UČEBNÍ TEXTY OSTRAVSKÉ UNIVERZITY

Přírodovědecká fakulta

UMĚLÁ INTELIGENCE

Hashim Habiballa

Ostravská Univerzita
2004

UMĚLÁ INTELIGENCE
KIP/UMINT

distanční studijní opora

PaedDr. Mgr. Hashim Habiballa, PhD.

UMĚLÁ INTELIGENCE

Hashim Habiballa

Cíl předmětu

Seznámit studenty se základními problémy umělé inteligence (UI). Vymezit UI jako obor i jako soubor unikátních problémů a přístupů k řešení. Absolvent by měl získat potřebný nadhled pro další studium v hlouběji zaměřených předmětech.

Obsah:

Úvod pro práci s textem pro distanční studium.

Průvodce studiem:

Účel textu

Tento text má sloužit pro potřeby výuky umělé inteligence jako přehledového kurzu na katedře informatiky a počítačů PřF Ostravské Univerzity. Má ukázat základní problémy umělé inteligence, a uvést studenty do studia vybraných disciplín, které budou studovat ve vyšších ročnících hlouběji.

Nepředpokládá se žádná předchozí znalost problematiky, předmět má rysy vstupního kurzu, ve kterém student získá základní přehled o otázkách umělé inteligence. Zároveň v něm získáte užitečné odkazy pro další hlubší studium partií, které vás zaujmou a z těchto oblastí pak budete zpracovávat korespondenční úkoly.

Struktura

V textu jsou dodržena následující pravidla:

- je specifikován cíl lekce (tedy co by měl student po jejím absolvování umět, znát, pochopit)
- výklad učiva
- důležité pojmy – otázky
- korespondenční úkoly (mohou být sdruženy po více lekcích)

Zpracujte prosím nejméně JEDEN korespondenční úkol z této opory a zašlete jeho řešení do konce semestru (nejpozději před zkouškou). Pozor! Zvolíte-li si úkol teoretický z poslední kapitoly, bude kriticky posuzována systematičnost a komplexnost zpracování. U ostatních úkolů prakticky orientovaných se bude přiměřeně přihlížet k náročnosti realizace (řešení nemusí mít tak komplexní charakter).

Zároveň Vás chci laskavě poprosit, abyste jakékoliv věcné i formální chyby v textu zdokumentovali a kontaktovali mne. Budu Vám za tuto pomoc vděčen a usnadní to učení i vašim kolegům.

1. Historie umělé inteligence

Cíl:

Získáte základní přehled o těchto otázkách:
- co se rozumí pojmem „umělá inteligence“
- jaká je historie a současnost tohoto oboru

Tento učební text by vám měl především otevřít obzor do celé škály problémů, postupů a otázek, které spadají do oblasti umělé inteligence. I když existuje mnoho definic umělé inteligence, je velice těžké vymezit ji jako disciplínu s přesnými hranicemi jako je teoretická informatika, numerická matematika a podobně. Proto se nejprve pokusíme podívat se do historie tohoto oboru.

1.1. Historie umělé inteligence

Inteligence

Inteligence je vlastností některých živých organismů. Vznikla a vyvíjela se v průběhu dlouhého časového intervalu a dnes umožňuje některým živým organismům efektivně reagovat na složité projevy prostředí a aktivně je využívat ve svůj prospěch a k dosažení svých cílů (Mikulecký, Ponce, 1996).

Zařazení umělé inteligence jako oboru je vcelku obtížné. Lze na ni pohlížet jako na matematickou disciplínu s aplikacemi, nebo také jako na technický obor. Závratný růst tohoto oboru ovlivňovalo mnoho faktorů, jako např. zvyšování požadavků v oblastech automatizovaného řízení, průzkumu nedosažitelných míst a řada dalších činností, kde je přítomnost člověka z technických či zdravotních důvodů vyloučena.

Umělá Inteligence

Umělá inteligence (UI) jako vědní disciplína se postupně formuje v posledních třiceti letech jako průsečík několika disciplín, jakými jsou např. psychologie, neurologie, kybernetika, matematická logika, teorie rozhodování, informatika, teorie her, lingvistika atd. Její vývoj není zdaleka ukončen. Dnes jí však již nikdo neupírá právo na samostatnou existenci. Přesto má však mezi ostatními vědními disciplínami poněkud specifické postavení, a to ze dvou důvodů. Za prvé, dosud neexistuje všeobecně přijímaná definice umělé inteligence; za druhé, umělá inteligence neposkytuje jednotící teorie - spíše volně sdružuje různorodé teorie, metody a techniky, které lze úspěšně používat k počítačovému řešení některých složitých úloh rozhodování, plánování, diagnostiky apod.

Počítačová UI byla ve větším měřítku použita poprvé v NASA, a to k řízení a kontrole letu dálkových družic. Asi všem je jasné, že základem UI je počítač. Bez počítače by nebylo UI a podle toho, jakou roli hraje počítač

v existenci umělé inteligence, rozdělujeme UI na slabou - tam je počítač jen užitečným nástrojem a silnou - počítač není pouhým nástrojem, nýbrž přímo myslí a pomocí programu chápe. Možná někdy v budoucnu se objeví tak sofistikovaný počítač, který zvládne myslet sám o sobě.

Rok 1941, poprvé byl použit elektronický počítač. Bylo to v Německu. Počítač zabíral obrovské místnosti a byl chlazen leteckými motory. Převážně byl využíván pro vojenské účely. Jeho programování bylo pro programátory hotová noční můra.
Počátky roku 1950 Norbert Wiener přišel na to, že vlastně všechna inteligentní rozhodnutí jsou založena na principu zpětné vazby (feedback). Princip zpětné vazby použil už James Watt ve svém parním stroji, i když on asi netušil, že tento objev velmi ovlivní počátky vývoje UI.

Později v roce 1955, Newell a Simon vyvinuli The Logic Theorist, mnohými považovaný za první UI program. Tento program, který reprezentoval každý problém jako tree (stromový) model, se pokoušel vyřešit ho tak, že našel větev, jejíž výsledek byl s největší pravděpodobností ten správný. Dopad Logic Theorist na veřejnost a na pole UI, měl také rozhodující vliv na vývoj UI.

V roce 1956 John McCarthy, který je považován za otce UI, zorganizoval konferenci pro všechny zajímající se o strojovou inteligenci. Pozval je do Vermontu na The Dartmouth summer research project on artificial intelligence. Od této chvíle se oboru začalo říkat UI. I přes trochu rozpačitý průběh dartmouthská konference přivedla k sobě všechny objevitele UI a stala se výchozím bodem pro budoucnost UI.

Sedm let poté se pole UI dalo znovu do pohybu. Ideje z konference byly přezkoumány a přepracovány. Centra výzkumu se zformovala na Carnegie Mellon a MIT. Byla stanovena nová výzva: Vytvořit systém, který by efektivně řešil problémy, podobně jako Logic Theorist. A za druhé, vytvořit systém, který by se dokázal sám učit.

V roce 1957 se testovala první verze programu The General Problem Solver (GPS) . Program vytvořili ti samí lidé jako Logic Theorist. GPS byl rozšířením principu zpětné vazby a byl schopen řešit velké množství běžných problémů. Pár let nato, IBM vytvořilo tým na výzkum umělé inteligence. Herbert Gelerneter strávil 3 roky práce na řešení geometrických teorémů.

V roce 1958 John McCarthy uvedl svůj nový objev, jazyk LISP, který se užívá dodnes. LISP (LISt Processing - zpracování seznamů) se stal brzy jazykem UI vývojářů.

Dalším mezníkem ve vývoji UI byl rok 1963, kdy MIT dostala grant v hodnotě 2.2 milionů dolarů od Ministerstva obrany USA, aby zajistila technologickou výhodu proti Sovětskému svazu. Projekt posloužil k zvyšování tempa vývoje UI a k jeho financování.

V dalších letech vznikalo velké množství programů. Koncem šedesátých let vznikl třeba program STUDENT, který dokázal řešit algebraické problémy, nebo program SIR, který rozuměl jednoduchým anglickým větám. Výsledkem těchto programů bylo zdokonalení v porozumění jazykům a logice.

Sedmdesátá léta znamenala příchod expertních systémů. Expertní systémy předvídají pravděpodobnost řešení ve stanovených podmínkách. V těchto letech vzniklo mnoho nových metod. Například David Marr předložil novou teorii o strojovém vidění. Týkala se rozeznávání obrazů na základě odstínu, základních informací o tvarech, barvě, hranách a texturách. Další novinkou v roce 1972 byl jazyk PROLOG.

Osmdesátá léta znamenaly pro UI zrychlení vývoje. Zvýšila se poptávka po expertních systémech pro jejich účinnost. Podniky jako Digital Elektronics užívaly XCON, expertní systém navržený pro programování na velkých VAX počítačích. Také firmy DuPont, General Motors a Boeing se silně spoléhaly na expertní systémy.

Samozřejmě že zároveň s vývojem se začalo využívat UI i v praxi. Například práce Minského a Marra měla za následek vznik kamer a počítačů pro kontrolu kvality na výrobních linkách. Jak jinak, také armáda měla zájem o UI. Ministerstvo obrany USA financovalo projekt na vývoj smart truck. Cílem bylo vytvořit robota, který by dokázal provést mnoho bitevních úkolů. Projekt byl velkým rozčarováním a proto Pentagon brzy jeho financování zastavil.

I přes některé neúspěchy se UI pomalu vyvíjela dál. Byly objeveny fuzzy logika a neuronové sítě a začalo se ukazovat, že UI se dá použít i v reálném životě (mimo jiné, UI systémy byly použity také ve válce během Písečné bouře - raketové systémy, heads-up-displays a jiné novinky byly prověřeny v praxi.)

1.2. Pojem umělé inteligence

V literatuře můžeme najít různé definice umělé inteligence. Pokusme se zde uvést alespoň nejdůležitější a nejvýstižnější z nich.

Marvin Minsky tvrdí, že „… umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který – kdyby ho dělal člověk – bychom považovali za projev jeho inteligence.“ (1967)

Tato definice vychází z Turingova testu.

Vyplývá z ní, že úlohy jsou tak složité, že i u člověka by vyžadovaly použití inteligence. Otázkou však je, *jaké vlastnosti má složitost a ‚inteligentní‘ řešení* ? Složitost lze ohodnotit počtem všech řešení, které

připadají v úvahu. Ale hledání řešení pouhým prohledáváním stavového prostoru možných řešení není možné u složitějších úloh ani prostřednictvím superrychlých počítačů. Tento postup navíc není možno nazvat inteligentním. Je tedy nutno omezit velikost množiny prohledávaných řešení, což se děje na základě *využívání znalostí*.

E. Richová se domnívá, že „… umělá inteligence se zabývá tím, jak počítačově řešit úlohy, které dnes zatím zvládají lidé lépe.“ (1991)

Tato definice se bezprostředně váže na aktuální stav v oblasti počítačových věd a je možno očekávat, že se v budoucnosti bude ohnisko této vědy posouvat a měnit. Nevýhodou této – jinak velmi výstižné – definice je fakt, že nezahrnuje úlohy, které dosud neumí řešit počítače, ale ani člověk.

Alternativní definice

1 UI je označení uměle vytvořeného jevu, který dostatečně přesvědčivě připomíná přirozený fenomén lidské inteligence.

2 UI označuje tu oblast poznávání skutečnosti, která se zaobírá hledáním hranic a možností symbolické, znakové reprezentace poznatků a procesů jejich nabývání, udržování a využívání.

3 UI se zabývá problematikou postupů zpracování poznatků – osvojováním a způsobem použití poznatků při řešení problémů.

Umělá inteligence je interdisciplinární vědou, která nemá pevně vymezený předmět zkoumání ani teoretický základ – jde spíše o soubor metod, teoretických přístupů a algoritmů, sloužících k řešení velmi složitých úloh.

TŘI ZÁKLADNÍ PROUDY UMĚLÉ INTELIGENCE

SYMBOLICKÝ FUNKCIONALISMUS

Symbolický funkcionalismus

Symbolický funkcionalismus je založen na dvou základních hypotézách: funkcionalistické hypotéze a hypotéze fyzikálního systému symbolů. Zjednodušeně řečeno, základními předměty výzkumu symbolického funkcionalismu je problém reprezentace znalostí a inteligentním prohledáváním stavového prostoru.

Se symbolickými způsoby náhledu na umělou inteligenci se při svém studiu setkáte poměrně zblízka. Jednak se budete hlouběji zabývat především typickou představitelkou symbolismu – logikou a také se blíže seznámíte ve vyšších kurzech s problematikou reprezentace znalostí různými metodami. Tento text by rozhodně neměl suplovat tyto vyšší kurzy. Přesto už nyní můžete přemýšlet o způsobech reprezentace znalostí. Pravděpodobně jste se na střední škole setkali aspoň se základy výrokové logiky. Víte, že věty přirozeného jazyka jsou poměrně složité. Přirozený jazyk (v našem případě zřejmě čeština), díky kterému se umíme dorozumět s lidmi okolo nás, je tím nejpřirozenějším způsobem, jak reprezentovat znalosti. Jenže takovýto přístup není rozhodně optimální pro počítač, který má pak s těmito znalostmi „inteligentně“ pracovat. To aby počítač (stroj)

porozuměl znalosti vyžaduje, aby tato znalost byla formulována mnohem jednoduššími prostředky – například výrokovou logikou.

Vezměme v úvahu následující zápis tvrzení v přirozeném jazyce.

Student složil úspěšně přijímací zkoušku, pokud udělal test z matematiky a zároveň i z informatiky.

V jazyce výrokové logiky bychom tuto bázi znalostí zapsali například následovně.

Výroky:

z – student složil úspěšně přijímací zkoušku,
m – student udělal zkoušku z matematiky,
i – student udělal zkoušku z informatiky,

$(m \,\&\, i) \rightarrow z$

Tento zápis znamená zjednodušeně „Když platí m a zároveň platí i, pak platí také z. Logika nám tedy pomocí symbolů umožňuje zapsat nějakou znalost pomocí prostředků, se kterými už počítač může poměrně efektivně pracovat. Klasické počítače jsou totiž na tomto „primitivním“ černobílém uvažování založeny.

KONEKCIONISMUS

Konekcionismus

Konekcionismus, jako směr bádání v rámci umělé inteligence předpokládá, že esence inteligence plyne ze statického propojení velkého počtu jednoduchých výpočetních jednotek. Myšlenka je inspirována mozkem jako médiem, které zosobňuje inteligentní uvažování. Základní výpočetní jednotkou rozumíme model neuronu, který se na základě hodnoty součtu vážených vstupů excituje do aktivního stavu nebo nikoliv. Mezi hlavní odvětví, které se z řadí ke konekcionismu patří neuronové sítě.

Neuronové sítě na rozdíl od klasických symbolických metod nejsou založeny na explicitní reprezentaci znalostí. Stejně jako v našem mozku jsou díky obrovským množstvím propojení neuronů uloženy nepřeberné dovednosti a informace jen díky těmto propojením, tak i v počítači je možné takové informace uložit nebo takové dovednosti realizovat. Rozdíl od symbolismu však je, že zdánlivě jde jen právě o chaotický shluk propojení. Nelze tu tedy již přímo vidět zápis jako v předchozím příkladě.

Robotický funkcionalismus

ROBOTICKÝ FUNKCIONALISMUS

Klíčová filosofie robotického funkcionalismu je založena na implementaci behaviorismu jako psychologické školy. Raději než se zabývat

representací inteligence, robotičtí funkcionalisté se koncentrují na funkcionalitu modelovaného systému. Jako inteligentní chování je zde chápána jako rozumná interakce mezi třemi entitami: systém, prostředí, úloha. V případě že bude agent na danou úlohu a v daném prostředí reagovat inteligentně (jako by reagoval člověk), je považován za agenta disponující schopností vykazovat inteligentní chování.

TŘI DRUHY POROZUMĚNÍ

Stupeň implementace inteligence pomocí umělého media závisí v principu na tom do jaké míry se podaří namodelovat klíčový faktor lidského uvažování a chápání. Chápání, stejně tak jako spousta jiných kvalit, lze charakterizovat ve smyslu silné nebo slabé míry. Řekneme-li o myšlence, systému nebo tvrzení, že je slabé, myslíme tím že je obecné. Na druhé straně silná myšlenka je ve své podstatě specifická.

Nejdůležitější probrané pojmy:

- inteligence
- minského definice, definice Richové a další alternativní definice
- symbolický funkcionalismus
- konekcionismus
- robotický funkcionalismus

Úkoly a otázky k textu:

1. Pokuste se odpovědět na otázku, jak souvisí vývoj umělé inteligence s obecnou historií informatiky.
2. Vyjmenujte tři úlohy, které podle vás vyžadují „inteligenci“.

2. Turingův test

Cíl:

Seznámíte se s:
- nejtypičtějším problémem UI
- jeho modifikacemi i praktickými aplikacemi
- dílem Alana Turinga – klíčové postavy teoretické informatiky

Turingův test je typickou úlohou umělé inteligence. I přestože je definován poměrně neurčitě a těžko ho lze považovat za exaktní definici umělé inteligence, pokusy o jeho řešení jsou pro pochopení základních východisek umělé inteligence velice potřebné. Pokuste se prosím nebrat tuto kapitolu, která vám bude možná připadat poměrně filozoficky orientovaná, jako zbytečnou. Měla by vám přinést jistou motivaci pro další studium konkrétních okruhů problémů, kterými se zabývá UI.

2.1. Turingův test

Turing se konstrukcí svého testu snažil otázku "mohou stroje myslet" převést z oblasti filozofických spekulací na exaktnější úroveň. Za myslící podle něj prohlásíme počítač tehdy, když jeho chovaní nebudeme schopni rozeznat od chování člověka.

Motivací pro vývoj umělé inteligence existovala už v polovině 20. století celá řada. Jedná se jistě o obrovskou teoretickou výzvu. Zkoumání inteligence umělé nám zřejmě dokáže mnoho říct o inteligenci vlastní. Dalším důvodem je prostá radost z konstrukce stále dokonalejších mechanismů. A neposlední řadě zde jsou motivy ryze praktické: Řadu úkolu musí plnit inteligentní bytosti, nicméně lidem se do nich příliš nechce (jsou např. stereotypní, nebo naopak nebezpečné).

Turingův test

Ve své původní podobě vychází Turingův test z tzv. imitační hry, ve které jde o to odlišit dva lidi podle pohlaví. Pozorovatel, na jehož pohlaví nezáleží, má proti sobě např. ženu a muže, který předstírá, že je žena. Trojice lidí spolu nepřijde do fyzického kontaktu, sedí v oddělených místnostech a zprostředkovatel mezi nimi přenáší popsané lístky. Turingův test spočívá v tom, že imitátorem člověka by byl počítač. Vyvstává tedy otázka, zda-li dokáže počítač simulovat chování ženy stejně dobře jako muž. Test jako celek se však dosud žádnému programu splnit nepodařilo (Houser, 2003).

Námitky a polemiky

Nejstarší námitka předchází samotný Turingův test o více než 100 let a pochází od Ady Lovelaceové, dcery lorda Byrona a spolupracovnice konstruktéra mechanických parních počítačů Charlese Babaggea. Ta přišla již v 1. polovině 19. století s tvrzením, že počítač nemyslí, protože myšlení je nějak spojeno s původností. Z počítače však dostaneme pouze modifikaci toho, co do něj vložíme (výsledek rovnice apod.). Za současného stavu vývoje IT není tato námitka příliš relevantní .Předně nevíme, jak vlastně definovat onu "původnost" - např. báseň je taktéž variací již existujících slov. Programy dnes dokáží psát povídky nerozpoznatelné od lidských nebo odhalit taková tajemství šachové hry, která zůstávala jejich tvůrcům zcela skryta. Program tedy svým chováním dokáže své tvůrce překvapit (nejen ve smyslu chyby), což je snad možné chápat jako ekvivalent původnosti.

Rozsáhlá literatura existuje o tzv. paradoxu čínského pokoje, s nímž přišel filozof John Searl. Searlova námitka uvádí, že splnění Turingova testu ještě nemusí znamenat nějaké myšlení a uvědomování. V původní formulaci Searl ukazuje, že člověk může být např. schopný smysluplně třídit tabulky s čínskými znaky, aniž rozumí čínsky a chápe, co text na tabulkách znamená. Podobně počítač, i když projde Turingovým testem, bude stále pouze něco imitovat.

Zajímavé námitky proti umělé inteligenci se týkají zdánlivě tak "strojového" oboru, jakým je matematika. V původních Turingových poznámkách se uvádí, že počítač by mohl byt v testu odhalen právě tím, že nedělá chyby, eventuálně počítá rychleji než člověk, tedy by paradoxně byl nápadný tím, že je dokonalejší, ale program by se zřejmě dal nějak modifikovat, aby se občas spletl, poskytoval výsledky se zpožděním apod..
Hawkingův spolupracovník Roger Penrose přišel s další matematickou námitkou, která má dokázat, že lidské myšlení má v sobě kromě algoritmu ještě něco jiného - počítače podle něj nejsou schopny matematického důkazu, např. odhalit, že prvočísel je nekonečně mnoho. Penrose se domnívá, že nejde jen o otázku softwaru a výpočetní kapacity, některé matematické úlohy jsou neřešitelné i pro obecný Turingův stroj.

Je možné, že v tuto chvíli ještě nevlastníme nějakou klíčovou znalost, která by nám umožnila konstruovat programy tak, aby prošly Turingovým testem. Může se stát, že Turingův test bude splněn až díky technologiím, o nichž dosud nemáme ani tušení.
Nakonec je zde ale ještě jeden problém. Zdá se, že jakkoliv splnění Turingova testu garantuje inteligenci, opak se již říct nedá. Proto je docela dobře možné, že se podaří vyvinout systémy, kterým neupřeme vlastní "myšlení", ale Turingovým testem tyto aplikace přesto neprojdou.

2.2. Modifikace Turingova testu a aplikace

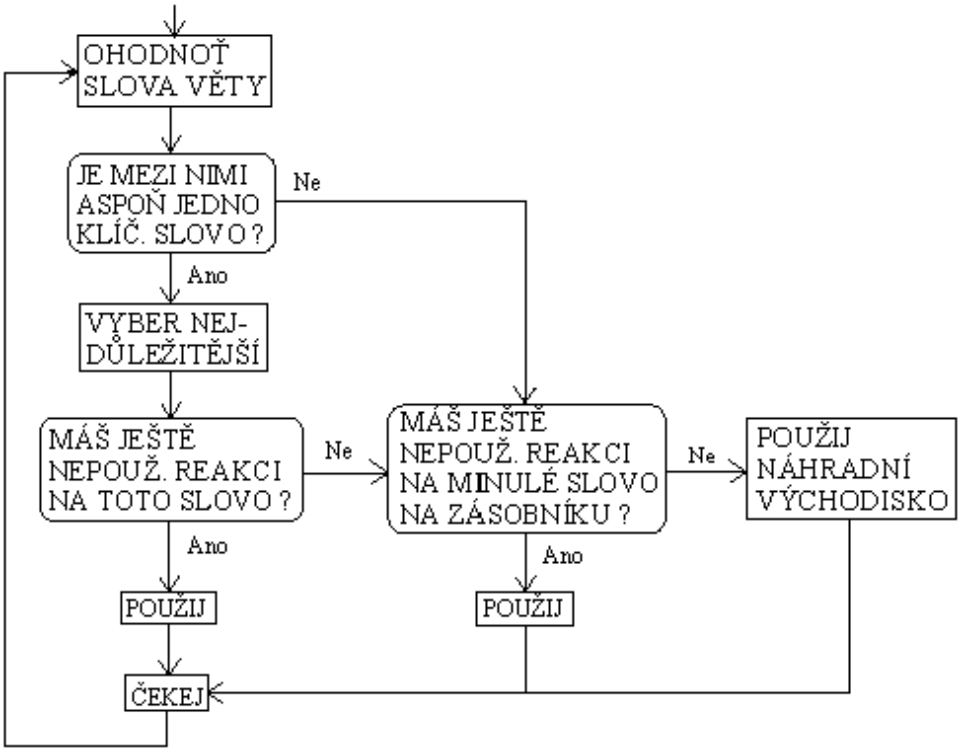
Během více než 50-ti let, které uplynuly od myšlenky původního Turingova testu, se samozřejmě objevila řada návrhů na jeho modifikace a zdokonalení.

První námitka vychází z toho, že počítač v Turingově testu simuluje pouze jednu z mnoha schopností člověka - schopnost rozumět jazyku a verbálně se vyjadřovat, oboje navíc pouze v psané formě. V modifikaci Stevana Harnada by počítač měl simulovat všechny lidské schopnosti - je jasné, že tento požadavek nesplní "program", ale pouze pokročilý humanoidní robot.

Eliza

Problémem modifikovaných verzí Turingova testu je obvykle jejich obtížná praktická realizovatelnost. Na rozdíl od původní varianty zůstanou proto zřejmě v rovině experimentů pouze myšlenkových.
Turingův test, částečně modifikovaný, vyvrátil - *alespoň podle některých pramenů* - program ELIZA. Člověk komunikující s tímto programem může uvěřit, že komunikuje s jiným člověkem. ELIZA se snaží imitovat psychiatra, formulace 'absolvovat Turingův test' znamená za 5 minut se neukázat být průhlednějším podvodníkem, než průměrný člověk, který si hraje na psychiatra, ačkoliv jím není.

