah centrální paměti přepínače (blok CP). Její rozměr je $p \times w$. Prvky m_{11} až m_{pw} představují paměťová místa v centrální paměti, kde jsou uloženy rámce.

- $\boldsymbol{P}$ - priority rámců zasílané z bloku MP do neuronové sítě. Její rozměr je $p \times w$. Prvky p_{11} až p_{pw} představují paměťová místa v bloku MP, kde jsou uloženy priority rámců.

- $\boldsymbol{C}$ - cílové adresy rámců zasílané z bloku MCA do neuronové sítě. Její rozměr je $p \times w$. Prvky c_{11} až c_{pw} představují paměťová místa v bloku MCA, kde jsou uloženy cílové adresy rámců.

- $\boldsymbol{Z}$ - zdrojové adresy rámců zasílané z bloku MZA do neuronové sítě. Její rozměr je $p \times w$. Prvky z_{11} až z_{pw} představují paměťová místa v bloku MZA, kde jsou uloženy zdrojové adresy rámců.

- $\boldsymbol{N}$ - výstup z neuronové sítě. Její rozměr je $p \times w$. Hodnoty prvků v této matici na pozicích n_{ij} korelují s rámci uloženými v matici M (centrální paměti) na pozicích m_{ij}.

- $\boldsymbol{B}$ - pozice rámců vybraných blokem BSD, které se nacházejí v centrální paměti a jsou určeny k odeslání do spojovacího pole. Její rozměr je $p \times w$. Prvky na pozicích b_{ij} korespondují s paměťovými místy (rámci) v matici $\boldsymbol{M}$ (centrální paměti) na pozicích m_{ij}.

- $\boldsymbol{R}$ - binární hodnoty, na základě kterých se ovládají stavy spínačů (rozepnutý / sepnutý) ve spojovacím poli. Její rozměr je $p \times p$. Hodnoty prvků na pozicích r_{ij} odpovídají stejným pozicím spínačů ve spojovacím poli.

Pokud byly učiněny všechny kroky popsané v oddílu 7.3.1, tak ke spuštění simulace stačí napsat do okna *Command Window* název hlavního souboru:
`model_sitoveho_prvku` (bez přípony m) a potvrdit klávesou *Enter*. Tím se spustí vlastní simulace, která bude probíhat v následujících krocích.

V prvním kroku je na výzvu programu nutno zadat parametry síťového prvku. Pro vysvětlení funkce síťového prvku byly zvoleny tyto parametry:

- Počet portů: **4**.

- Velikost okna: **5**.

- Rozsah priorit: **1** až **8** (rámec s hodnotou priority 1 má nejvyšší prioritu).

Následně se na základě zadaných parametrů vytvořil model síťového prvku, který měl 4 porty a centrální paměť měla kapacitu pro uložení 20 rámců. Dále se ze vstupních vyrovnávacích pamětí jednotlivých portů načetly rámce do centrální paměti (matice $\boldsymbol{M}$). Její obsah je vidět na výstupu z programu, který je umístěn na následující stránce.

Výpis programu 7.1: Obsah centrální paměti (matice **M**)

```
Obsah centrální paměti (CP):
```

Cílová adresa	Zdrojová adresa	Priorita	Datová část rámce [B]
1	1	3	1325
3	2	1	1431
2	3	5	333
2	4	1	650
1	1	1	1219
3	2	1	314
1	3	7	730
3	4	3	1121
3	1	7	569
3	2	4	264
2	3	2	1024
1	4	1	714
2	1	1	135
1	2	5	970
4	3	5	562
0	0	8	0
2	1	5	680
0	0	8	0
0	0	8	0
0	0	8	0

Každý řádek této tabulky odpovídá jednomu rámci. Je zde vidět cílová adresa, zdrojová adresa, priorita a datová část všech rámců. Pro zvýšení přehlednosti jsou všechny hodnoty v dekadické soustavě. Jednotlivé části rámců jsou náhodná celá čísla s **rovnoměrným rozložením**, přičemž cílové a zdrojové adresy byly generovány z rozsahu 1-4, priority z rozahu 1-8 a velikost datových částí rámců mohla být maximálně 1500B. Řádky s nulovou cílovou adresou představují "**virtuální rámce**" (prázdné paměťové místo v centrální paměti) a jsou jim přiřazeny nejnižší priority.

