AKADÉMIA OZBROJENÝCH SÍL
GENERÁLA MILANA RASTISLAVA ŠTEFÁNIKA

KATEDRA INFORMATIKY

Študijný odbor: Elektronika a komunikačná technika
Špecializácia: Komunikačné a informačné technológie
Evidenčné číslo:

NÁVRH A OPTIMALIZÁCIA KYBERNETICKÉHO SYSTÉMU POUŽITÍM EXPERTNÝCH SYSTÉMOV

DIPLOMOVÁ PRÁCA

Vypracoval: **Ľuboslav IVANKO**
Vedúci diplomovej práce: **kpt. Ing. Michal TURČANÍK, PhD.**

Liptovský Mikuláš máj 2005

Ďakujem kpt. Ing. Turčaníkovi Michalovi, PhD. za tolerantný prístup, Ing. Staroňovej Jane za odbornú pomoc pri písaní diplomovej práce a Böhmerovi Marianovi za pomoc pri jej spracovávaní.

OBSAH

In the beginner's mind there are many possibilities,
in the expert's mind there are few.

Shunryu Suzuki

V mysli začiatočníka je mnoho možností,
v mysli experta je ich len pár.

Shunryu Suzuki

ÚVOD

Prostredie, v ktorom pôsobia organizácie, sa stáva čoraz zložitejším a komplexnejším. K efektívnemu rozhodovaniu sú potrebné čoraz lepšie znalosti, a to nielen „učebnicové“ teoretické, ale najmä praktické, získané skúsenosťou a dlhodobou praxou. Práve táto skupina znalostí je vlastná ľuďom, ktorých nazývame experti.

Cieľom tejto práce je demonštrovať použitie expertných systémov pri optimalizácii kybernetického systému. Expertný systém bude realizovaný ako program vo vyššom programovacom jazyku na osobnom počítači v prostredí Microsoft Windows. Kybernetickým systémom bude jednoduchý dvojkolesový robot so sústavou snímačov pohybujúci sa v bludisku. Úlohou expertného systému bude nájsť čo najkratšiu cestu týmto bludiskom. Rozhodovacia činnosť systému bude zabezpečená znalosťami, ktoré sa budú približovať znalostiam človeka. Systém možno považovať za úspešný, ak dokáže nájsť cestu bludiskom aspoň tak rýchlo ako človek.

Keďže robot nie je autonómna jednotka a jeho pohyb bude ovládaný z nadradeného počítača s expertným systémom, je nutné zabezpečiť výmenu informácií medzi robotom a počítačom. Komunikácia robota s počítačom bude realizovaná cez štandardné rozhrania počítača.

1 SÚČASNÝ STAV RIEŠENEJ PROBLEMATIKY

Približne v roku 1950 vznikol prvý program, ktorý imitoval ľudské myslenie. Program *Logic Theorist* obsahoval už dokázané axiómy a keď sa mu bol zadaný nový logický výraz, dokázal prehľadať všetky možné operácie a použitím heuristiky bol schopný objaviť dôkaz daného logického výrazu. Program bol schopný vyriešiť 38 z 55 matematických dôkazových problémov. Približne v tom istom čase boli vyslovené názory, že počítač by mohol dokázať hrať šach. Pár rokov potom – v r. 1956 vznikol termín *Umelá inteligencia.*

Prvou komerčnou aplikáciou Expertných systémov bol program *Dendral*, ktorý dokázal simulovať analytické myslenie experta – chemika a jeho schopnosť robiť rozhodnutia. Neskôr vzniklo množstvo ďalších programov využívajúcich expertné systémy, ktoré našli uplatnenie aj v ďalších vedeckotechnických odvetviach. [Kri96]

Po rozmachu expertných systémov v 80. rokoch minulého storočia je badať citeľný útlm v tejto oblasti výskumu a vývoja, spôsobený sklamaniami vyplývajúcimi z prehnaných očakávaní od tejto technológie. Expertné systémy sa však stále môžu uplatniť, pokiaľ sa bude na ich možnosti pozerať triezvo, bez prehnaných očakávaní a budú sa hľadať cesty na ich vhodné spojenie s inými technológiami, možno že aj diametrálne odlišnými. [Bab04]

Umelá inteligencia (AI – Artificial Intelligence) a technológia expertných systémov spolu s nástrojmi ako sú *genetické algoritmy* a *neurónové siete prinášajú* nové techniky pre simuláciu inteligencie pri tvorbe rozhodnutí, evolúcii a učení sa počítačov.

V neposlednom rade je nutné spomenúť aplikáciu AI v oblasti počítačových hier. Aj keď by sa mohlo zdať, že sa jedná iba o formu zábavy, práve komerčný vývoj počítačových hier dáva priestor pre nové aplikácie či metódy použitia umelej inteligencie. Práve v tejto oblasti je v súčasnosti vývoj v oblasti umelej inteligencie (a teda samozrejme aj ES) najmarkantnejší.

Ďalšou veľkou skupinou sú expertné systémy tvorené na zákazku, pre riešenie veľmi špecifických problémov. To že expertné systémy sú stále populárnym a silným prakticky použiteľným nástrojom ukazuje aj množstvo špecializovaných vývojových prostredí pre ich vývoj. Tieto vývojové prostredia uľahčujú vývoj a neskôr aj nasadenie expertných systémov v inžinierskej oblasti. Medzi najznámejšie z nich patria OPS5, OPS83, INSIGHT, Level 5 Object, Prolog, VP-Expert, Teknowledge a mnohé ďalšie. [Kri96]

2 CIELE PRÁCE

V rámci diplomovej práce:

1. Zoznámte sa s problematikou a spôsobmi realizácie kybernetických systémov.
2. Zoznámte sa s problematikou vytvárania expertných systémov.
3. Zoznámte sa so spôsobmi používania rozhraní číslicového počítača.
4. Navrhnite a prakticky realizujte kybernetický systém podľa zadania vedúcim diplomovej práce.
5. Navrhnite prostredníctvom jazyka C++ programový systém pre optimalizáciu kybernetického systému pomocou expertných systémov podľa zadania vedúcim diplomovej práce.

3 KYBERNETIKA

Kybernetika sa zaoberá kvantitatívnymi a štrukturálnymi zákonitosťami riadenia, oznamovania a kontroly samoregulujúcich sa sústav. Pod sústavou je potrebné si predstaviť takú, ktorá je schopná reagovať na podnety z okolia i zvnútra sústavy samotnej. Zvláštnym prípadom takejto sústavy môže byť aj živý organizmus.

Kybernetika, ako veda o riadení je základným kameňom teórie automatizácie, zasahuje do mnohých vedeckých oblastí a vytvára v nich svoje nové zvláštne odvetvia.

Kybernetika je vedná disciplína, ktorá skúma riadiace a regulačné procesy v biologickej sfére, v technike a v spoločnosti a navrhuje modely na znázornenie, transformáciu a spracovanie informácií. Všetky automatické zariadenia na spracovanie dát sú v tomto zmysle kybernetickými strojmi a samotná informatika je náuka o kybernetických strojoch a metódach.

Kybernetika neskúma úplne všetky systémy, ale len tie, v ktorých sa uplatňuje proces riadenia. Kybernetické systémy sú teda zároveň riadiacimi systémami. Môžu byť biologické, technické, spoločenské aj iné. Tieto systémy obsahujú prvky, medzi ktorými prebieha vzájomná výmena predmetov, energie, informácií a podobne.

Pre riadenie a kybernetiku je podstatná výmena informácií, pričom každá informačná výmena je viazaná na fyzikálneho (látkového či energetického) nositeľa. Každý riadiaci proces prebieha na základe prijímania, prenosu, ukladania a spracovania informácií.

Kybernetika dokázala vysvetliť problém účelného a cieleného chovania umelých aj prírodných systémov. Všade, kde sa prejavuje cielené chovanie, kde sa napriek rušivému vplyvu prostredia usilujeme o dosiahnutie určitého stavu, musia existovať spätnoväzobné mechanizmy. Sú to funkcie usmerňovania systému, pri ktorých je určitá veličina trvale sledovaná a porovnávaná s riadiacou veličinou, a ovplyvňovaná tak, aby sa približovala k tejto riadiacej veličine. Popri tomto na systém pôsobia vzájomné vzťahy vonkajších vplyvov a vnútorných stavov systému.

Kybernetika sa delí na teoretickú a aplikovanú. Tieto však nie sú od seba prísne oddelené, naopak prelínajú sa a dopĺňajú.

Teoretická kybernetika sa zaoberá teóriou informácií, teóriou systémov, teóriou algoritmov, teóriou dát, teóriou hier, teóriou automatov, teóriou učenia a teóriou riadenia.

V aplikovanej kybernetike sa ako základ používajú výsledky práce teoretických disciplín, avšak tieto teoretické nástroje sa aplikujú na technické, reálne systémy s praktickým využitím v bežnom živote. Prenos je obojsmerný, pretože aplikovaná kybernetika kladie spätne nové požiadavky na skúmanie v teoretických oblastiach ako sú technická kybernetika, robotika, ekonomická kybernetika, organizačná kybernetika atď. [Kyb05]

3.1 Metódy kybernetiky

Keď sa hovorí o metódach kybernetiky, rozlišuje sa systémový prístup a modelovanie. Systémový prístup je metodologický smer vo vede, ktorý si kladie ako hlavnú úlohu vypracovávať metódy skúmania a konštruovania zložito organizovaných objektov - systémov rozličných typov a tried.

Systémový prístup je spôsob myslenia, resp. riešenia problémov, pri ktorom sa javy chápu komplexne v ich vnútorných a vonkajších súvislostiach.

Pri systémovom prístupe sa definuje ako systém jednak predmet skúmania, jednak proces skúmania, a to so zameraním na vnútorné i vonkajšie väzby medzi prvkami (úlohami) systému. Na základe systémových vzťahov sa v závislosti od zvolenej rozlišovacej úrovne rozdeľuje systém na jednotlivé časti, pri skúmaní ktorých sa vychádza zo systému ako celku, vrátane jeho cieľov. Predmet skúmania sa musí chápať dynamicky, to znamená, že sa berú do úvahy aj zmeny stavu, správania i štruktúry systému.

Pri modelovaní ide o skúmanie objektov pomocou iných, spravidla umele konštruovaných objektov, v ktorých sa vyjadrujú, charakterizujú a definujú iba vybrané vlastnosti, stránky a vzťahy originálneho objektu. Reprodukcia charakteristík určitého objektu prebieha na inom objekte, špeciálne vytvorenom na ich štúdium. Tento druhý objekt sa nazýva modelom. [Fil05]

3.2 Významové útvary kybernetiky

Informácia je obraz jedného objektu v druhom využívaný pri formovaní riadiaceho zásahu.

Regulácia je udržiavanie určených veličín systému riadenia na vopred predpísaných, najčastejšie konštantných hodnotách, pričom sa v priebehu regulácie zisťujú skutočné hodnoty týchto veličín a porovnávajú sa s predpísanými hodnotami. Podľa zistených odchýlok sa do procesu zasahuje tak, aby sa vzniknuté odchýlky odstránili, alebo aspoň minimalizovali.

Stav systému je množina definovaných podmienok a údajov, ktoré charakterizujú skúmaný systém a spolu so vstupom systému vymedzujú jeho výstup. Stav k určitému časovému okamihu sa určuje vstupom a stavom systému v predchádzajúcom časovom okamihu. Z hľadiska správania systému sa rozlišuje rovnovážny a nerovnovážny stav.

Spätná väzba je návrat malej časti energie z výstupu nejakého systému s cieľom poskytnúť systému informácie, pomocou ktorých možno výstup ovládať alebo riadiť. [Fil05]

3.3 Robotika

Robotika je vedné odvetvie, ktoré sa zaoberá robotmi. Definícia robota však už nie je tak jednoduchá. Najjednoduchšia definícia pravdepodobne znie: „Robot je stroj, ktorý je možné preprogramovať.” Práve fakt, že robot je preprogramovateľný je veľmi dôležitý. Ďalšia dôležitá vlastnosť robota je to, že by mal byť schopný robiť skoro všetko to čo človek. Súčasné roboty síce dokážu iba veľmi málo, avšak vývoj postupuje nezadržateľne vpred.

Robotika sa zaoberá mechanickým a elektrickým inžinierstvom, teóriou riadenia, programovaním a v súčasnosti aj umelou inteligenciou. Najväčší dôraz však kladie na matematické modelovanie pohybu fyzikálnych telies. Na celom svete je v súčasnosti vynakladané veľké úsilie o zostrojenie androida so strojovým videním, teda robota, ktorý by mal vzhľad a simuloval chovanie sa človeka. [Sel92]

Teóriu robotiky a robotov bližšie rozoberá Šolc a Žalud. [Sol02]

4 ZNALOSTNÉ A EXPERTNÉ SYSTÉMY

4.1 Definícia expertných systémov

Hoci počítač, najzložitejší a zároveň najvšeobecnejší stroj vymyslený človekom, je univerzálne zariadenie na spracovanie symbolov, doteraz sa využívajú prevažne len jeho schopnosti obrovskou rýchlosťou vykonávať aritmetické operácie alebo vytvárať a prehľadávať súbory údajov. Výskum v oblasti umelej inteligencie sa zameriava na skúmanie prostriedkov počítačovej (symbolovej) reprezentácie poznatkov a znalostí, ako aj metód ich používania. Vedie to k rozvoju programových prostriedkov, ktorými je možné obohacovať spôsobilosť počítača riešiť problémy postupmi založenými na využívaní znalostí. Expertné systémy sú jedným z konkrétnych prejavov realizácie takýchto spôsobilostí na súčasných počítačoch.