Vývojový diagram Elizy:



Eliza pracuje se seznamem klíčových slov a s ním porovnává slova věty. Nalezne-li ve větě klíčové slovo, najde v tabulce reakcí odpovídající a nepoužitou odpověď a použije ji. Není-li mezi slovy věty klíčové slovo, reaguje Eliza na nepoužité klíčové slovo předchozí věty. Pokud už Eliza použila všechny reakce na klíčová slova na zásobníku, použije se náhradní východisko. Je to seznam reakcí, použitelných na téměř všechny situace, jako např.*"Máte pro svůj názor nějaký důvod?"*, *"Skutečně?"*, *"To je velmi zajímavé..."*, *"Řekněte mi o tom více."* a podobně.

Část dialogu mezi programem ELIZA (E) a pacientem (P): (přeloženo do češtiny)

P : Všichni lidé jsou stejní.
E : V jakém smyslu?
P : Pořád nás do něčeho nutí.
E : Máte na mysli nějaký určitý příklad?
P : No, můj chlapec mě přinutil, abych sem přišla.
E : Váš přítel vás donutil, abyste sem přišla!
P : Říká, že jsem pořád tak skleslá.
E : Je mi líto, že jste skleslá.
P : Je to pravda. Jsem nešťastná.
E : Myslíte, že vám pomůže, že jste sem přišla, abyste nebyla nešťastná?
P : Potřebuji nějakou pomoc, to je jisté.
E : Co by to znamenalo, kdybyste dostala nějakou pomoc?
P : Možná, že bych se naučila nějak vyjít se svou matkou.
E : Řekněte mi něco víc o své rodině!
P : Moje matka se o mě stará.
E : Kdo z vaší rodiny se ještě o vás stará?
P : Můj otec.
E : Váš otec.
P : V něčem se podobáte mému otci.
E : V čem vidíte podobnost?
...

Vidíte, že sestrojit program, který se zdánlivě chová jako člověk není tak těžké. Problémem však je, že v tomto případě se autor programu spíše snažil vyřešit problém Turingova testu. Samozřejmě, že praktické použití tak jednoduchého algoritmu je zřejmě sporné. Pravdou ale zůstává, že z tohoto příkladu rozhovoru bychom mohli nabýt dojmu, že ELIZA je člověk nikoliv stroj.

2.3. Život a dílo Alana Turinga

Alan Mathison Turing byl jednou z největších osobností a pionýrů počítačového světa. Osobně se podílel na vývoji prvních prototypů digitálních počítačů, je autorem obecně používaných termínů jako "Turingův stroj", nebo "Turingův test". Jako jeden z prvních uvažoval o konstrukcích a aplikacích umělé inteligence a pracoval na matematických podkladech pro algoritmizaci umělého života. V řadě případů byly jeho práce publikovány až mnoho let po jeho smrti, v nedokončené, přesto přínosné a často citované podobě (Zapletal, 2001).

Alan Turing se narodil v Londýně 23. června 1912. Profesionální kariéru v matematice zahájil na Cambridge University v roce 1931. V této době byl pověstný tím, že než by stavěl na předchozích pracích svých kolegů, raději si teoretické základy znovuvybudovával sám. Jakmile odgraduoval, přestěhoval se na Univerzitu v Princetonu.

Během druhé světové války pracoval Alan Turing pro britskou vládu na dekódování německých šifer. Německo používalo stroj s názvem Enigma, který byl schopen generovat průběžně měnící se kód, označovaný za nerozluštitelný. Realitou ovšem bylo, že jeho britský protějšek Collossus, prakticky jen hromada servomotorků a relátek (ale přece jen další krok k digitálním počítačům), jej po větší část války louskal bez sebemenších problémů.

Po válce Alan Turing pracoval pro National Physical Laboratory (NPL) na Automatic Computing Engine (ACE), prvním skutečném digitálním počítači. Během této doby se začal také zajímat o vztah mezi počítači a přírodou a v článku "Intelligent Machinery" vznesl otázku umělé inteligence. Turing věřil, že stroje mohou imitovat procesy, které probíhají v lidském mozku, i když připouštěl obtížnost této cesty a mimochodem anticipoval také psychologické aspekty, odmítání akceptování děl vytvořených strojovou cestou. Často uváděl příklady televizních kamer či mikrofonu, které v podstatě mohou simulovat funkčnost lidských vjemových orgánů. Ještě před ukončením projektu ACE se Turing přesunul do Univerzity v Manchesteru, kde vyvinul Manchester Automatic Digital Machine (MADAM).

Turingův stroj

Alan Turing byl skutečným pionýrem konceptu digitálního počítače. Turingův stroj (abstraktní matematická formalizace pojmu algoritmu) není v podstatě nic jiného než víceúčelový počítací stroj, který je schopen číst posloupnost jedniček a nul z pásky. Je to automat, který má schopnost čtení i zápisu na vstupní pásce, spolu s možností vracet se po potenciálně nekonečné pásce na libovolné místo na ní. Jde tedy o stroj, který na vstup dostane slovo v určité abecedě, může slovo modifikovat a vydat tuto modifikaci jako výsledek. Důležitá je pro jeho funkci správně sestavená přechodová funkce, jenž je formálně zobrazením stavů a symbolů na nový stav, nový zapsaný symbol a příznak posunu čtecí hlavy na pásce (hlava se může posouvat doleva a doprava, případně zůstat na místě). Takovýto Turingův stroj pak můžeme chápat jako prostředek, který realizuje zobrazení, stejně jako jej může realizovat program v Pascalu či strojový jazyk procesoru.

Často přehlíženým aspektem Turingova života jsou jeho práce na biologickém poli, které většinou zůstaly nedokončené (nicméně i v této podobě byly později publikovány). Převážně se zajímal o důvody vývoje určitých organických tvarů, postavených na bázi společenství primitivních buněk. Předešel tak o desítky let vznik oboru Alife.

Turing byl ve světě počítačů skutečným radikálem a byl pod značným psychologickým tlakem okolí. Časté "diskuse" s kolegy vedl na téma, zda počítač může být vytrénován na generování odpovědí mimo rámec omezené množiny možností. Reakcí v roce 1950 byl článek popisující dnes klasický lakmusový papírek inteligence Turingův test. ("Oxford Companion to the Mind" z roku 1987 jej popisuje jako nejlepší současný test, schopný potvrdit přítomnost inteligence ve stroji.) Bostonské Museum počítačů již několikrát pořádalo soutěž o Lebnerovu cenu, ve které se utkávají programy v jednodušší variantě Turingova testu v diskusí na vymezená témata. Turing zemřel za poněkud nejasných okolností (sebevražda, nebo nebezpečný pokus?) 7. června 1954. Škoda, že se

nedočkal vzniku prvních skutečných počítačů (tím spíše těch osobních), protože i když dnešní svět informatiky není výsledkem práce jednotlivce, jméno Alana Turinga má v dějinách jeho historie právo na skutečně čelní místo, a i v novodobých pracích na téma algoritmizace či umělé inteligence najdeme odkazy na jeho články.

Nejdůležitější probrané pojmy:

- Turingův test
- ELIZA
- Alan Turing

Úkoly a otázky k textu:

1. Najděte na Internetu libovolnou implementaci ELIZY a proveďte s ní krátký rozhovor. Pokuste se zpochybnit, že ELIZA se chová jako člověk.
2. Najděte nebo sestrojte příklad Turingova stroje pro jednoduchý vámi zvolený problém.

3. Řešení úloh a prohledávání stavového prostoru

Cíl:

Seznámíte se:
- se základními metodami řešení úloh
- s prohledáváním stavového prostoru
- metodami hraní her

Naučíte se:
- se programovat základní metody řešení úloh

Je-li dán počáteční model prostředí a požadovaný koncový model prostředí, pak je úkolem inteligentních strojů a systémů vyhledat vhodnou posloupnost akcí, jejichž aplikací lze přejít od počátečního modelu k cílovému. Taková posloupnost se nazývá **plán** a metody vytváření plánů se označují jako **metody řešení úloh**. Každému modelu odpovídá jistý stav prostředí, množina všech stavů tvoří **stavový prostor**.
Metody řešení úloh se dělí na:
- metody prohledávání stavového prostoru
- metody řešení rozkladem na podproblémy
- metody hraní her

Tyto metody využijete i v praxi. Mnoho úloh, které v informatice najdete nebo budete řešit, vyžaduje právě nějakou metodu prohledávání stavového prostoru (např. hledání cesty, optimalizační úlohy atd.)

3.1. Metody prohledávání stavového prostoru

Stavový prostor si můžeme představit jako orientovaný graf (strom), jehož uzly představují jednotlivé stavy úloh a jehož hrany představují přechody mezi těmito stavy. Cesta z počátečního stavu do některého cílového stavu je řešením úlohy. Metody jak najít řešení ve stavovém prostoru jsou v principu nastíněny níže.
Pozn. Pro následující ukázky prohledávání stavového prostoru mějme hrací plochu o 9 polích, obsahující kameny s čísly 1 až 8. Kameny jsou na hrací ploše náhodně rozmístěny a úkolem je posunem kamenů dosáhnout stavu na obr. 1.

Obr. 1

První metoda prohledávání stavového prostoru se nazývá „prohledávání do šířky“. Spočívá v tom, že na každé úrovni vždy vygenerujeme všechny možnosti a ty teprve řešíme.

a) Slepé prohledávání do šířky

Cena (ohodnocení) uzlu se rovná jeho hloubce. Uzly se expandují v pořadí, v jakém byly vygenerovány. Metoda nalezne vždy nejkratší cestu k cíli, pokud tato cesta existuje.

Obr. 2 naznačuje, jak je strom vytvořen:

Prohledávání „do šířky“

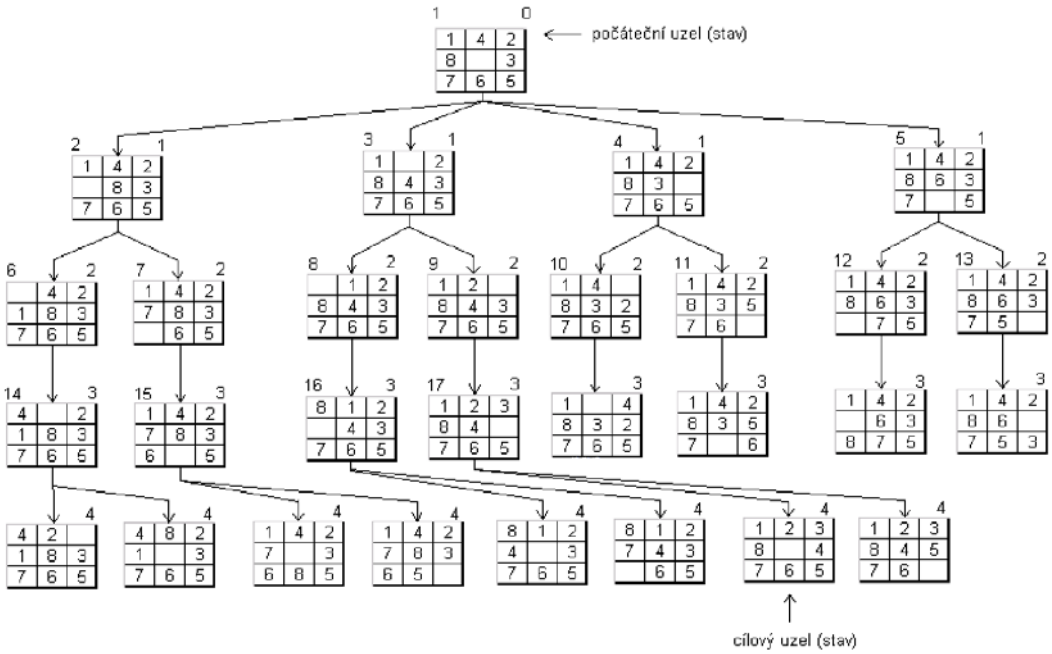
- všechny uzly mají označení, na které úrovni se nacházejí (číslo vpravo)
- číslo vlevo značí hodnotu stavu
- prochází se postupně celá úroveň, pokud se nenalezne cíl, pokračuje se na další úrovni

V některých případech je výhodnější prohledávat stavový prostor od cílového stavu(cílový stav je kořenem a počáteční stav hledaným listovým stavem), nebo oba postupy kombinovat.

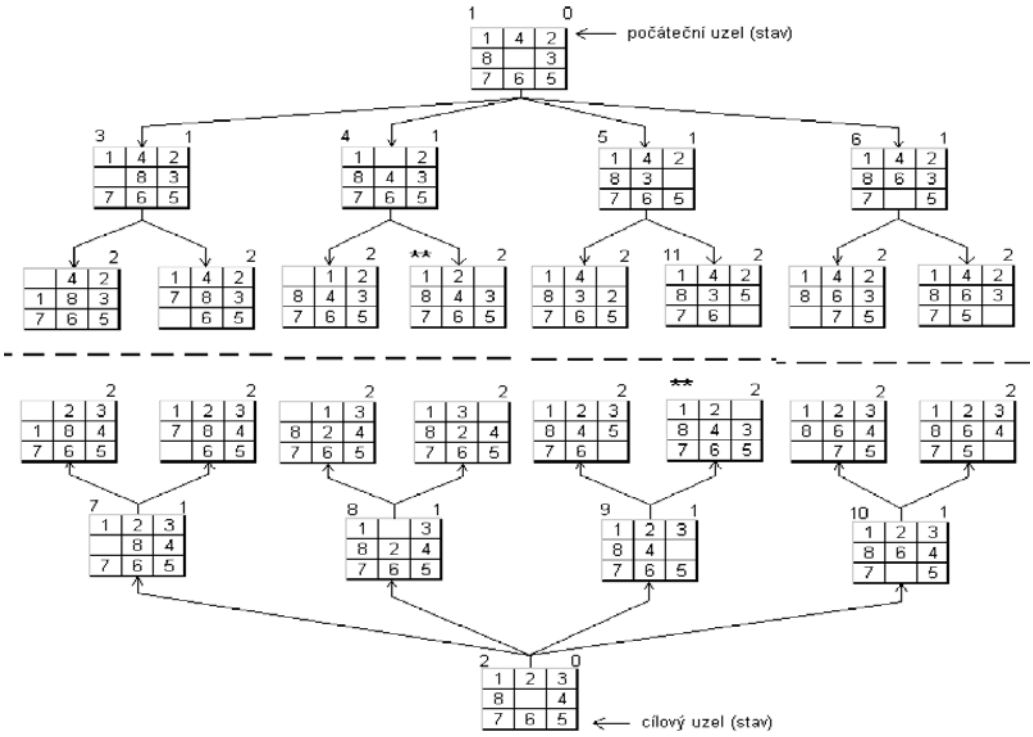
- Kombinovaný způsob:

(uzly označené ** představují hledaný společný uzel)

- obrázek 3



Obr. 2



Obr.3

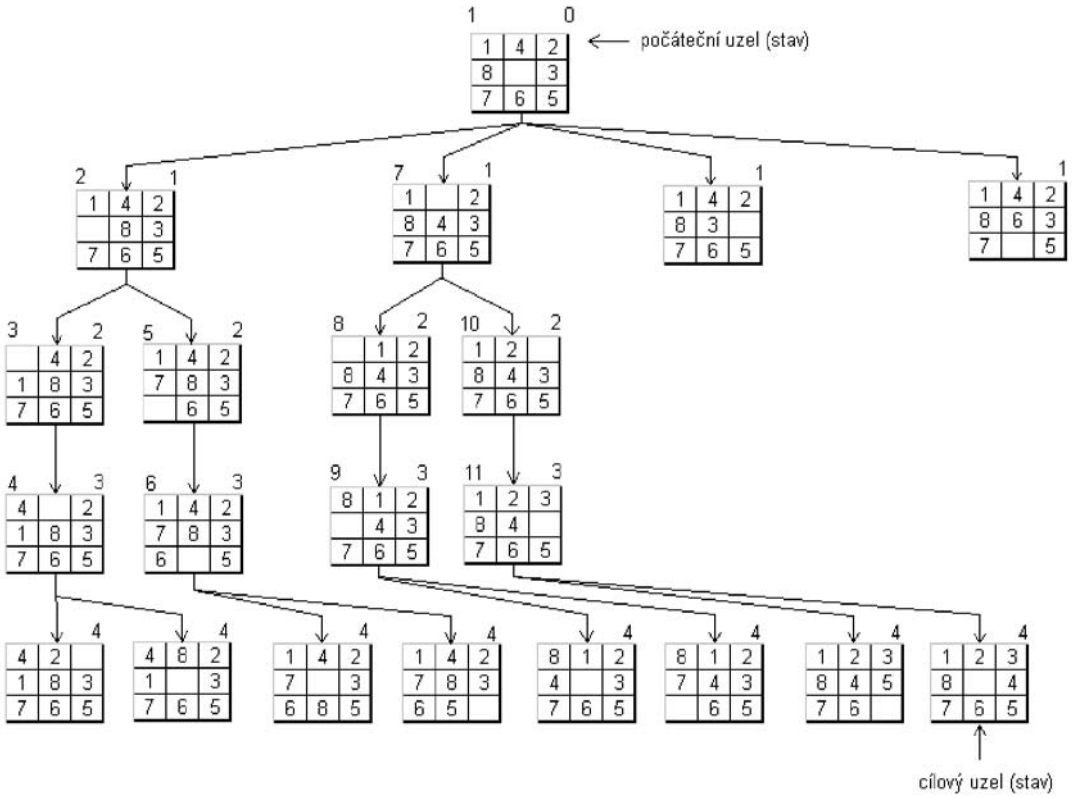
V jistém smyslu je další metoda duální k metodě prohledávání do šířky. Negeneruje všechna řešení na dané úrovni, ale jde „do hloubky“ vždy na nižší úroveň.

Prohledávání „do hloubky“

b) <u>Slepé prohledávání do hloubky</u>

Cena uzlu se rovná jeho hloubce. Expandují se uzly v největší hloubce v pořadí, v jakém byly vygenerovány. Často se tato metoda doplňuje omezující podmínkou – maximální hloubkou, aby se zabránilo prohledávání neperspektivních větví. Určení této podmínky je věcí intuice a může nalezení cílového stavu jak uspíšit, tak i oddálit.

<u>Pozn.</u> V našem příkladu max. hloubka=4.



Obr. 4

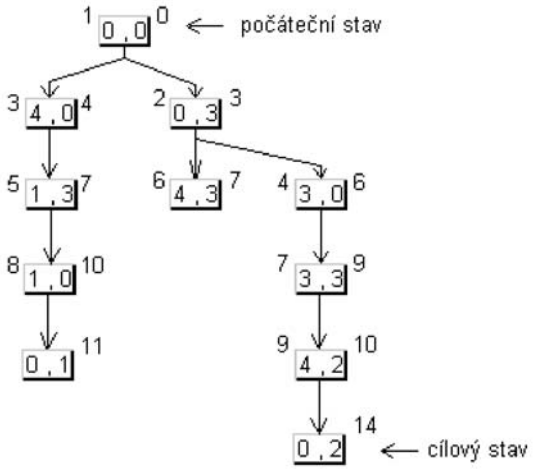
Prohledávání metodou stejných cen

c) <u>Slepé prohledávání metodou stejných cen</u>

Metoda vychází z předpokladu ohodnocení uzlů cenami, které odpovídají vynaloženým nákladům na vygenerování těchto uzlů. K expanzi se vždy vybírá uzel s minimálním ohodnocením. Jsou-li ohodnocení uzlů rovna jejich hloubce, je metoda stejných cen shodná s metodou prohledávání do šířky.

<u>Př.</u> Přelévání vody

Jsou dány dvě nádoby. Větší z nich má objem 4 litry(A), menší 3 litry(B). Na počátku jsou obě nádoby prázdné. Cílem je dosáhnout toho, že nádoba A je prázdná a v nádobě jsou 2 litry vody. K dispozici je neomezený zdroj vody, nádoby nemají žádné označení míry. Stav řešení úlohy je možné popisovat skutečným množstvím vody v nádobách A a B. Přechod mezi stavy je dán některou z těchto operací: vylití nádoby, naplnění nádoby a přelití vody z nádoby do nádoby.



Obr. 5

Pozn. Čísla v uzlech označují množství vody ve větší a v menší nádobě.

Postup: Z počátečního stavu se můžeme dostat jen do dvou stavů – naplníme buď nádobu A nebo nádobu B. Ceny uzlů se na této úrovni budou rovnat objemu nádob, tedy 4 a 3. Expandovat se bude uzel s menší cenou, tzn. uzel (0,3). Zde se opět můžeme dostat do dvou stavů – buď naplníme nádobu A a následující uzel bude (4,3) nebo přelijeme nádobu B do A a pak bude uzel (3,0). Nyní se bude expandovat uzel, který má větší cenu než uzel 2, tzn. uzel 3. Zde připadá v úvahu expanze do uzlu (1,3) – do nádoby B nalijeme 3 litry a v A zůstane 1 litr. Nyní musíme ohodnotit všechny uzly na 2. úrovni. Uzel (1,3) bude mít cenu 4 litry z předešlého uzlu + přelijeme 3 litry z A do B, tedy cena bude 7. Cena uzlu (4,3) bude 3+4=7. Uzel (3,0) má cenu 3+3=6. Analogicky pokračujeme v expandování a ohodnocování dalších uzlů.

Společnou nevýhodou metod slepého prohledávání je obrovské množství uzlů, které musí být generovány a testovány. To klade nejen neúnosné požadavky na paměť počítače, ale i na čas výpočtu. Pro složitější úlohy jsou tyto metody nepoužitelné.

Heuristické prohledávání

d) Heuristické prohledávání

Pro redukci počtu generovaných uzlů se mohou použít jisté znalosti o řešené úloze, které jsou často intuitivní. Tyto znalosti se nazývají heuristické znalosti nebo také heuristiky.

Heuristické znalosti můžeme využít dvěma způsoby:

1) první způsob je zřejmější a spočívá v zahrnutí těchto znalostí do pravidel. Například v úloze o nádobách je zřejmé, že naplnit oba džbány současně je nesmyslné, protože pak je nutné opět jeden z nich vyprázdnit.

2) Druhý způsob je založen na doplnění ohodnocující funkce o předpokládanou cenu k cíli:

$$f^*(i) = g^*(i) + h^*(i)$$

kde: f*(i) je ohodnocení uzlu i
g*(i) je dosavadní cena cesty z počátečního uzlu do uzlu i
h*(i) je předpokládaná cena cesty z uzlu i do cílového uzlu

3.2. Metody řešení rozkladem na podproblémy

Metody řešení rozkladem na podproblémy znáte pravděpodobně již z vlastní zkušenosti z programování. Například třídící metoda „Quicksort" je typickým příkladem rozkladu na podproblémy stejného typu. Najdeme dělící prvek a všechny hodnoty menší umístíme na jednu stranu pole a větší na druhou stranu. Tím nám vzniknou dva úseky u nichž je zaručeno, že jsou navzájem jako celky setříděny. Tyto dva úseky pak můžeme chápat jako podproblémy a aplikujeme na ně naprosto stejný princip.

Rozklad na podproblémy

Rozklad úlohy na podproblémy lze opět znázornit grafem, ale na rozdíl od předcházejících metod neodpovídají nyní uzly grafu stavům úlohy, ale podúlohám (podproblémům). Každý podproblém se opět rozkládá na jednodušší podproblémy až konečně listy grafu odpovídají buď elementárním úlohám nebo úlohám neřešitelným.

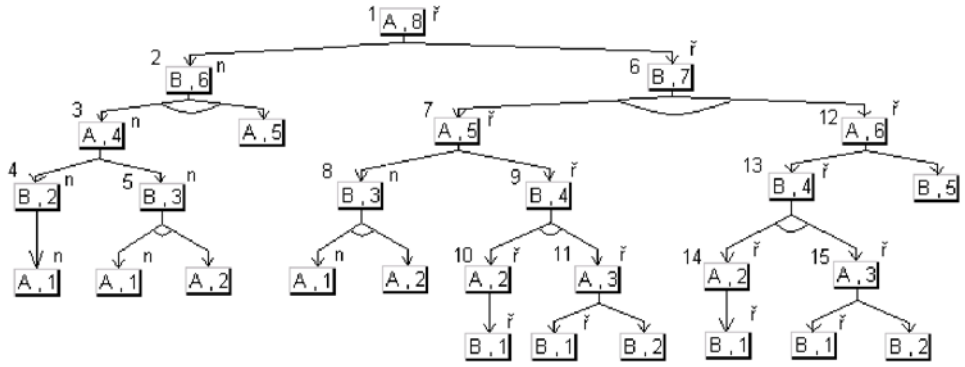
Druhý rozdíl od grafů stavových prostorů spočívá v tom, že rozeznáváme dva typy bezprostředních následníků – uzly AND a uzly OR (větve k následníkům typu AND se spojují obloučkem). Uzel je řešitelný pokud jsou řešitelní všichni jeho bezprostřední následníci typu AND, resp. je-li řešitelný alespoň jeden bezprostřední následník typu OR. V opačném případě je uzel neřešitelný.

Metody řešení rozkladem na podproblémy jsou vlastně metody prohledávání AND/OR grafů a můžeme je opět rozdělit na slepé (do šířky a do hloubky) a heuristické.