Zdrojový kód 7.2: Algoritmus generování rámců

```
for i=1:n_portu
    pocet_gen_jednotek = round(free_pamet(i)*rand(1));
    % generování nových rámců
    new_mat_cil_adr = ceil(n_portu*rand([1 pocet_gen_jednotek]));
    new_mat_zdroj_adr = repmat(i,[1 pocet_gen_jednotek]);
    new_mat_prior = ceil(low_prior*rand([1 pocet_gen_jednotek]));
    new_mat_dat = ceil(max_vel_dat*rand([1 pocet_gen_jednotek]));
```

Po uložení rámců do centrální paměti se vypsala její obsazenost. Hodnota 1 představuje obsazenou a 0 volnou paměťovou pozici v centrální paměti.

Výpis programu 7.3: Obsazenost centrální paměti

```
 Matice znázorňující obsazené paměťové pozice v centrální paměti:
```

```
     1     1     1     1     1
     0     1     1     1     1
     0     1     1     1     1
     0     0     1     1     1
```

Následně se z jednotlivých polí rámců načetly potřebné údaje do bloků MP (matice $\boldsymbol{P}$), MCA (matice $\boldsymbol{C}$) a MZA (matice $\boldsymbol{Z}$), které byly použity jako vstup pro neuronovou síť. Následující výpis znázorňuje ve společné tabulce všechny údaje zaslané z bloků MP, MCA a MZA do neuronové sítě.

Výpis programu 7.4: Obsah matic **P**, **C** a **Z** ve společné tabulce

```
      2     2     3     1     1
      5     1     7     1     3
1.ř. ──────────────────────────
      0     1     3     3     3
      8     5     4     1     1
2.ř. ──────────────────────────
      0     4     2     1     2
      8     5     2     7     5
3.ř.──────────────────────────
      0     0     1     3     2
      8     8     1     3     1
4.ř.──────────────────────────
```

V prvním řádku jsou údaje rámců z prvního vstupního portu, ve druhém řádku z druhého vstupního portu atd, přičemž spodní čísla na jednotlivých řádcích jsou hodnoty priorit rámců a horní představují cílové adresy rámců. Z této tabulky lze vyčíst i pořadí, ve kterém rámce přišly do přepínače. Rámce, které přišly na příslušný vstupní port přepínače jako první, jsou vždy uloženy na první pozici v každém řádku úplně vpravo a vlevo od nich se postupně nacházejí údaje rámců, které přišly do přepínače až jako další v pořadí.

Hodnoty uložené v blocích MP, MCA a MZA (priority rámců, cílové a zdrojové adresy) byly odeslány do bloku s neuronovou sítí. Zde byly tyto hodnoty nejdříve normovány na rozsah inicializačních hodnot -1 až $+1$ a předány na vstup Hopfieldovy neuronové sítě. Její funkce již byla popsána v oddílu 7.2.3 a její struktura je

vidět na obrázcích 7.8 až 7.11. Výstupem NS je matice výstupních hodnot $\boldsymbol{N}$, která obsahuje ohodnocení jednotlivých rámců hodnotami z intervalu -1 až $+1$ (což je dáno použitou přenosovou funkcí **satlins**, obr. 7.7). Tato matice byla zaslána do bloku BSD k dalšímu zpracování. Pozice prvků v matici $\boldsymbol{N}$ odpovídají pozicím rámců uložených v matici $\boldsymbol{M}$ (centrální paměti). Velikost jednotlivých ohodnocení vyjadřuje nutnost odeslání daného rámce do spojovacího pole v dané periodě tak, aby byla maximálně využita přenosová kapacita spojovacího pole při současném zohlednění priorit jednotlivých rámců. Forma výstupu z neuronové sítě je vidět na výpisu z programu.

Výpis programu 7.5: Výstup z neuronové sítě (matice **N**)

```
Matice výstupních hodnot z neuronové sítě:
```

−0.0222	0.1652	−0.0238	0.0731	−0.0207
−0.3749	−0.0866	0.1446	0.2851	0.2851
−0.3749	0.2176	0.1739	−0.1526	0.0333
−0.3749	−0.3749	0.1563	0.2469	0.2485