Vysloviť výstižnú definíciu expertného systému nie je ľahké. Popper [Pop89] uvádza nasledujúce definície:

- Expertný systém je počítačový systém hľadajúci riešenie problému v rozsahu určitého súboru tvrdení alebo istého zoskupenia znalostí, ktoré boli formulované expertmi pre danú špecifickú aplikačnú oblasť.
- Expertný systém je systém založený na reprezentácii poznatkov expertov, ktoré využíva pri riešení zadaných problémov.
- Expertný systém je systém kooperujúcich programov na riešenie vymedzenej triedy úloh, v jednotlivých problémových oblastiach zvyčajne riešených expertmi.
- Expertný systém je počítačový systém vybavený znalosťami odborníka (experta) zo špecifickej oblasti, v rozsahu ktorých je schopný uskutočňovať rozhodnutia rýchlosťou a kvalitou vyrovnávajúcou sa najmenej priemernému špecialistovi.

4.2 Princípy programovania „mysliacich počítačov“

Pri riešení akéhokoľvek problému je najdôležitejšie *definovať si problém*. Keď je problém definovaný, je potrebné zvoliť *vhodnú techniku riešenia.* Práve v tomto bode musí byť problém dôkladne analyzovaný a musia byť pochopené všetky súvislosti a vlastnosti, ktoré ovplyvňujú riešenie. Je nutné poznamenať, že aj keď spôsobov riešenia môže byť niekoľko, jeden z nich dokáže vyriešiť problém najefektívnejšie. Všetky programy AI využívajú

vedomosti na to aby vyriešili konkrétny problém. Keďže riadiace stratégie sú spoločné a môžu využívať vedomosti na riešenie problémov v rôznych oblastiach, musia byť vedomosti reprezentované samostatne, tak aby k nim bolo možné ľahko pristupovať, a aby bolo možné ich jednoducho upravovať počas riešenia problému. Nakoniec je len potrebné zvoliť čo najefektívnejšiu riadiacu stratégiu.

Podstatou všetkých expertných systémov je určitý druh programov. Spôsob ich vytvárania a ich princípy, na základe ktorých sa realizujú, majú svoje osobitosti. Pri výraze „mysliace počítače“ vzniká predstava, že sú to stroje, ktoré dokážu vykonať viac, ako mohol programátor pri vytváraní zodpovedajúcich programov predvídať, na čo pri svojej činnosti myslel. Človek, to čo vie, dokáže spravidla efektívne situačne používať a naviac dokáže svoje vedomosti a zručnosti rozšifrovať, prehlbovať a zdokonaľovať. Obdobné spôsobilosti sa očakávajú aj od fyzikálneho systému, ktorý si nárokuje na označenie „mysliaci počítač“.

Tradičná predstava o prednostiach počítačov sa jednoznačne viaže na ich veľmi rýchlu, presnú a dokonalú činnosť, zároveň však aj na ich neschopnosť vykonať viac než to, čo možno očakávať na základe programov, ktorými sú vybavené. Druhú časť tejto tradičnej predstavy je však potrebné prekonať.

Existujú tri princípy, alebo štýly programovania, ktoré by mohli pomôcť tomu, aby počítače dokázali vykonať oveľa viac, ako je im predpísané pri tvorbe programov. Tieto tri štýly programovania možno označiť: *„vykonaj teraz“, „vykonaj, keď môžeš“ a „vykonaj čosi zmysluplné“.* Prvý prípad zodpovedá klasickému štýlu procedurálneho programovania, druhý štýlu situačných programov vychádzajúcich z princípu deklaratívneho programovania, a tretí princípom evolučných, t.j. samo sa zdokonaľujúcich programov.

Klasický procedurálny štýl programovania spočíva na jednoduchom princípe. Program je tvorený postupnosťou príkazov, z ktorých niektoré môžu zodpovedať opakovanému vykonávaniu činností – cyklom. V zjednodušenej slovnej podobe možno takýto program vystihnúť takto: „Urob toto. Potom urob tamto. Urob to a potom toto, pokiaľ nenastane tamto.“ Takýto štýl programovania núti tvorcu programu predstaviť si do všetkých podrobností situácie, do ktorých sa spracúvanie údajov môže dostať, a spôsoby ako sa v týchto situáciách bude správať proces zodpovedajúci programu. Riadiace princípy sú veľmi jednoduché a možno ich slovne formulovať takto: *ak ďalší krok postupu nie je explicitne definovaný programovou konštrukciou, tak vykonaj nasledujúcu inštrukciu.*

Základná predstava, na ktorej je založený *princíp deklaratívneho (situačného) programovania* je iná. Neorientuje sa na procedurálne hľadiská, ale namiesto nich kladie dôraz na *zákonitosti* (pravidlá) riešenia jednotlivých problémov, ktoré oddeľuje od vlastných riešiacich procesov. Vyjadruje ich v niektorom z reprezentačných (najčastejšie logických) formalizmov. Nevyhnutným komplementom deklaratívnych programov je *interpretátor,* ktorý

v procese riešenia problému interpretuje symbolovo vyjadrené zákonitosti, spolu s ich dôsledkami, ktoré z nich vyplývajú.

Už v prípade jednoduchých programových prostriedkov stačí, keď programátor vhodnými pravidlami predpíše akcie (operácie či ich zoskupenia) zodpovedajúce jednotlivým situáciám, do ktorých sa riešenie problému môže dostať. Nemusí pritom predvídať, kedy takéto situácie nastanú. Vyspelejšie prostriedky situačných programov umožňujú ešte viac: programátor nemusí predvídať ani len jednotlivé možnosti riešenia problému. Stačí ak dokáže formulovať úlohu natoľko jednoznačne, aby použitému programovému systému bolo jasné, *čo* sa má riešiť, bez toho aby musel formulovať, *ako* dospieť k výsledku. Systém sám odvodí vzájomné väzby medzi možnými situáciami a akciami, ktoré je v nich účelné vykonať.

Pri situačnom riešení problémov sa teda nevyžaduje, aby boli vopred deterministicky určené jednotlivé akcie a poradie ich vykonávania. Stanovujú a vykonávajú sa vtedy, keď systém dospeje k podmienkam (situáciám), ktoré ich vykonávanie vyžadujú alebo umožňujú.

Existujú taktiež programy, ktoré sa dokážu učiť a usudzovať nové fakty na základe analógií. V súvislosti s tvorbou takýchto programov hovoríme o štýle resp. princípe *evolučného programovania*. Slovná podoba vyjadrujúca tento štýl by mohla znieť napr. takto: “Ak je zadaný problém analogický s problémom, ktorý si v minulosti úspešne vyriešil metódou *M*, pokús sa túto metódu použiť aj pri riešení daného problému.” Princíp takéhoto prístupu tkvie v schopnosti naučiť sa rozpoznať, ktoré predchádzajúce “skúsenosti“ (uchované v pamäti) sú na základe najvhodnejšej analógie použiteľné pri riešení daného problému. [Pop89]

Prostriedky umožňujúce realizovať systémy situačného riešenia problémov vytvárajú jeden z hlavných predpokladov vzniku expertných systémov.

Väčšina netriviálnych programov, ako je známe, sa skladá z určitého počtu podprogramov. Tvorba klasických programov vyžaduje od programátora presne predvídať spôsoby ich vzájomného volania a spôsoby vzájomného odovzdávania údajov prostredníctvom programových premenných. Toto vedie k určitej nepružnosti programov. Metódy situačných programov využívajú iný princíp, a to *spontánne procedúry riešenia problémov*. Zodpovedajúce procesy sa realizujú procedúrami bez priameho volania. Namiesto toho, aby sa procedúry vzájomne volali svojimi menami, k ich volaniu dochádza následkom výskytu istých obsahov údajových štruktúr charakterizujúcich cieľ alebo pod ciele, ktoré sú jednotlivé procedúry schopné dosiahnuť.

Iným dôležitým princípom umožňujúcim situačné riešenie problémov je *rozklad problémov* na jednoduchšie podproblémy. Rozklad problému na podproblémy môže byť často významný z hľadiska *hierarchických postupov riešenia:* problém sa rieši najprv iba zhruba a keď je takto riešiteľný, riešenie sa postupne zjemňuje, detailizuje.

Ďalším mechanizmom, ktorý je v programových systémoch situačného riešenia problémov včlenený je mechanizmus *prehľadávania* možností a im *zodpovedajúcich riešiacich postupov.* Mechanizmy *prehľadávania* umožňujú systémom do značnej miery riešiť problémy samostatne, t.j. bez toho, aby sa im špecifický deterministický postup vopred predpísal. Medzi najelementárnejšie postupy tohto druhu patrí *exhaustívne* t.j. určitou metódou určené *systematické prehľadávanie* všetkých štruktúr, ktoré prichádzajú do úvahy. Takéto prehľadávanie je však vzhľadom na jeho mimoriadne vysoké časové a pamäťové nároky len zriedkakedy reálne použiteľné. Preto sú *exhaustívne postupy* nahradzované *postupmi cielenými.*

Jedným z veľmi rozšírených prostriedkov reprezentácie zákonitostí, t.j. poznatkov a znalostí z problémovej oblasti, je vyjadrenie vzťahov medzi situáciami v riešení problémov a zodpovedajúcimi akciami v tvare explicitne formulovaných *situačno-akčných pravidiel*. Sú to pravidlá, ktorých charakteristická štruktúra má takýto tvar:

Ak vznikne situácia *S,* **tak** vykonaj akciu *A.*

Dostatočne veľký počet takýchto pravidiel, pokiaľ sú konzistentné (t.j. neprotirečivé) a úplné (t.j. umožňujúce nájsť riešenie, pokiaľ existuje) po určitom počte cielených pokusov ich použitia, dáva možnosť dospieť k správnemu vyriešeniu daného problému. V netriviálnych prípadoch, môže však vyžadovať neprípustne veľký počet exhaustívnych (alebo náhodných) pokusov hľadania vhodnej postupnosti aplikovateľných pravidiel, čo môže byť aj s pomocou najvýkonnejších počítačov nepoužiteľné.

Ak je však možné pravidlá na základe ich vlastností *hierarchizovať* a (situačne) *štrukturovať,* potom je možné exhaustívne alebo náhodné riešiace postupy obísť. [Pop96]

4.3 Znalostné systémy

Jedným z príznakov ľudskej spôsobilosti riešiť problémy charakterizovateľné nedeterministickými postupmi je používanie heuristík. *Heuristika* je návod na rozhodovanie v nedeterministických krokoch riešiaceho postupu. Často umožňuje nájsť správny postup, no nezaručuje ho. To znamená, že taký návod môže aj zlyhať. Neexistuje teda metóda, ktorá by dokázala ohodnotiť vhodnosť použitia heuristiky. Na druhej strane, použitie rôznych heuristík v kontexte s primeraným objemom štruktúrovaných *poznatkov* môže nedeterminizmus natoľko výrazne znížiť, že zvyšujúci ohraničený počet alternatívnych postupov možno už aj "naslepo" preverovať, hoci aj na základe "princípu pokusu a omylu".

Pod pojmom *poznatok* obvykle rozumieme reprodukciu určitej vymedzenej časti objektívneho sveta vrátane zákonitostí, ktoré v ňom platia.

Znalosti sú viac než len statické zoznamy poznatkov. Pod týmto pojmom rozumieme vzájomne previazané (meniteľné, doplniteľné) štruktúry súvisiacich poznatkov. *Znalosť* niečoho znamená vlastniť tomu zodpovedajúcu reprezentáciu v podobe dostatočne verného a presného kognitívneho modelu vrátane spôsobilosti vykonávať s tým, čo je reprezentované, rôzne kognitívne operácie.

Znalosti sú tvorené *deskripciami* (opismi) entít, *reláciami* (vzťahmi) medzi nimi a *procedúrami*, ktoré s nimi možno vykonať.

Deskripcie identifikujú a rozlišujú entity a ich triedy. Sú tvorené vetami určitého jazyka, ktorých prvky sú tvorené primitívnymi pojmami.

Relácie vyjadrujú závislosti a asociácie medzi entitami. Spravidla vyjadrujú definičné, empiricko-asociatívne, štrukturálne, priestorové, časové, príčinné, funkčné, ohraničujúce sa regulačné vzťahy, ako aj úlohy, v ktorých sa majú alebo môžu vyskytovať.

Procedúry špecifikujú operácie, ktoré sa v priebehu riešenia problému majú vykonať. Ide o používanie a interpretovanie deskripcií a relácií pri špecifických aplikáciách znalostí.