K efektivně řešitelným úlohám patří například:
- jednoduché hry dvou protihráčů (u kterých lze AND/OR graf sestrojit)
- symbolické integrování a derivování
- úloha hanojské věže

Příklad Hra NIM
V této hře odebírají hráči z hromádky zápalek střídavě jednu nebo dvě zápalky – prohrává ten hráč, na kterého zbyla poslední zápalka.
V našem případě nechť počáteční počet zápalek je 8 a začíná hráč A. Prohledávat budeme **do hloubky** bez stanovení mezní hloubky.

Pozn. Z pohledu hráče A stačí, aby k jeho výhře vedl alespoň jediný z jeho možných tahů, ale musí k ní vést všechny možné tahy hráče B.



Obr. 6

V příkladu je vidět, že některé uzly jsou generovány i expandovány vícekrát. To sice zvyšuje rozsah výpočtů, ale na druhé straně i procedura rozpoznávající ekvivalentní uzly může být časově náročná a navíc v algoritmu mohou vzniknout nežádoucí cykly.

3.3. Metody hraní her

K nejznámějším metodám hraní her patří Minimaxová procedura a její modifikace Alfa-beta procedura.

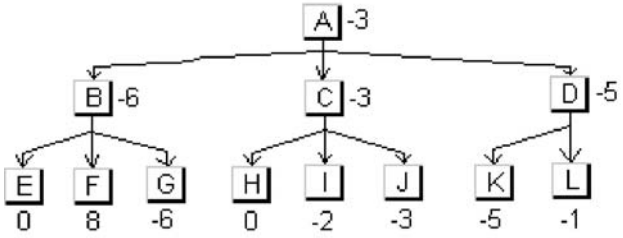
a) Minimaxová procedura

Minimax

Tato metoda je ve svém principu metodou prohledávání do hloubky s omezením hloubky prohledávání. Specifický je pro ni především způsob volby statické ohodnocovací funkce f, která se vypočítává pro daný uzel na i-té úrovni grafu využitím tohoto iterativního algoritmu:

- daný uzel se expanduje a pro všechny následovníky se určí hodnota f
- z takto určených hodnot se vybere ta nejlepší a použije se zpětně jako nové ohodnocení rodičovského uzlu na i-té úrovni

Pro účinnost algoritmu je zřejmě rozhodujícím faktorem definice „nejlepší“ hodnoty ohodnocující funkce na dané úrovni grafu popisující prohledávání. Je zřejmé, že pokud se bude jednat obecně o lichou úroveň, je nutné brát do úvahy **maximální hodnotu** ohodnocující funkce na nejbližší nižší úrovni, jde tedy o maximalizaci zisku hráče A. Naopak, bude-li se jednat o úroveň sudou, musí se určovat **minimální hodnota** ohodnocující funkce na nejbližší nižší úrovni, neboť výběr operátoru převádějícího stav ze sudé úrovně na lichou je plně v rukou protihráče B. Ten se bude samozřejmě snažit minimalizovat zisk hráče A.



Obr. 7

Na obr. 7 je zobrazeno vyhodnocování hodnot jednotlivých uzlů:

Pro uzly B,C a D, které jsou na sudé úrovni se vybírá minimální hodnota z nejbližší nižší úrovně, tedy pro B se vybere –6, pro C –3 a pro D –5. Následuje uzel A na liché úrovni a pro tento uzel se vybírá maximální hodnota z úrovně B,C,D – tedy uzel A bude mít hodnotu –3.

Alfa-beta

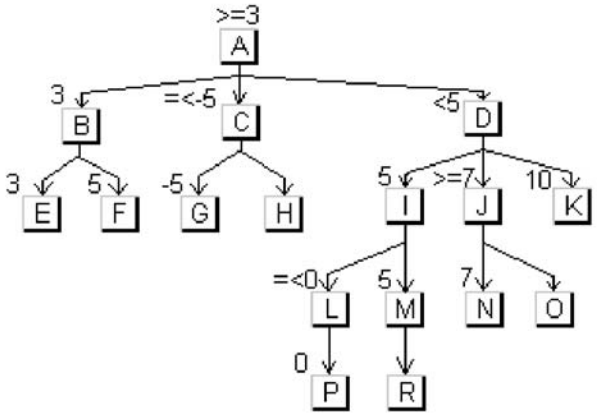
b) Alfa-beta prořezávání

Účinnost klasického prohledávání bývá umocňována technikami *branch-and-bound* (větev-mez), při nichž jsou ve velmi ranných stadiích prohledávání zavrhována řešení evidentně horší než již dosud nalezená. K zavrhování řešení je k dispozici jistá mezní, ještě akceptovatelná mez ohodnocení uzlu. V případě her dvou hráčů musí být tyto techniky schopny pracovat s dvěmi mezními hodnotami odhodnocení, tzv. **mezí** – každý hráč disponuje v daném okamžiku jistou svojí mezní hodnotou, která mu pomáhá rozhodnout. Strategie větví a mezí modifikovaná pro hru dvou hráčů se nazývá metodou prořezávání alfa-beta. Tato metoda využívá dvě mezní hodnoty:

- hodnotu **alfa** představující dolní mez ohodnocení uzlu, v němž je na tahu hráč A (tj. uzlu, ve kterém je maximalizováno ohodnocení)
- hodnotu **beta** představující horní mez ohodnocení uzlu, v němž je na tahu hráč B (tj. uzlu, ve kterém je minimalizováno ohodnocení)

Obr. 8: Byl-li ohodnocen uzel B hodnotou 3, bude hodnota uzlu A minimálně také 3 – **alfa hodnota**, protože hráč A si vybírá maximum. Zjistíme-li nyní, že hodnota uzlu G je –5 (tzn. že je menší než alfa hodnota), nemá smysl určovat hodnotu uzlu H, protože hráč B si v uzlu C určitě vybere své minimum (provede se tzv. **alfa řez**). Stejným způsobem bude zamezeno určování hodnoty uzlu R.

Po zamezení určování hodnoty uzlu R se určí hodnota uzlu M, kterou přes uzel I přeneseme až k uzlu D a označíme ji jako **hodnotu beta** (tj. beta=5). Skutečná hodnota uzlu D může být stejná nebo menší, protože hráč B si vybírá minimum. Nyní expandujeme uzel J. První z jeho bezprostředních následníků – uzel N – má hodnotu větší než je hodnota beta. Hráč B si proto nevybere uzel J (s ohodnocením minimálně 7), protože již má k dispozici výhodnější tah (uzel I) a není tedy potřeba určovat hodnotu uzlu O (provede se **beta řez**). Hodnota alfa může být pouze zvětšována (nalezne-li se podstrom s vyšším ohodnocením), hodnota beta naopak pouze snižována.



Obr. 8

Účinnost prořezávání alfa-beta do značné míry závisí na pořadí, v němž jsou větve stromu expandovány. Pokud jsou náhodou přednostně vyšetřovány nejhorší cesty, k prořezávání nedochází. Pokud by ovšem v daném systému prohledávání do hloubky byla nejlepší cesta generována jako první, nemuselo by vůbec dojít k expanzi uzlů ležících mimo tuto cestu.

Nejdůležitější probrané pojmy:

- stavový prostor
- prohledávání do šířky
- prohledávání do hloubky
- prohledávání metodou stejných cen
- heuristické prohledávání
- metoda rozkladu na podproblémy
- metoda hraní her (minimaxová procedura, alfa-beta prořezávání)

Úkol k textu:

Porovnejte základní metody prohledávání stavového prostoru.

Korespondenční úkol:

Navrhněte, vytvořte a odlaďte program pro řešení Vámi zvoleného jednoduchého problému pomocí nejméně tří metod prohledávání stavového prostoru. Do programu integrujte také možnost měřit čas řešení všemi metodami (porovnání efektivity). Vámi zvolený problém nejprve konzultujte s tutorem.
Příklady:

- vybraná hra (tic-tac-toe, hra kameny z příkladu atd.)
- vybraný optimalizační problém (sestavení rozvrhu hodin, hledání cesty atd.)

4. Plánování

Cíl:

Po prostudování této kapitoly si uvědomíte:
- co je problém plánování

Seznámíte se s:
- metodami plánování

Co vlastně pojem **plánování** představuje? Dalo by se říci, že každý z nás vědomě či nevědomě se s ním dnes a denně setkává. Představme si kupříkladu studenta, který se přihlašuje na zkoušku ve zkouškovém období. Musí brát na zřetel eventuální možné termíny, jejich obsazení, své splněné či nesplněné předpoklady k ukončení daného předmětu, stejně tak jako své vědomostní schopnosti a celkovou připravenost, aby věděl, kdy bez problémů dokáže danou zkoušku vykonat.

Plánování

Takže na začátku je zapotřebí si důkladně promyslet (**naplánovat**) určitou posloupnost akcí (např. u našeho studenta splnění jeho úkolů, nastudování teorie apod.), abychom posléze mohli zdárně dospět k požadovanému výsledku – v případě onoho studenta k úspěšnému vykonání jeho zkoušky.

Pro člověka je otázka **plánování** nepříliš složitá, samozřejmá, téměř každodenní činnost. Ale z hlediska **umělé inteligence** představuje jeden z problémů, které bychom mohli zařadit mezi tzv. obtížné úlohy.

Při řešení úloh, které představují vlastně klasické metody umělé inteligence, se používají různé způsoby prohledávání stavového prostoru. V něm každý bod odpovídá jedné možné situaci, která se může ve skutečném reálném světě vyskytnout. Systémy umělé inteligence hledají vhodnou posloupnost akcí (nazvanou **plánem**), díky které lze přejít od počátečního stavu do stavu koncového.

1	2	3
8		4
7	6	5

obr. 1 - Cílový stav dané hry - stav jediný

U jednoduchých úloh typu Problém 8 (viz. obr. 1), kdy je třeba dosáhnout cílového stavu postupným posouváním kamenů, je dostatečně vyhovující **algoritmus A*–** *algoritmus uspořádaného prohledávání*. U komplikovanějších typů úloh toto již nestačí.

4.1. Zjednodušení složitějších úloh

Máme-li úlohu poněkud složitější, je důležité mít možnost si daný problém rozdělit na dílčí části. Pracujeme pak s menšími úlohami, jejichž výsledky nakonec zkombinujeme do jednoho celkového řešení. V případě, že nemáme možnost takového dílčího rozdělení, se naše úloha stane v podstatě neřešitelná. Získáme příliš velké množství dat, příp. podmínek, které je nutno v každém kroku uvažovat. Je tedy nutné úlohu určitým způsobem zjednodušit. K tomu nám dopomůžou dva způsoby.

První možností, jak toho dosáhnout, je přestat přepočítávat celý stav úlohy na stav nový a to při každém kroku ve stavovém prostoru. Namísto toho bychom měli jenom zaznamenávat změny stavu. Představme si příklad z praxe. Vstoupíme-li do kuchyně, abychom připravili oběd, pak touto naší akcí se nezmění rozmístění spotřebičů v kuchyni ani poloha kuchyně vůči bytu. Ale vstupem do kuchyně (pojmenování naší akce) může být změněna naše poloha např. vůči kuchyňské lince. Změn samozřejmě bude více, začneme-li vařit. Ty složky stavu řešené úlohy, které jsou příslušnou akcí změněny, přesně popisuje **rámec akce**. A právě s rostoucí složitostí stavu úlohy se zvyšuje důležitost mechanizmu rámců.

Vezměme si na pomoc další příklad. Takový obyčejný úklid domácnosti (např. vysávání koberců) rovněž představuje značně složitou situaci. Stav této úlohy musí zaznamenávat nejen polohu a vzájemné vztahy všech předmětů v blízkém okolí, ale i lokalizaci dotyčné osoby. Ovšem činnost tohoto člověka ovlivňuje pouze velmi malou část tohoto celkového stavu. Dá se předpokládat, že se nezmění poloha dveří, oken, skříní a větších kusů nábytku vůči celému bytu, ale třeba umístění vysavače (příp. židlí či jiných menších kusů nábytku) dozná jistě určitých změn. Při použití rámců se budeme snažit zachytit změny pouze těch složek stavu úlohy, které budou ovlivněny příslušnou akcí, přičemž ostatní složky stavu je možné považovat za neměnné.

Dekompozice na jednodušší podúlohy

Dekompozice

Nyní jsme se dostali ke druhé možnosti, jak si usnadnit řešení složitých úloh. Jedná se o rozklad (**dekompozici**) daného problému na skupinu menších jednodušších podúloh. Ve většině případů se jedná o podúlohy skoro nezávislé, mezi kterými existuje relativně málo vzájemných interakcí. Rozklad na úplně samostatné a vzájemně nezávislé podúlohy bychom našli jen ztěží.

Ukažme si pro názornost příklad dekompozice. Máme vytvořit program pro práci se zásobníkem. Bylo by jistě složité vše řešit v hlavním programu dohromady. Zde použitá dekompozice znamená vytvořit procedury (podprogramy), které vykonávají jednodušší činnosti. Může se jednat o zjištění, zda je zásobník prázdný nebo plný, uložení prvku do zásobníku, případně vybírání prvku ze zásobníku. Díky tomu se stane hlavní program přehlednějším. A co teprve bude-li zapotřebí vícekrát využít těchto procedur. Tady jistě oceníme možnost dekompozice.

Abychom mohli vůbec s podobnými rozložitelnými úlohami pracovat, je nutné znát metody (dekompozice) umožňující řešit jednotlivé podúlohy samostatně a zároveň zaznamenat do celkového řešení i přehled možných interakcí. Použití těchto metod se nazývá **plánování**. Jak už bylo uvedeno, plánování je pouze promyšlení určité posloupnosti kroků potřebných k dosažení zvoleného cíle, aniž by bylo třeba jakýkoliv krok během plánování uskutečnit.

Plánovací systém

U systémů pro řešení úloh, které jsou založeny na elementárních postupech, musíme nutně provádět každou z těchto funkcí :

- *vybrat nejvhodnější pravidlo příštího kroku na základě dostupné heuristické informace*

Nejpoužívanější postup k výběru vhodného pravidla je postup, kdy se nejprve určí množina rozdílů mezi požadovaným stavem cílovým a stavem aktuálním. Potom se identifikují pravidla významná pro redukci těchto rozdílů. Bude-li nalezeno takových pravidel více, lze použít pro výběr mezi nimi dostupné heuristické informace. Ty sice nebývají podloženy hlubší teorií, ale velmi často pomáhají účinně nalézt řešení, ačkoli obecně jej negarantují.

- *aplikovat vybrané pravidlo a vypočíst nový stav úlohy vycházející z použití tohoto pravidla*

U jednoduchých systémů popisujících omezený rozsah světa by byla aplikace pravidel snadná. Stačí specifikovat stav úlohy, který vyplynul z jeho použití. Chceme-li pracovat s pravidly, která specifikují jen malou část úplného stavu úlohy, je třeba použít jiný způsob, jak toho dosáhnout. Jedna z možností je pro každou akci popsat všechny změny, které tato akce způsobuje na stavovém popisu, přičemž všechno ostatní zůstává fixní.

Další zajímavé řešení lze nalézt v následující podkapitole (STRIPS – plánování se zásobníkem cílů).

- *zjistit, zda bylo nalezeno řešení*

Za úspěšné je považováno to řešení plánovací úlohy, kdy je nalezena posloupnost operátorů, která transformuje počáteční stav úlohy na stav cílový. Aby systém mohl tuto situaci rozpoznat, je zapotřebí přímá unifikace stavových popisů. Jsou-li však úplné stavové popisy popsány množinou požadovaných vlastností, je tento problém podstatně složitější.

- *najít slepé cesty, které je možno vynechat, aby se systém mohl ubírat přijatelnějším směrem*

V tomto bodě se jedná o tentýž mechanismus usuzování jako pro detekci

správného řešení. Plánovací systém musí být schopen zjistit, zda zkoumaná cesta není k nalezení řešení nemožná, resp. nepravděpodobná.

- *detekovat, zda nebylo nalezeno téměř správné řešení, a pokud ano, tak použít zvláštních technik za účelem získání zcela správného řešení*

V případě, že nebylo nalezeno to správné řešení, máme k dispozici celou řadu akcí k získání zcela správného řešení. Úplně nejjednodušší možností je zapomenout na právě provedené úkony a pustit se do hledání jiného řešení. Tento směr je ale značně neefektivní a nemusí vždy vést ke správným výsledkům. Další lepším přístupem je rozbor situace, která vznikne provedením posloupnosti operátorů odpovídajících návrhovému řešení a tu porovnat s žádaným cílem. V každém případě (má-li navržené řešení alespoň nějaký smysl) bude rozdíl mezi těmito dvěma novými stavy menší než rozdíl mezi počátečním a cílovým. Třetí možnou variantou opravy skoro správného řešení je přímo využít znalosti o tom, co je špatně a posléze tuto nesrovnalost dát do pořádku. Jako nejúčinnější způsob opravy se jeví ta skutečnost, že vlastně se nic neopraví. Tudíž ponecháme neúplně specifikována řešení až do posledního možného okamžiku. Dalším postupem lze získat mnohem více doplňujících informací, kterých můžeme využít a dokončit specifikaci tak, aby nenastal žádný spor.

4.2. Systém STRIPS

STRIPS

Zásobník cílů byl použit jako jedna z prvních technik vyvinutá k řešení složitých cílů s možnou interakcí. Tohoto přístupu využívá **systém** Nilssonův a Fikesův **STRIPS** (*Standford Research Institute Problem Solver*), jenž vznikl jako výsledek robotického projektu. Systém STRIPS sám zjišťuje diference mezi stavy, interpretuje je jako cíle a sám vybírá vhodná pravidla. Stavy jsou v systému STRIPS formulovány jako množiny formulí predikátové logiky. Každý takový operátor je tvořen trojicí, kde první člen (tzv. **PRECONDITION-seznam**) udává podmínku aplikovatelnosti operátoru. Druhým členem je množina formulí (tzv. **DELETE-seznam**), která má být ze stávající množiny vypuštěna. Člen třetí (tzv. **ADD-seznam**) představuje množinu formulí, které mají být přidány.

Funkce systému spočívá v tom, že uživatel zadá pouze počáteční množinu platných formulí, cílovou formuli a množinu operátorů. Je-li cílová množina obsažena v počáteční množině, je úloha vyřešena. U netriviálních úloh jsou hledány diference mezi množinou platných formulí a cílovou formulí, přičemž je hledán vhodný operátor k odstranění některé z diferencí. Při jeho nalezení je podmínka operátoru aplikovatelnosti považována za nový podcíl, vzhledem k němuž se celý přístup opakuje.

I když systémy STRIPS (i jiné obecné systémy např. GPS, PLANNER) nedosáhly praktického rozšíření, jsou dodnes důležitým teoretickým zdrojem prací v oblasti řešení úloh a poskytují teoretické zdůvodnění či teoretický základ pro konstrukci modernějších systémů umělé inteligence. Kladou však obvykle mnohem větší důraz na využívání vysoce speciálních znalostí.

4.3. Plánování nelineární a hierarchické

Předcházející metoda plánování se zásobníkem cílů řeší dané úlohy tím způsobem, že je vyřizuje jeden po druhém. Plán generovaný touto metodou obsahuje posloupnost operátorů pro dosažení prvního cíle s následnou úplnou posloupností pro druhý cíl atd. Ve většině úloh je však zapotřebí komplikovanějšího plánu, kdy na násobných podúlohách se pracuje souběžně. A právě takové plánování se nazývá **plánování nelineární**. Zde výsledná sekvence akcí se nevytváří jako lineární řetězení dílčích plánů určených k úplnému řešení jednotlivých podúloh.

V Sussmanově plánovacím systému HACKER bylo nastíněno mnoho základních myšlenek o nelineárním plánování. Ale prvním takovým skutečným nelineárním plánovacím systémem se stal až NOAH, který byl dále vylepšen programem NONLIN. Následující plánovací systémy (např. MOLGEN a TWEAK) používaly jako svoji ústřední metodu tzv. **techniku přídavných omezení**. Hlavní myšlenkou tohoto omezení je inkrementální tvorba **hypotetického plánu**. Ten je složen z elementárních posloupností operátorů, kdy jsou kladena určitá omezení na jejich pořadí, a argumenty operátorů se váží na konkrétní hodnoty, resp. množiny hodnot. Za hypotetický plán je považována částečně uspořádaná a částečně na konkrétní hodnoty vázaná posloupnost operátorů. Skutečný plán lze potom vygenerovat takovým způsobem, že se převede toto částečné uspořádání na jedno z přípustných úplných uspořádání.

V nelineárním plánování je použito několik heuristických pravidel. Jsou to kroky, které byly převzaté ze systému TWEAK, přičemž první čtyři kroky bývají často nazývány **operacemi pro modifikace plánu**. Tyto heuristiky ve většině případů dostačují pro řešení libovolné úlohy nelineárního plánování. Ovšem výjimečně je nutno přidat i krok pátý. Tedy heuristiky systému TWEAK jsou :

- *přidání kroku*

Vytvoří se nové kroky plánu. Je to zcela základní metoda použitá již v systému GPS

- *předsunutí*

Zde se provede vazba jednoho kroku na jiný takovým způsobem, aby v konečném plánu předcházel kroku jinému.

- *Přizpůsobení*

Mezi dva kroky se vloží pomocný krok tak, že tento zabezpečí splnění takových předpokladů původně druhého kroku, které byly porušeny původním prvním krokem. Vkládání operací pomocí **přizpůsobování** bylo poprvé použito v plánovacím systému NOAH a později i v NONLIN.

- *Přiřazení*

Přiřadí se hodnota nějaké proměnné k zajištění předpokladů pro uskutečnění nějakého kroku.

- *Inhibice*

Jedná se o zábranu v přiřazení některých hodnot určité proměnné.

Pro řešení jakékoli řešitelné úlohy nelineárního plánování se jeví jako postačující výše zmiňovaných pět operací plánovacího systému TWEAK. Pro využití operací modifikace plánu byl sestaven jednoduchý algoritmus nelineárního plánování :

1. *Inicializovat množinu výroků V, které popisují cílový stav.*
2. *Vybrat z ní některý nesplněný výrok P a odstranit ho z množiny výroků V.*
3. *Splnit P použitím modifikačních operací (přidání kroku, předsunutí, přizpůsobení, přiřazení a inhibice)*
4. *Prozkoumat všechny kroky plánu (i ty zavedené v bodě 3) a zjistit, které z jejich předpokladů nejsou splněny. Posléze nesplněné předpoklady přidáme do množiny V.*
5. *Pokud ještě není množina V prázdná, pokračujeme bodem 2.*
6. *Nyní již dokončíme plán převedením částečného uspořádání kroků na úplně uspořádanou posloupnost kroků a je-li potřeba, inicializujeme všechny proměnné.*

Systém TWEAK – systém s prokazatelně správnou činností – byl pozoruhodným úspěchem formální teorie umělé inteligence. Objasnil mnohé komplikované a nepřesné pojmy používané dřívějšími plánovacími systémy a zpřístupnil tak spolehlivý plánovací systém.

Hierarchické plánování

Jedná-li se o těžkou úlohu, bude muset pro její vyřešení plánovací systém generovat dlouhé plány. Pro efektivitu této činnosti je třeba, aby byl systém schopný eliminovat některé detaily úlohy do té doby, dokud není nalezeno řešení vztahující se na hlavní otázky. Pak může být učiněn pokus upřesnit patřičné detaily. Dřívější pokusy v tomto ohledu využívaly **makrooperátorů**, kdy velké operátory byly složeny z menších. Tímto přístupem však ve skutečnosti žádné detaily nebyly eliminovány. Pouze se skryly uvnitř makrooperací. Mnohem lepší přístup umožňoval až systém ABSTRIPS, jenž v podstatě vytvářel plány v hierarchii **úrovní abstrakce**. Zde byly požadavky z nižší abstrakce ignorovány.

Celkový přístup systému ABSTRIPS byl přesně takový, jaký můžeme pozorovat u systému STRIPS. Rozdíl spočívá v tom, že se ignorují takové předpoklady, které mají nižší **ohodnocení důležitosti** než maximální (nejvyšší úroveň detailů). Je-li takový plán vytvořen, je použit jako náčrt plánu úplného a nato uvažujeme předpoklady na nejblíže nižší úrovni ohodnocení důležitosti. Posléze rozšíříme stávající plán o operátory splňující tyto předpoklady, přičemž znovu ignorujeme při výběru operátorů všechny předpoklady, které mají opět nižší ohodnocení důležitosti než je momentálně uvažovaná úroveň. Dále pokračujeme tímto způsobem k nižším a nižším úrovním ohodnocení důležitosti, dokud nebudou zahrnuty všechny předpoklady původních pravidel.

Protože tento postup zkoumá celý plán nejprve na jedné úrovni detailů a až teprve pak se obrací k detailům nižší úrovně, byl pojmenován **prohledáváním do délky**. Vůbec nejpodstatnější pro úspěch této hierarchické metody plánování je určení správných ohodnocení důležitosti jednotlivých předpokladů, přičemž nejkritičtější jsou předpoklady nesplnitelné žádnými operátory.

4.4. Reakční systémy a jiné techniky

Dostali jsme se ke zcela odlišnému způsobu, jak přistoupit k problému rozhodování o tom, co má být provedeno. Dosavadní plánovací systémy popisovaly cílené plánovací procesy, kdy před vykonáním každé akce předcházelo vytvoření plánu pro realizaci celé úlohy. Ale **reakční systémy** disponují zcela jiným pohledem na plánovací proces jako takový.