Následně byly v bloku BSD hledány v jednotlivých řádcích matice $\boldsymbol{N}$ **maximální hodnoty**. Pořadí řádků, ve kterých se maximální hodnoty hledaly, bylo určeno pomocí funkce `randperm`, která vygenerovala náhodnou permutaci z množiny vstupních portů {1, 2, 3, 4}. Náhodnost je v programu zajištěna použitím funkce `rand`, která generuje čísla s náhodným rozložením. Díky této strategii výběru je zajištěno **zrovnoprávnění** vstupních portů. Ve zde uváděném příkladu simulace bylo vygenerováno pořadí řádku **3**., **2**., **4**., **1**. Algoritmus výběru:

1. Ve 3. řádku matice $\boldsymbol{N}$ byla nalezena maximální hodnota u prvku $n_{32} = 0,2176$. Tomuto prvku odpovídal rámec (*cílová adresa* = 4, *zdrojová adresa* = 3, *priorita* = 5 a *velikost datové části* = 562B), který byl uložený v matici $\boldsymbol{M}$ (centrální paměti) na pozici m_{32}. Tento rámec byl tedy vybrán k odeslání do spojovacího pole (jeho pozice v centrální paměti byla uložena do matice $\boldsymbol{B}$ v bloku BSD). Protože tento rámec měl vstoupit do spojovacího pole přes port 3 a měl být přepnut na výstupní port 4, tak již žádný rámec v dané přepínací periodě nemohl této cesty využít. Toto omezení je způsobeno použitím jednocestného spojovacího pole u navrženého modelu síťového prvku, které pracuje na principu křížového přepínače (obr. 7.6). Z tohoto důvodu byly ve výstupní matici neuronové sítě $\boldsymbol{N}$ vymazány (byla zde uložena hodnota NaN) všechny prvky, jejichž indexy odpovídaly indexům rámců v matici $\boldsymbol{M}$ (centrální paměti), které měly stejnou cílovou adresou (tj. 4) nebo zdrojovou adresou (tj. 3) jako vybraný

rámec. Jednalo se tedy o prvky $n_{31} = -0,3749$, $n_{33} = 0,1739$, $n_{34} = -0,1526$ a $n_{35} = 0,0333$. Dále si můžeme všimnout prvků n_{21}, n_{31}, n_{41}, n_{42}, které měly shodnou velikost, a to $-0,3749$. Na stejných pozicích v matici $\boldsymbol{M}$ se nacházely „virtuální rámce“ (prázdná paměťová místa), které již byly zmíněny dříve. Jak vidíme, byly neuronovou sítí ohodnoceny nejmenšími čísly, a to z toho důvodu, aby nesnižovaly efektivnost spojovacího pole (v případě přepnutí neexistující jednotky) a tím i celého přepínače. Výstupní matice $\boldsymbol{N}$ z neuronové sítě vypadala po prvním rozhodovacím kroku v bloku BSD takto:

Výpis programu 7.6: Obsah matice **N** po 1. kroku výběru

```
Matice výstupních hodnot z neuronové sítě (po 1. kroku výběru):

  −0.0222    0.1652   −0.0238    0.0731   −0.0207
  −0.3749   −0.0866    0.1446    0.2851    0.2851
   0         0         0         0         0
  −0.3749   −0.3749    0.1563    0.2469    0.2485
```

2. Ve 2. řádku matice $\boldsymbol{N}$ byla nalezena maximální hodnota u dvou prvků $n_{24} = 0,2851$ a $n_{25} = 0,2851$. Těmto prvkům odpovídaly rámce (*cílová adresa* $= 3$, *zdrojová adresa* $= 2$, *priorita* $= 1$ a *velikost datové části* $= 314$B u rámce m_{24} a 1431B u rámce m_{25}). Oba rámce tedy měly stejné priority a směřovaly ze společného vstupu na stejný výstup spojovacího pole. Protože je ovšem pro vyhledávání maximální hodnoty v řádcích použita funkce `max`, která v případě nalezení stejných (maximálních) hodnot v řádku (vektoru) vrací index prvku, který je na pozici s menším indexem, tak byl vybrán rámec uložený na pozici n_{25}[1], který přišel na vstup přepínače dříve. V těchto případech je tedy využito principu **FIFO** fronty. K odeslání do spojovacího pole byl tedy vybrán rámec na pozici m_{25}. Protože tento rámec měl vstoupit do spojovacího pole přes port 2 a měl být přepnut na výstupní port 3, tak již nemohl žádný rámec v dané přepínací periodě této cesty využít. Z tohoto důvodu byly ve výstupní matici neuronové sítě $\boldsymbol{N}$ vymazány stejně jako v předešlém bodě všechny prvky, jejichž indexy odpovídaly indexům rámců v matici $\boldsymbol{M}$, které měly stejnou cílovou adresu (tj. 3) nebo zdrojovou adresu (tj. 2) jako vybraný rámec. Jednalo se tedy o prvky $n_{13} = -0,0238$, $n_{21} = -0,3749$, $n_{22} = -0,0866$, $n_{23} = 0,1446$, $n_{24} = 0,2851$ a $n_{44} = 0,2469$. Stav výstupní