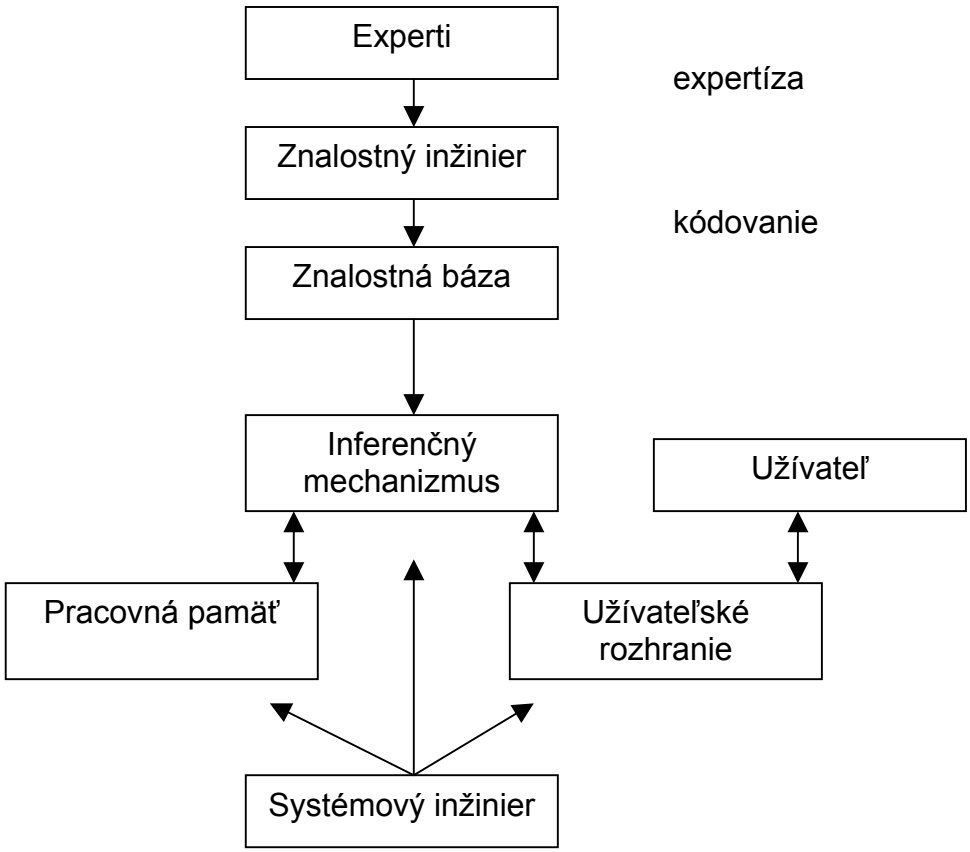
Počítače vybavené programami umožňujúcimi riešiť problémy na základe poznatkov produktívnymi postupmi, teda také, ktoré sú vybavené istou znalosťou, sa nazývajú *znalostné systémy*. Pre znalostné systémy sú podstatné znalosti. Tieto musia byť vyjadrené v tvare *symbolických* výrazov zodpovedajúcim deskripčným a relačným zákonitostiam.

Rozsah možností vystrojiť počítače znalosťami je podmienený tým, či a do akej miery sa nám darí vytvárať vhodný konceptuálny systém na opis problematiky a formu jeho symbolovej reprezentácie. Poslaním znalostných systémov je pomáhať pri riešení problémov, ktoré, hoci pre ne nepoznáme algoritmické riešiace postupy, sú riešiteľné použitím produktívnych metód využívajúcich znalosti. Pre znalostné systémy na riešenie odborných problémov postupmi vyžadujúcimi na rozdiel od všeobecných predovšetkým špecializované znalosti sa zaužívalo osobitné pomenovanie *expertné systémy.*

Expertné systémy sú teda, vzhľadom na špecifickú povahu v nich zahrnutých odborných poznatkov, považované za podtriedu znalostných systémov. Ich špecifickosť spočíva v tom, že používajú znalosti a riešiace postupy, ktorých zdrojom sú odborníci, spravidla experti, špecialisti vo svojej profesii. Expertné systémy sa vytvárajú na riešenie problémov, ktorými sa zvyčajne poverujú odborníci. Súčasťou „inteligencie“ expertného systému je aj jeho spôsobilosť vysvetliť a zdôvodniť svoje riešiace postupy na základe použitých znalostí. Je to významná vlastnosť, ktorá umožňuje používateľovi expertného systému posúdiť úroveň expertízy stelesnenej systémom a na základe toho sa s jeho riešením stotožniť, modifikovať ho, alebo ho odmietnuť. Vďaka tomuto, na rozdiel od klasických programov a programových

systémov, prestáva byť činnosť expertného systému pre používateľa „čiernou skrinkou“. [Pop96]

4.4 Architektúra expertných systémov



Obr. 1 Štruktúra expertného systému

Štruktúru expertného systému, jeho vznik a prevádzku znázorňuje obr. 1. *Experti* z danej oblasti dávajú expertízou dohromady súbor znalostí. Znalostný inžinier tieto znalosti dá do formy zrozumiteľnej programu. Inferenčný mechanizmus bude potom vyberať vhodné znalosti a bude radiť užívateľom. [Rac95]

Expertný systém je realizovaný (rozsiahlou) sústavou kooperujúcich programov, ktorých činnosť sa opiera o špecifické údajové štruktúry. Jednotlivé programové celky tejto sústavy sú prvkami funkčne vymedzených a svojim poslaním odlišných modulov.

Vzájomné väzby medzi modulmi a ich programovými celkami vytvárajú *architektúru expertného systému*.

Obr. 2 Základné zložky expertného systému

Na obr. 2 sú znázornené tri základné zložky tvoriace *minimálnu konfiguráciu* expertného systému. Sú to:

- BU - báza údajov resp. báza faktov;
- IM - inferenčný (odvodzovací, riešiaci) mechanizmus;
- BZ - báza znalostí.

Inferenčný mechanizmus je tvorený systémom kooperujúcich programov zabezpečujúcich procedurálnu zložku činnosti expertného systému. Báza znalostí a báza údajov sú pasívne údajové štruktúry. Toto členenie odráža kľúčovú myšlienku deklaratívneho (situačného) programovania, tvoriaceho jeden zo základných princípov expertných systémov. Báza znalostí a báza údajov nemusia byť vzájomne oddelené a v niektorých systémoch sú tieto pasívne systémy realizované jedinou údajovou štruktúrou.

Báza znalostí však predstavuje štruktúru údajov reprezentujúcich všeobecne platné a prijímané poznatky o pravidlách a zákonitostiach, kým báza znalostí je nositeľkou konkrétne zadaných či odvodených faktov, alebo predpokladaných údajov o špecifickom probléme, ktorý je predmetom riešenia. V praxi sa mnohokrát uchováva báza znalostí a báza údajov fyzicky rôzne. Kým báza údajov je zvyčajne v operačnej pamäti, báza znalostí sa často nachádza na diskovom pamäťovom médiu. [Pop89]

4.4.1 Inferenčný mechanizmus

Hlavnou časťou každého expertného systému je odvodzovací (inferenčný) mechanizmus. Tento mechanizmus určuje spôsob využívania znalostí uložených v BZ a ich aplikácie na údaje v báze údajov. Vždy keď expertný systém zaháji inferenčný proces, je nutné aby

fakty, ku ktorým dospeje niekam uložil, tak aby ich bolo možné neskôr použiť. Od kvality inferenčného systému závisia vlastnosti celého expertného systému. [Bab04]

IM je väčšinou realizovaný ako samostatný modul a zabezpečuje interpretáciu znalostí na konkrétny prípad. Umožňuje podľa dopredu známeho mechanizmu (tzv. riadiacej stratégie používania znalostí) používať znalosti z bázy znalostí na analýzu údajov nachádzajúcich sa v BU ako aj na odvodenie výsledného riešenia (väčšinou rozhodnutia).

Základné funkcie inferenčného mechanizmu sú realizované dvoma metódami, alebo ich kombináciou. Nazývajú sa *spätný* a *priamy chod.* [Pop89]

Spätný chod spočíva v tom, že užívateľ zadá nejakú hypotézu a expertný systém sa snaží rozkladať túto hypotézu na podproblémy tak dlho, kým tieto nie sú elementárnymi faktami, ktoré pozná, alebo elementárnymi faktami, na ktoré sa môže opýtať užívateľa. Takýmto spôsobom vznikne strom, zložený z:

- koreňových uzlov, ktoré reprezentujú hypotézy (ciele);
- medziľahlých uzlov, ktoré reprezentujú dielčie hypotézy;
- listových uzlov, ktoré reprezentujú elementárne fakty a otázku pre užívateľa.

Spätný chod teda zodpovedá cieľom riadenému odvodzovaniu: riešenie problému spočíva v nachádzaní vhodného a efektívneho spôsobu dosahovania určitého, vopred stanoveného cieľa. Efektívne riešiace postupy sa musia *hľadať*, a musia sa *produkovať*. V prípade spätného chodu to znamená, že od daného (zvoleného) cieľa je potrebné spätne odvíjať požiadavky jeho dosiahnuteľnosti. Keďže nie je väčšinou možné rozhodnúť o splniteľnosti všetkých požiadaviek, určuje sa splniteľnosť alternatívnych podmienok určením nového podcieľa, alebo podcieľov. [Rac95]

Priamy chod zodpovedá údajmi riadenému odvodzovaniu: riešenie vyplýva z určitého objemu známych faktov, ktoré je potrebné interpretovať. Je teda potrebné odvodiť čo z nich vyplýva, k akým dôsledkom je možné na ich základe dospieť.

Priamy chod je výpočtovo omnoho zložitejší ako spätný chod. Na druhej strane priamy chod umožňuje odvodiť všetky zmysluplné dôsledky vyplývajúce so známych faktov. Pri spätnom chode, determinovanom vopred explicitne daným cieľom, to tak nie je – odvodzovanie je zamerané len na stanovený cieľ.

V pracovnej pamäti programu s priamym chodom sú pravidlá tvaru

ľavá strana (ĽS) => pravá strana (PS)

kde ĽS je súbor podmienok a PS je súbor akcií. Na tieto pravidlá sa potom opakovane aplikuje nasledujúci postup:

1. Výber pravidla, ktorého ĽS zodpovedá súčasnému stavu pracovnej pamäti.
2. Vykonanie PS pravidla (toto väčšinou zmení stav operačnej pamäte).
3. Opakuj kroky 1 a 2, pokiaľ to je možné.

Tento postup dokáže pri vhodne formulovaných pravidlách vyriešiť zadaný problém. [Rac95]

4.4.2 Báza znalostí

Bázu znalostí tvorí znalostný inžinier. Tento pri tvorbe expertného systému komunikuje so skutočným expertom a snaží sa implementovať znalosti experta do programovej podoby. Báza znalostí je zväčša tvorená skupinou pravidiel typu AK – POTOM. Spravidla je možné v jednom okamžiku činnosti expertného systému aplikovať viacero odvodzovacích pravidiel a vzniká tzv. konflikt výberu pravidiel. Inferenčný mechanizmus určuje, ktoré z pravidiel sa v danej chvíli použije (hovorí sa o "odpálení pravidla"). Výber pravidla sa môže riadiť viacerými spôsobmi. Najjednoduchším spôsobom určenia použitého pravidla je vopred stanovené poradie pravidiel.

Sofistikovanejší spôsob výberu inferenčného pravidla môže byť založený na očakávanej hodnote informačného zisku pre jednotlivé pravidlá: vždy sa vyberá to pravidlo, od ktorého vyhodnotenia sa očakáva najväčší informačný zisk.

Expertné systémy sa svojou povahou, architektúrou i vlastnosťami líšia od konvenčných programov pre vedecko-technické výpočty. Zatiaľ čo v konvenčných programoch sú znalosti špecialistov "roztrúsené" v jednotlivých častiach algoritmu, pri expertných systémoch sú znalosti experta vyjadrené úplne explicitne v báze znalostí.

V každom expertnom systéme je nevyhnutným komplementom inferenčného mechanizmu práve báza znalostí. Ku každému inferenčnému mechanizmu, je možné vytvárať množstvo báz znalostí líšiacich sa obsahovo aj použitím. Takto je možné expertný systém použiť aj vo vzájomne vzdialených problémových oblastiach na riešenie rôznych kategórií problémov.

Báza znalostí je adekvátom deklaratívneho programu. Jej obsah je tvorený pasívnymi údajovými štruktúrami v tom zmysle, že tie netvoria vykonávateľné inštrukcie programu. Reprezentácia poznatkov môže spočívať na viacerých druhoch formálnych reprezentačných prostriedkov. Práve reprezentácia poznatkov je jeden z ťažiskových predmetov výskumu v oblasti umelej inteligencie.