Základní ideou reakčních systémů je vyhnout se úplnému a komplikovanému plánování. Zde se bere v úvahu pouze pozorovaná momentální situace, k níž přísluší příslušná reakce. Aby bylo zřejmé jaké akce mají být podniknuty za daných okolností, musí mít reakční systém přístup k nějaké **bázi znalostí**, která je obvykle implementována, spravována a udržována jako samostatný soubor. Báze znalostí představuje vlastně znalosti převzaté od experta (včetně znalostí neurčitých).

Reakční systém se tedy značně odlišuje od ostatních druhů plánovacích systémů. Nesnaží se predikovat a nebuduje celou posloupnost akcí dříve než udělá první krok. Je zřejmé, že právě reakční systémy jsou schopny překvapivě složitého chování, obzvláště v úlohách z reálného světa.

Hlavní výhodou reakčních systémů s porovnáním s tradičními plánovacími systémy je, že dokáží pracovat robustně v oblastech, kde je úplné a přesné modelování obtížné. Reakční systémy se dovedou obejít bez úplného modelování a zakládají své akce přímo na svém vnímání reálného světa. Další výhodou reakčních systémů je vysoká rychlost jejich odezvy a právě proto jsou vhodné pro úlohy v reálném čase (např. automatizované řízení vozidel).

V průběhu vývoje plánovacích systémů bylo zkoumáno a vytvořeno značné množství nejrůznějších principů a technik. Výše zmiňované techniky patří ke klasickým metodám v umělé inteligenci. Ale existují i jiné způsoby řešení plánovacích úloh :

- *trojúhelníkové tabulky (Triangle tables)*

Umožňují zaznamenávání cílů, které může každý operátor splnit, a cílů, které naopak musí být splněny, aby mohl být operátor správně vykonán. Stane-li se během vykonávání plánu něco neočekávaného, tabulka poskytne potřebné informace pro nápravu plánu.

- *metaplánování (Metaplanning)*

Tato technika uvažuje nejen o řešené úloze, ale i o samotném plánovacím postupu.

- *makrooperátory (Macro-operators)*

Dovolují plánovači vytvořit nové operace reprezentující časté sekvence operátorů.

- *case-based planning*

Toto je plánování založené na znalosti konkrétních případů. K tvorbě nových plánů jsou použity staré ověřené plány.

Plánovací činnosti a rozvrhování jsou výsadou inteligentních živých bytostí. Samozřejmě kdo jiný než člověk je ze všech takových tvorů na tom nejvyšším stupni inteligence. I když jsou i takoví zástupci svého druhu, o kterých lze jistě hovořit jako o tvorech inteligentních (např. pes, delfín). Ale rozhodně nikdo jiný než člověk nedokáže natolik využít svých znalostí, aby uměl „vdechnout“ jistou dávku inteligence i neživému objektu. Opět se dostáváme k pojmu **umělá inteligence**, tedy něco, co se svým umem (myšleno v oblasti plánování) přibližuje člověku. Úloha plánování je tudíž považována jako netriviální problém **umělé inteligence**.

Nejdůležitější probrané pojmy:

- plánování
- plánovací systém
- rámec akce
- dekompozice
- STRIPS
- nelineární plánování
- hierarchické plánování
- reakční systémy

Úkol k textu:

Jak souvisí plánování s metodami prohledávání stavového prostoru?

Korespondenční úkol:

Najděte na Internetu vybraný systém pro plánování, stručně jej popište a vytvořte v něm řešení zvoleného plánovacího problému. Systém i problém konzultujte s tutorem.

5. Expertní systémy

Cíl:

Po prostudování této kapitoly pochopíte:

- Co je to expertní systém
- Jakou má architekturu
- Kde se používají tyto systémy

Expertní systémy mají poměrně bohatou historii. Jejich nasazení je ve velkém spektru problémů z průmyslu, zdravotnictví a dalších oblastí. Jejich pojmenování vychází ze snahy simulovat chování experta při určité činnosti. Například může jít o práci lékaře-diagnostika, který má podle příznaků pacienta stanovit diagnózu.

5.1. Charakteristika expertního systému

Expertní systém

Expertní systémy jsou počítačové programy, simulující rozhodovací činnost experta při řešení složitých úloh a využívající vhodně zakódovaných, explicitně vyjádřených znalostí, převzatých od experta, s cílem dosáhnout ve zvolené problémové oblasti kvality rozhodování na úrovni experta (Feigenbaum - 1988).
Další definice postuluje, že expertní systém je počítačový systém hledající řešení problému v rozsahu určitého souboru tvrzení nebo jistého seskupení znalostí, které byly formulované experty pro danou specifickou oblast .

V prvním období výzkumu v oblasti expertních systémů se kladl důraz na kvalitu rozhodování na úrovni experta; tomu ostatně napovídá už sám název expertní systémy. V polovině 70.let totiž převládal (optimistický) názor, že bude možné vytvořit „instantního experta“, které by po nahrání na počítač byl schopen řešit i ty nejsložitější úlohy v dané oblasti. Tato představa narazila jak na meze samotných systémů (expertní systémy řešily úspěšně spíše rutinní problémy), tak i na skepsi uživatelů („přece nebudu poslouchat nějaký počítač“). S jistou nedůvěrou uživatelů souvisejí i etické a právní otázky používání expertních systémů; kdo ponese odpovědnost za chybná rozhodnutí učiněné na základě doporučení expertního systému. Proto se postupně slevuje z pohledu na roli expertního systému v procesu rozhodování. Z původně zamýšlené (a deklarované) role experta, tedy někoho, kdo ví více než uživatel (a uživatel by tedy měl slepě poslechnout) se přechází k chápání expertního systému jako kolegy uživatele (tedy někoho, kdo ví přibližně stejně jako uživatel, ale jehož výkonnost nepodléhá stresu, vlivem okolí apod.) nebo dokonce jako „pouhého“ asistenta (tedy někoho, komu uživatel-expert svěřuje rutinní úlohy, aby se mohl plně soustředit pouze na komplikované případy). S tímto posuvem v pohledu na roli expertních systémů souvisí i posuv v terminologii. Dnes se spíše hovoří o znalostních systémech; důraz se tedy

klade na využívání vhodně zakódovaných speciálních znalostí převzatých od experta.

Znalostní systém

Znalostní systém (*knowledge-based system*) je podle staršího pojetí obecnější pojem než expertní systém. Expertní systémy tedy lze chápat jako zvláštní typ znalostních systémů, který se vyznačuje používáním expertních znalostí a některými dalšími rysy, jako je např. vysvětlovací mechanismus.
V poslední době dochází ke stírání rozdílů mezi těmito pojmy.
Expertní systémy tvoří znalostní inženýři, kteří společně s odborníky v daném oboru studují chování reálného systému. Snaží se formulovat pravidla a zákonitosti, aby je mohli nadeklarovat v podobě pravidel, kterými se bude řídit budovaný systém.

Charakteristické rysy expertních systémů

1. Oddělení znalostí a mechanismu pro jejich využívání

Znalosti experta jsou uloženy v bázi znalostí odděleně od inferenčního mechanismu. To umožňuje vytvářet problémově nezávislé (prázdné) expertní systémy (expert system shells), kde jeden inferenční mechanismus může pracovat s různými bázemi znalostí. Předem je dána pouze strategie využívání znalostí z této báze - řídící mechanismus neboli inferenční mechanismus.

2. Neurčitost (nejistota) v bázi znalostí a neurčitost (nejistota) v datech

Způsob zpracování znalostí a dat v expertním systému musí mít některé rysy podobné způsoby uvažování experta. Expert pracuje velmi často s nejistými znalostmi - heuristikami - a s nejistými daty (např. se subjektivním popisem potíží pacienta, s výsledky přibližných metod vyhodnocování apod.). I expertní systém musí být schopen využívat nejistých znalostí, tj. znalostí s přidělenou mírou důvěry v jejich platnost - nejistota v bázi znalostí. Zde se pak objevují pojmy jako „často“ „většinou“, které je potřeba kvantifikovat (např. v nějaké škále od „určitě ano“ přes „nevím“ až k „určitě ne“). Takovou znalostí s neurčitostí může například být „jestliže má pacient teplotu, obvykle je mu předepsán acylpyrin"; pacient totiž „často" má teplotu v důsledku chřipky, ale „někdy“ může mít teplotu, protože má zlomenou nohu. Systém také musí umět využívat odpovědí, zahrnujících nejistotu uživatele, způsobenou nepřesně určenými hodnotami nebo subjektivním pohledem uživatele (odpovědi typu "nevím", "asi ano" apod.) - nejistota v bázi dat.

3. Dialogový režim a báze dat

Expertní systémy jsou nejčastěji konstruovány jako tzv. konzultační systémy. Uživatel komunikuje se systémem způsobem „dotaz systému - odpověď uživatele“ obdobně, jako s lidským expertem. Tento systém práce byl do značné míry usnadněn nástupem osobních počítačů a s tím souvisejícím přechodem od dávkového zpracování na sálech výpočetních

středisek k interaktivnímu zpracování přímo na psacím stole uživatele. Báze znalostí popisuje znalosti z dané oblasti. Má tedy charakter obecného rozhodovacího pravidla (či systému rozhodovacích pravidel). Řešit konkrétní případ znamená "dosadit" data o daném případu do obecně formulovaných znalostí z báze znalostí. Data k danému konkrétnímu případu poskytuje obvykle uživatel sekvenčně, v dialogovém režimu s počítačem. Dialog uživatele s počítačem má charakter dialogu laika či méně zkušeného odborníka s expertem. Expertní systém se uživatele dotazuje na údaje, týkající se konzultovaného případu, a na základě odpovědí a svých obecných znalostí si postupně upřesňuje "představu" o případu a dochází k závěru, eventuálnímu řešení.

4. Vysvětlovací činnost

Aby se zvýšila důvěra uživatelů v závěry a doporučení expertního systému, měl by systém poskytovat vysvětlení svého uvažování. Obvykle systém vysvětluje právě položený dotaz, znalosti relevantní k nějakému tvrzení, právě zkoumanou cílovou hypotézu, právě probíhající odvozování.

5. Modularita a transparentnost báze znalostí

Pro účinnost expertního systému je rozhodující kvalita báze znalostí. Znalosti experta nemají statický charakter, nýbrž se postupně (a obvykle nikoliv pomalým tempem) vyvíjejí a rozrůstají. Modularita umožňuje snadnou aktualizaci báze znalostí, transparentnost umožňuje její snadnou čitelnost a srozumitelnost. Samotné vytváření báze znalostí probíhá iterativním způsobem při opakovaných konzultacích experta z dané problémové oblasti s odborníkem na tvorbu bází, tzv. znalostním inženýrem, kdy je báze znalostí postupně „laděna“, až chování expertního systému (alespoň při konzultaci pro vzorové příklady) odpovídá představám experta.

5.2. Typy úloh

Expertní systémy se hodí pro řešení úloh různých typů. Pokusme se nyní základní typy shrnout:

Diagnóza – proces nalezení chyb či chybných funkcí systému (např. MYCIN nebo INTERNIST – systém pro oblast vnitřního lékařství; problémy: dysfunkce mohou být maskovány symptomy jiných dysfunkcí, dysfunkce mohou být jen občasné, některá data mohou být nepřístupná, chování systému nemusí být plně známo.

Interpretace – analýza dat s cílem určení jejich významu (např. DENDRAL, PROSPECTOR; problémy: data mohou být chybná, chybějící, cizorodá, zatížená šumem).

Monitorování – průběžná interpretace signálů a dat a určení okamžiku, kdy je nutná intervence (např. VM-systém, který v reálném čase monitoroval

pacienta napojeného na „umělé plíce“; problémy: nebezpečí falešných alarmů).

Plánování – nalezení posloupnosti akcí k dosažení cíle (např. MOLGEN – systém pro plánování experimentů v molekulární genetice; problémy: plánovací problémy jsou obvykle široké, komplikované a nedostatečně strukturované; ten kdo plánuje nemusí vždy chápat možné důsledky plánovaných akcí).

Návrh (design) – vytváření konfigurací objektů vyhovujících daným podmínkám (např. R1/XCON; problémy: podobné jako při plánování, navíc hraje roli prostorové uspořádání).

Predikce – předpověď běhu budoucích událostí na základě modelu minulosti a současnosti (např. GLAUCOMA – systém pro predikci vývoje šedého zákalu; problémy: predikce vyžaduje uvažování o čase a v čase).

Při nejjednodušším dělení vystačíme se dvěma základními skupinami – s expertními systémy diagnostickými (které řeší diagnózu, interpretaci a monitorování) a expertními systémy generativními (které řeší plánování, návrh a predikci). Diagnostické expertní systémy pracují s pevným počtem cílů (diagnóz, hypotéz), ze kterých vybírají svá doporučení. Naproti tomu generativní expertní systémy si hypotézy generují dynamicky až v průběhu konzultace. Generativní systémy tedy na rozdíl od diagnostických mění „stav světa“, se kterým pracují. Zatím převažují diagnostické systémy. Je to dáno jednak povahou problémů, kterou typicky řeší lidé – experti, jednak i tím, že vývoj diagnostické aplikace je relativně jednodušší.

Diagnostické úlohy

Diagnostický proces lze chápat jako získávání a interpretaci informací relevantních pro potvrzení přítomnosti nebo nepřítomnosti nějaké závady v systému. Cyklus stanovení diagnózy je tvořen třemi kroky: formulování hypotézy, testování hypotézy a přijetí nebo zamítnutí hypotézy. Konceptuální model diagnostické úlohy se opírá o možné typy znalostí použitých v dané aplikaci. Použité znalosti mohou mít charakter:

- popisu normálního chování systému
- popisu abnormálního chování systému
- výčtu závad a seznamu příznaků pro každou závadu (bez explicitních znalostí o chování systému)
- seznamu příznaků pro normální situaci

V reálných aplikacích se výše uvedené znalosti různě prolínají, při uvažování různých konceptuálních modelů je ale vhodné mezi nimi rozlišovat:

- diagnostikování odchylky od normálu
- diagnostikování porovnáním abnormálního chování
- diagnostikování klasifikováním abnormality

Generativní úlohy
Plánování, tedy schopnost sestavit sekvenci akcí, která povede k dosažení požadovaného cíle, patří v umělé inteligenci k tradičním oblastem zájmu. Pohled na generativní úlohy (a plánování v UI obecně) se postupně vyvíjel. V prvním období bylo plánování chápáno jako dokazování teorémů nebo prohledávání. Dnes je plánování spíše chápáno jako splňování omezení, kladených na požadované řešení úlohy. Požadavky na chování systému se porovnávají s chováním navrhovaného řešení. V případě, že řešení nevyhovuje, musí se modifikovat. V situacích, kdy je k dispozici dostatek znalostí pro stanovení, co se má v dané chvíli provést, se postupuje způsobem „navrhni a aplikuj". Klade se zde důraz na postupné rozšiřování dílčího řešení, a lze tedy postupovat poměrně přímočaře. V situacích, kdy dostupné znalosti jsou neúplné, nebo kdy existuje řada suboptimálních řešení, se strategie poněkud mění. Postupuje se způsobem „navrhni a reviduj", který umožňuje navrácení k předcházejícím variantám řešení.
Řešení je tedy u generativních úloh vytvářeno (poskládáno) z dílčích komponent.

5.3. Typy architektur

Architektura expertního systému umožňuje rovněž rozdělit typy expertních systémů:

Diagnostické expertní systémy
Jejich úlohou je provádět efektivní interpretaci dat s cílem určit, která z hypotéz z předem stanovené konečné množiny cílových hypotéz nejlépe koresponduje s reálnými daty týkající se daného konkrétního případu. U diagnostických systémů tedy řešení případu probíhá formou postupného ohodnocování a přehodnocování dílčích a cílových hypotéz v rámci víceméně pevně daného vnitřního modelu řešeného problému, který je předem navržen expertem.
Jádrem tohoto systému je řídící mechanismus, který využitím báze znalostí a báze dat po každé odpovědi upřesňuje aktuální model konzultovaného případu. Řídící mechanismus je odpovědný za výběr dotazu, od jehož zodpovězení očekává největší přínos k upřesnění aktuálního modelu, a za úpravu aktuálního modelu po obdržení odpovědi. Báze dat může být obecně tvořena jak přímými odpověďmi uživatele, tak i hodnotami automaticky odečtenými z měřících přístrojů.

Plánovací expertní systémy
Jsou jimi obvykle řešeny takové úlohy, kdy je znám cíl řešení a počáteční stav a systém má využitím dat o konkrétním řešeném případu nalézt posloupnost kroků (operátorů), kterými lze cíle dosáhnout. Podstatnou částí takovýchto expertních systémů je generátor možných řešení, který automaticky kombinuje posloupnost kroků. Lze ukázat, že s rostoucím počtem operátorů, a především s nutným počtem kroků řešení, roste velmi rychle počet kombinací při vytváření posloupnosti kroků – hovoří se o

„kombinatorické explozi“. Znalost experta i data o reálném případu jsou v plánovacích expertních systémech užívány k výraznému omezení kombinatorické exploze zkoumaných řešení, navrhovaných generátorem. Výsledkem činnosti plánovacího systému je seznam navrhovaných řešení, obvykle ohodnocený jistou měrou optimality, k jejímuž výpočtu se též užívá údajů z báze dat. Tento seznam je uchováván v zásobníku vhodných řešení, který se v průběhu řešení úlohy dynamicky mění.

Hybridní systémy
Vyznačují se kombinovanou architekturou, poněvadž částečně využívají principů diagnostických, částečně plánovacích systémů. K hybridním systémům řadíme například inteligentní výukové systémy či systémy monitorovací. Od samého počátku rozvoje expertních systémů byla zdůrazňována opakovatelná využitelnost vyvíjených expertních systémů či jejich komponent. Vysoká znovupoužitelnost je patrná zejména u obecných problémově nezávislých částí expertních systémů, tj. u řídících a vysvětlovacích mechanismů.

Prázdné expertní systémy
Expertní systémy – ať již diagnostické či plánovací – bez problémově závislých částí, tj. bez báze znalostí a bez báze dat. Doplněním báze znalostí k prázdnému systému se systém teprve orientuje na řešení příslušné problematiky. Dodáním báze dat je pak vždy řešen konkrétní případ.

5.4. Historie ES

Expertní systémy jsou dnes užívány k řešení nejrůznějších úloh, ať již charakteru analytického (klasifikace, interpretace dat, porozumění složitým signálům, identifikace systémů, lékařská a technická diagnostika), syntetizujícího (plánování, technické návrhy, návrhy terapie v medicíně, automatická syntéza programů) či smíšeného (aplikace při výuce, monitorování). Jednotlivým typům úloh odpovídají příslušné třídy expertních systémů: systémy diagnostické, plánovací a hybridní.
V posledních letech se objevily stovky, ale spíše tisíce, systémů v nejrozmanitějších aplikačních oblastech. Zajímavé je, že nejméně polovina aplikací je zaměřena na oblast medicíny – je tomu tak zřejmě proto, že znalosti jsou v této oblasti nejlépe strukturovány, ale možná i proto, že pod pojmem expert si lidé nejspíš vybavují představu experta-lékaře.
Je nad lidské síly všechny existující systémy katalogizovat , tj. objektivně posoudit, porovnat, správně zařadit a zhodnotit jejich přínos v té které aplikaci. V přehledech je totiž každoročně uváděno skutečně obrovské množství nových systémů. Mnohdy jsou referovány vlastně jen báze znalostí spojené s opakovatelně využívaným řídícím mechanismem jako

samostatné expertní systémy, jindy se setkáváme spíše se systémy pseudo-expertními než expertními.
Výčet jednotlivých systémů by byl značně rozsáhlý, proto je lepší se podívat do historie expertních systémů. Ukazuje se, že několik málo poměrně raných systémů ovlivnilo a ovlivňuje hlavní směry ve vývoji expertních systémů i v architektuře, reprezentaci a využívání znalostí v silně aplikačně zaměřených systémech. U téměř každé nové aplikace můžeme vystopovat přímou či nepřímou vazbu na některý z „klasických" systémů.

Ve vývoji expertních systémů můžeme zřetelně odlišit čtyři významné etapy:

- počáteční fáze (1965 – 1970): DENDRAL, MACSYMA
- etapa výzkumných prototypů (1970 – 1975): MYCIN, PROSPECTOR, HEARSAY –II
- etapa experimentálního nasazení (1975 – 1981): PUFF, SACON, ONCOCIN, HEADMED, CLOT, AL/X, HASP, INTERNIST, CADUCEUS
- etapa komerčně dostupných systémů (od roku 1981 po současnost): XCON, XSEL, DIPMETER, ADVISOR…..

Systém **MACSYMA** (Bogen, 1975) poskytuje soubor nástrojů pro manipulaci s matematickými výrazy a vzorci. Stále je hojně využíván, např. nukleárními fyziky, kteří jsou často nuceni řešit soustavy velkého počtu rovnic.
MYCIN (Buchanan, Shortliffe, 1984) patří ke skupině nejčastěji citovaným „klasických" expertních systémů. Byl vyvinut ve SRI (Standford Research Institute). Jeho úkolem je na základě jednoduše dostupných dat rychle určovat typy bakteriální infekce, kterými by mohl být nově hospitalizovaný postižen, a navrhovat vhodnou léčbu antibiotiky tak, aby se stav pacienta stabilizoval do doby, než jsou dokončena časově náročná, podrobná laboratorní vyšetření. MYCIN je významný spíše z hlediska umělé inteligence než z hlediska medicínského. Metody zpracování neurčité informace, zavedené v tomto systému, se v modifikované verzi užívají u řady nově vyvíjených systémů dodnes.
Úkolem systému **PROSPECTOR**, který byl obdobně jako MYCIN vyvinut v SRI (Duda a kol., 1978), je vyhodnocování jednoduše dostupných geologických dat s cílem rychle rozhodnout, zda v dané lokalitě provádět podstatně dražší hloubkové vrty. Typická konzultace se systémem PROSPECTOR trvá několik minut a stojí několik dolarů. Může však ušetřit několikaměsíční čekání na posouzení dat a vzorků expertem. Při prvním formálním testu se systém ukázal velice úspěšným, neboť se mu podařilo odhalit molybdenové ložisko v hodnotě sta milionu dolarů. Tento výsledek upoutal pozornost veřejnosti a přispěl k výrazné finanční podpoře výzkumu v oblasti expertních systémů jako celku. Z hlediska teoretického je významným přínosem systému PROSPECTOR pseudobaysovský model pro práci s neurčitou informací.

HEARSAY-II (Erman, 1977) byl prvním systémem, který ukázal, že počítač by v brzkém budoucnu mohl spolehlivě a rychle rozumět přirozené řeči v úzce vymezené předmětné oblasti. Stimuloval tak rozvoj výzkumu v oblasti rozpoznávání a porozumění přirozené řeči. Jeho přínos pro rozvoj teorie znalostních systémů obecně byl taktéž značný, protože v systému HEARSAY-II bylo poprvé použito architektury tabule.
Systémy **MYCIN, DENDRAL a HEARSAY** mají zcela odlišnou architekturu, nicméně jejich úspěšné prototypy prokázaly životaschopnost myšlenky expertních a znalostních systémů. Uvedené systémy začaly být posléze experimentálně aplikovány v jiných oblastech, než v těch, pro něž v těch, pro něž byly původně určeny. Cílem bylo vyzkoušet eventuální širší aplikovatelnost a opakovatelnost použití. Postupně byly od systémů MYCIN a PROSPECTOR vyčleněny a eventuálně dokončeny problémově nezávislé části, které byly už beze změny užívány jako „jádro“ v nových aplikacích expertních systémů. Tyto problémově nezávislé části, popř. doplněné algoritmy pro formalizaci a údržbu znalostí, jsou obvykle nazývány prázdné expertní systémy či pouzdra. Tak ze systému MYCIN vznikl prázdný systém EMYCIN (van Melle, 1980) s mírně odlišným modelem pro práci s neurčitou informací, ze systému PROSPECTOR pak systém KAS (Waterman, 1986) atd.