[1] „index (2,5)“ - z pohledu programu Matlab se jedná ve skutečnosti o index (2,1), protože všechny matice jsou ve výpisech simulačního modelu síťového prvku pro lepší názornost zrcadlově otočeny zleva doprava (pomocí funkce `fliplr`).

matice $\boldsymbol{N}$ z neuronové sítě po druhém rozhodovacím kroku v bloku BSD můžeme vidět zde.

Výpis programu 7.7: Obsah matice **N** po 2. kroku výběru

```
Matice výstupních hodnot z neuronové sítě (po 2. kroku výběru):
-0.0222    0.1652    0         0.0731   -0.0207
 0         0         0         0         0
 0         0         0         0         0
-0.3749   -0.3749    0.1563    0         0.2485
```

3. Ve 4. řádku matice $\boldsymbol{N}$ byla nalezena maximální hodnota u prvku $n_{45} = 0.2485$. Tomuto prvku odpovídal rámec (*cílová adresa* = 2, *zdrojová adresa* = 4, *priorita* = 1 a *velikost datové části* = 650B), který byl uloženy v matici $\boldsymbol{M}$ (centrální paměti) na pozici m_{45}. Tento rámec byl tedy vybrán k odeslání do spojovacího pole. Protože měl tento rámec vstoupit do spojovacího pole přes port 4 a měl být přepnut na výstupní port 2, tak již nemohl žádný rámec v dané přepínací periodě této cesty využít. Z tohoto důvodu byly stejně jako v předešlých případech ve výstupní matici neuronové sítě $\boldsymbol{N}$ vymazány všechny prvky, jejichž indexy odpovídaly indexům rámců v matici $\boldsymbol{M}$, a které měly stejnou cílovou adresu (tj. 2) nebo zdrojovou adresu (tj. 4) jako vybraný rámec. Jednalo se tedy o prvky $n_{11} = -0,0222$, $n_{12} = 0,1652$, $n_{41} = -0,3749$, $n_{42} = -0,3749$ a $n_{43} = 0,1563$.

Výpis programu 7.8: Obsah matice **N** po 3. kroku výběru

```
Matice výstupních hodnot z neuronové sítě (po 3. kroku výběru):
 0         0         0         0.0731   -0.0207
 0         0         0         0         0
 0         0         0         0         0
 0         0         0         0         0
```

4. Nakonec byla v 1. řádku matice $\boldsymbol{N}$ nalezena maximální hodnota u prvku $n_{14} = 0.0731$. Tomuto prvku odpovídal rámec (*cílová adresa* = 1, *zdrojová adresa* = 1, *priorita* = 1 a *velikost datové části* = 1219B), který byl uložen v matici $\boldsymbol{M}$ na pozici m_{14}. Tento rámec byl tedy vybrán k odeslání do spojovacího pole. Následně došlo k vymazání zbývajících prvků matice $\boldsymbol{N}$ (tj. $n_{15} = -0.0207$). V tomto kroku bylo vidět, že byl upřednostněn rámec, který

sice přišel do síťového prvku později než rámec uložený na pozici m_{15} v centrální paměti (a který měl stejnou cílovou adresu). Tento rámec měl ovšem prioritu 3, která je nižší než u vybraného rámce. Proto byl rámec na pozici m_{14} vybrán ke vpuštění do spojovacího pole jako první, tzn.**prioritně přepnutý**.

Na následujícím výpisu z programu (matice $\boldsymbol{B}$) můžeme vidět pozice prvků v centrální paměti, které byly vybrány blokem BSD k odeslání do spojovacího pole. Na příslušných pozicích se nachází hodnota 1. Na základě těchto hodnot byly vybrané rámce odeslány na příslušné vstupy spojovacího pole.