Báza znalostí je spravidla implementovaná ako množina pravidiel tvaru

AK PLATÍ [podmienka] **POTOM** PLATÍ [záver]

Báza znalostí obsahuje znalosti experta alebo expertov potrebné k riešeniu zvoleného problému. Zachytáva znalosti od najzákladnejších, „učebnicových“ až po špeciálne heuristiky a neraz iba tušenia získané skúsenosťou a dlhodobou praxou, pre ktoré často neexistuje lepšie vysvetlenie ako to, že „niekedy dobre fungujú“ a expert by si ich ani nedovolil publikovať. Znalosti experta nemajú statický charakter, ale postupne sa môžu vyvíjať a rozrastať. Prirodzenou požiadavkou kladenou na bázu znalostí expertného systému je vysoká modularita, ktorá umožní bázu znalostí upravovať a dopĺňať. [Pop89]

4.4.3 Báza údajov

Riešiť konkrétnu úlohu s pomocou expertného systému znamená aplikovať v ňom uložené znalosti na informácie a fakty o konkrétnej úlohe. Množinu všetkých údajov relevantných k danému prípadu nazývame – na rozdiel od bázy znalostí – bázou údajov. Aj táto je tvorená pasívnymi údajovými štruktúrami. Kým báza znalostí obsahuje symbolovú reprezentáciu všeobecne platných poznatkov z danej problémovej oblasti, báza údajov uchováva symbolovú reprezentáciu konkrétnych faktov súvisiacich s práve riešeným problémom. Zodpovedajúce údaje v báze dát používa inferenčný mechanizmus pri svojej činnosti, keď ich dopĺňa, modifikuje, prípadne aj ruší. Údaje v báze údajov sú okrem inferenčného mechanizmu prístupné aj ďalším prípadným aktívnym programovým modulom expertného systému. [Bab04]

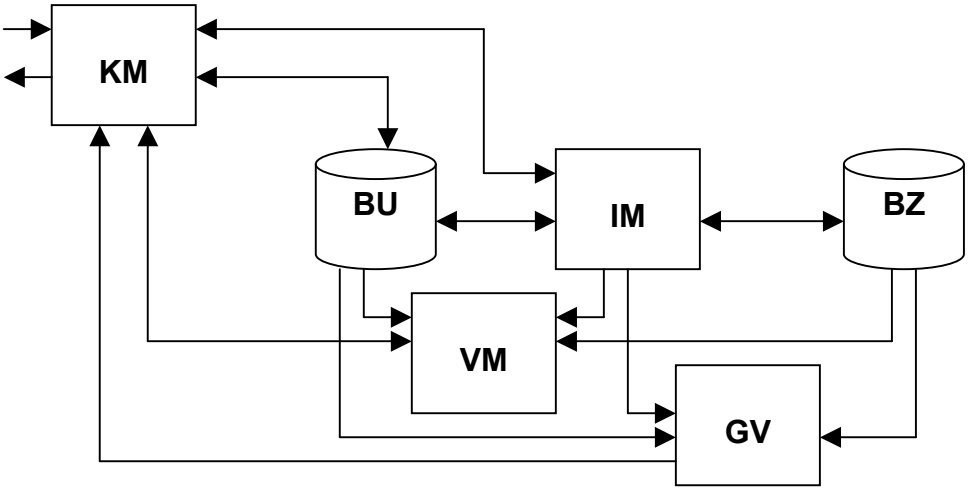
Frekvencia pristupovania k položkám bázy údajov je v porovnaní s prístupom k položkám bázy znalosti mnohonásobne vyššia. Z tohto hľadiska je organizácia obsahu bázy údajov veľmi dôležitým faktorom.

Báza údajov je na rozdiel od bázy znalostí dynamická údajová štruktúra. Toto značí, že jej obsah sa v priebehu odvodzovania značne mení. Najčastejšie sa dopĺňa – vytvárajú sa v nej nové položky. Môže však dochádzať aj k modifikovaniu resp. rušeniu položiek. Niektoré položky vznikajú pri začiatočnej špecifikácii problému, iné počas riešenia problému. Výskyt alebo obsah niektorých položiek v báze údajov môže mať len podmienenú platnosť. Sú to položky, ktoré nemožno získať priamo, a vznikajú iba *odhadom*, ako očakávateľné alebo predpokladateľné a ich platnosť je prípustná len dovtedy kým sa nedostanú do rozporu s údajmi zodpovedajúcimi realite.

Zadávanie údajov o konkrétnej úlohe môže byť riešené formou odpovede používateľa na požiadavku systému (tzv. dialógový režim) alebo výberom z vopred pripravenej databázy

faktov o úlohe (tzv. dávkový alebo batch režim). Niekedy je vhodné požadované údaje získať ich priamym meraním, čím sa odstráni potreba interakcie s používateľom. [Pop89]

4.4.4 Ďalšie zložky expertných systémov



Obr. 3 Základné a prídavné zložky expertných systémov

Obr. 3 zobrazuje ďalšie – prídavné zložky expertného systému. Medzi najpodstatnejšie patria *komunikačný modul (KM)*, ktorý zabezpečuje komunikáciu medzi používateľom a expertným systémom, *vysvetľovací modul (VM)* resp. mechanizmus, ktorý vysvetľuje a zdôvodňuje stav a priebeh riešenie problému, a *generátor výsledkov (GV)*, ktorého funkcia spočíva v zostavovaní čiastkových výsledkov. Na obrázku sú samozrejme znázornené aj základné zložky expertného systému. BU je báza údajov, IM je inferenčný mechanizmus a BZ je báza znalostí.

Pomerne významnú úlohu tvorí práve vysvetľovací systém resp. mechanizmus. Vždy keď expert dospeje k nejakému rozhodnutiu, nezaujatý pozorovateľ by mohol byť zvedavý ako expert dospel k takémuto rozhodnutiu. Taktiež, vždy keď sa expert opýta užívateľa nejakú otázku, musí byť expertný systém schopný vysvetliť mu prečo sa spýtal práve túto otázku. V expertnom systéme túto úlohu plní vysvetľovací modul. Vďaka nemu by mal byť ES schopný v ktorejkoľvek fáze spracovania úlohy podať používateľovi informáciu o tom, čo už bolo odvodené, čo je aktuálnym cieľom odvodzovania (t.j. vysvetlenie, "prečo bol používateľ požiadaný práve o túto informáciu") a ako bol odvodený niektorý zo záverov expertného systému.

Vysvetľovací mechanizmus je počas činnosti expertného systému používaný len veľmi zriedkavo, ale pocit jeho existencie zvyčajne značne zvyšuje dôveru používateľov k systému. Má nenahraditeľnú úlohu pri návrhu, úpravách, kontrole a ladení systému. Pomocou neho

ľudský expert testuje činnosť systému a kontroluje spôsob, akým systém došiel k svojím záverom. [Pop89]

4.5 Neurčitosť systému

Mnoho údajov o reálnom svete sa nedá vyjadriť presne. Expertný systém sa musí vedieť vysporiadať hneď s niekoľkými druhmi neurčitostí. Prvým druhom je neurčitosť v báze znalostí, spôsobená domnienkami a tušeniami experta o vzťahoch platiacich v danej problémovej oblasti a (alebo) znalosťami platiacimi iba v obmedzenom rozsahu. Neurčitosť v báze údajov vyplýva z toho, že ani používateľ expertného systému nie je vždy schopný odpovedať na jeho otázky kategoricky "áno" alebo "nie", ale dokáže vyjadriť určitú mieru svojho presvedčenia o platnosti alebo neplatnosti určitého faktu (odpoveď "asi áno", "takmer isto nie", "možno" a pod.).

Na danom mieste je potrebné poukázať na rozdiel chápania neurčitosti v expertných systémoch a vo fuzzy systémoch. Pokiaľ neurčitosť v expertných systémoch je založená na presvedčení o miere platnosti či neplatnosti určitého faktu alebo na miere všeobecnosti platnosti údajov v báze znalostí, neurčitosť vo fuzzy systémoch vychádza z prirodzenej vágnosti pojmov ako sú "vysoký", "rýchly", "vzdialený" a pod. Fuzzy systémy umožňujú výpočtovej technike pracovať s pojmami, ktoré sú človekom bežne používané a v ľudskej komunikácii majú veľkú informačnú hodnotu.

Našťastie sa tieto dva spomínané chápania neurčitosti môžu kombinovať; niekedy sa vágnosť uvádza ako jeden z typov neurčitosti, s ktorou dokážu expertné systémy pracovať. V takomto prípade sa hovorí o tzv. fuzzy expertných systémoch; tieto sú jedným z príkladov veľmi úspešnej kombinácie metód symbolickej ("klasickej") a subsymbolickej umelej inteligencie. [Pop89]

4.6 Ukladanie dát v expertných systémoch

Najťažším a najdôležitejším krokom implementácie expertného systému je tvorba bázy znalostí. Získavaním znalostí od experta sa zaoberá znalostné inžinierstvo. Kľúčovým faktorom, na ktorom môže proces získavania znalostí zlyhať, je porozumenie a dobrá komunikácia medzi expertom a znalostným inžinierom, implementujúcim znalosti experta do výpočtového systému. [Pop89]

Zavedenie znalostí ako informačných objektov spôsobilo prechod od bázy dát k báze znalostí. Základnou otázkou pri vytváraní bázy znalostí sa stáva výber vhodného spôsobu

ich reprezentácie. V bežnom živote ľudia používajú k vyjadreniu znalosti prirodzeného jazyka. Umelo vytvorený systém, u ktorého je predpokladané inteligentné chovanie, nemôže pracovať bez ľudských znalostí reprezentovaných vhodným spôsobom. Spôsob reprezentácie znalostí výrazne ovplyvňuje schopnosť systému riešiť zadané problémy. Z tohto dôvodu je problematika reprezentácie znalostí považovaná za kľúčovú oblasť umelej inteligencie a tým aj expertných systémov.

Dáta v expertných systémoch sa ukladajú v rôzne „inteligentných“ štruktúrach, ako sú *fakty, relácie, záznamy*. Najbežnejšie používanou štruktúrou sú takzvané rámce. Tieto obsahujú rysy známe z objektového programovania, najmä dedičnosť. Príklad rámcovej uvádza napr. Račanský [Rac95]. Ďalšími možnosťami reprezentácie sú deklaratívne a procedurálne schémy.

Problematiku reprezentácie dát bližšie rozoberá Hala [Hal03] a Zbořil. [Zbo90]

5 KYBERNETICKÝ SYSTÉM SERGEJ

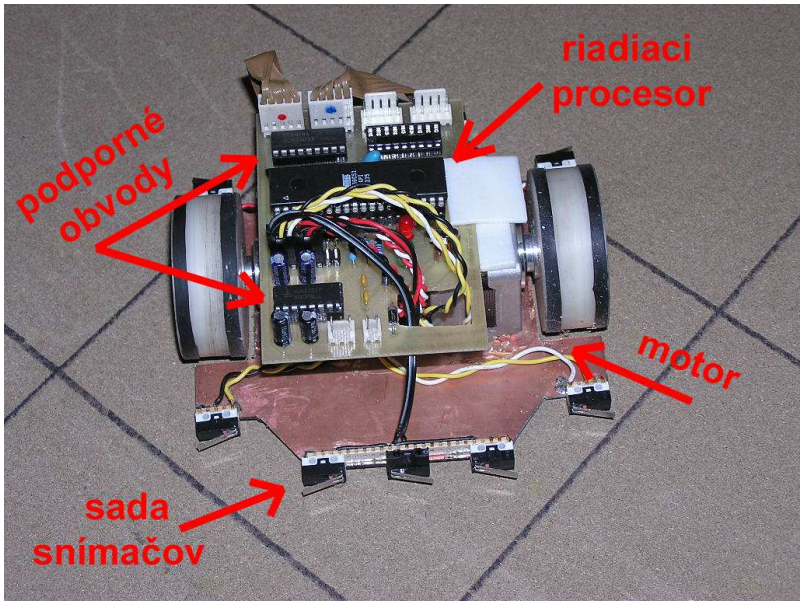
Úloha, na ktorej je demonštrovaná činnosť a vlastnosti expertného systému je veľmi jednoduchá. Jedná sa o hľadanie cesty v bludisku, ktoré pozostáva zo štvorcov. Hľadanie prebieha v nezmapovanom bludisku a cestu (pokiaľ možno optimálnu) hľadá mobilný robot. Robot je jednoduché dvojkolesové vozidlo, so sústavou senzorov, prostredníctvom ktorých robot získava informácie o prostredí. Keďže sa jedná iba o veľmi jednoduchú demonštráciu, pohyb robota je obmedzený na krok vpred, krok vzad, otočenie doľava a doprava. Robot sa teda v pomyselnom bludisku pohybuje iba v rozsahu štvorcov.

Návrh robota vychádza z predstavy, že v pamäti robota budú uložené len základné funkcie pre pohyb v jednotlivých smeroch a návrat hodnôt snímačov nadradenému systému. Nadradeným systémom, ktorý vykonáva rozhodovaciu činnosť je osobný počítač. Toto rozdelenie úloh je tu najmä kvôli komfortu, ktorý poskytuje programovanie vo vyššom programovacom jazyku na osobnom počítači, a taktiež preto aby bolo možné ľahko demonštrovať proces “myslenia” prostredníctvom obrazovky osobného počítača.

Na osobnom počítači je vytvorený program vo vyššom programovacom jazyku, v ktorom je implementovaný jednoduchý expertný systém. Program je vytvorený v programovacom jazyku C++ v prostredí Microsoft Windows.

Komunikácia medzi robotom a počítačom prebieha po sériovom kanáli cez rozhranie RS232.

5.1 Konštrukcia robota



Obr. 4 Robot Sergej

Pre svoj pohyb využíva robot dvojicu krokových motorov. Sú to motory typu KP4M4 s krokom 3.6 stupňa a impedanciou 150 ohmov na vinutie. Stredové konce jednotlivých vinutí sú spojené a vyvedené ako samostatný vývod. Tento typ motorov sa používal v starších 5,25 palcových disketových mechanikách a v robotovi je použitý najmä kvôli nízkej nadobúdacej cene. Budenie krokových motorov je zabezpečené budičmi ULN2803 firmy SGS-Thomson Microelectronic. Jedná sa o integrovaný obvod skladajúci sa z ôsmich Darlingtonových polí.

Ako senzory sú použité mechanické mikrospínače, opäť kvôli čo najväčšej jednoduchosti. Senzory sú umiestnené na prednej aj zadnej časti robota, teda robot spoľahlivo určí prekážku pri pohybe vpred aj vzad. Senzory sú umiestnené na troch miestach vpredu a troch miestach vzadu takže robot dokáže detekovať aj prekážku, ktorá neleží priamo v imaginárnom štvorci po ktorých sa robot pohybuje.