Na bázi prázdných expertních systémů byly vytvořeny další poměrně úspěšné problémově orientované experimentální expertní systémy. Zde se pro dokreslení zmiňme alespoň o několika málo z nich:
PUFF (Aikins a kol., 1983) byl realizován využitím ulity EMYCIN. Poskytuje konzultace týkající se možných příčin obstruktivních potíží dýchacích cest.
SACON (Bennet, Engelmore, 1979) poskytuje konzultace uživatelům rozsáhlého softwarového balíku MARC pro analýzu metodou konečných prvků. Využívá přístupu i architektury systému MYCIN.
Z mnoha stovek dalších systémů, které vznikly použitím systému **EMYCIN** nebo jsou založeny na architektuře systému MYCIN, jmenujme ještě některé z historicky prvních: např. **ONCOCIN** je určen k řízení léčby pacienta na jednotce onkologické péče, **HEADMED** se využívá v klinické psychofarmakologii, **CLOT** diagnostikuje srážlivost krve, **DART** byl určen k diagnostice závad počítačů atd.
AL/X (Patterson, 1981), jenž identifikuje závady na technologickém zařízení platformy pro těžbu nafty v Severním moři, vychází důsledně ze systému PROSPECTOR.
HASP (Nii a kol., 1982) využíval architektury a některých modulů systému HEARSAY-II. Byl určen k interpretaci sonarových dat s cílem lokalizovat ve vymezeném sektoru lodě, jejich typy a orientaci.
V období experimentování však vznikaly i zcela nové, svoji architekturou i rozsahem zajímavé systémy. Za zmínku stojí především **INTERNIST** (Miller a kol. 1984), který byl vyvíjen s cílem pokrýt znalostmi celou oblast interního lékařství a který je považován za jeden z vůbec nejrozsáhlejších expertních systémů v historii. Později byl modifikován a přejmenován na systém **CADUCEUS**, obsahující údajně 85% veškerých

znalostí z interního lékařství. Oba systémy se, přes problémy s jejich monstróznostmi, dodnes v lékařské praxi využívají.
První šířeji a komerčně úspěšněji nasazované systémy se objevují na počátku osmdesátých let. K velmi úspěšným a často zmiňovaným patří systémy **XCON** a **XSEL**, vycházející ze své rané verze **R1** (McDermott, 1982) a užívané firmou DEC ke zpracování objednávek zákazníků na počítače řady VAX. Pracovníci firmy DEC vyčíslili úspory dosažené díky těmto systémům na více než 10 miliónů dolarů ročně.
DIPMETER ADVISOR byl vyvinut firmou Teknowledge pro potřeby několika naftařských společností. Poskytuje radu obsluze vrtných zařízení, jak postupovat v případě problémů s hloubkovým vrtem. Náklady na vývoj systému přesáhly částku 21,5 miliónů dolarů. Uživatelské firmy však dokumentovaly rychlou návratnost této investice, měřitelnou v měsících či dokonce týdnech.

Oblast	Příklady systémů
Medicína	MYCIN, CASNET/GLAUCOMA, INTERNIST/CADUCEUS, PIP, RHEUM, PUFF, ONCOCIN, MEDICO, PROTIS, DM, CADIAG, NEUREX, VM, HEADMED, EEG, …
Chemie	DENDRAL, CRYSALIS, SECS, SYNCHEM
Genetika	MOLGEN
Geologie	PROSPECTOR
Mechanika	SACON, MECHANO
Matematika	MACSYMA, AM
Výuka	GUIDON, WHY, BLAH, SOPHIE
Právo	LEGOL, TAXMAN, LRS, MATRIM

Nejdůležitější probrané pojmy:

- expertní systém
- úlohy, architektura expertního systému
- historie používání expertních systémů

Úkol k textu:

Vyjmenujte základní typy úloh řešených ES a pokuste se sami najít příklad z praxe, kde by bylo možné tuto úlohu řešit.

Korespondenční úkol:

Vyberte si libovolný expertní systém (nejlépe volně šiřitelný), vytvořte a otestujte vlastní bázi znalostí na zvolený problém. Výběr systému a problému konzultujte s tutorem.

6. Distribuovaná umělá inteligence

Cíl:

Po prostudování této kapitoly pochopíte:

- pojem distribuované umělé inteligence
- multiagentní systémy

V dnešní době je distribuované a paralelní zpracování informace prakticky nezbytností při řešení mnoha problémových úloh. Nesetkáváme se s ním pouze v oblasti UI, ale i při programování. V následující kapitole se seznámíte se základními problémy tohoto přístupu v UI.

6.1. Distribuované řešení úloh

Sdružování jednotlivců do větších skupin je jednou z charakteristických vlastností živé hmoty. Schopnosti vzniklé skupiny výrazně převyšují prostý součet všeho, co je schopen dosáhnout každý z jejích členů. Jak je to možné? V čem spočívá tajemství tohoto zázraku? Jaké jsou metody součinnosti zúčastněných jednotlivců? Takové otázky si klade distribuovaná umělá inteligence (Distributed Artifical Intelligence – **DAI**). Zcela obecně se DAI zabývá součinností několika totožných či různorodých systémů při řešení společného problému. DAI se soustřeďuje především na problematiku reprezentace, zpracování a využití znalostí v těchto systémech a hledá vhodné modely způsobu rozhodování.

Problematika DAI se dělí na dvě oblasti:

Distribuované řešení úloh

Řeší se pevně zvolená úloha a její vhodné rozdělení znalostí mezi více modulů a efektivní sdílení informací o řešení. Důraz klade na distribuci činností v rámci pevně definované struktury, často přesně (centrálně) řízené.

Multiagentní systémy (MAS)

Multiagentní systémy

Úlohu řeší skupina volně propojených autonomních systémů spolupracujících v zájmu společného cíle. Tyto autonomní systémy se nazývají **agenti**, čímž se zdůrazňuje jejich částečná samostatnost. Přitom **agentem** může být počítačový systém, senzor, robot nebo i člověk. Multiagentní systémy kladou důraz na hledání mechanizmů zajišťujících účinnou spolupráci mezi libovolnými samostatnými agenty. Autonomnost agentů je žádoucí, vazby mezi agenty jsou volnější, flexibilita systému vyšší a vnitřní život komunity bohatší.

Multiagentní systémy jako technické řešení těží z těchto výhod týmové práce:

- zkrácení doby řešení (vzhledem na paralelní postup),
- snížení nároků na komunikaci (předávání pouze závěrů),
- zvýšení operativnosti a spolehlivosti (rozšiřitelnost týmu, vzájemná zastupitelnost).

Podle toho, zda a na jaké úrovni je schopen zvažovat rozličné varianty řešení svého cíle rozlišujeme:

- **Reaktivní agent** vždy pouze reaguje na podněty z vnějšího světa. Zná pouze omezenou množinu akcí, na něž je schopen reagovat. Upřesňuje svůj obraz světa na základě vnějších podnětů. Své výsledky může sdělovat ostatním. Za jistých okolností mu hrozí zacyklení. Typickým příkladem je klasický expertní systém 1. generace.

- **Intencionální (deliberativní) agent** zvažuje své možnosti dosáhnout cíle. Ve volbě cíle se odráží motivace agenta. Koordinace skupiny spočívá ve vzájemném informování o tom, čemu momentálně věří a co plánují.

- **Sociální agent** pracuje s explicitními modely chování ostatních agentů. Samozřejmě má schopnost upravovat a aktualizovat modely, neboť je používá k přizpůsobení plánované činnosti jiných agentů.

- V reálném systému se vyskytují agenti s **hybridním chováním**.

Definice agenta (Bond,Gasser):

Agent je výpočetní proces s jedním centrem řízení a s určitým (lokálním) cílem. Agenti koordinují hlavně své znalosti, cíle, schopnosti a plány tak, aby je mohli využívat společně při akcích a řešení úloh. Agenti v MAS mohou ve své činnosti směřovat k jednomu globálnímu cíli nebo k jednotlivým odděleným cílům, které spolu mohou, ale nemusí souviset. Agenti sdílení znalosti o úlohách a řešeních.

Definice agenta (Farhoodi, Graham):

Inteligentní agent je entita, která je zodpovědná za rozhodování, zda a jak reagovat na externí podněty. Agent může posílat a přijímat informace od jiných agentů za použití vhodných protokolů, může zpracovávat přijaté informace a uvažovat o nich, má soubor schopností provádět akce, které se mohou i dynamicky měnit, akce charakterizující úkoly, které dokáže agent provádět.

Inteligentní agent:

Dovede uvažovat o svých schopnostech a schopnostech ostatních agentů, generovat cíle nebo plány pro sebe a jiné agenty, účastnit se složitých

interakcí s ostatními, dynamicky se zapojovat do skupin nebo takové skupiny opouštět, získávat informace a používat jejich zdroje, udržovat explicitní modely důvěry pro sebe a ostatní agenty.

Obecnější definice agenta (Wooldridge, Jennings):

Agent = hardwarový nebo softwarový systém, který je autonomní (pracuje bez přímého zásahu člověka, řídí své akce a vnitřní stav), sociální (interaguje s ostatními), reaktivní (vnímá své okolí a reaguje včas na změny v něm), proaktivní (reakce dána i cíli – převzetí iniciativy při řešení).

Základním úkolem organizace MAS je zajistit koordinaci, kooperaci, komunikaci. Koordinace je patrná zejména při přidělování nedostatkových zdrojů, předávání mezivýsledků. Vzor si bere z organizování lidských kolektivů. Příkladem je vzájemné přizpůsobení, direktivní velení, metody standardizace některých aktivit. Cílem kooperace je zkvalitnění práce MAS (zrychlení řešení využitím paralelního zpracování, rozšíření třídy řešitelných úloh vyplývající z výhod sdílení zdrojů, zvýšení počtu úspěšně završených úloh - paralelní řešení několika metodami, snížení počtu kolizí mezi vzájemně soutěžícími agenty). Stupeň kooperace mezi agenty se pohybuje mezi vztahy plně kooperativními až vztahy antagonistickými. Mezi metody kooperace patří vyjednávání a multiagentní plánování. Komunikace je nutná podmínka pro kooperaci a koordinaci. Struktura předávaných zpráv a použité metody závisí na složitosti jednotlivých agentů a celého systému. Komunikace výrazně určuje vlastnosti distribuovaného systému. Rozeznáváme:

- **přímou komunikaci** v synchronním či asynchronním režimu,
- **nepřímou komunikaci** prostřednictvím sdílené paměti (např. tabule).

Metody přímé komunikace
- adresné posílání zpráv (point-to-point),
- všesměrové vysílání zpráv (broadcasting),
- selektivní vysílání zpráv skupinám.

Architektura **tabule** je příkladem nepřímé komunikace. Důležité zprávy od všech agentů jsou zasílány na jedinou adresu (do tabule). Z pohledu MAS rozeznáváme tři hlavní komponenty:

1. **Agenti (zdroje znalostí).** Agent = zdroj znalostí musí znát své podmínky aplikace. Zdroje znalostí modifikují vlastní obsah tabule, a to explicitně a viditelně.
2. **Datová struktura tabule** informuje o aktuálním stavu řešení úlohy, jejím prostřednictvím agenti komunikují a zde se ukládá vznikající řešení.
3. **Řídící část tabule.** Řízení zaměřuje pozornost při řešení na následující položku pro zpracování. Aktivuje buď zdroje znalostí

nebo objekty na tabuli – „ostrůvky“ řešení, které mají být dále sledovány. Může navrhnout také kombinaci obou přístupů. Řešení je vytvářeno postupně krok za krokem. Celý proces je dynamický a oportunistický (zaměřuje se nejvhodnějším směrem, kde očekává největší přínos k řešení), nikoliv pevný a předem naprogramovaný.

Základní principy koordinace a kooperace

Cílem koordinace i kooperace je vhodně rozdělovat úlohy mezi agenty celého systému. Koordinace vnáší řád do jednání společenství agentů – vytváří společenské normy systému. Metody kooperace přinášejí principy organizace činnosti skupiny agentů realizujících danou úlohu. Koordinace a kooperace spolu těsně souvisí – rozhodnutí v jedné oblasti vyžadují určité postupy v oblasti druhé a naopak.

Prostředky koordinace

Předpokladem koordinace aktivit agentů je existence agentů, kteří dovedou uvažovat o důsledcích jednání svého i předpokládaných akcí ostatních agentů. MAS bez těchto agentů nemá smysl. Úroveň vzájemných informací o záměrech a plánech se pohybuje mezi těmito extrémy:

- všichni jsou informováni o všech, včetně všech průběžných změn => nedochází ke kolizím, ale agent se soustředí na komunikaci a zpracování změn a nepracuje na úkolu. Problémy převáží nad výhodami použití MAS.
- existence jediného centrálního metaagenta, „vševěda“. Komunikace je více ekonomická, ale na metaagenta to klade nesplnitelné nároky (hlavně v oblasti řízení).

Pro skupiny bez centrálního agenta platí nejen, že každý z agentů musí nést svou část zodpovědnosti za koordinaci akcí jednotlivců, ale dokonce tam není jiného místa, kde by mohl být skryt klíč ke koordinaci.
Z hlediska dosažení výchozích cílů společenství agentů je důležitá jistá stálost v jednání agenta, jeho schopnost dokončit započatou práci. Z toho vyvozujeme jako jedno z měřítek „zodpovědnost“ jedince za svěřené úkoly. N.R.Jennings dokonce formuloval smělou hypotézu, že „veškeré koordinační mechanismy mohou být redukovány na závazky a různé typy odpovídajících úmluv.“
Závazek je v tomto kontextu příslib agenta vykonat určitou činnost spolu se záměrem tomuto dostát. Závazky slouží nejen k návrhu schématu řešení úlohy, ale v případě formulace závazku s vysokou pravděpodobností splnění slouží pozorujícímu agentovi k odvození dlouhodobých záměrů zavázaného agenta bez další komunikace.
Závazky tvoří kostru řešení úlohy distribuovaným systémem. K jejich přijetí dochází ovšem za jistých předpokladů, které se však v dynamickém prostředí mění. Pak není dané závazky nutné dodržet. Prostředky sloužící k hodnocení dodržování a změn závazků jsou **smlouvy.** V nich se stanoví podmínky pro dodržení závazků.

S DAI se budeme setkávat stále více, protože poskytuje silné nástroje k řešení problematiky zatížení sítí, řízení výroby, výpočetně náročných

úkolů. Určitě se setkáme s použitím těchto přístupů při simulaci lidského chování, i když zde je limitující problém vyjádření komplexnosti lidské mysli. Každopádně má tato mladá odnož umělé inteligence před sebou zajímavý vývoj.

6.2. Agenti

Reaktivní agent je entita vykonávající akce bezprostředně a výlučně na základě stimulů přijatých z prostředí. Racionalita jeho chování je důsledkem interakce s prostředím. Svou funkcionalitu je přizpůsoben ke vnímání omezené množiny podnětů z prostředí a může vykonávat akce vymezeného typu. Nemá žádný modul pro tvorbu plánů ani modul pro rozhodování, který cíl z potencionální množiny cílů bude sledovat. Jeho součástí jsou tzv. kompetenční moduly, což jsou funkce nebo procedury, které se vykonávají na základě splnění aktivační podmínky.

Nejdůležitější probrané pojmy:

- distribuovaná umělá inteligence
- multiagentní systémy
- koordinace, komunikace, kooperace

Úkoly a otázky k textu:

Najděte (například na Internetu) reálnou ukázku použití distribuované umělé inteligence v praxi.

7. Optimalizační a herní problémy

Cíl:

Získáte základní přehled o těchto otázkách:

- co je to problém optimalizace
- typy optimalizačních úloh
- metody umělé inteligence v herních problémech

Optimalizační a herní problémy jsou v informatice často řešenou úlohou. V návrhu sofistikovaných metod řešení těchto problémů spatřuje mnoho informatiků svou realizaci – vždyť hledání efektivních algoritmů je vlastně typickou úlohou informatiky. Proto se v následující skromné kapitole alespoň dotkneme této problematiky.

7.1. Problém optimalizace

Optimalizační algoritmy jsou mocným nástrojem pro řešení mnoha problémů inženýrské praxe. Obvykle se používají tam, kde je řešení daného problému analytickou cestou silně nevhodné či nereálné. Při vhodné implementaci mohou být použity tak, že dokonce není potřeba častého uživatelského zásahu v činnosti příslušného zařízení, kde jsou použity.

Většina problémů inženýrské praxe může být definována jako optimalizační problém jako např. nalezení optimální trajektorie robota, optimální tloušťky stěny tlakové nádoby, optimální nastavení parametrů regulátoru atd. Jinými slovy, řešený problém lze převést na matematický problém daný vhodným funkčním předpisem, jehož optimalizace vede k nalezení argumentů tzv. účelové funkce, což je cílem optimalizace.

Příkladů lze nalézt nespočetně. Řešení takových problémů obvykle vyžaduje práci s argumenty optimalizovaných funkcí, přičemž definiční obor těchto argumentů může být různorodého charakteru jako např. obor celočíselný, reálný, komplexní apod. Navíc se může stát (případ od případu), že pro určité subintervaly z povoleného intervalu hodnot může příslušný argument optimalizované funkce nabývat různých typů hodnot (opět celočíselný, reálný, komplexní apod.). Navíc v rámci optimalizace mohou být uplatněny různé penalizace a omezení nejen na dané argumenty, ale také na funkční hodnotu optimalizované funkce. Řešení takového optimalizačního problému analytickou cestou je mnohdy možné, nicméně značně komplikované a zdlouhavé.

Pro úspěšné řešení takových problémů byla v posledních dvou desetiletích vyvinuta množina velmi výkonných algoritmů, které umožňují řešit velmi složité problémy efektivním způsobem. Tato třída algoritmů má svůj specifický název a to "evoluční algoritmy". Tyto algoritmy jsou schopny

řešit velmi složité problémy tak elegantně, že se staly velmi oblíbené a používané v mnoha inženýrských oborech.

Typickým rysem pro evoluční algoritmy je, že pracují s tzv. populacemi možných řešení, jimž se říká jedinci. Tito jedinci navzájem ovlivňují svou kvalitu na základě určitých evolučních principů v cyklech, které obvykle nesou jméno "Generace". Cílem celého evolučního procesu je nalézt nejlepší řešení.

7.2. Typy optimalizačních metod

Optimalizační algoritmy

Optimalizační algoritmy slouží k nalezení minima dané účelové funkce tak, že hledají optimální numerickou kombinaci jejich argumentů. Tyto algoritmy lze rozdělit podle principů jejich činnosti tak, jak je naznačeno na další stránce. Toto rozdělení není samozřejmě jediné možné, nicméně vzhledem k tomu, že vcelku dobře vystihuje současný stav, lze jej brát jako jeden z možných pohledů na klasické, ale i moderní optimalizační metody.

Možné uspořádání optimalizačních algoritmů

DETERMINISTICKÉ	STOCHASTICKÉ	SMÍŠENÉ
GREEDY	RANDOM SEARCH-WALK	MATEMATICAL PROGRAMMING
HILL-CLIMBING	SIMULATED ANNEALING	ANT COLONY OPTIMIZATION
BRANCH & BOUND	MONTE CARLO	IMMUNE SYSTEM METHODS
DEPTH-FIRST	TABU SEARCH	MEMETIC ALGORITHMS
BROADTH-FIRST	EVOLUTIONARY COMPUTATION	SCATTER SEARCH & PATH REL.
BEST-FIRST	STOCHASTIC HILL-CLIMBING	PARTICLE SWARM
CALCULUS BASED		GENETIC ALGORITHMS
		DIFFERENTIAL ALGORITHMS
		SOMA

Jednotlivé třídy algoritmů představují obecně způsob řešené daného problému metodami s různým stupněm efektivity a složitosti. Jejich vlastnosti včetně optimálního použití jsou:

Enumerativní

Jde o výpočet všech možných kombinací daného problému. Tento přístup je vhodný pro problémy, u nichž jsou argumenty účelové funkce diskrétního charakteru a nabývají malého množství hodnot. Pokud by byl

použit obecně, zcela reálně by mohl potřebovat na úspěšné dokončení čas, který je delší nežli existence našeho vesmíru.

Deterministické

Tato skupina algoritmů je postavena pouze na rigorózních metodách klasické matematiky. Algoritmy tohoto charakteru obvykle vyžadují předběžné předpoklady, jenž "umožní" této metodě podávat efektivní výsledky. Tyto předpoklady obvykle jsou, že

- problém je lineární
- problém je konvexní
- prohledávaný prostor možných řešení je malý
- prohledávaný prostor možných řešení je spojitý
- účelová funkce je pokud možno unimodální (pouze jeden extrém)
- mezi parametry "uvnitř" účelové funkce nejsou nelineární interakce
- jsou dostupné informace typu gradient apod.
- problém je definován v analytickém tvaru.

Výsledkem deterministického algoritmu je pak pouze jedno jediné řešení.

Stochastické

Algoritmy tohoto typu jsou založeny na využití náhody. Jde v podstatě o čistě náhodné hledání hodnot argumentů účelové funkce s tím, že výsledkem je vždy to nejlepší řešení, jenž bylo nalezeno během celého náhodného hledání. Algoritmy tohoto typu jsou obvykle pomalé, vhodné jen pro prohledávané prostory možných řešení, jenž jsou malé a vhodné pro hrubý odhad

Smíšené

Tato třída algoritmů představuje "rafinovanou" směs metod deterministických a stochastických, které ve vzájemné spolupráci dosahují překvapivě dobrých výsledků. Poměrně silnou podmnožinou těchto algoritmů jsou již zmíněné evoluční algoritmy.

Shrnou-li se tedy tyto vlastnosti, lze stručně konstatovat že:

- Enumerativní a stochastická optimalizace není vhodná na problémy, u nichž se prohledává rozlehlý prostor možných řešení.

- Deterministická optimalizace pracuje dobře na problémech, u nichž se prohledává úzký prostor možných řešení.

- Smíšená optimalizace je vhodná na problémy "bez omezení" velikosti jejich prostoru možných řešení.

Nástin principu Ant Colony Optimization ACO

(optimalizace mravenčí kolonií)

Jak už to tak bývá, když si člověk neví rady se svými problémy, obrátí se na matku přírodu. Nemyslí se tím návrat k přírodě ve smyslu zahození

všech technických vymožeností, ale právě naopak, využití přírodních mechanismů v dalším vylepšení technologií.

Mravenec jako jedinec ani nevykazuje nějaké příliš složité chování, vlastně jen přímo reaguje na své okolí a ostatní mravence. Ale když se takových jedinců dá dohromady více, můžeme se divit jaké složité problémy dokáže taková kolonie řešit. Typickým příkladem může být hledání potravy, kdy si mravenčí kolonie dokáže velice rychle najít optimální cestu k potravě a jedinci pak po dané trase pochodují jako po dálnici. Žádná překážka nedokáže tomuto putování zabránit, mravenci si vždy najdou další nejlepší cestu.

Princip je následující. Nechť existuje zdroj mravenců (mraveniště) a cíl jejich snažení (potrava). Když jsou vypuštěni, tak po nějaké době dojde k tomu, že všichni mravenci se pohybují po kratší (optimální) cestě mezi zdrojem a cílem. Tento efekt, kdy mravenci naleznou optimální cestu je dán faktem, že mravenci si svou cestu značkují feromonem. Jeho intenzita pak ovlivňuje rozhodnutí mravence. Pokud dorazí k rozcestí dvou cest vedoucích ke stejnému cíli první mravenci, pak jejich rozhodnutí, po které cestě se vydají je náhodné. Ti, kteří zvolí kratší cestu ji označkují a při návratu jsou díky těmto značkám při rozhodování ovlivnění ve prospěch kratší cesty. Při návratu ji označkují podruhé, což opět zvyšuje pravděpodobnost rozhodnutí dalších mravenců v její prospěch. Tyto principy jsou použity v ACO algoritmu. Feromon je zde zastoupen vahou, která je přiřazena dané cestě vedoucí k cíli. Tato váha je aditivní, což umožňuje přidávat další "feromony" od dalších "mravenců". V ACO algoritmu je zohledněn i fakt vypařování feromonů tak, že váhy u jednotlivých spojů s časem slábnou. To zvyšuje robustnost algoritmu z hlediska nalezení globálního extrému.

Mravenčí putování přirozeně našlo nejdříve odezvu v řešení technických problémů, které jsou mu podobné. Snad poprvé se algoritmus založený na chování mravenčí kolonie objevil při řešení problému obchodního cestujícího (cestující musí co nejrychleji projet danou síť měst). Telekomunikační sítě typu internet jsou totiž jako stvořené pro aplikaci "mravenčích" algoritmů. Při hledání spojení jde přece také o nalezení optimální trasy a v případě, že je nějaká část sítě postižena výpadkem, je potřeba rychle najít další cestu, aby spojení nebylo přerušeno. Další aplikace můžeme najít v logistice či plánování výroby (nezáživné? jen na první pohled!), kde je opět potřeba hledat optimální cesty (výrobků přes stroje či sklady) a být schopen je adaptovat na změněné podmínky.

Hledáním a obnovováním optimálních cest ovšem nejsou mravenčí možnosti vyčerpány. Pozornost vzbuzuje také schopnost přepravovat neuvěřitelně velké náklady i po dosti strmých plochách. Mravenci, na rozdíl třeba od lidí, při takové práci nekomunikují přímo, ale prostřednictvím přepravovaného objektu. V praxi to znamená, že takový mravenec sám pozná, kde má "zatlačit", aniž by se o tom domlouval s

ostatními. První pokusy s roboty přemisťujícími velké bedny podobnou technikou již probíhají.

7.3. Implementace metod umělé inteligence v počítačových hrách

Sledujeme-li vývoj implementace metod umělé inteligence ve hrách, je vidět několik trendů. Při jejich sledování narážíme na obtíž - hry jsou komerčními produkty firem, které většinou nezveřejňují principy jimi publikovaných her a to hlavně u úspěšných titulů. Tituly neúspěšné bývají naproti tomu obohaceny občas klamavými informacemi (např. se dosti často udává „neuronový oponent") ačkoliv se ve fázi finálního testování ukážou významné problémy se stabilitou adaptivního oponenta a aby bylo zabráněno naprostému fiasku je adaptivní oponent nahrazen klasickou optimalizací, obohacenou například modelováním náhodných odchylek.