Výpis programu 7.9: Pozice rámců (matice **B**) v centrální paměti vybraných blokem BSD k odeslání do spojovacího pole

```
Matice znázorňující pozice paměťových míst v centrální paměti,
kde se nacházejí rámce určené k odeslání do spojovacího pole:
```

```
    0     0     0     1     0
    0     0     0     0     1
    0     1     0     0     0
    0     0     0     0     1
```

Dále blok BSD poslal do bloku ŘM informace o zdrojových a cílových adresách vybraných rámců. Blok ŘM na jejich základě sestavil řídicí matici $\boldsymbol{R}$, pomocí které se nastavil stav (sepnuto / rozepnuto) jednotlivých spínacích bodů ve spojovacím poli.

Výpis programu 7.10: Řídicí matice **R** udávající pozici sepnutých spínačů ve SP

```
Řídicí matice R udávající pozici sepnutých spínačů ve SP:
```

```
Řádky matice odpovídají vstupům do spojovacího pole.
Sloupce matice odpovídají výstupům ze spojovacího pole.
```

```
    1     0     0     0
    0     0     1     0
    0     0     0     1
    0     1     0     0
```

Hodnota uložená na pozici $r_{11} = 1$ v této matici odpovídá stejné pozici spínače ve spojovacím poli (spínač byl tedy uveden do sepnutého stavu). I ostatní prvky s hodnotou 1 v matici $\boldsymbol{R}$ způsobily sepnutí příslušných spínačů a došlo k přepnutí

rámců na požadované výstupy. Názorně je celý proces vidět na obrázku 7.6 i s podrobným popisem. Všechny rámce, které byly odeslány na výstupy přepínače, jsou vidět na výstupu z programu.

Výpis programu 7.11: Rámce odeslané přes SP na příslušné výstupní porty

```
Rámce odeslané přes SP na příslušné výst.porty:
```

Cílová adresa	Zdrojová adresa	Priorita	Datová část rámce [B]
1	1	1	1219
3	2	1	1431
4	3	5	562
2	4	1	650

Po odeslání těchto rámců z centrální paměti byla vypsána její aktuální obsazenost.

Výpis programu 7.12: Obsazenost centrální paměti po odeslání rámců do SP

```
Matice znázorňující obsazené paměťové pozice v centrální paměti
po odeslání vybraných rámců na příslušné výst. porty přepínače:
Číslo 1 značí obsazené a 0 prázdné paměťové místo v paměti
    1     1     1     0     1
    0     1     1     1     0
    0     0     1     1     1
    0     0     1     1     0
```

Následně došlo k **dekrementaci priorit** u všech rámců, které zůstaly v centrální paměti o hodnotu **1**. Tímto krokem se zvýšila pravděpodobnost jejich odeslání v budoucnu. Také byla provedena **defragmentace** volného místa v paměti, při které došlo k **realokaci rámců**. Posledním krokem bylo načtení nových rámců ze vstupních vyrovnávacích pamětí a jejich uložení na volné pozice v centrální paměti. Obsah paměti po provedení těchto kroků je vypsán na následující stránce.

Výpis programu 7.13: Obsah centrální paměti (matice **M**) po načtení nových rámců

```
Obsah centrální paměti (CP):
```

Cílová adresa	Zdrojová adresa	Priorita	Datová část rámce [B]
1	1	2 -	1325
3	2	1 =	314
2	3	4 -	333
3	4	2 -	1121
3	1	6 -	569
3	2	3 -	264
1	3	6 -	730
1	4	1 =	714
2	1	1 =	135
1	2	4 -	970
2	3	1 -	1024
3	4	7	756
2	1	4 -	680
3	2	7	903
3	3	1	1467
3	4	5	1361
0	0	8	0
0	0	8	0
4	3	7	347
3	4	7	186

Změny, které nastaly po odeslání rámců do spojovacího pole a provedení předchozích kroků, jsou ve výpisu obsahu centrální paměti vyznačeny následovně:

- Zeleně podbarvené řádky - takto jsou označeny rámce, které zůstaly na původních pozicích v centrální paměti.
- Modře podbarvené řádky - takto jsou označeny rámce, které byly přemístěny na uvolněné pozice (po odesláných rámcích) v centrální paměti.
- Červeně podbarvené řádky - takto jsou označeny nové rámce, které byly načteny ze vstupních vyrovnávacích pamětí portů přepínače.
- Šedě podbarvené řádky - takto jsou označeny „virtuální rámce“ (volné paměťové pozice v centrální paměti).
- Znakem (-) jsou označeny priority, u kterých došlo k dekrementaci. K tomuto procesu došlo u rámců, které zůstaly v paměti a jejich hodnoty priorit byly větší než 1. Tímto se zvýšila pravděpodobnost jejich odeslání v budoucnu.