Vnútornú logiku robota zabezpečuje mikroprocesor rady 8051 firmy ATMEL typu 89C51, v ktorom je napálený ovládací program, vďaka ktorému robot dokáže robiť elementárne pohyby a hlásiť stav svojich senzorov. Celý systém ešte obsahuje integrovaný obvod MAX232 firmy MAXIM, čo je prevodník logických úrovní z rozhrania RS – 232C (V24) na TTL logiku a naopak.

Schéma zapojenia celej riadiacej časti vrátane návrhu dosky plošného spoja je uvedená v prílohe A.

5.2 Komunikácia robota s nadradeným počítačom

Ako už bolo spomenuté v úvode, robot nie je autonómna jednotka, a pre svoju *rozumnú* činnosť musí komunikovať s nadradeným počítačom. Oddelenie robota a počítača má význam v tom, že je možné kedykoľvek nahradiť robota novším modelom, pričom nie je nutné meniť ovládací program. Rovnako je možné jednoducho doplniť v programe nové vlastnosti (napr. modifikovať bázu znalostí) bez nutnosti robiť zásahy do konštrukcie robota. Tabuľka použitých riadiacich kódov je uvedená v prílohe B.

Množstvo údajov, ktoré posiela počítač robotovi a naopak robot počítaču je veľmi malé a tieto údaje prichádzajú v nepravidelných časových úsekoch – podľa potreby. Komunikácia je teda realizovaná nie po bitoch, ale po znakoch. Každý pokyn robotovi znamená jeden prenesený znak po sériovom porte, a každé hlásenie robota o úspešnom či neúspešnom vykonaní požadovanej akcie, či zopnutí niektorého zo senzorov je oznamované nadradenému počítaču taktiež jedným znakom. Jednou z výhod tejto komunikácie po znakoch je to, že otvára možnosti použitia robota na rôzne účely, bez nutnosti hlbšie

skúmať komunikačné protokoly. Robota je takto dokonca možné jednoducho ovládať aj cez obyčajný hyperterminál.

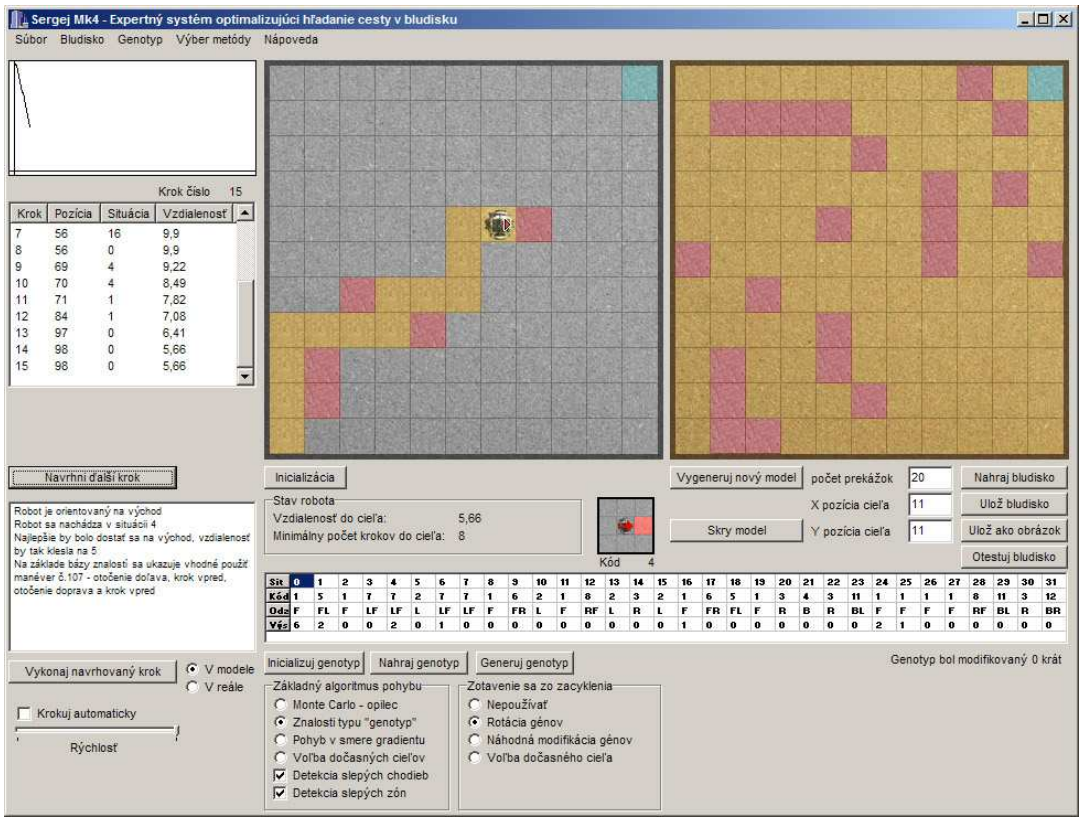
Nadradený počítač a robot medzi sebou komunikujú prenosovou rýchlosťou 2400 b/s, 8 údajových bitov, 1 stop bit, bez parity a riadenia toku. Prenosová rýchlosť je na dnešné komunikačné zariadenia relatívne nízka, avšak pre požiadavky systému plne postačuje a s použitým typom procesora na strane robota je táto rýchlosť veľmi ľahko technicky realizovateľná.

Príklad komunikácie robota s nadradeným počítačom je uvedený v prílohe C.

6 Programové riešenie expertného systému

6.1 Programové prostredie

Program je vytvorený vo vývojovom prostredí C++ Builder. Obsahuje v sebe nielen možnosť priamo optimalizovať pohyb robota v bludisku, ale aj simulačné prostriedky, nutné na to aby bolo možné pozorovať činnosť jednotlivých algoritmov, a sledovať účinnosť jednotlivých metód – druhov znalostí.



Obr. 5 Užívateľské rozhranie expertného systému v prostredí MS Windows

Funkcie, možnosti programu a jeho ovládanie sú bližšie popísané v prílohe F.

6.2 Programové riešenie

Štruktúra programu vychádza z architektúry ES. V programe je oddelená báza znalostí, báza údajov a výber vhodných znalostí a vytváranie znalostí, ktoré nie sú implicitne obsiahnuté v báze znalostí zabezpečuje inferenčný mechanizmus.

Pre lepšiu demonštráciu je v programe implementovaných viacero typov znalostí, ktoré predstavujú algoritmy hľadania cesty k cieľu. Program využíva priamy chod inferenčného mechanizmu, a teda hľadá riešenie v rozsahu známych faktov. Keď robot (príp. simulácia) zistí nový fakt o probléme, poznačí si tento fakt do bázy údajov a hľadá nové riešenie v rámci nových faktov.

Program pracuje v dvoch fázach. V prvej fáze navrhne na základe doterajších údajov a príslušných znalostí ďalší krok a druhej fáze navrhovaný krok vykoná. Ak sa tieto dve fázy budú opakovať dostatočne dlho program nájde riešenie.

Báza údajov je reprezentovaná statickým dvojrozmerným poľom, ktorého prvkami sú celočíselné hodnoty. Pole má rozmer 13x13, čo vyplýva z rozmeru reálneho bludiska, ktoré bolo použité. Hranice poľa sú pri inicializácii označené ako prekážka, čo taktiež zodpovedá reálnemu bludisku, ktoré má po obvode hranicu.

2	2	2	2	2	2	2	2	2	2	2	2	2
2	0	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	0	0	0	2
2	2	2	0	0	0	0	0	0	0	0	0	2
2	1	1	0	0	0	0	0	0	0	0	0	2
2	1	1	1	0	0	0	0	0	0	0	0	2
2	2	2	2	2	2	2	2	2	2	2	2	2

Obr. 6 Vnútorná reprezentácia bázy údajov pomocou statického dvojrozmerného poľa

Hodnoty prvkov poľa môžu nadobúdať tieto hodnoty:

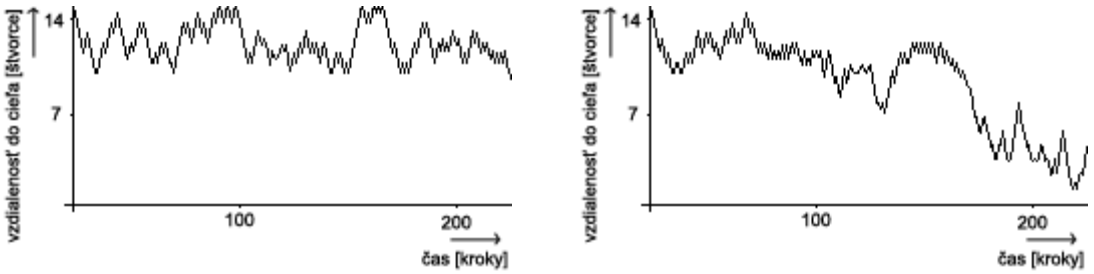
- 0 – políčko je nezmapované, systém nevie či je políčko voľné, alebo je na ňom prekážka.
- 1 – políčko je zmapované a je voľné.
- 2 – políčko je zmapované a je na ňom prekážka.
- 12 – políčko je voľné, alebo obsadené, avšak systém usúdil, že cesta do cieľa nevedie cez toto políčko.

Ako znalosti program využíva nasledujúce metódy a ich kombinácie:

- náhodný pohyb
- znalosti typu genotyp
- pohyb v smere gradientu
- detekciu a zotavenie sa z uviaznutia (zacyklenia)
- detekciu „slepých chodieb“ a „slepých zón“

6.2.1 Algoritmus náhodného pohybu

Tento algoritmus vychádza z metód pre stochastické modelovanie metódou Monte Carlo, i keď samotná metóda Monte Carlo rieši úplne iný problém. V každom okamihu sa robot pohne náhodne zvoleným smerom, pričom šanca pre každý jednotlivý smer je presne 25%. Pri tomto algoritme nemožno hovoriť o nejakých znalostiach, pretože sa jedná o čisto stochastický pohyb. Aby algoritmus predsa len vykazoval nejakú inteligenciu, je určená obmedzujúca podmienka, aby sa robot nikdy nepresunul na políčko, na ktorom je prekážka. Je zrejmé, že metóda je stále príliš slabá a nemôže nájsť riešenie v prijateľnom čase.



a) bez detekcie slepých zón a chodieb b) s detekciou slepých zón a chodieb

Obr. 7 Grafy charakterizujúce náhodný pohyb

Grafy na obr. 7 charakterizujú pohyb robota v bludisku pri použití metódy náhodného pohybu vychádzajúcej z metódy Monte Carlo. Ako testovacie bludisko bol použitý súbor TestLab2.lab. Ako ukazujú grafy, algoritmus nedokázal nájsť cestu bludiskom v prijateľnom čase. Z grafu však možno pozorovať vplyv detekcie slepých zón a chodieb, vďaka čomu sa dostal robot k cieľu rýchlejšie. Je nutné podotknúť, že keďže algoritmus je stochastický výsledky nie sú reprodukovateľné a v každom pokuse dosiahne robot iné výsledky. Ďalšia dôležitá vlastnosť, ktorá platí pri všetkých stochastických metódach, je že algoritmus nikdy

neuviazne v lokálnom extréme. Keďže algoritmus nie je deterministický, nemôže pri ňom dôjsť k zacykleniu. Táto metóda teda určite nájde riešenie, aj keď možno v nekonečnom čase (po nekonečnom počte krokov).

6.2.2 Znalosti typu genotyp

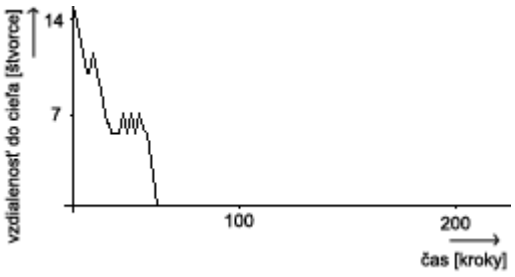
Problematiku použitia evolučných algoritmov na optimalizáciu pohybu robota v bludisku riešil vo svojej práci Böhmer. [Boh05] Genotyp použitý v spomínanej práci sa skladá z 32 génov, ktoré popisujú všetky elementárne situácie, v ktorých sa môže robot nachádzať. Výsledky Böhmerovej práce sú aplikované aj v tomto expertnom systéme ako jedna z báz znalostí. Použitý je genotyp, ktorý podľa Böhmerovych testov vykazoval najlepšie vlastnosti.

Genetické algoritmy však vo všeobecnosti predstavujú iba veľmi slabý nástroj, a genotyp skladajúci sa z 32 génov je príliš krátky na to, aby jeho použitie dokázalo vyriešiť všetky typy bludísk. Pri vyhodnotení situácie robot analyzuje iba 5 najbližších políčok v jeho okolí, a riziko uviaznutia v lokálnom extréme je veľmi vysoké. Bližší popis metódy ako aj jej obmedzenia je možné nájsť priamo v Böhmerovej práci. [Boh05]

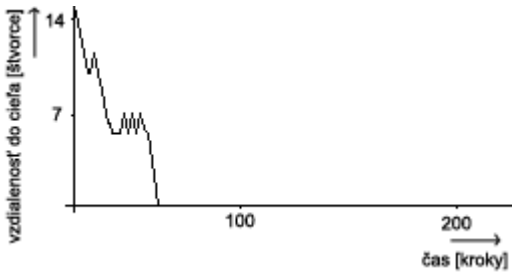
Aby dokázal algoritmus vyriešiť ešte väčšiu skupinu bludísk, je nutné použiť aj detekciu zacyklenia (v prípade uviaznutia v lokálnom extréme) a metódy, ktoré dostanú systém zo zacyklenia.