Pevně naprogramované chování

Nejjednodušší variantou je pevně naprogramovaný oponent tvořený sekvenční sadou podmínek, což přináší četné nevýhody jako například stereotypii a nemožnost dosažení některých řešení. Zvláště druhý problém se ukázal jako fatální, protože u komplexních her často nastávají autory neočekávané situace. Zatímco šachy jsou hrou relativně „triviální", neboť je k dispozici úplná informace, šachovnice je diskrétní, konečná a „velmi malá", u mnoha her je prostor (v obecnosti) spojitý, rozsáhlý a často vytvářený jinou osobou než vytvářela algoritmus oponenta. To vede k množství speciálních případů, kdy algoritmus selže a počítačem řízené monstrum se zasekne nebo zacyklí v nějakém herním prostoru či situaci a nemůže se dostat ven. Zatímco stereotypní chování oponentů je pokládáno hráči za „nepříjemnost", zasekávání se je pokládáno přímo za chybu. Je ale jasné, že se u pevně zapsaných podmínek jen s krajní obtíží ošetřují všechny možné případy.

O něco lepším řešením se ukázaly pravidlové systémy, které mohou být simulovány i v pevně vytvořeném kódu tak, že algoritmus prohledá všechny alternativy, přidělí jim míry uspokojivosti a z nich nakonec vybere optimální. To do značné míry eliminuje nedostatky prosté sekvence podmínek, ale chování oponentů je stále vnímáno jako stereotypní, neboť ve stejné situaci počítač vybírá stejné řešení. Tento nedostatek byl následně odstraňován vnesením nedeterminismu, kdy počítač náhodně volí mezi přibližně stejně výhodnými alternativami. Výsledek je percipován jako mnohem živější a metoda je hojně užívána.

Optimalizační metody

Velmi důležité jsou optimalizační metody, které slouží k řešení mnoha významných okruhů problémů jako:

- Hledání nejkratší nebo optimální cesty a to i v situaci, kdy se graf možných cest dynamicky mění v průběhu času.
- Optimální alokaci zdrojů, výroby nových jednotek.
- Určení optimální struktury útoku - hledání specifikace, která jednotka vlastní zaútočí na kterou jednotku nepřátelskou a v jakém pořadí.
- Hledání optimální prostorové konfigurace jednotek a základen tak, aby bylo riziko při napadení minimální při největším palebném pokrytí co do plochy nebo intenzity.
- Hledání strategických bodů na mapě - kde vede mnoho důležitých cest, kde jsou zdroje surovin a zdroje pro výrobu jednotek.

Je patrné, že se ve hrách řeší podobné okruhy problémů jako v reálném světě, např. můžeme uvést problematiku pokrývání plochy chráněného území protiletadlovými bateriemi se specifickými vlastnostmi a dosahem. Rozdíl vidíme jen ve složitosti problému - u her je do značné míry zajištěna konstantnost vlastností a při optimalizaci se uvažuje méně kritérií.

Fuzzy algoritmy

Ve stejné době se začaly zavádět fuzzy algoritmy pro rozhodování, zvláště u her s neúplnou informací (např. s „fog-of-war" na mapě). Zavedení „fog-of-war" přináší pro umělou inteligenci značné potíže, protože je nutné přijít s algoritmy, které odhadují ve velmi nejisté a proměnlivé situaci. Zde již opouštíme pole algoritmů klasických znalostních systémů, které při vyhodnocování informaci spíše sbírají a nebo dokážou navíc minimalizovat počet nutných dotazů.

Je patrné, že inferenční systém operuje v podmínkách značné nejistoty a aplikace fuzzy algoritmů je zcela na místě. Jednotlivé jednotky vcházející do kontaktu s nepřítelem přinášejí parciální poznatky, které je možné převádět na informaci s nejistotou, kde lze informaci ohodnocovat např. podle poměru známé a neznámé plochy nepřátelské základny.

Adaptivní oponenti

Adaptivita je zajišťována různými způsoby. Je zajímavé, že přes rozsáhlé informace o implementaci neuronových oponentů do her různých typů (simulátory, strategie) na začátku roku 1997 je odhadovaný počet reálných implementací mnohem nižší.

Poloexperimentální počítačové hry

Mezi softwarovými tituly se objevují produkty, které není možné považovat za hry v pravém smyslu slova. Obvykle se rozlišuje „computer game" a „computer toy". Tyto pojmy odrážejí rozdíl v cíli činnosti hráče, v češtině krásně rozlišované slovesy „hrát" a „hrát si" a podstatnými jmény „hra" a „hračka".

Nejdůležitější probrané pojmy:

- optimalizace
- enumerativní, deterministické, stochastické a smíšené metody
- metody implementace UI v počítačových hrách

Úkol k textu:

Najděte reálnou aplikaci pro tři vybrané metody optimalizace.

Korespondenční úkol:

Pro zvolený optimalizační problém navrhněte algoritmus a naprogramujte jej ve zvoleném vývojovém prostředku. Volbu konzultujte s tutorem.

8. Vybrané aplikace UI

Cíl:

V této kapitole si uděláte představu o některých problémech spadajících do oblasti UI:

- Šachové algoritmy
- Teorie chaosu
- Počítačové vidění
- Zpracování přirozeného jazyka
- Rozpoznávání hlasu a řeči
- Rozpoznávání písma

8.1. Šachové algoritmy

Šachové počítače jsou z hlediska laika obestřeny jakousi tajemnou aurou. Právě na šachách se často demonstrují možnosti i omezení dnešních počítačů, otázky, zda počítač "myslí". Jak algoritmus, umožňující počítači hrát šachy, vůbec vypadá?

Prohledávání stromu hry

Nejdůležitější a nejzajímavější částí šachového programu je schopnost samostatně vymyslet tah. Dnešní šachové programy těží z několika desítek let evoluce, jejímž výsledkem je téměř shodná základní struktura všech úspěšných programů. Obdobně jsou postaveny i programy jiných deskových her, např. dámy. Typický algoritmus vychází z tzv. statického hodnocení pozice. Program nejdříve sečte hodnotu kamenů na šachovnici možné hodnoty jsou např. 100 bodů za pěšce, 350 za jezdce a střelce, 550 za věž a 1 000 za dámu. Za bílé figury tyto hodnoty k celkovému hodnocení přičítá a za černé odečítá. Dále přidává k celkovému hodnocení různé hodnoty poziční. Může například hodnotit pohyblivost figur, pěšcové slabiny, vazby figur, bezpečnost králů a mnoho dalších faktorů. Právě v pozičním hodnocení mají autoři prostor k uplatnění vlastních šachových znalostí. Pouze je musejí popsat tak přesně, aby výsledkem bylo číselné hodnocení. A to vyžaduje o něco víc, než jen umět dobře hrát šachy.

Minimax

Statického hodnocení nyní můžeme využít k volbě vhodného tahu. Jednoduše vygenerujeme všechny tahy a necháme počítač ohodnotit výslednou pozici po každém z nich. Pak už jen vybereme ten, který vede do pozice s nejlepším hodnocením. Nemusíme se ovšem omezit na hloubku jediného půltahu. Místo okamžitého vrácení statické hodnoty můžeme opět generovat tahy, všechny je zkusmo provést, vybrat z výsledných hodnot hodnotu nejvýhodnější pro stranu na tahu a tu vrátit.

Když algoritmus dosáhne předem dané hloubky propočtu, vrátí statické hodnocení. Je-li na tahu bílý, vrací rekurzivní algoritmus hodnotu maximální, je-li na tahu černý, vrací hodnotu minimální. Proto se tomuto algoritmu říká minimax. Pro zjednodušení zápisu můžeme použít malý trik: při rekurzivním volání prostě překlopíme znaménko výsledku a vždy hledáme nejvyšší hodnotu. Prohledávaná struktura připomíná strom, a také se jí tak říká. Úvodní pozici se běžně říká kořen (root). Pozicím umístěným ve stromě se říká uzly (nodes). Uzlům, ve kterých propočet končí a z nichž se vrací statické hodnocení, se říká listy (leaves, tips), ostatním se říká vnitřní uzly (internal nodes). Uzlům, ve kterých hra podle pravidel končí (mat, remíza), se říká uzly terminální.

Alphabeta

Časová náročnost algoritmu minimax roste exponenciálně vzhledem k hloubce prohledávaného stromu. V 50. letech byla vynalezena jednoduchá technika, která snižuje počet navštívených větví v některých vnitřních uzlech, přičemž výsledná hodnota a tah v kořenu zůstávají nezměněny. Tím se výrazně sníží počet uzlů, které je nutné navštívit v propočtu s danou hloubkou. Tento algoritmus jde jednoduše vysvětlit na příkladu přemýšlení lidského šachisty: když se mu podaří vyvrátit postup soupeře nějakým tahem, už své další tahy v této pozici prostě neuvažuje a využije svůj čas ke zkoumání jiných pozic.

Iterativní prohlubování

Vlastnosti alphabety vedou k tomu, že je výhodnější prohledávat iterativně, tj. nejdříve se prohledává do hloubky 1, pak 2, 3, 4 a dále. Ve vyšších iteracích pak máme k dispozici důležité informace o tom, jak třídit tahy, a to nám přináší další výrazné časové úspory. Další výhodou iterativního prohledávání je jednodušší kontrola času. Po ukončení iterace se program podívá na hodiny a může se rozhodnout, zda bude v iterování pokračovat nebo provede zatím nejlepší nalezený tah.

Hashovací tabulky

Další technikou, kterou používají všechny úspěšné programy, je tabulka killerů a transpozic. Pozice, které již program prošel, jsou uloženy do paměti. Pokud se ve stromu vyskytne pozice shodná s některou uloženou, máme většinou k dispozici vhodný killer a za splnění jistých předpokladů můžeme použít výslednou hodnotu předchozího výpočtu a rovnou ji vrátit bez dalšího prohledávání. Do políčka transpoziční tabulky se většinou ukládají tyto hodnoty: Hashovací klíč pozice, jakýsi "podpis", podle kterého se určí shoda pozice v tabulce s aktuální pozicí. Při použití vhodných technik tvorby klíče lze vystačit se čtyřmi nebo osmi byty. Nejlepší nalezený tah, což je pozdější vhodný kandidát na killera.

Extenze a prořezávání

Podíváme-li se blíže na pozice ve stromu prohledávaném běžnou alphabetou, zjistíme že většina pozic jsou pozice jednoznačně vyhrané pro jednu ze stran a na výsledek prohledávání mají minimální vliv. Autoři programů tedy zkoumají, jak ze stromu bezpečně odstranit nezajímavé větve a v těch zajímavých zase prohloubit propočet. Odstraňování nezajímavých větví říkáme prořezávání (pruning). Prohloubení propočtu říkáme extenze.
Často používanou metodou prořezávání je tzv. metoda nulového tahu. Strana na tahu vždy před zahájením prohledávání vlastních tahů zkusí předat právo tahu soupeři, při tomto pokusu výrazně sníží hloubku prohledávání. Jestliže je výsledek nulového tahu větší nebo roven hodnotě beta, předpokládáme, že máme k dispozici i jiný tah s hodnotou alespoň beta a běžné tahy vůbec nepočítáme. Výhodou této metody je jednoduchá implementace a znatelné zvýšení hloubky propočtu. Nevýhodou je možnost taktických přehmatů v pozicích, kterým šachisté říkají "zugzwang", kde povinnost táhnout vede k prohře.

Knihovna zahájení

Na knihovně zahájení není nic moc složitého. Většinou je to databáze pozic, ke každé pozici se váže jeden nebo více tahů a k tahu jeho hodnota, která určuje, s jakou pravděpodobností má být zahrán.
Zajímavější to začne být až ve chvíli, kdy vymýšlíme, jak do knihovny dostat dobré tahy. V dávných dobách počítačového šachu byla knihovna zahájení téměř ruční prací autora programu. Dnes jsou k dispozici obsáhlé elektronické databáze velmistrovských partií a ruce Kena Thompsona si konečně mohou odpočinout. Práci s převáděním tahů do knihovny zahájení za něj udělá počítač. Navíc může stroj o každém tahu zpracovat statistiku jeho četnosti a úspěšnosti a na jejím základě pak spočte vhodnou pravděpodobnost pro pozdější vybírání z knihovny. Chytřejší programy pak umějí pravděpodobnostní hodnoty tahů v knihovně upravit podle výsledků vlastních partií. Jde vlastně o jednoduchou formu učení se.

Databáze koncovek

Stále více programů dnes využívá dokonalé znalosti některých typů koncovek s malým materiálem. Program má v databázi uloženy všechny možné pozice koncovky spolu s jejich přesným hodnocením to znamená buď počet tahů do matu, nebo teoretickou remízu. Dnes jsou zpracovány všechny tří, čtyř a pětikamenové koncovky a některé koncovky šestikamenové. Na Internetu jsou volně dostupné databáze od Eugene Nalimova spolu s programem v C++, kterým je možné je vygenerovat.
Časová a paměťová náročnost generování výrazně roste s počtem kamenů. Všechny tříkamenové koncovky zaberou 70 KB, čtyřkamenové 30 MB a pětikamenové 7 GB. Z toho je vidět, že touto cestou nemáme šanci dospět k úplné analýze šachové hry od základní pozice, což by vlastně odpovídalo analýze dvaatřicetikamenové "koncovky".

Díky počítačovým databázím šachových koncovek byly odhaleny chyby ve výsledcích práce nejlepších lidských teoretiků šachových koncovek. Počítač tak za několik hodin až dnů práce s generováním databáze odhalí to, na co lidským mozkům nestačila staletí. Známým příkladem je koncovka dvou střelců proti jezdci. V roce 1851 byl vydán podrobný rozbor této koncovky zpracovaný teoretiky Klingem a Horowitzem. Výsledkem tohoto rozboru je mimo jiné typ pozice, ve které se slabší strana ubrání. Tuto pozici pak převzali do svých děl pozdější renomovaní autoři učebnic a příruček šachových koncovek. Podle objevitelů se jí běžně říká Kling-Horovitzova pozice. Až výsledky databázového zpracování této koncovky ukázaly, že obrana je mnohem obtížnější, a dokonce že ukázková Kling-Horowitzova pozice je vyhraná pro silnější stranu.

O reálné herní síle šachových počítačů toho bylo napsáno opravdu hodně. Přitom o ní víme až překvapivě málo. Počítače se totiž prakticky nezúčastňují turnajů, ve kterých by se mohly utkat s lidmi. Máme tak k dispozici pouze výsledky několika specializovaných turnajů a zápasů. Z řady důvodů nejsou výsledky navíc příliš reprezentativní.
Použít můžeme partie ze zápasu mezinárodního mistra Deena Hergotta proti programu Hiarcs 6 na Pentiu MMX/200 (z roku 1997) a 4 partie programu Rebel 2 proti velmistru Jusupovovi (1997) a 2 proti velmistru Anandovi (1998).
Mnoho se spekulovalo o síle počítače po zápase Kasparov Deep Blue v roce 1997, který vyhrál počítač v poměru 3,5:2,5. Nic víc než spekulovat se z výsledku pouhých šesti partií proti jedinému protivníkovi stejně nedá. V laické veřejnosti se vytvořil dojem, že počítač Deep Blue je dnes silnější než lidský mistr světa.

8.2. Teorie chaosu

Není jednoduché stručně říci, co je teorie chaosu. Až do vzniku teorie chaosu zaměřovali badatelé svou pozornost na systémy a jevy, které bychom mohli z jistou nadsázkou nazvat uspořádané, protože se daly s dobrým přiblížením popsat nějakou soustavou rovnic a vyhýbali se takovým jevům, které se z tohoto hlediska zdáli být nahodilé, neuspořádané, prostě chaotické. Takové jevy, jsou ale ve skutečnosti velice časté. Teorie chaosu, ale není teorií o nepořádku.
Myšlenka chaosu se opírá o následující tvrzení:

- Malé změny v systému mohou způsobit velké fluktuace.
- Nelze přesně určit stav nějakého systému (polohy a rychlosti jednotlivých atomů ap.).
- Naopak je jednoduché popsat chování systému jako celku.

Pojem chaosu byl zaveden u dynamických soustav americkým meteorologem Lorenzem (1963) v souvislosti se snahou o popis konvektivního proudění v atmosféře. Soustava tří jednoduchých nelineárních diferenciálních rovnic (Lorenzovy rovnice) vykazovala neočekávaně složité chování v závislosti na hodnotách charakteristických

parametrů a názorně demonstrovala nemožnost dlouhodobé předpovědi počasí. Byl velmi překvapen nestabilitou a chaotickým vývojem tohoto meteorologického modelu, protože s velmi nepatrnými změnami výchozích dat se dostavovaly zcela rozdílné meteorologické výsledky. Od té doby se chaos stal fenoménem, používaným v celé řadě oborů lidského poznání v souvislosti s popisem dynamiky chování systémů v čase.

Od chvíle, kdy se badatelé začali rozhlížet po této oblasti, bylo jasné, že se chaos skutečně vyskytuje všude, v chování počasí, v chování letadel za letu, v chování aut, hromadících se na dálnici, v chování ropy, tekoucí podzemním porubím i ve vývoji cen na burze. Ať už se ale takovéto jevy vyskytují v jakémkoli prostředí, vždy se řídí stejnými (v rámci teorie chaosu zkoumanými a objevovanými) principy. Chaos si vyžádal vznik nových badatelských technik (například zvláštní způsob využívání počítačů, zvláštní druhy zobrazení, za jejichž složitostí se skrývají fantastické jemné struktury. Stručně řečeno jedná se tedy o vědu, která se zabývá globálním chováním nejen přírodních, ale i sociálních systémů v nejširším smyslu.

S teorií chaosu je spojen pojem atraktor. Slovo atraktor je odvozeno od anglického "to attract" (přitahovat) a vyjadřuje stav, ke kterému systém směřuje (například : kyvadlo hodin podléhá tření a tak se jednou zastaví; kyvadlo tedy směřuje do stavu klidu v nejnižší poloze, kterým v tomto případě je těžiště, kterým prochází). Různé atraktory mohou mít různý tvar, například bod, periodicky se opakující smyčka, ale i daleko složitější a obtížněji představitelné tvary. Můžeme tedy říct, že atraktor je něco jako cílový/konečný stav systému.

Atraktor je tedy důležitým pojmem v rámci teorie chaosu, která se vesměs zabývá popisem nelineárních a složitě předvídatelných systémů. Jedním z případů podobných výše zmíněnému kyvadlu je chladnutí čaje. Atraktorem je v tomto případě samozřejmě teplota místnosti (pro zjednodušení předpokládejme, že ta je konstantní a díky čaji se nijak nezvýší) a časem, v němž se teploty vyrovnají, pak je nekonečná hodnota. Jakou však bude mít čaj teplotu v určité konkrétní budoucí době? Některé úvahy na toto téma jsou dosti elementární, například to, že rychlost chladnutí bude záviset na rozdílu teplot (na gradientu), takže čaj bude nejrychleji chladnout na počátku. Ukazuje se však, že udělat nějakou obecnější předpověď je dosti obtížné a pokud čaj ještě ke všemu budeme míchat, stává se náš úkol téměř nemožným.

Teorie chaosu popisuje právě takovéto "nedeterministické" systémy a odhaluje ve zmatku mnohdy překvapivé stopy řádu. Řád má obvykle podobu matematického jazyka, kromě atraktorů jsou dílem teorie chaosu například i populární fraktály.

Chaos se zkoumá v oblasti matematiky a co víc, má velmi pěkné aplikace v počítačové grafice a v umění, vytvářeném počítačem. Onou zajímavou a velmi vzhlednou aplikací chaosu u počítačů je fraktálová grafika fractal

graphics. Fraktálovou grafiku viděl určitě každý a řada lidí kolem počítačů se zejména s fraktálovou grafikou setkává více než vědomě, neboť existují i archivy pěkných, povedených fraktálových grafických objektů. Fraktál je členitý nebo-li fragmentovaný geometrický tvar, který je sám sobě podobný při různých zvětšeních. Fraktál bude vypadat skoro stejně, ať na něj budete dívat z jakékoli blízkosti. Fraktály jsou tedy obecně samopodobné a nezávislé na měřítku.

Chaos v praxi

Chaos je jako každá jiná teorie ukryt v pozadí celé řady různých vědních oborů. Přináší pro ně především nový a zajímavý pohled - něco odlišného od klasických newtonovských představ - a kromě jiného i např. nové směry v zobrazování vědeckých údajů (místo běžných funkčních závislostí lze interpretovat křivky ve fázovém prostoru ap.)
Přímé praktické aplikace:

- Modelování biologických systémů (růst populace, epidemie, arytmický pacemaker)
- Modelování dalších systémů (obchody na burze, kapající kohoutek)
- Grafické aplikace (Fractal Design Painter, filmové efekty - realistické mraky, skály, stíny)
- Fraktálová komprese obrazu (zatím ve vývoji, slibuje kompresní poměr 1:600, protože složitý obrazec popíše jednoduchou rovnicí, z níž lze všechno vypočítat)

Stále více vědců se dnes kloní k názoru, že ke dvěma nesporným „revolucím“ dvacátého století, teorii relativity a kvantové mechanice, patří i třetí – teorie chaosu. Je zajímavé, byť v jistém smyslu nepříliš potěšující, že všechny tři zmíněné oblasti stanovují určité meze našemu poznání okolního světa: prvá udává mez rychlosti přenosu energie a informací (danou rychlostí světla ve vakuu), druhá pak dolní mez velikosti interakcí mezi systémy (určené Planckovou konstantou). Chaos a nelineární dynamika konstatují zásadní omezení přesnější předpovědi chování jednoduchých systémů v delším časovém horizontu.
Nakonec je také nutno říci, že fraktální geometrie určitě v budoucnosti zaznamená velké úspěchy a otevře nám dveře k dalším tajům matematiky a nejen jí, které nebyly dosud popsány. Mnohé z metod konstruování fraktálů byly již úspěšně použity pro simulaci vývoje některých biologických struktur a na jejich další použití se čeká, protože nalezení pravidel, které by obsahovaly i vlivy vnitřních procesů, je i v současné době velice složité a obtížné.

8.3. Počítačové vidění

Úlohou počítačového vidění je napodobit pomocí technických prostředků schopnosti lidského oka. Jde nejen o samotnou schopnost zpracování obrazu ale také o rozpoznání co obraz představuje. V tom hraje podstatnou roli inteligence člověka a jeho předchozí zkušenosti.

Počítačové vidění lze rozdělit do dvou oblastí:

- rozpoznávání 2D obrazu (např. textu a znaků na papíře)
- rozpoznávání 3D obrazu (např. rozpoznávání součástky pro její následné uchopení)

Vizuální systém člověka je značně provázaný s jeho nabytými zkušenostmi a chápáním okolního světa. Dvě kamery by byly schopné emulovat oči, avšak člověk má navíc mnoho dalších schopností jako je například schopnost pohybovat hlavou a tím sledovat libovolný objekt v prostoru. Navíc mozek využívá své abstraktní znalosti při analýze obrazu. Při pohledu na nějaký objekt může předpokládat, srovnávat a vyhodnocovat určité jeho vlastnosti což počítač by mohl zvládat jen ve velmi omezeném rozsahu. Například člověk snadno rozpozná průhledný či neprůhledný objekt a rozliší daný objekt od okolního prostředí podle různých znaků které nelze počítačově jednoznačně vyhodnotit. Proto důležitou částí počítačového vidění jsou techniky zpracování 2D obrazů.

Zpracování 2D obrazu

Postup zpracování 2D obrazu lze rozdělit do těchto kroků:

1. snímaní, digitalizace a uložení obrazu v počítači
2. předzpracování,
3. segmentace obrazu na objekty,
4. popis objektů,
5. rozpoznávání (porozumění obsahu obrazu).

Při snímání se převádějí vstupní spojité fyzikální veličiny na spojitý elektrický signál. Vstupní informací může být jas, intenzita rentgenového záření, intenzita úměrná teplotě v termovizi apod. Snímat se může v jednom nebo více spektrálních pásmech. Barevný signál lze skládat ze tří samostatných spektrálních složek RGB.
Digitalizací obrazu rozumíme převod vstupního spojitého signálu do diskrétního tvaru. Vstupní analogový signál je popsán funkcí f(i,j) představující souřadnice v obrazu. Funkční hodnota odpovídá jasu(intenzitě). Dále se signál vzorkuje a kvantuje. Při vzorkování je třeba určit interval vzorkování což je obvykle polovina rozměru nejmenšího detailu obrazu. Pro vzorkování se také musí vybrat plošné uspořádání bodů. Nejčastěji se používá čtvercová nebo hexagonální mřížka. Kvantováním se spojité hodnotě jednoho vzorku přidělí diskrétní hodnota, obvykle jedna z 256 možných jasových úrovní.
Po digitalizaci obrazu dostaneme matici přirozených čísel popisujících obraz. Digitalizace poněkud zvyšuje složitost rozpoznávání obrazu. Například po digitalizaci dvou protínajících se úseček se může stát že nemají žádný společný bod.

Cílem předzpracování je potlačit šum a zkreslení vzniklé při digitalizaci a přenosu obrazu. Odstranění šumu se provádí průměrováním malého okolí zpracovávaného bodu. Ovšem po takovém zpracování se obraz rozmazává.