- Znakem (=) jsou označeny priority, které zůstaly beze změny. Toto se týkalo rámců, které zůstaly v paměti a měly hodnoty priorit *1*. U nich již nemohlo dojít ke snížení priorit, protože priority mohou nabývat pouze hodnot z intervalu *1* až *n*. Přičemž hodnota *n* je nastavena uživatelem na začátku simulace a hodnota 1 je nejvyšší priorita, která může být rámcům přidělena.

Pokud by byla stisknuta klávesa *Enter*, tak by simulace pokračovala s nově načtenými rámci. Stiskem klávesy *q* byl program ukončen.

Tímto byl dokončen popis funkce navrženého simulačního modelu aktivního síťového prvku. Všechny zdrojové kódy jsou okomentovány a umístěny v příloze na CD.

8 ZÁVĚR

V integrovaném prostředí MATLAB byl vytvořen simulační model síťového prvku (přepínače). Konkrétně se jedná o model přepínače s jednocestným spojovacím polem, které pracuje na principu křížového přepínače. Tato architektura byla zvolena z toho důvodu, že se vyznačuje řadou zajímavých vlastností, jako např. nehrozí u ní vnitřní blokování, má jednoduchou architekturu a je modulární. Také je vhodná pro demonstraci funkce v případě použití simulačního modelu při laboratorní výuce, což byl jeden z dílčích cílů této práce. Celkové blokové schéma simulačního modelu přepínače je na obrázku 7.2 v kapitole 7.

Dále byla pro potřeby simulace vytvořena datová jednotka (rámec), jejíž struktura je znázorněna na obrázku 7.1 v kapitole 7.

Rámce ze vstupních vyrovnávacích pamětí jsou ukládány do centrální paměti. Její velikost je dána nastavenými parametry modelu na začátku simulace. Konkrétně se jedná o parametry: počet portů přepínače a velikost okna. Velikost okna udává maximální počet rámců, které mohou být načteny z jednotlivých vstupních vyrovnávacích pamětí portů přepínače během jedné periody. Z jednotlivých polí rámců uložených v centrální paměti bývají načítány informace do bloků MP, MCA a MZA. Těmito informace jsou údaje o jejich prioritě, cílové a zdrojové adrese. Následně bývají zaslány do bloku NS, kde dochází k jejich normování na inicializační hodnoty, které slouží jako vstup do Hopfieldovy neuronové sítě.

Hopfieldova neuronová síť byla vytvořena s použitím NEURAL NETWORK TOOLBOXu a její struktura v prostředí SIMULINK je zachycena na obrázcích 7.8 až 7.11 v kapitole 7. Tato neuronová síť byla použita pro optimalizaci přepínání ve spojovacím poli přepínače tak, aby bylo toto pole efektivně využito, a současně, aby prováděla prioritní zpracování rámců. Výstupem z neuronové sítě je matice výstupních hodnot $\boldsymbol{N}$, která představuje ohodnocení jednotlivých rámců hodnotami z intervalu -1 až $+1$. Tento interval hodnot je dán použitou přenosovou funkcí satlins (obr. 7.7 v kapitole 7). Velikost jednotlivých ohodnocení vyjadřuje nutnost odeslání daného rámce do spojovacího pole v dané periodě tak, aby byla maximálně využita přenosová kapacita spojovacího pole při současném zohlednění priorit jednotlivých rámců.

Výstup z neuronové sítě je následně zpracován v bloku BSD. Na jeho základě je v bloků ŘM sestavena řídicí matice $\boldsymbol{R}$, která slouží k ovládání stavů jednotlivých spínacích bodů ve spojovacím poli. Po sepnutí daných spínačů dochází k odeslání rámců do příslušných vyrovnávacích pamětí výstupních portů.

Závěrem lze konstatovat, že použití neuronové sítě pro optimalizaci řízení přepínače se jeví jako velmi perspektivní. Ve většině případů došlo po postupné excitaci neuronů v neuronové síti k nalezení „výhodného“ stabilního stavu (atraktoru)

ve vstupním prostoru, který reprezentoval některá z lokálních minim energetické funkce. Slovo „výhodného“ je zde myšleno s ohledem na efektivní využití spojovacího pole při prioritním přepínání.