Pri genotype je možnosť manipulovať len s jednotlivými génmi, takže zotavenie prebieha modifikáciou génov, a to buď rotáciou alebo náhodne. Pri rotácii sa gén ktorý zapríčinil zacyklenie nahradí iným povoleným génom, cyklicky podľa tabuľky povolených génov uvedenej v prílohe D. Pri náhodnej modifikácii je gén, ktorý zapríčinil zacyklenie nahradený úplne náhodne (teda nový gén môže byť aj zakázaný).

Metóda vychádza z predpokladu, že k modifikácii génov nebude dochádzať až tak často, a veľká časť pôvodnej *kvalitnej* genetickej informácie ostane zachovaná. V prípade zložitejšieho bludiska, nijaká konfigurácia genotypu nedokáže problém hľadania cesty vyriešiť, kvôli obmedzeniam vyplývajúcim z dĺžky génu.



a) bez detekcie slepých zón a chodieb



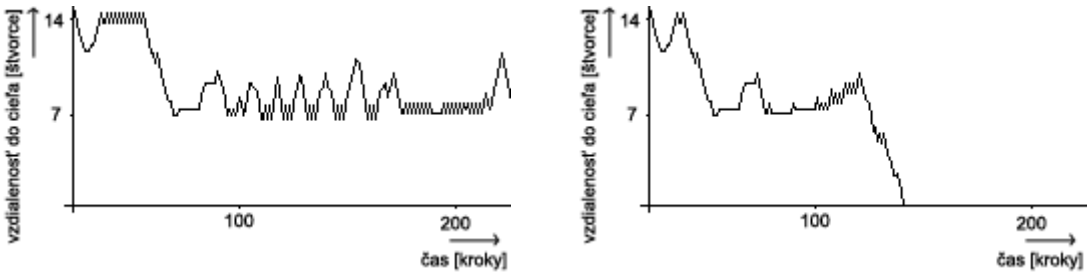
b) s detekciou slepých zón a chodieb

Obr. 8 Grafy charakterizujúce pohyb využívajúci znalosti typu genotyp

Grafy na obr. 8 charakterizujú pohyb robota v bludisku pri použití znalostí typu genotyp so zotavením sa zo zacyklenia rotáciou génov. Ako testovacie bludisko bol použitý súbor TestLab2.lab. Z obr. 8 je vidieť, že algoritmus dokázal nájsť cestu bludiskom v prijateľnom čase. Vplyv detekcie slepých zón a chodieb sa neprejavil, pretože testovacie bludisko TestLab2 je veľmi jednoduché. Zotavenie zo zacyklenia prebiehalo deterministicky cyklickou rotáciou génov. V prípade náhodnej modifikácie génov by výsledok mohol a nemusel byť lepší.

6.2.3 Pohyb v smere gradientu

Gradient je vektorová veličina charakterizujúca priestorovú zmenu (spád) skalárnej veličiny. Skalárnou veličinou je v tomto prípade vzdialenosť do cieľa a *pohybom v smere gradientu* je myslený pohyb smerom, ktorým je zmena skalárnej veličiny maximálna – teda smerom čo najbližšie k cieľu. Môže sa stať, že takýchto smerov (políčok) je viac. V tomto prípade systém navrhne ísť smerom, ktorý ešte nebol zmapovaný. Takýmto spôsobom je aspoň čiastočne motivovaný k tomu, aby zmapoval čo najväčšiu časť poľa, pretože čím viac toho program vie a čím obsiahlejšia je báza údajov, tým menšia je množina možných riešení a o to ľahšie je nájsť konečné riešenie. Tento postup – snaha čo najviac sa priblížiť k cieľu, je veľmi jednoduchá a efektívna metóda, avšak hrozí riziko uviaznutia v lokálnom extréme a následného zacyklenia. Zotavenie zo zacyklenia v tomto prípade nastáva voľbou náhradného cieľa – najvhodnejšieho nezmapovaného políčka do ktorého je možné sa dostať. Postup voľby dočasného cieľa bude vysvetlený neskôr.



a) bez detekcie slepých zón a chodieb b) s detekciou slepých zón a chodieb

Obr. 9 Grafy charakterizujúce pohyb v smere gradientu

Grafy na obr. 9 charakterizujú pohyb robota v bludisku v smere gradientu so zotavením sa zo zacyklenia voľbou dočasného cieľa čo najbližšie k robotovi. Ako testovacie bludisko bol použitý súbor TestLab4.lab. Obr. 9 ukazuje chovanie sa systému v pomerne ľahkom

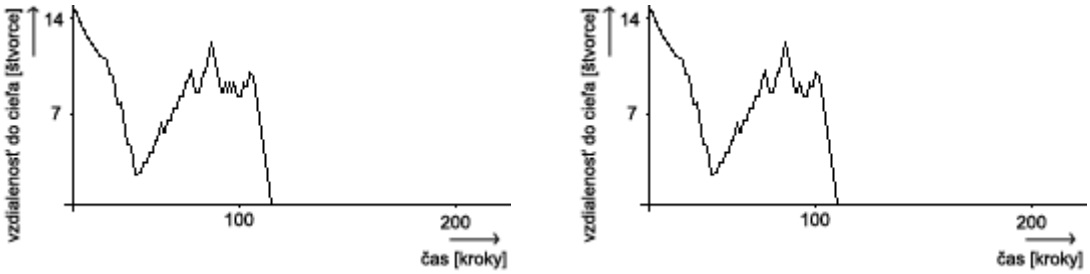
bludisku. Použitie detekcie slepých chodieb má pri tejto metóde veľký význam, pretože práve slepé chodby predstavujú pre systém lokálne extrémy, v ktorých môže systém uviaznuť.

6.2.4 Voľba dočasných cieľov

Všetky doteraz popisované metódy (okrem náhodného pohybu samozrejme) sa snažili nájsť riešenie na základe toho, že sa pokúšali čo najviac priblížiť k cieľu. Toto prináša so sebou riziko uviaznutia v lokálnom extréme a následného zacyklenia. Pozorovaním činnosti človeka – experta, pri riešení problému hľadania cesty bludiskom, je vidieť, že človek nikdy neuviazne v lokálnom extréme. Toto je spôsobené tým, že ak nájde cestu, ktorá nevedie k riešeniu, pokúsi sa hľadať iné riešenie. V okamihu keď človek (v tomto prípade je expertom aj obyčajný človek) nájde lokálny extrém prestáva sa sústrediť na to aby sa dostal čo najrýchlejšie do cieľa, ale snaží sa zmapovať si okolité prostredie aby si vybral nové potencionálne riešenie. Volí si teda *dočasný cieľ*. Implementovanie takéhoto typu znalostí do programu prináša ďalšie priblíženie sa k rozhodovaniu experta, a čo je najdôležitejšie – minimalizuje riziko uviaznutia. Práve tento algoritmus poskytuje najlepšie výsledky a dokáže spolu s detekciou slepých chodieb, vyriešiť hocijaké bludisko.

Pri voľbe náhradného riešenia sa expert môže vydať rôznymi cestami – volí dočasný cieľ podľa svojich kritérií. Jeden expert sa môže rozhodnúť ísť cestou, ktorá je k nemu najbližšie, iný zvolí cestu ktorá je najbližšie cieľu, rozhodovací proces tretieho môže byť úplne iný. Voľba dočasného cieľa je v programe riešená použitím rôznej metriky pre ohodnotenie *kvality* nového potencionálneho riešenia (ktoré závisí od voľby dočasného cieľa).

Ako už bolo povedané metóda síce riziko uviaznutia minimalizuje, avšak neeliminuje ho úplne. Toto je spôsobené tým, že aj pri pohybe do dočasného cieľa, sa robot pohybuje v smere, ktorý ho maximálne priblíži k dočasnému cieľu. Dočasné ciele sa však väčšinou nachádzajú veľmi blízko aktuálnej pozície robota, a riziko uviaznutia v lokálnom extréme je preto minimálne. Ak by však predsa len došlo ku zacykleniu, robot si zvolí náhodne úplne nový dočasný cieľ, o ktorom predpokladá, že bude možné sa doňho dostať. Toto konanie už síce celkom nezodpovedá konaniu experta, ale je pravdepodobné, že aj expert by v bludisku, ktoré je veľmi zložité urobil podobné rozhodnutie. V prípade, že by zistil, že kvôli veľkej zložitosti bludiska nevie kam má presne ísť, pokúsil by sa vydať aspoň do nejakého iného (pravdepodobne náhodného) známeho miesta z ktorého by cestu našiel ľahko.



a) bez detekcie slepých zón a chodieb b) s detekciou slepých zón a chodieb

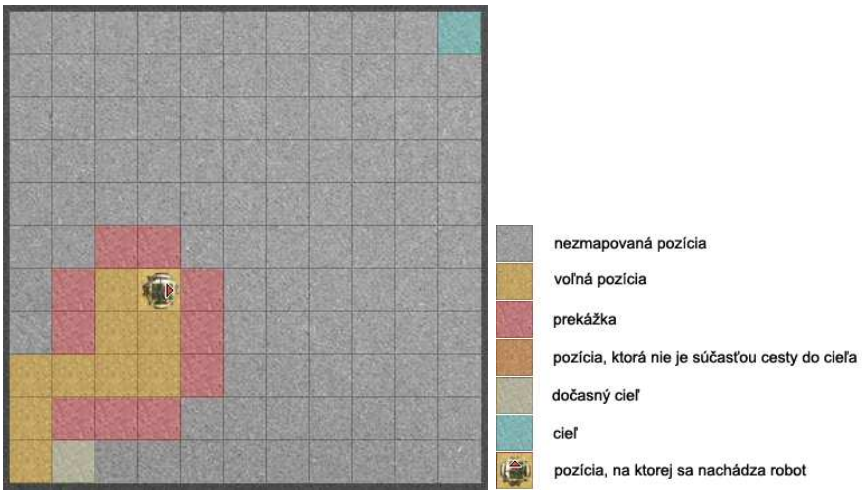
Obr. 10 Grafy charakterizujúce pohyb s voľbou dočasných cieľov

Grafy na obr. 10 charakterizujú pohyb s voľbou dočasných cieľov čo najbližšie k robotovi. Ako testovacie bludisko bol použitý súbor TestLab6.lab. Veľmi dôležitou vlastnosťou pri hľadaní dočasných cieľov skutočnosť, že ak systém nemôže navrhnúť nijaký ďalší dočasný cieľ, dokáže správne určiť že bludisko nemá riešenie. Metódou dočasných cieľov je teda možné veľmi rýchlo vyriešiť aj problém určenia riešiteľnosti bludiska. Práve tento spôsob je použitý aj v tomto programe pri testovaní riešiteľnosti bludiska.

6.2.5 Detekcia zacyklenia

Expert pri hľadaní cesty v neznámom prostredí môže, v prípade, že je prostredie príliš komplexné a zložité usúdiť že uviazol v lokálnom extréme a pohybuje sa stále dookola v tej istej oblasti. Toto zistí obvykle na základe toho, že prostredie, v ktorom sa pohybuje sa mu zdá povedomé a dokáže prehlásiť *„tu som už predsa bol“* a *„tieto kroky som už predsa raz podnikol“.* Ďalším faktom, ktorý ho môže utvrdiť v tom, že pravdepodobne uviazol v lokálnom extréme, môže byť skutočnosť, že jeho vzdialenosť do cieľa sa v poslednom čase príliš nemení. V takomto prípade pravdepodobne podnikne kroky, ktoré mu pomôžu sa z tejto situácie dostať.

Program sa snaží kopírovať rozhodovanie experta a pracuje podobne. V operačnej pamäti si systém vedie „denník“ o tom, kde sa v ktorom kroku robot nachádzal. Tento denník možno považovať za súčasť bázy údajov, pretože aj tieto informácie pomôžu nájsť systému riešenie. Na základe denníka dokáže systém určiť či už v pozícii, v ktorej sa nachádza už niekedy nebol, a či sa do tejto pozície nedostal krokmi ako niekedy v minulosti. V takomto prípade dokáže usúdiť, že došlo k zacykleniu.



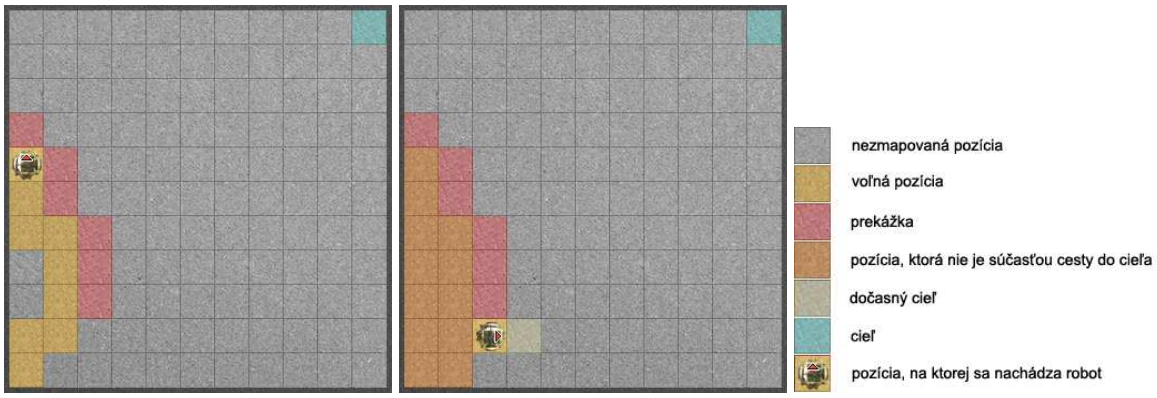
Obr. 11 Príklad triviálneho uviaznutia v lokálnom extréme.