Proto některé metody používají takové okolí daného bodu které náleží jednomu objektu.
V předzpracování často potřebujeme zvýraznit určité rysy obrazu podstatné pro další zpracování. Takovým rysem můžou být například hrany v obraze, tj. obrazové body, ve kterých se obrazová funkce náhle mění.
Při segmentaci se snažíme nalézt v obrazu nepřekrývající se oblasti odpovídající objektům. Jde například o hledání černých písmen na bílém podkladu. Segmentaci rozdělujeme na úplnou, kdy se zcela povede přiřadit oblasti objektům a částečnou, kdy některé nalezené části neodpovídají objektům. U segmentace se používá několik metod. První z nich je prahování, lze ji použít tehdy, když se objekty, které hledáme, výrazně jasem odlišují od pozadí. Druhá metoda je založena na hledání hran a jejich vzájemné propojování do hranic objektů. Třetí metoda se snaží automaticky vyhledávat oblasti, které mají stejné vlastnosti. V této metodě se obvykle používá narůstání oblastí(je to obdobný postup jako při semínkovém vyplňování). Čtvrtá metoda segmentace používá postupy srovnávání se vzorem, který je předem znám. Postupně se prochází obrazem a každá pozice se porovná se vzorem. Pokud je míra shody dostatečně vysoká, je objekt v obrazu nalezen.
Popsat objekty lze kvantitativně pomocí souboru číselných charakteristik nebo kvalitativně pomocí relací mezi objekty. Za jednoduchý způsob popisu lze považovat stanovení velikosti objektů.

V nejjednodušším případě můžeme za porozumění považovat klasifikaci objektů v obraze podle jejich velikosti tj. podle počtu obrazových bodů. Jinou klasifikací objektů je rozdělení do několika předem známých tříd, např. na hranaté a kulaté. V obecném případě představuje porozumění interpretaci obrazových dat, o kterých se předem nic nepředpokládá. Porozumění obrazu je potom založeno na znalosti, cílech, tvorbě plánu k jejich dosažení a využití zpětných vazeb mezi různými úrovněmi zpracování. Pro porozumění v počítačovém vidění se používá také technik znalostních systémů avšak technicky lze dnes realizovat jen jednoduché úlohy pro porozumění obrazu.
Na těchto příkladech lze ukázat co se porozuměním rozumí v technicky zvládnutelných systémech.

Jednoduchý příklad - jednoduchá automatická analýza transparentního obrazu buněčného preparátu pozorovaného optickým mikroskopem. Cílem je spočítat buňky a roztřídit je podle tvaru buněčných jader na podlouhlé a ostatní. Úloha je z principu dvourozměrná. Předzpracováním se odstraní případný šum a pro segmentaci lze využít skutečnosti, že buněčná jádra jsou mnohem tmavší než zbytek. Pak už jen zbývá popsat body obrazu, příslušné každému buněčnému jádru, číselnými charakteristikami a podle nich buňky třídit.

Složitý systém - porozumění obrazu používané v autonomních vozidlech. Cílem je vybavit navigací terénní vozidlo pohybující s v blíže neomezené krajině bez řidiče. Poměrně dokonalé experimentální verze již byly

vytvořeny. Systém analýzy obrazu je ale jen jeden z mnoha senzorů, na jejichž základě se řídící systém vozidla rozhoduje. Jinými zdroji informací jsou např. radar, laserový dálkoměr, informace o absolutní poloze vozidla odvozená z gyroskopu, digitální mapa. Scéna je trojrozměrná a navíc v čase proměnná. Obrazový systém obvykle pracuje s informací ze dvou kamer, aby pomocí techniky stereovidění mohl odhadovat hloubku objektů ve scéně od pozorovatele. Vozidlo si musí vytvářet a průběžně modifikovat vnitřní model okolního světa.

Jednou z oblastí využití počítačového vidění je dálkový průzkum země (lesnictví, geodézie, meteorologie, archeologie, ...), lékařské aplikace (rozpoznávání rakovinných buněk, texturní analýza myokardu v echokardiografie, ...) a aplikace ve výrobě (počitadlo lahví, obsluha textilních strojů).

8.4. Zpracování přirozeného jazyka

Automatická analýza přirozeného jazyka počítačem vyžaduje - koneckonců jako každý problém, který řešíme - rozdělit práci na několik menších, dobře definovaných podproblémů, které pak řešíme nezávisle. V oblasti zpracování přirozeného jazyka se mluví o tzv. rovinách popisu jazyka. Pro účely analýzy jazyka jsou tyto roviny uspořádány zdola nahoru, od roviny nejjednodušší (zabývající se ortografií či akustickou stránkou věci) po rovinu nejsložitější, rovinu významu. Každá rovina má své jednotky popisu, definice vztahů na této rovině, a navazuje bezprostředně na rovinu nižší a vyšší, tzn. výstup z nižší roviny je vstup do následující vyšší roviny. Obvykle se hovoří o pěti až šesti rovinách:

– pragmatika (znalost světa…), logika aj., mezivětné vztahy…
– sémantika (hloubková syntaxe, význam)
– syntaxe (povrchová)
– morfologie
– fonologie
– fonetika

Někdy je vhodné některé roviny dále rozdělit, nebo naopak sloučit či přeskočit. Závisí to na spousty faktorech řešeného problému, např. jaký jazyk zpracováváme, tím myslím angličtinu či češtinu, nebo zda se jedná o řešení zpracování mluvené řeči, nebo jen překladu z jednoho jazyka do druhého atd.

Fonetická rovina

Vstupem fonetické roviny je akustický signál, výstupem posloupnost fónů (zvuků — vektorů různých charakteristik)

Předzpracování

Vstupem je posloupnost znaků čili písmen, výstupem pak posloupnost slov a interpunkce. Předzpracování můžeme rozdělit na roviny: pravopis, fonologie, morfonologie

Morfologická rovina

Vstupem je jak se dá předpokládat posloupnost slov a výstupem dvojice [lemma, značka].

Morfologické analýze se budu podrobněji věnovat v další části.

Syntaktická (povrchová) rovina

Vstupem je posloupnost dvojic [lemma, značka].

Výstupem větná struktura (strom) s označením větných vztahů

Sémantická (tektogramatická, hloubková) rovina

Vstup: větná struktura (strom) s pojmenováním vztahů

Výstup: rovněž stromová struktura, ale: hloubkové funkce, odstraněná pomocná slova

Pragmatická (logická) rovina

Vstup: hloubková struktura věty (propozice)

Výstup: logická forma, která může být vyhodnocena (pravda/nepravda)

Korpus

Textový korpus se většinou definuje jako: rozsáhlý vnitřně strukturovaný a ucelený soubor textů daného jazyka elektronicky uložený a zpracovávaný.

Tvorba korpusů

Korpusy mohou vznikat pouhým převodem dostupných textů v elektronické podobě, to je velice levné řešení. Pro rozsáhlejší korpusy, které kladou velký důraz na reprezentativnost, existují poměrně rozsáhlá kritéria, co a jak se má do korpusu vložit. Například se setkáváme s problémem duplicit textů, nebo problém zastaralých textů. Všechny větší korpusy většinou obsahují kromě čistě psaného textu také texty vzniklé přepisem mluvené řeči. Existují i mluvené korpusy obsahující výhradně mluvené slovo, u kterých je primární složkou zvukový záznam promluv. Takové korpusy slouží ke studiu výslovnosti nebo jako trénovací či testovací data pro syntézu či analýzu řeči.

Pozice, tokenizace

Každý rozsáhlejší systém, který má být snadno použitelný, musí vytvářet nějakou abstraktní úroveň s dostatečně jednoduchými prvky.

V korpusu by takový základní prvek mohlo být slovo. Texty ovšem obsahují i jiné řetězce znaků: čísla a interpunkce (tečky, čárky, uvozovky apod.). Základním stavebním prvkem by tedy mělo být něco, co může být slovem, číslem nebo nějakým znaménkem. Tento prvek se označuje jako pozice. Korpus tedy tvoří posloupnost pozic. Často se místo termínu pozice používá termín token a proces rozdělení textu na pozice je označován jako tokenizace.

Morfologická analýza

Morfologická (tvaroslovná) analýza vstupuje do hry až v okamžiku, kdy ve vstupním textu jsou identifikována slova, mezery, interpunkce, a pokud možno i začátky a konce vět, tzn. že je již dokončena tokenizace a jednotka zpracování pro morfologickou analýzu je tedy již jednoznačně určena. Tento předpoklad je i z praktického hlediska nepříliš omezující, neboť většina existujících textových **korpusů** je tokenizována.
Na střední škole se učí, že úkolem morfologické analýzy slova je určit morfologické kategorie danému slovu v textu příslušné. Pro člověka je tato definice přijatelná,. Při počítačovém zpracování je však situaci třeba definovat a popsat mnohem přesněji.
Především je třeba jasně rozlišovat mezi morfologickou kategorií a její hodnotou. **Číslo** je morfologickou kategorií, **singulár** (jednotné číslo) její hodnotou. V češtině a slovenštině je možno rozlišovat mnoho kategorií, např: slovní druh, slovní "poddruh", rod, číslo, pád, přivlastňovací rod, přivlastňovací číslo, osobu, čas, slovesný rod, negaci, stupeň a variantu. Hodnotami jsou např. čísla 1 až 7 pro české pády, "aktivní" a "pasivní" pro slovesný rod, atd. Nejbohatší kategorií je slovní poddruh, který má celkem 75 možných hodnot, nejvíce z nich pro zájmena.
Morfologická analýza pracuje bez ohledu na kontext, tj. zpracovává **izolovaně** vždy jen jedno slovo (slovní tvar). Tím "odsouvá" řešení některých problémů na pozdější dobu, a jakkoli je to z lingvistického pohledu bolestné, je tento přístup (vyplývající z dělení popisu a zpracování jazyka na jednotlivé roviny) jediný možný, neboť umožňuje nemíchat dohromady věci, které k sobě nepatří a byly by tudíž těžko formalizovatelné a zpracovatelné.
Pro počítačové zpracování se zavádí tzv. množina morfologických značek (**tagset**). Každá značka shrnuje hodnoty morfologických kategorií pro jeden slovní tvar. Pro vlastní zpracování se používá několik typů notací, z nichž nejrozšířenější je notace tzv. **poziční**. V této notaci se každé kategorii přiřadí pozice ve značce, a každé hodnotě jeden znak, který se zapisuje na příslušnou pozici. Slovní druh je tedy např. na první pozici, a jeho hodnoty jsou reprezentovány např. znaky **N** (pro podstatné jméno, noun), **A** (pro adjektivum), atd. Hodnoty pro daný slovní tvar irelevantních kategorií jsou označeny speciálním znakem, obvykle pomlčkou. Např. tedy pro obyčejné podstatné jméno rodu mužského neživotného ve 4. pádě jednotného čísla v pozičním systému s 15 kategoriemi má příslušná značka tvar **`NNIS4-----A----`** (první pozice je slovní druh (**N**), druhá slovní poddruh (zde **N**), třetí rod (**I** pro mužský neživotný, masc. inanim.), čtvrtá

číslo (**S** pro singulár), čtvrtý pád (**4** pro akuzativ), atd. (**A** na jedenácté pozici specifikuje, že dané slovo není negováno příslušnou předponou).
Co tedy (počítačová) morfologická analýza vlastně dělá? Pro každý slovní tvar určí všechny možnosti kombinací hodnot morfologických kategorií, které danému tvaru vůbec mohou příslušet. Že i to je obrovská pomoc pro další zpracování, je vidět z prostého číselného srovnání: zatímco všech možných značek (kombinací hodnot morfologických kategorií) je pro češtinu přes 4400 (pokud myslíme 13 výše jmenovaných kategorií), průměrný počet značek po morfologické analýze je menší než 5 (na jedno slovo v běžném textu).
Počítačová morfologická analýza však musí řešit ještě jeden problém, tzv. problém lematizace. Lematizace určuje pro každý slovní tvar jeho základní podobu (obvykle tvar, ve kterém slovo najdeme ve slovnících). Ani lematizace není obecně při zpracování izolovaného slova jednoznačná. Navíc je nutno rozlišovat mezi slovy, která jsou v základním tvaru homonymní - např. *stát* (jako státní útvar) a *stát* (jako sloveso). Počítačová lematizace proto ještě navíc tato slova rozlišuje a jednoznačně identifikuje (např. připojením číselného indexu k základnímu tvaru slova, např. *stát-1*, *stát-2* atd.).
Formálně tedy můžeme popsat morfologickou analýzu jako matematickou funkci, která posloupnosti znaků (písmen) jazyka přiřazuje množinu možných výsledků, složených vždy z dvojic <lema,značka>:

Ma(f) --> { <l,t>; l ∈ L, t ∈ T },

kde f je slovní tvar složený z písmen abecedy analyzovaného jazyka (např. *stát*), L je množina identifikací lemat (obvykle ve formě řetězce nějakých znaků, považovaného ovšem za nedělitelný) v daném případě bude jedním z možných výsledků např. *stát-1*), a T je množina značek používaná pro daný jazyk (jako např. **NNIS4-----A----**;).
Prakticky morfologická analýza pracuje s (tokenizovaným) textem, v dohodnutém formátu, a na výstupu je tentýž text obohacený o lemata a morfologické značky.

Vstup do morfologické analýzy - tokenizovaný text:

```
<f cap>Pekař
<f>peče
<f>housky
<D>
<d>.
```

Výstup z morfologické analýzy (zjednodušeno):

```
<f cap>Pekař<MMl>pekař<MMt>NNMS1----A----
<f>peče<MMl>péci<MMt>VeYS------A----<MMt>VB-S---
3P-AA---
<f>housky<MMl>houska<MMt>NNFP1-----A----
<MMt>NNFP4-----A----<MMt>NNFS2-----A----
<D>
```

```
<d>.<MMl>.<MMt>Z:-------------
```

Proces morfologické analýzy

Morfologická analýza, jejíž definici jsme uvedli v předchozí sekci, je ovšem realizována v počítači nikoli jako matematická funkce, ale jako výpočetní procedura. Jako základní datová struktura slouží pro daný přirozený jazyk jeho **morfologický slovník**, který je používán vlastním **algoritmem** morfologické analýzy (v zásadě pak již na jazyce nezávislým). Způsobů, jak efektivně provádět morfologickou analýzu, se používá několik, já zde popíši systém "přímé" analýzy. Ten potřebuje ke své práci jednak morfologický slovník, a samozřejmě i příslušný algoritmus, který vlastní morfologickou analýzu na základě slovníku realizuje.

Morfologický slovník

Morfologický slovník obsahuje ke každému lematu informaci o **kmeni** slova (v našem případě je za kmen slova považována ta část slova, která se při ohýbání nemění), a o přípustných koncovkách. Množina koncovek tvoří **vzor**. U každé koncovky je navíc informace o tom, které značky (kombinace hodnot morfologických kategorií) jí pro daný vzor odpovídají.

Příkladem vzoru je např. následující množina koncovek a jejich značek:

```
"" NNIS1-----A----, NNIS4-----A----
"u" NNIS2-----A----, NNIS3-----A----, NNIS6-----A-
--1
"e" NNIS5-----A----
"ě" NNIS6-----A----
"em" NNIS7-----A----
"y" NNIP1-----A----, NNIP4-----A----, NNIP5-----A-
---, NNIP7-----A----
"ů" NNIP2-----A----
"ům" NNIP3-----A----
"ech" NNIP6-----A----
```

Pro každý vzor je dále ve slovníku uvedeno, zda připouští negaci slova pomocí předpony "*ne-*" (tj. negaci) a u každé koncovky dále informace o tom, zda připouští připojení předpony "*nej-*" (stupňování). Pro velmi nepravidelná slova jsou pak ve slovníku uvedenu všechny jejich tvary i s příslušnými značkami.

Algoritmus morfologické analýzy

Tzv. "přímá" analýza slovních tvarů je založena na vyčerpávající analýze slova z hlediska možné segmentace na kmen a koncovku (případně i předpony *ne-* a *nej-*). Pro každou takto získanou dvojici kmene a koncovky je nutno ověřit, zda se ve slovníku vyskytuje jak kmen, tak i koncovka a

zda kmen i koncovka náleží ke stejnému vzoru. Všechny dvojice lemat (příslušných ke kmeni/kmenům) a značek (nalezených ve slovníku u příslušných koncovek) jsou pak prohlášeny za výsledek morfologické analýzy.
Příkladem může být slovo (slovní tvar) *housky*. Toto slovo je možno rozdělit na kmen *housky* + nulovou koncovku, nebo na *housk* + *y*, nebo na *hous* + *ky*, atd. až k *h* + *ousky* (kmen nulové délky se nepřipouští). Z těchto možností nakonec bude správná jen možnost *hous* + *ky*, neboť ve slovníku je neměnná část základu (zde jen *hous*, neboť 2. p. mn. čísla je *hous*+*ek*). Koncovky *y*, *sky*, a nulová koncovka jsou sice ve slovníku koncovek uvedeny také, ale kmen *housk (hou)* je nepřipouští (resp. nejsou uvedeny v seznamu koncovek pro vzor příslušný danému kmeni).

Značkování (zjednoznačňování morfologické analýzy)

Značkování (anglicky poněkud nevhodně nazývané "Part-of-Speech tagging") je v rámci popisu a zpracování jazyka pomocí rovin jakýsi "krok stranou": snažíme se totiž na úrovni morfologické analýzy o něco, co alespoň teoreticky přísluší až rovině syntaktické. Nicméně je to problém velmi praktický, jehož výsledky jsou použitelné ve třech směrech: jednak jako (zatím) finální krok při značkování korpusů pro lexikografické účely, dále jako krok výrazně zrychlující syntaktickou analýzu (byť do ní vnáší jistou míru chyb), a v neposlední řadě i pro některé aplikace, které mohou s výhodou využít i jen částečnou jazykovou analýzu (např. pro vyhledávání v elektronických slovnících, pro vyhledávání informací obecně, a dokonce i pro strojový překlad pro blízké flektivní jazyky).
Značkování již může využít pro zjednoznačnění výstupu morfologické analýzy (na rozdíl od ní samé) kontext, ve kterém se analyzované slovo nachází. Dnes se téměř výhradně používají pro značkování metody statistické, založené na strojovém učení. Počítač se tedy naučí, že po určitých předložkách následují jen některé pády, že na začátku věty nalezneme nejspíše pád první než jakýkoliv jiný, nebo že slovo *při* je téměř vždy předložka, jen velmi málokdy tvar slova pře, a téměř nikdy rozkazovací způsob od slovesa *přít* (a k tomu se, doufejme, naučí i to, kdy jde přeci jen o (soudní) *při*).
Jak se však může počítač takovou věc naučit? Potřebuje k tomu (alespoň v dosud nejúspěšnějších metodách) předem **ručně** označkovaný korpus. Takový korpus je samozřejmě velmi pracnou záležitostí; pro spolehlivé naučení, kdy procento chyb klesá (pro češtinu) pod 5%, bylo třeba označkovat přes 1.5 miliónu výskytů slov v textu (přitom každé zdvojnásobení tohoto počtu přinese jen několik desetin procenta zlepšení, a jistou hranici úspěšnosti zřejmě nelze překročit vůbec). Označkované korpusy jsou proto velmi cenným zdrojem lingvistických informací.
Učení z ručně označkovaného korpusu (takovému korpusu se říká **trénovací data**) může probíhat několika způsoby. Velmi jednoduchý a účinný (a dosud prakticky nepřekonaný) je postup, při kterém se spočítají relativní četnosti značek následujících po dvojici bezprostředně předcházejících značek v textu (takový způsob se nazývá **HMM tagging**).

Pro každou dvojici značek (tzv. **historii**) se tak vytvoří menší či větší tabulka, ve které jsou uvedeny relativní četnosti značek po ní následujících v trénovacích datech. Jakkoli je tento systém lingvisticky jasně neadekvátní, značkování založené na efektivním algoritmu aplikace těchto tabulek na kontinuální text dává velmi dobré výsledky: pro angličtinu se dosahuje i méně než 3% chyb na prakticky libovolném textu, pro češtinu pak okolo 5%.
Jako konkrétní příklad uveďme opět větu *Pekař peče housky*. Na základě vstupu z obr. 2 obdržíme následující výstup, ve kterém je pro každé vstupní slovo už jen jedna značka a jedno lema:

```
<f cap>Pekař<MDl>pekař<MDt>NNMS1----A----
<f>peče<MDl>péci<MDt>VB-S---3P-AA---
<f>housky<MDl>houska<MDt>NNFP4-----A----
<D>
<d>.<MMl>.<MMt>Z:-------------
```

U slova *Pekař* nebylo nutno rozhodovat o ničem, neboť již bylo jednoznačně určeno morfologickým analyzátorem[5]. Slovo *peče* je samozřejmě v této větě v přítomném čase a 3. osobě (nikoli jako přechodník!), a *housky* jsou zde ve 4. pádě množného čísla. A nyní je vše připravené pro vstup do roviny syntaktické.

V některých případech použití jako je např. strojový překlad podobných jazyků (čeština a slovenština) již není třeba syntaktické roviny, protože jazyky jsou si tak podobné, že není třeba analyzovat souvislosti mezi jednotlivými slovy a hledat jejich význam. Překlad spočívá v transferu dvojic <lemma, značka> pomocí tabulky, kdy každé této dvojici ve zdrojovém jazyce je přiřazena dvojice nebo několik dvojic v cílovém jazyce.

8.5. Rozpoznávání hlasu a řeči

V dnešní informační společnosti se předávání informací odehrává zpravidla mluveným slovem. Je to efektivní, přirozený a pohodlný způsob komunikace.
Zdrojem řečových kmitů jsou lidské řečové orgány. Ty se skládají z hlasivek, dutiny hrdelní, ústní a nosní, měkkého a tvrdého patra, zubů, jazyka a čelisti. Tyto orgány můžeme rozdělit do tří částí:

· *Ústrojí dýchací:* Řečový projev se realizuje při výdechu. Také je důležité vytvářet trvalý tlak vzduchu v hrtanu, které je důležitý pro vznik hlasu.

· *Ústrojí hlasové:* Nachází se v hrtanu, nejdůležitější částí jsou hlasivky. Ty fungují tak, že je proud vzduchu zkrze zúženou hlasivkovou štěrbinu zbržděn, tím dojde ke zvyšení tlaku vzduchu pod hlasivkami. Jeho působením se ale štěrbina rozšíří a dojde k průchodu nahromaděného vzduchu a tím ke snížení tlaku. To vede opět ke zúžení štěrbiny. Vzniká tak periodické kmitání. Velikost štěrbiny se dá hlasivkovými svaly

ovlivnit, což vede k ovlivnění periody kmitů hlasivek. Frekvence kmitaní hlasivek je různá a pohybuje se v rozmezí 150 až 400 Hz. Frekvence kmitů hlasivek F0 charakterizuje základní tón (tj. výška) lidského hlasu.

· *Ústrojí modifikační:* vytváří takzvanou šumovou složku hlasu. Skládá se ze tří dutin: *dutina hrdelní, dutina nosní, dutina ústní (*podílí se na realizaci každé hlásky – *pohybem jazyka, rtů a čelisti)*

Akustická skladba jednotlivých hlásek souvislé řeči je modifikována v rámci delších časových úseků (tj. slova, taktu, větného úseku, věty) tzv. **modulačními faktory,** jimiž se vytvářejí **prozodické rysy** promluvy - modifikace časové, intenzitní a melodické. **Prozodie** - zvuková stránka řeči, součástí je intonace, rytmus a prodlevy řeči – dodává řeči její přirozenost.

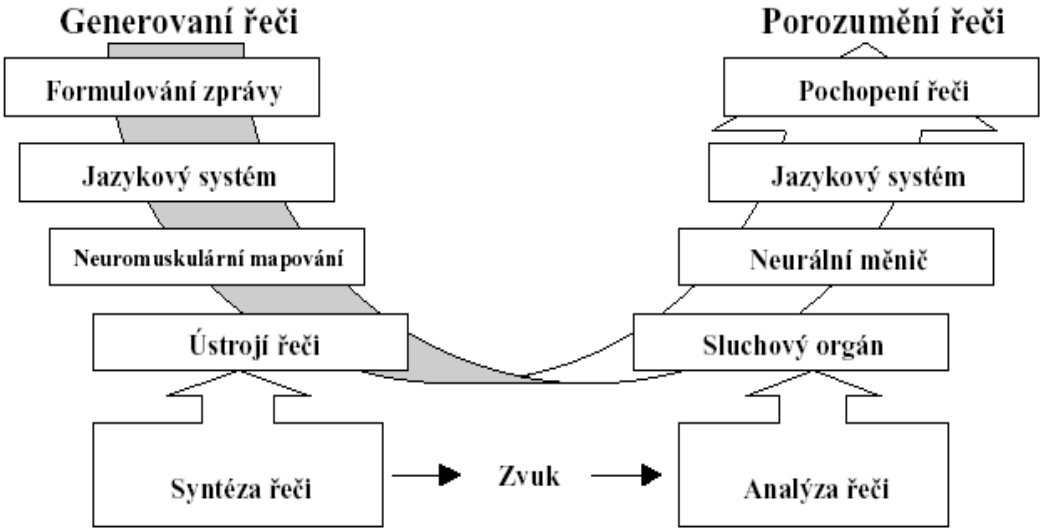
Popis řeči jako zvukové formy

Základní jednotkou popisu je **elementární zvukový segment**. Volba jeho délky ovlivňuje další zpracování řeči. Segment musí být dostatečně krátký, musí být schopen tvořit další jednotky tzv. **řetězením** (*concatenation*). Nutná je znalost vztahu segmentu k jazyku jako lingvistické formě. Při zpracování řeči lze kombinovat segmenty o různých velikostech.