Vytyčené cíle diplomové práce byly splněny. Vytvořený model síťového prvku byl zpracován tak, aby mohl být případně využit pro demonstraci možnosti využití neuronové sítě pro optimalizaci řízení u síťovém prvku v laboratorní výuce.

Možným pokračováním práce může být rozšíření softwarového simulačního modelu síťového prvku o realizaci hardwarové implementace, např. pomocí programovatelných logických obvodů. Před tímto krokem by bylo vhodné otestovat navržený softwarový model v reálném provozu. Vhodnou hardwarovou realizací se může radikálně zvýšit rychlost zpracování, protože paralelní zpracování dat v neuronové síti je během simulace na běžných sekvenčních počítačích v případě síťového prvku s větším počtem portů poměrně pomalé.

LITERATURA

[1] NOVÁK, M. a kolektiv *Umělé neuronové sítě: teorie a aplikace*. 1. vydání. Praha: C. H. Beck, 1998. 382 s. ISBN 80-7179-132-6.

[2] NOVÁK, M. *Neuronové sítě a neuropočítače*. 1. vydání. Praha: VÝBĚR, 1992. 192 s. ISBN 80-901245-0-X.

[3] JAN, J. *Číslicová filtace, analýza a restaurace signálů*. 2. opravené a rozšířené vydání. Brno: VUTIUM, 2002. 427 s. ISBN 80-214-2911-9.

[4] KECMAN, V. *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*. 1. vydání. London: The MIT Press, 2001. 541 s. ISBN 0-262-11255-8.

[5] ŠNOREK, M. - JIŘINA, M. *Neuronové sítě a neuropočítače*. 1. vydání. Praha: Vydavatelství ČVUT, 1998. 124 s. ISBN 80-01-01455-X.

[6] TUČKOVÁ, J. *Úvod do teorie a aplikací umělých neuronových sítí*. 1. vydání. Praha: Vydavatelství ČVUT, 2005, 103 s. ISBN 80-01-02800-3.

[7] ŠÍMA, J. - NERUDA, R. *Teoretické otázky neuronových sítí*. 1. vydání. Praha: MATFYZPRESS, 1996, 390 s. ISBN 80-85863-18-9.

[8] BURDA, K. *Návrh, správa a bezpečnost počítačových sítí*. Elektronická skripta. Fakulta elektrotechniky a komunikačních technologií VUT v Brně.

[9] VONDRÁK, I. *Umělá inteligence a neuronové sítě*. 2. vydání. Ostrava: VŠB - TECHNICKÁ UNIVERZITA OSTRAVA, 2002, 140 s. ISBN 80-7078-949-2.

[10] NOVOTNÝ, V. *Architektura sítí*. Elektronická skripta. Fakulta elektrotechniky a komunikačních technologií VUT v Brně.

[11] BÍLA, J. *Umělá inteligence a neuronové sítě v aplikacích*. Praha: Vydavatelství ČVUT, 1998, 135 s. ISBN 80-01-01769-9.

[12] MAŘÍK, V. - ŠTĚPÁNKOVÁ, O. - LAŽANSKÝ, J. a kolektiv *Umělá inteligence (4)*. 1. vydání. Praha: Academia, nakladatelství Akademie věd České republiky, 2003. 475 s. ISBN 80-200-1044-0.

A PŘÍLOHA

Obsah CD

- `DP.pdf` – Tento dokument.
- `readme.txt` – Soubor obsahující informace o požadavcích na použitý software a úkony nezbytné pro spuštění simulace.
- `metadata.pdf` – Dokument obsahující metadata diplomové práce.

+ \`LaTeX` – Obsahuje zdrojové kódy tohoto dokumentu.
+ \`Matlab` – Obsahuje zdrojové kódy funkcí a skriptů prostředí MATLAB.
  - `model_sitoveho_prvku.m` – Hlavní soubor. Nastavení parametrů modelu přepínače. Je nezbytné spustit tento skript pro spuštění simulace.
  - `generovani_dat.m` – Generování rámců. Zobrazování výsledků v průběhu simulace. Správa paměti přepínače.
  - `neuronova_sit.m` – Vytvoření neuronové sítě a řídicí matice. Řízení spojovacího pole.
  - `neuronova_sit_model.mdl` – Model přepínače v prostředí Simulink (obr.7.8 až 7.11).