Obr. 11 demonštruje najjednoduchší typ uviaznutia v lokálnom extréme a následného zacyklenia. Robot sa snažil dostať do cieľa avšak uviazol v lokálnom minime. Na základe toho, že si „uvedomil“, že sa stále pohybuje v jednej oblasti zvolil nový dočasný cieľ. Zacyklenie tohto typu väčšinou nastáva pri znalostiach typu genotyp, alebo pri pohybe v smere gradientu. Pri silnejších metódach dochádza k zacykleniu iba v omnoho komplexnejších situáciách a riziko, že k nemu dôjde je veľmi malé.

6.2.6 Detekcia „slepých chodieb“

Pri pohybe bludiskom, si expert často dokáže uvedomiť, že určitá cesta nevedie k riešeniu, pretože je to slepá chodba a vylúči ju z množiny prípustných ciest vedúcich do cieľa. Ak expert správne vylúči všetky nevhodné cesty, musí mu ostať správna cesta – ak existuje. Kybernetický systém, ak chce úspešne konkurovať živému expertovi si musí takéto skutočnosti uvedomovať tiež. Záznam o tom, že určitá cesta nevedie do cieľa si urobí v báze údajov, a políčka označí podobne ako obsadené, a nebude sa do nich už v budúcnosti vracať.

V tejto chvíli je dobré poznamenať, že tu získava expertný systém oproti človeku obrovskú výhodu, pretože si dokáže zapamätať úplne všetky slepé chodby, ktoré objaví. Rýchlosť s akou tieto chodby objaví je tiež mnohonásobne vyššia ako u živého experta a teda najmä pri zložitejších a rozsiahlejších bludiskách ho táto schopnosť stavia do výhodnejšej pozície oproti živému expertovi.



a) b)

Obr. 12 Detekcia slepých chodieb

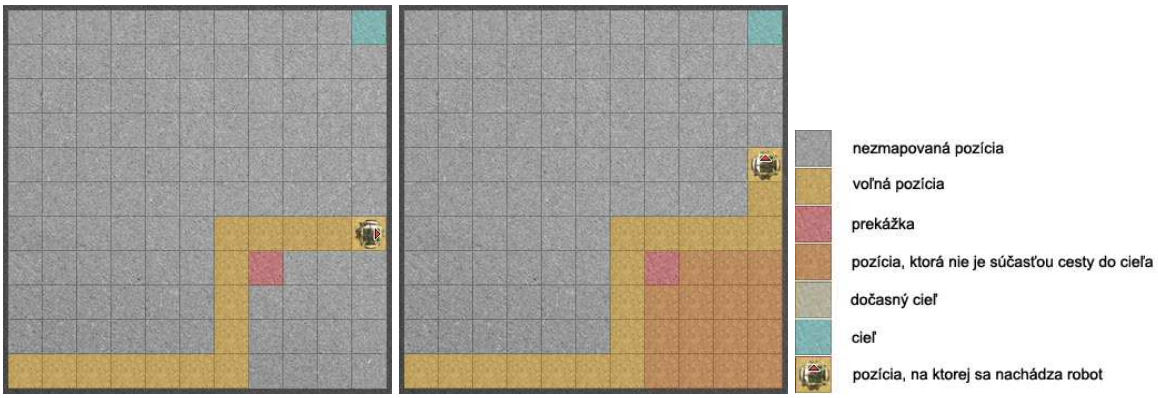
Na obr. 12 je vidieť ako prebieha rozhodovanie systému v prípade objavenia slepej chodby. Na obrázku a) robot dorazil do slepej chodby. Na obrázku b) je vidieť, že pri ceste naspäť si robot označil slepú chodbu, a v budúcnosti sa už do nej nevráti.

6.2.7 Detekcia „slepých zón“

Expert, pri hľadaní cesty bludiskom občas objaví celé oblasti, ktoré môže považovať za oblasti, ktorými určite nevedie cesta k cieľu. Takéto oblasti sa budú vyskytovať najmä v bludiskách, ktoré sú ohraničené zo všetkých štyroch strán a to najmä v oblastiach rohov. Keďže expertný systém by mal čo najvernejšie kopírovať činnosť experta, musí byť aj program byť schopný odhaliť tieto oblasti. V prípade, že program takúto zónu objaví, poznačí to do bázy údajov a políčka v zóne označí podobne ako obsadené a nebude sa do nich už v budúcnosti vracať.

Program na rozdiel od experta označuje aj zóny vnútri bludiska, ktoré nie sú potrebné pre nájdenie cesty bludiskom. Takéto riešenie síce že určitých okolností môže cestu do cieľa predĺžiť, avšak výrazne zmenší množinu v ktorej ES bude hľadať riešenie. V mnohých prípadoch použitie detekcie slepých zón zníži počet krokov potrebných k nájdeniu riešenia a použitie tejto techniky má určite význam.

Kombináciou detekcie slepých chodieb a zón dokáže systém správne vylúčiť prakticky všetky „neužitočné“ cesty, čím sa celé bludisko značne zjednoduší (veľakrát ostane len jediná možná cesta) a proces hľadania cesty sa stáva triviálnym.



a) b)

Obr. 13 Detekcia slepých zón

Obr. 13 zobrazuje ako prebieha detekcia slepej zóny. Na obrázku a) robot objavil slepú zónu. Na obrázku b) systém zónu zaznačil, ako náhle sa robot bezpečne vzdialil od tejto zóny, a v budúcnosti sa už do nej nevráti.

6.2.8 Metrika pre voľbu dočasných cieľov

Pri každej voľbe dočasného cieľa, či už v prípade uviaznutia, alebo len pri voľbe nového podcieľa, je dôležité tento nový cieľ zvoliť čo najvhodnejšie. Tento nový cieľ bude pravdepodobne nezmapované políčko, také do ktorého je možné sa dostať. Nezmapované bude preto, aby bol systém motivovaný k prehľadávaniu bludiska a dopĺňal nové informácie do bázy údajov. V prípade, že takýchto políčok bude viac, je nutné vybrať to najvhodnejšie. Každý potencionálny nový cieľ je nutné nejakým spôsobom ohodnotiť, a nakoniec vybrať ten najvhodnejší.

Ak je ako metrika použitá vzdialenosť k robotovi vyberá sa ako dočasný cieľ to políčko, ktorého vzdialenosť k robotovi je najmenšia. V prípade, že je takýchto políčok viac vyberá sa to, ktoré je z nich najbližšie k cieľu. Tento typ metriky pre dočasný cieľ predstavuje najnižšie riziko uviaznutia v lokálnom extréme, keďže do cieľa, ktorý je blízko robota je väčšinou možné dostať sa bez problémov.

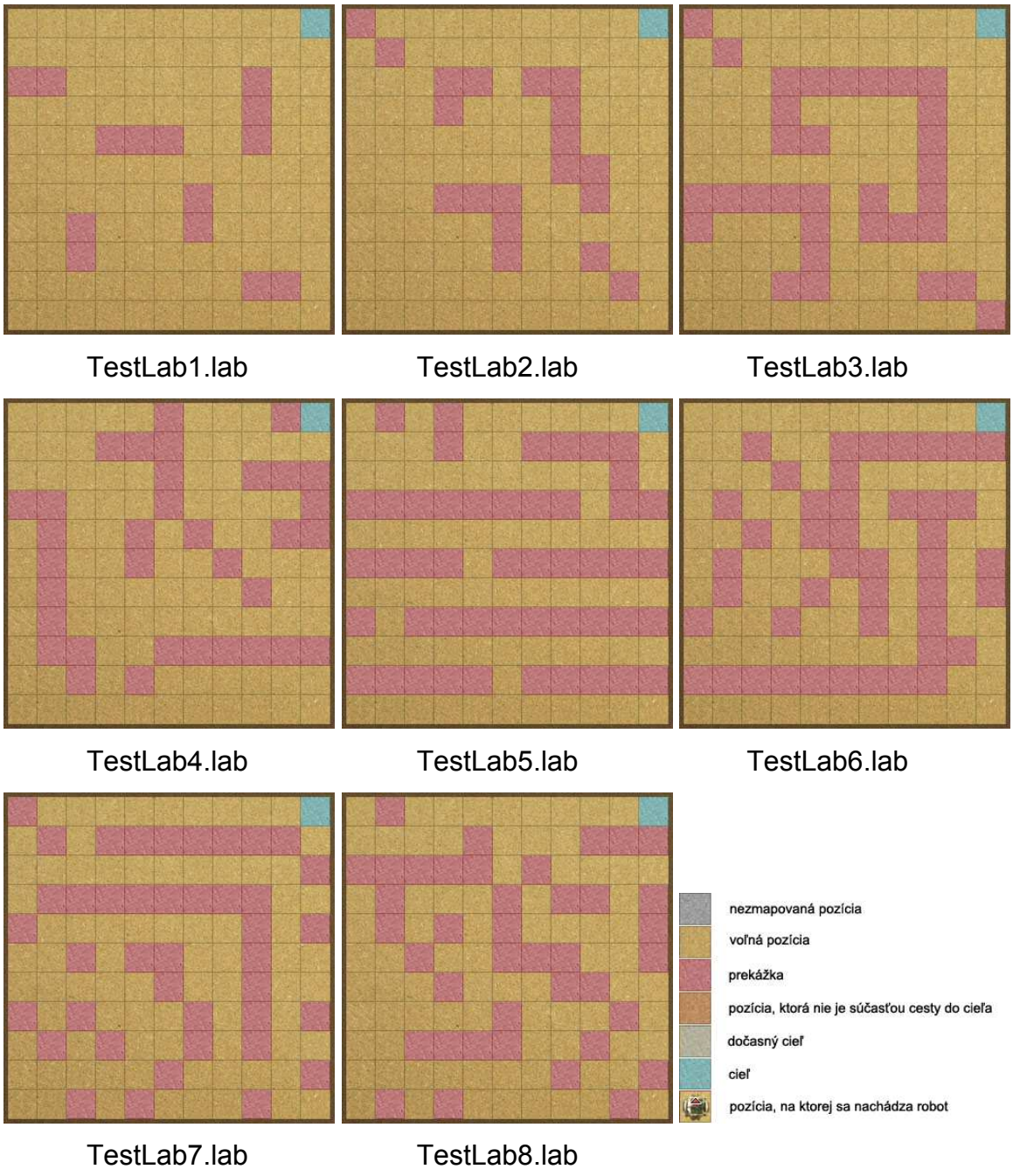
Ak je ako metrika použitá vzdialenosť do cieľa vyberá sa ako dočasný cieľ to políčko, z ktorého je vzdialenosť do cieľa najmenšia. V prípade, že je takýchto políčok viac vyberá sa to, ktoré je z nich najbližšie k robotovi. Tento typ metriky pre dočasný cieľ zaistí najrýchlejší pohyb k skutočnému cieľu, avšak zároveň prináša väčšie riziko uviaznutia v lokálnom extréme, keďže nový dočasný cieľ môže byť príliš ďaleko od súčasnej pozície robota a robot sa nemusí do tohto cieľa vždy jednoducho dostať.

Ak je ako metrika použitý stred medzi robotom a cieľom, tak potencionálne dočasné ciele sú ohodnotené podľa súčtu ich vzdialenosti k robotovi a ich vzdialenosti k cieľu. Najvhodnejšie políčka sú teda blízko k robotovi, a zároveň blízko k cieľu. Toto predstavuje kompromis medzi prvými dvoma spôsobmi určovania metriky.

Posledným spôsobom je náhodné určenie dočasného cieľa. Táto metóda zvolí dočasný cieľ bez ohľadu na jeho vzdialenosť k cieľu či k robotovi. Takáto voľba väčšinou prinesie horší výsledok ako deterministické určenie dočasného cieľa, avšak práve vďaka faktoru náhody môže tento spôsob znížiť riziko uviaznutia v lokálnom extréme.

6.3 Testovanie vlastností jednotlivých metód

Pri návrhu jednotlivých metód resp. znalostí nebolo možné presne určiť, ktorá metóda bude tá najefektívnejšia a bude použitá vo finálnom expertnom systéme. Aby bolo možné ohodnotiť efektivitu jednotlivých metód, jednotlivé algoritmy a ich kombinácie boli podrobené sérii testov. Metódy boli testované na ôsmich bludiskách rozličnej náročnosti.



Obr. 14 Bludiská použité v testoch

Po otestovaní všetkých alternatív bol vyhodnotený priemerný počet krokov ktoré robot, pri určitej metóde potreboval na prejdenie jedného bludiska a hodnoty boli zapísané do tabuľky najvhodnejších metód.