Základní řečová jednotka je **foném.** Fónem označuje minimální fonetickou jednotku identifikující jednotlivé primitivní zvuky. Všechny odlišné fóny určitého fonému se nazývají **alofóny.** Kupříkladu pro foném [s] existuje množství alofónů [s], závislých na kontextu, jako např. ve slovech “sup”, “eso”, “osa”, “srp” atd. Foném se mění vlivem svého umístění mezi ostatními fonémy. Tomuto jevu se říká **koartikulace**. Je to vliv předcházejícího a následujícího fonému, souvisí s trváním, s tempem řeči a s intonací.

Počet fonémů v existujících světových jazycích se pohybuje od 12 do 60. V českém jazyce je jich 36, v anglickém 42 a např. v ruském 40. Fonémy se skládají do větších celků – slabik. Libovolná promluva je potom vlastně opakování různých slabik.

Přenos informací pomocí řeči



Přenos informací řečí se pro komunikaci člověk-člověk rozděluje na dvě části. První část je generování řeči (vytváření), druhá část je rozpoznání řeči (porozumění) viz obrázek. Mezi těmito dvěma celky vystupuje řeč (zvuk) jako médium pro přenos informace (informační kanál). Generování řeči se dále skládá z následujících částí:

1. *Formulování zprávy:* Vytvoření myšlenky, kterou chceme sdělit okolí.
2. *Jazykový systém:* Převedení myšlenky na smysluplnou posloupnost slov a vět včetně intonace atd.
3. *Neuromuskulární mapování:* Vytvoření artikulačních parametrů pro vokální ústrojí.
4. *Ústrojí řeči:* Z artikulačních parametrů se vytvoří řeč (zvuková vlna).

Proces porozumění obsahuje podobné části jako generátor řeči, jenom v opačném pořadí:

1. *Sluchový orgán:* Povede frekvenční analýzu zvuku (chová se jako banka filtrů).
2. *Neurální měnič:* Frekvenční charakteristiku převede na posloupnost rysů.
3. *Jazykový systém:* Z rysů sestaví slova a věty.
4. *Pochopení řeči:* Ze slov se pokusí vytvořit myšlenku, která měla být předána.

U zařízení je to obdobné.

Rozpoznání řeči

Faktory ovlivňující hlasový vstup

Hlasový vstup je ovlivněn mnoha faktory. První je řečová složitost vstupu. Na aplikace pro zadávání příkazů a ovládání (command and control) jsou kladeny nejmenší nároky. Jedná se o přístup, kdy člověk ovládá zařízení pouze jednoduchými (jedno nebo více slovnými) příkazy. Tyto příkazy se rozpoznají ze slovníkové databáze čítající několik desítek slov, na jejich základě se provedou případné povely a pokud to bylo vyžádáno, tak se v posloupnosti slov či v jednoduchých větách podá uživateli zpráva o výsledku příkazu. Vyšší nároky jsou kladeny na aplikace pro diktát. Tyto aplikace se musejí vyrovnat s plynulou řečí a nezávislosti na mluvčím. Nejvyšší nároky jsou na aplikace pro diktát s integrovaným porozuměním mluvenému jazyku.

Další hledisko je prostředí použití. Nejjednodušší je použití v tichém prostředí a pro jednoho mluvčího. Obtížnost stoupá s použitím plynulé řeči, nezávislosti na prostředí a na mluvčím.

Charakteristika akustického signálu

Informace obsažená v mluvené řeči je obvykle získávána mikrofonem. Pro další zpracování je zapotřebí tyto analogové kmity převést do číslicových údajů. Tento proces se nazývá **pulsní kódová modulace** nebo též **digitalizace a** zahrnuje provedení dvou kroků, a to vzorkování a kvantizaci s kódováním.

Důležitým předpokladem při zpracovávaní akustického signálu je fakt, že se nám jeho vlastnosti v průběhu času mění pomalu, což umožňuje dále signál zpracovávat pomocí metod **krátkodobé analýzy**, při níž se úseky řeči vydělují a zpracovávají tak, jako by to byly krátké, na sobě nezávislé zvuky (**mikrosegmenty)**. Délka mikrosegmentu se většinou pohybuje v rozmezí 10 – 30 milisekund. Výsledkem analýzy je potom číslo, či vektor čísel, který jistým způsobem charakterizuje daný mikrosegment.

Fáze zpracování signálu

Fáze předzpracování, jak už název říká, zahrnuje metody vylepšující zpracování signálu na vstupu ve dvou oblastech: 1) **zpracování v časové oblasti** a 2) **zpracování ve frekvenční oblasti.**
ad 1) Krátkodobá analýza – zpracování signálu na časovém intervalu, o němž se předpokládá, že na něm nedochází k výraznějším dynamickým změnám. Tento interval se, jak již bylo řečeno, nazývá **mikrosegment** (někdy také stručneji segment) a jeho velikost se obvykle od 10 do 40 ms.
ad 2) Nejčastěji používané postupy jsou založeny na aplikaci **krátkodobé diskrétní Fourierovy transformace** (resp. diskrétní Fourierova transformace (DFT), rychlá Fourierova transformace (FFT)).

Rozpoznávání signálu

Klasické metody rozpoznávání jsou založeny na matematických metodách, a to na

- **porovnávání obrazců** (*pattern recognition*) - na základě vzdálenosti mezi obrazem (slovo, slabika, segment…) a vzorem ze slovníku; vzdálenost je vyjádřená vektory vypočtenými pomocí:

LPC – lineární predikční kódování – nové vzorky řečového signálu odhadujeme jako lineární kombinace určitého počtu vzorků předchozích a vzorků buzení. Každý nový vzorek tak určujeme ze vzorků předchozích, predikujeme jej. proto predikční, bývá zařazována mezi metody zpracování signálu ve frekvenční oblasti
FFT – Fast Fourier Transform - FFT je speciálním případem diskrétní Fourierovy transformace (DFT) aplikovatelným pro případy, kdy délka mikrosegmentu je mocninou 2
KEPSTRÁLNÍ ANALÝZA - Vychází z modelu činnosti hlasového ústrojí. Kepstrum se často používá při stanovení základního hlasivkového tónu a pro klasifikaci řeči na znělé a neznělé segmenty

- **stochastickém modelování** – pomocí:

DTW *(Dynamic Time Warping – dynamické borcení času*) se používá pro porovnání dvou úseků promluv (např. slov), vyjádřených posloupností akustických vektorů, vzniklých rozdělením slov do mikrosegmentů a jejich klasifikací souborem krátkodobých charakteristik. Tato metoda ošetřuje časovou neshodu, tj. různou délku, dvou vstupů (slov) tím, že je převede na stejný časový úsek.
HMM (*Hidden Markov Model – skryté Markovovy modely*) Model představy: hlasové ústrojí je během krátkého časového intervalu (např. odpovídající době trvání mikrosegmentu) v jednom z konečně mnoha stavů artikulačních konfigurací, generuje hlasový signál, a přejde do následujícího stavu hlasového ústrojí, je počítána pravděpodobnost přechodů mezi jednotlivými mikrosegmenty. Během rozpoznání je posloupnost segmentů neznámého slova identifikována pomocí pravděpodobnostní matice přechodů, která je generována v etapě učení.Tato činnost je chápána statisticky. Výhodou tohoto přístupu je, že nejsou potřebné rozsáhlé fonetické znalosti, nevýhodou je nutnost pracovat s velkým množstvím řečového materiálu. Mezi stochastické modelování řadíme i umělé neuronové sítě. (UNS)
UNS – *Umělá neuronová síť* - rozpoznávání probíhá na základě podobnosti si jednotlivých vzorků promluvy. Čím méně se budou jednotlivé hodnoty vektorů lišit, tím větší bude pravděpodobnost zařazení do stejné skupiny neuronů. Jedná se tedy o jakousi "vzdálenost" mezi porovnávanými vektory. Pro účely zařazování navzájem si podobných vzorků se jeví jako nejvhodnější dvouvrstvá neuronová síť. Spodní vrstva je jakýmsi vstupem sítě. Tam je přivedeno např. celé, právě namluvené slovo. Síť tyto data zpracuje a na horní vrstvě (výstupní), která bude reprezentovat jednotlivé rozpoznané slova, se zobrazí výsledek. Tím je třída slov, kterým se vstup nejvíce podobá.

Syntéza řeči

Systémy, které pracují s hlasovou syntézou se již dnes staly nedílnou součástí běžného života, kdy nám např. "neživý" operátor sděluje výši našeho kreditu na mobilním telefonu či hlásí zpoždění vlaku. Jde také o efektivní způsob přijímání informací.

V úloze syntézy řeči se musíme soustředit na tři oblasti. Je to volba fonetických a lingvistických jednotek, způsob zpracování a vlastní syntéza akustických kmitů.

Princip syntézy řeči

Dnes nejrozšířenější metodou syntézy je tzv. „řetězení segmentů z řečového korpusu (databáze)". Pro kvalitní syntetickou řeč je třeba řetězit co největší segmenty (důvodem jsou plynulost a dobré prozodické vlastnosti), to ale klade nároky na velikost řečového korpusu a na rychlost prohledávání. Je třeba hledat kompromis, kterým je korpus složený z menších segmentů, které dovolují větší variabilitu, a to i za cenu horší prozodie. Tu je nutné „přidávat" k monotónní syntetické řeči instrukce „jak" spolu souvisí text na jedné a výslovnost, intonace a další vlastnosti mluvené řeči na druhé straně, zprostředkovává fonetická transkripce.

Syntéza řeči z textu

Jednou z řešených úloh je i syntéza řeči z textu (*Text-to-Speech Synthesis*, TTS), někdy nazývaná **konverze textu na řeč**. Je to nejen nejobecnější, ale také nejnáročnější způsob syntézy. Část posluchačů se mylně domnívá, že to, co slyší často z počítače, je skutečně syntetická řeč vzniklá převodem z textu. Ve většině případů se však jedná o tzv. „resyntézu" neboli reprodukci původní promluvy. Skutečný TTS systém je však schopen generovat promluvu z libovolného textu. Cílem výzkumu v této oblasti je vytvořit systém, který převede automaticky libovolný text (korespondence, e-maily, SMS-zprávy, novinové a časopisecké články, knihy apod.) na mluvenou řeč. Tato syntetická řeč však nesmí, ani po delší době, posluchače unavovat, ani od něj vyžadovat přílišnou pozornost, musí být tedy co nejvíce přirozená. Přirozenosti řeči napomáhá dobrá prozodie. Je třeba si však uvědomit, že syntéza je jazykově závislá. Postup při TTS syntéze je následující:

- zpracování textu (fonetická transkripce),
- navržení prozodických charakteristik,
- vyhledání odpovídajících řečových jednotek,
- generování syntetického akustického signálu.

TTS má řadu výhod. Je operativní, oborově nezávislá, je možné pracovat v reálném čase, s malými nároky na paměť a s vysokou srozumitelností. Její nevýhodou je špatná prozodie, zvláště při použití parametrického způsobu syntézy (na základě LPC resp. kepstrálních koeficientů), kdy nejsou zachovány ani charakteristické rysy mluvčího.

Fáze syntézy z textu

Prvním úkolem je **segmentace,** tj.volba akustické jednotky (foném, difón, trifón, …). Čím menší je zvolená akustická jednotka, tím více se projeví vliv nesprávné koartikulace (ovlivňování okolními řečovými jednotkami). Špatná koartikulace mezi slovy má vliv na plynulost řeči, koartikulace mezi slabikami ovlivňuje srozumitelnost. Čím větší jednotka, tím více je nutné mít variant promluv. Zvolené segmenty extrahované z řečového signálu se uloží v paměti a tím se vytvoří **inventář.** Vlastní syntéza je založena na spojování segmentů, na tzv. **řetězení** *(concatenation).* Při každé syntéze řeči nastává problém s větší či menší přirozeností prozodie. Zejména při použití parametrických metod dochází ke ztrátě informací, které se týkají melodie, tempa či dalších vlastností charakterizujících mluvčího. Dnes nám ale pouhá srozumitelnost (i když velmi dobrá) nestačí. Ke zkvalitnění syntetické řeči je nutné do syntezátoru přidat modul pro **modelování prozodie**.
Fonetická transkripce

Pro TTS syntézu musíme zpracovávat nejenom řečový signál, ale také text. Informace obsažené v textu se musí převést do formy vhodné pro počítačové zpracování. Zvukovou formu jazyka vyjadřuje **fonetická transkripce.**
Pojem transkripce se používá pro dvě úlohy: pro zápis spisovné výslovnosti (např. ve slovnících) a pro zápis reálného jazykového projevu (dětský – ďecký, výkvět – víkvjet, včera – fčera). Pro automatickou transkripci je vypracována mezinárodní fonetická abeceda (*International Phonetic Alphabet – IPA*) pro fonémy. Pro transkripci češtiny, podobně i pro jiné jazyky, není možné tuto mezinárodní transkripci použít beze zbytku. Je třeba ji modifikovat z ohledem na specifika výslovnosti českých hlásek. Výzkumná pracoviště v ČR, která se zabývají zpracováním řeči, vyvinula českou variantu IPA, která se nazývá SAMPA (vytvořen na katedře teorie obvodů FEL ČVUT v Praze).

Typy syntezátorů

Syntezátory řeči můžeme rozdělit na tři skupiny. Jsou to syntezátory založené na zpracování řečového signálu v kmitočtové oblasti, v časové oblasti a pomocí UNS.

Prvně jmenovaný způsob, **syntéza řeči v kmitočtové oblasti** vychází z modelování procesu vytváření řeči (matematický model artikulačního ústrojí). Kvalita syntetické řeči je sice dobrá, ale řešení je technicky náročné

Druhým typem je **syntéza řeči v časové oblasti**. Spojitý akustický signál je převeden na digitální kód s možností jeho rekonstrukce v určitém časovém okamžiku. Tento způsob syntézy je relativně jednoduchý, technicky nenáročný a produkuje kvalitní syntetickou řeč.V mnoha případech lze zachovat charakteristiky mluvčího (nejsou-li editovány). Řeč

je kódována, digitalizována a vzorky jsou uloženy v paměti. Po provedení syntézy řetězením je zpětně dekódována na analogový tvar. Nevýhoda syntézy založené na řetězení izolovaně promluvených a zakódovaných slov je především velikost slovníku, ale také nemožnost pracovat s koartikulací a s intonací věty. Metoda je vhodná pro méně rozsáhlé slovníky (např. pro slovníky profesně zaměřené), pro povely, příkazy a hlášení.

Zmíněné metody byly jen nástin do oblasti rozpoznání a syntézy řeči. Celá tato problematika je velice náročným tématem, kterým se neustále zabývají vědci ze špičkových ústavů (patří zde i liberecká laboratoř Speechlab), protože je stále co zdokonalovat. Použití výsledných aplikací totiž velice pomůže v některých oblastech života. Např. „pomalá" komunikace s PC z klávesnice, pomůcka pro nevidomé, informační systémy…

8.6. OCR

OCR je technologie převodu textu uloženého v bitmapovém formátu do formátu textového. OCR je speciálním případem vektorizace (Optical Character Recognizing), tedy rozpoznávání písma.Text uložený v bitmapě není chápán jako text, je to jen sada tmavých a světlých bodů v obrázku. OCR program tedy musí identifikovat v bitmapě různé tvary a porovnat je s předlohou a rozhodnout jaké písmenko, ten který shluk představuje. Situace je navíc zkomplikována tím, že texty bývají napsány v různých fontech a dokumenty bývají často nekvalitní. Zvláště xeroxované dokumenty bývají „zašpiněné", tzn. obsahují rozmazaná písmenka a šmouhy. Program se tedy musí snažit i určit zda tečka poblíž identifikovaného písmenka „c" je háček a nebo jen nějaké smítko. Novější trasovací programy pracují tak, že dokument procházejí několikrát za sebou a při posledních průchodech už spolupracují se spell-checkerem. Mnohé programy se umí „učit". Takže když chcete převést do textového formátu sadu dokumentů psaných na jednom psacím stroji, můžete OCR program naučit, že dotyčnému stroji ustřelovalo písmenko „z" a „k" bylo trochu rozmazané.

Vektorizace

Vektorizací rozumíme převod dat z rastrového formátu do formátu vektorového. Jedná se o úlohu obtížnou, neboť informací uložených v rastrovém formátu je méně, než informací uložených ve formátu vektorovém, a tak je potřeba nové informace automaticky generovat, nebo je ručně do dat doplnit.

- **Ruční**, tedy obsluhovaná uživatelem. Na zobrazovacím zařízení (monitoru) se zobrazí rastrový obrázek a podle něj uživatel zadává pomocí vstupního digitalizačního zařízení (tablet nebo myš) jednotlivé vektorové entity. Ruční vektorizace je nejpřesnější, ale

velmi zdlouhavá a náročná. Zvláštním přikladem ruční vektorizace může být přímá digitalizace papírové předlohy digitizérem bez použití mezistupně rastrového obrázku.

- **Automatická**, obsluhovaná programem. Program musí být schopen na základě informací o barvě jednotlivých pixelů určit základní entity, ze kterých se obraz skládá. V mnoha případech se pro takový postup používají metody příbuzné umělé inteligenci. Automatickou vektorizaci lze v současné době použít jen u jednoduchých a zřetelných předloh.

- **Poloautomatická** vektorizace je dnes nejčastější. Samotnou vektorizaci provádí program, který je v případě sporných situací korigován a opravován uživatelem. Kvalita vektorizace a její rychlost závisí na stupni automatizace.

Části OCR programu

V současné době se rozpoznávání ručně psaných znaků potýká s řadou problémů, které snižují potenciální popularitu OCR. Tyto problémy se týkají zejména odstraňování pozadí, korekce sklonu a velikosti písma, digitální způsob přemýšlení, který se liší od lidského. Některé z těchto problémů se podařilo do jisté míry odstranit tzv. předprocesními a poprocesními úpravami (preprocessing and postprocessing), které se vykonávají před nebo po samotném OCR. Je rozumné oddělovat fáze předprocesní a poprocesní od algoritmů OCR, a to z důvodu, že většina známých algoritmů OCR umí pracovat jen s černými znaky na bílém pozadí.

Preprocessing

Typickým příkladem předprocesních algoritmů je odstraňování vzorů na pozadí, vypreparování textu a srovnání šikmo psaného textu, korekce velikosti a sklonu znaků. Text je vlastně často psán na vzorovaném pozadí (papíře), například při posílání dopisů lidé často používají pěkně graficky zpracované obálky, proto by dobrý systém měl umět rozeznat text od okrasné grafiky. Lidé toto samozřejmě zvládnou díky přirozené intuici, ale podíváme-li se na tento problém z hlediska počítače, je to nelehký oříšek. Není výjimkou, že se celý blok textu rozpozná jako grafika a je tak vyloučen z rozpoznávací procedury.

Vypreparování textu (**skeletonizace**) je velice důležitou úlohou (možný postup je naznačen na obrázku), jelikož informace o tvaru je tou nejdůležitější pro algoritmus OCR. Skeletonizace může pomoci odstranit nepravidelnosti ve znacích, tím zjednoduší celý rozpoznávací algoritmus, který pak uvažuje jen znakové úhozy, jež jsou pouze pixel široké. Rapidně se taktéž sníží paměťové nároky na uchovávání informace o vstupních

znacích. Toto je jedna z hojně užívaných metod, protože mimo jiné zkracuje čas samotného rozpoznávání. Kostry se získávají tzv. ztenčovacími metodami nebo transformacemi vzdáleností.

Zjednodušení algoritmu OCR můžeme dosáhnout také tím, že algoritmus bude předpokládat, že všechna vstupní data jsou již normalizována na standardní rozložení (míněno, že text není „do kopce“ ani „z kopce“), velikost a sklon. Existuje řada úspěšných algoritmů, které umožňují vytvořit horizontální rozložení, otáčet písmena a utvořit konzistentní velikost. Algoritmy OCR, pak mohou dostávat normalizované tvary znaků. Jiným příkladem normalizace je případ, kdy může být znak psán více způsoby (např písmeno „a“).

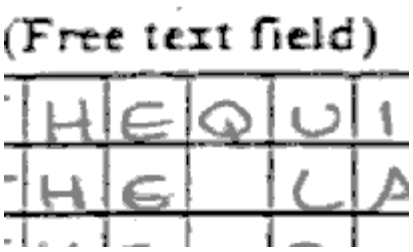
Postprocessing

Poprocesní zpracování je velice důležité, protože slouží k napravování chyb, kterých se případně dopustí algoritmus OCR. Nejtypičtějším případem poprocesního zpracování je kontrola pravopisu (spell checking), která automaticky opraví drobné chybičky a u větších chyb se zeptá uživatele na správný tvar. Musíme si uvědomit, že poprocesní systémy by neměly znehodnotit původně správně rozpoznaná data. Poprocesní systém opravující chyby OCR algoritmu, je možné napojit na jeho výstup.

Techniky získávání dat

Pořídíme-li obrázek běžným barevným scannerem, musíme provést práhování šedi (thresholding). Každému pixelu na pozadí přiřadí hodnotu 0 a pixelu na popředí hodnotu 1.

Jako první tuto metodu popsal **Ahmed.** Použil nedoručitelné dopisy z poštovního úřadu U.S.A. Obálky digitalizoval s rozlišením 166 dpi. Po digitalizaci provedl **segmentaci** číslic a převedl obrázek na binární strukturu.



Algoritmus hledá na obrázku sloupcovité skrumáže bílých pixelů (je proveden thresholding, máme pouze 2 barvy), které znamenají oddělení jednotlivých znaků. Pokud uvažujeme PSČ, pak jsou evidentně úspěšné ty případy, u kterých segmentace odhalila pět shluků (tzv. **blobs** - nepopsatelné objekty shluku informací), a tedy pětimístné PSČ. To nám zaručuje, že se dané vzorky skládají z jednotlivých shluků. Tento případ vylučuje přítomnost dvoutahové pětky, jelikož by se jednalo o dva shluky, což je vcelku nevýhoda.

OCR algoritmy

Nejčastěji používané techniky OCR jsou založeny na Markovových modelech, neuronových sítích, filtrech na extrakci tahů a bodů a srovnávacích metodách tvaru a vzoru. Markovovy modely zaznamenaly velký úspěch v rozpoznávání řeči, jsou to vlastně stavové stroje, které využívají kontextové informace. Počítače obecně nemají problémy s rozpoznáváním dobře napsaných znaků, které se moc neliší od daných vzorů, ale psané znaky jsou mnohdy víceznačné a nečitelné ani pro člověka. Lidé jsou schopni číst slova s nečitelnými znaky, a tak by to mělo být i u počítačů. Algoritmy založené na principu rozpoznávání znaku po znaku by na takovém slově neuspěly, ukazuje se, že kontextová informace je nejen užitečná, ale často i nutná.

Neuronové sítě nezadávají explicitně instrukce počítači, respektive metody na řešení problému, avšak bombardují síť tréninkovými vstupními daty, přičemž pokaždé je počítána chyba (odchylka) hodnot neuronů ve výstupní vrstvě. Základní myšlenkou celého tohoto algoritmu je zpětné šíření vypočtených odchylek do předcházejících vrstev. Čím více dat, tím větší a hlubší poznatky nabude systém a zvětší se tak šance na odstranění chybných vzorů. Neuronové sítě jsou tedy lépe připraveny na vstupy nízké kvality na rozdíl od klasických modelů. Vytrénované sítě jsou schopny vyvodit pravděpodobný odhad řešení, kdy se pak vytvořením patřičného výstupu stává důležitou složkou řešení daného problému.

Korespondenční úkol:

Zvolte si některou další zajímavou oblast aplikace umělé inteligence a zpracujte systematickou studii této oblasti. Studie by měla obsahovat motivaci oboru, základní přístupy a vymezení souvislostí s disciplínami a problémy UI naznačené v tomto textu. Zvolené téma konzultujte nejprve s tutorem.

Literatura

[1] Kelemen, J. a kol.: Základy umelej inteligencie. ALFA, Bratislava, 1992

[2] Mařík, V. a kol.: Umělá inteligence 1,2,3. Academia, Praha, 1993, 1995, 2001

[3] Sedláček, V.: Umělá inteligence I. UJEP Brno, 1983

[4] Russell, S., Norvig, P.: Artificial Intelligence. A Modern Approach. Prentice - Hall, 1995

[5] Peter Mikulecký, Daniela Ponce: Základy umělé inteligence. Studijní text. Katedra aplikované informatiky FŘIT VŠP Hradec Králové, 1996, 1997

[6] Pavel Houser: Paradoxy umělé inteligence: Turingův test 50 let poté, Computerworldu 11/2003.

[7] Jaroslav Zapletal: Kdo to byl Alan Mathison Turing? Scienceworld, 2001.

[8] Giarratano J., Riley G., Expert Systems, KENT Publishing Company, New York, 1989

[9] Nilsson N., Principles of Artificial Intelligence, Tioga Publishing Co., 1980

[10] Jan Hajič: Statistické modelování a automatická analýza přirozeného jazyka (morfologie, syntax, překlad). Slovenčina a čeština v počítačovom spracovaní (zborník referátov zo seminára Bratislava 26.-27.10.2001 (ed.A. Jarošová)), VEDA, vydavateľstvo SAV, Bratislava.

[11] Laboratoř zpracování přirozeného jazyka – http://www.fi.muni.cz/nlp/

[12] KUBÍK, A.:Agenty a multiagentové systémy, Slezská univerzita v Opavě, Opava, 2000