Tab. 1 Tabuľka najvhodnejších metód

Poradie metódy	Algoritmus	Priemerný počet krokov potrebných k prejdeniu bludiska
1	Voľba dočasných cieľov čo najbližšie k robotovi s detekciou slepých chodieb	75,8
2	Voľba dočasných cieľov čo najbližšie k robotovi s detekciou slepých chodieb a zón	76,5
3	Voľba dočasných cieľov medzi robotom a cieľom s detekciou slepých chodieb	77,8
4	Voľba dočasných cieľov čo najbližšie k cieľu s detekciou slepých chodieb a zón	92,3
5	Voľba dočasných cieľov medzi robotom a cieľom s detekciou slepých chodieb a zón	95,1
6	Voľba dočasných cieľov čo najbližšie k cieľu s detekciou slepých chodieb	127,8
7	Pohyb v smere gradientu so zotavením sa voľbou dočasného cieľa čo najbližšie k cieľu s detekciou slepých chodieb a zón	165,7
8	Pohyb v smere gradientu so zotavením sa voľbou dočasného cieľa čo najbližšie k cieľu s detekciou slepých chodieb	175,3
9	Pohyb v smere gradientu so zotavením sa voľbou náhodného dočasného cieľa s detekciou slepých chodieb a zón	185
10	Pohyb v smere gradientu so zotavením sa voľbou dočasného cieľa medzi robotom a cieľom s detekciou slepých chodieb a zón	190,1
11	Pohyb v smere gradientu so zotavením sa voľbou náhodného dočasného cieľa s detekciou slepých chodieb	194
12	Pohyb v smere gradientu so zotavením sa voľbou dočasného cieľa medzi robotom a cieľom s detekciou slepých chodieb	199,1
13	Znalosti typu „Genotyp“ so zotavením sa voľbou dočasného cieľa medzi robotom a cieľom s detekciou slepých chodieb	203,3
14	Znalosti typu „Genotyp“ so zotavením sa voľbou dočasného cieľa medzi robotom a cieľom s detekciou slepých chodieb a zón	203,3
15	Pohyb v smere gradientu so zotavením sa voľbou dočasného cieľa čo najbližšie k robotovi s detekciou slepých chodieb a zón	208,2
16	Znalosti typu „Genotyp“ so zotavením sa voľbou náhodného dočasného cieľa s detekciou slepých chodieb	212,6
17	Pohyb v smere gradientu so zotavením sa voľbou dočasného cieľa čo najbližšie k robotovi s detekciou slepých chodieb	213,2
18	Voľba náhodných dočasných cieľov s detekciou slepých chodieb a zón	221,0
19	Voľba náhodných dočasných cieľov s detekciou slepých chodieb	223,0
20	Znalosti typu „Genotyp“ so zotavením sa voľbou dočasného cieľa čo najbližšie k robotovi s detekciou slepých chodieb a zón	230,1

V tab. 1 je uvedených 20 najlepších metód zo všetkých použitých, ktoré dokázali vyriešiť všetkých osem testovacích bludísk. Priemerný počet krokov potrebných k prejdeniu bludiska bol vyrátaný ako priemer z počtu krokov, ktoré systém potreboval na vyriešenie bludiska pre jednotlivé testovacie bludiská.

Metóda náhodného pohybu skutočne nedokáže priniesť uspokojivé výsledky. Nepoužíva prakticky žiadne relevantné znalosti, a to či nájde riešenie je iba otázkou náhody. Použitie takejto metódy nemá reálny význam.

Použitie znalostí vo forme genotypu prináša akceptovateľné výsledky iba pri veľmi jednoduchých bludiskách. Pri komplexnejších problémoch nedokáže metóda nájsť riešenie ani pri rôznej modifikácii génov. Toto je samozrejme spôsobené aj malou dĺžkou genotypu, ktorý dokáže popísať iba veľmi málo a aj to iba elementárnych situácií. Kvôli úplnosti bol *ideálny* genotyp nahradený aj úplne náhodným genotypom, avšak výsledky boli, podľa očakávaní, ešte horšie.

Metóda pohybu v smere gradientu dokázala vyriešiť všetky testovacie bludiská. Jej najväčšou nevýhodou je však to, že veľmi často uviazne v lokálnom extréme a zotavenie zo zacyklenia trvá pomerne dlho. Kvôli tejto vlastnosti metóda nájde cestu bludiskom až po veľmi veľkom počte krokov.

Metóda voľby dočasných cieľov sa približuje najviac rozhodovaniu experta a aj preto prináša najlepšie výsledky. Táto metóda dokáže nájsť riešenie akéhokoľvek bludiska v prijateľnom čase. V kombinácii s detekciou slepých chodieb a zón ponúka najefektívnejšie riešenie problému hľadania cesty v bludisku z doteraz spomínaných metód.

Detekcia slepých chodieb má jednoznačne pozitívny vplyv na počet krokov nutných k prejdeniu bludiska a zároveň znižuje riziko uviaznutia. Vďaka jej použitiu dokázali bludisko vyriešiť aj slabšie metódy. Detekcia slepých chodieb má v systéme neoceniteľný význam pri bludiskách v ktorých existuje iba jediná cesta do cieľa.

Detekcia slepých zón mierne zvyšuje pravdepodobnosť vyriešenia bludiska. Vplyv na počet krokov nutných k prejdeniu bludiska je len veľmi malý. V niektorých prípadoch sa dokonca počet krokov potrebných k prejdeniu bludiska dokonca zvýšil.

Keďže expertný systém má nahradiť funkciu experta a má robiť rozhodnutia minimálne tak kvalitné ako expert, systém bol porovnaný aj so živými ľuďmi. Nakoľko problém hľadania cesty bludiskom je triviálny, nebolo nutné hľadať špecialistov a program bol otestovaný ľuďmi z môjho okolia. Tri osoby boli požiadané aby sa pokúsili nájsť cestu tými istými neznámymi testovacími bludiskami, ktoré boli použité pri teste systému. V prípade jednoduchých bludísk boli výsledky približne rovnaké, ale najmä pri komplexnejších bludiskách program dokázal nájsť riešenie tak isto rýchlo, ba až rýchlejšie (v zmysle počtu potrebných krokov) ako človek. Toto však nebolo spôsobené len tým, že by znalosti programu boli kvalitnejšie ako znalosti

človeka. Pri tak triviálnom probléme ako je hľadanie cesty bludiskom možno iba ťažko hovoriť o kvalifikovaných znalostiach. Výhoda osobného počítača spočíva samozrejme v jeho výpočtovej sile, rýchlosti a veľkosti pamäti. Možno predpokladať, že pri bludiskách väčších rozmerov by sa rozdiel medzi človekom a programom prejavil ešte výraznejšie. Samozrejme v prospech expertného systému.

Podrobné výsledky všetkých 600 testov sú uvedené v prílohe E.

ZÁVER

Práca si kládla za úlohu overiť možnosti použitia expertných systémov pre riešenie na prvý pohľad triviálnych problémov. Ukázalo sa, že aj takáto neštandardná aplikácia ES dokáže priniesť pozitívne výsledky, a možnosti nasadenia expertných systémov sú skutočne veľmi široké.

Bol zostrojený jednoduchý robot, ktorého je možné ovládať po sériovom kanáli. Komunikácia robota s počítačom je realizovaná cez štandardné rozhranie číslicového počítača – port RS232. Pri konštrukcii robota boli využité poznatky z elektrotechniky, elektroniky a programovania v jazyku symbolických adries. Robot použitý v tomto systéme môže byť využitý aj pre ďalšie prípadné projekty v oblasti automatizácie, kybernetiky, či umelej inteligencie.

Podarilo sa zostrojiť expertný systém, ktorý je pri hľadaní cesty bludiskom približne tak *inteligentný* ako človek. Je možné, že existujú ešte silnejšie algoritmy na riešenie tohto druhu problému. V tomto systéme však šlo o snahu čo najvernejšie kopírovať myslenie človeka – experta. Analýzou myšlienkových pochodov, a rozhodnutí experta boli formulované jednoduché znalosti, ktoré boli neskôr implementované do programu.

Existujú samozrejme mnohé ďalšie znalosti, ktoré človek dokáže využiť avšak tieto do systému neboli implementované nakoľko je problém ich vhodne reprezentovať. Jedná sa najmä o rozpoznávanie tvarov, detekciu slepých zón nepravidelného tvaru a pokročilé heuristické metódy, teda rozhodnutia, ktoré sám expert nevie racionálne vysvetliť, avšak často prinášajú úspech.

Práca rozhodne nájde svoje uplatnenie pri demonštrácii činnosti jednoduchého expertného systému a keďže sa jedná o skutočne veľmi triviálnu aplikáciu, poskytuje možnosť zoznámiť sa s problematikou expertných systémov aj človeku, ktorý s umelou inteligenciou nemá žiadne predošlé skúsenosti. Programové riešenie je taktiež tvorené veľmi modulárne prostredníctvom funkcií a procedúr a každý programátor by mal byť schopný program modifikovať a doplniť nové znalosti.

BIBLIOGRAFIA A BIBLIOGRAFICKÉ ODKAZY

[Ash57] ASHBY, W.R..: An Introduction to Cybernetics Second Impression. London: Chapman & Hall ltd., 1957, 156 s. ISBN 0416683002

[Kri96] KRISHNAMOORTHY, C.S. – RAJEEV, S.: Artificial Intelligence and Expert Systems for Engineers. Madras: CRC Press, 1996, 254 s. ISBN 0849391253

[Pop89] POPPER, M. – KELEMEN, J.: Expertné systémy 1. vyd. Bratislava: Alfa, 1989, 360 s. ISBN 80-05-00051-0

[Sel92] SELIG, J.M.: Introductory Robotics. Hertfordshire: Prentice Hall International, 1992, 157 s. ISBN 0-13-488875-8 (pbk.)

[Sol02] ŠOLC, F. – ŽALUD L.: Robotika: Výskumná práca. Brno: Vysoké učení technické v Brně, 2002, 61 s.

[Zbo90] ZBOŘIL, F. – HANÁČEK, P.: Umělá inteligence Brno: Ediční středisko VUT Brno, 1990, 125 s. ISBN 80-05-00051-0

[Boh05] BÖHMER M.: Návrh a optimalizácia kybernetického systému pomocou genetických algoritmov: Diplomová práca. Liptovský Mikuláš: Akadémia ozbrojených síl generála Milana Rastislava Štefánika Liptovský Mikuláš, 2005

[Gre00] GREENBERG, H.J: A Prospective on Mathematics and Artificial Intelligence: Výskumná správa. Denver: University of Colorado, 2000, 5 s.

[Hal03] HALA V.: Znalosti a reprezentácia znalostí v UI: Seminárna práca. Košice: Technická univerzita Košice, 2003, 7 s.

[Hut02] HUTTER M.: Towards a Universal Theory of Artificial Intelligence based on Algorithmic Probability and Sequential Decisions: Prezentácia. Manno Lugano: Instituto Dalle Molle di Studi sull'Intelligenza Artificiale IDSIA, 2002, 32 s.

[Len99] VAN LENT M. - LAIRD J.: Developing an Artificial Intelligence Engine: Výskumná správa. Michigan: University of Michigan, 1999, 11 s.

[Rac95] RAČANSKÝ V.: Umělá inteligence: Zápisy z prednášok. Brno:Universitas Masarykiana, 1995, 48 s.

[Sch00] SCHMOTZER M.: Znalostné systémy z hľadiska tradície umelej inteligencie: Interná výskumná správa. Košice: Technická univerzita Košice, 2000, 13 s.

[Bab04] BABJAK, J.: Ako pracujú expertné systémy In: Science World http://www.scienceworld.cz/sw.nsf/0/4EF3C683DB1C92D9C1256E970048FE70?OpenDocument&cast=1 (2004-10-04)

[Bus04] BUS, J: Konštrukcia expertného systému In: My Clips, jednoduchý výkonný expertný systém http://myclips.wz.cz/phprs/view.php?cisloclanku=2004042501 (2004-04-25)

[Bus14] BUS, J: Znalostné inžinierstvo In: My Clips, jednoduchý výkonný expertný systém http://myclips.wz.cz/phprs/view.php?cisloclanku=2004052001 (2004-05-20)

[Bus24] BUS, J: Znalostné expertné systémy In: My Clips, jednoduchý výkonný expertný systém http://myclips.wz.cz/phprs/view.php?cisloclanku=2004041801 (2004-04-18)

[Esp05] Expertní systémy: Příručka k přednáškám http://ui.fpf.slu.cz/diplomky/znalostni_a_expertni_systemy/ (2005-04-02)

[Exs05] Expertné systémy http://neuron-ai.tuke.sk/~krankill/ui/exp_sys.html (2005-05-02)

[Fil05] FILIT, Otvorená filozofická encyklopédia http://dent.ii.fmph.uniba.sk/~filit/fvk/kybernetika_-_odkazy.html (2005-01-02)

[Har04] Hardware server: RS232 http://rs232.hw.cz/ (2004-09-11)

[Jon04] JONES D.W.: A Worked Stepping Motor Example http://www.cs.uiowa.edu/~jones/step/example.html (2004-05-12)

[Joh04] JOHNSON J.: Working With Stepper Motors
http://eio.com/jasstep.htm (2004-05-13)

[Kyb05] Kybernetika
http://karlik.gymkc.cz/vyuka/m/inf_minerva/kybernetika.html (2005-01-03)

[Mce05] Monte Carlo Experiments: "Drunken Sailor's" Random Walk
http://www.chem.uoa.gr/applets/appletsailor/Text_Sailor2.htm (2005-04-02)

[Rez02] ŘEZÁČ K.: Krokové motory, princip funkce, metody řízení
http://robotika.cz/articles/steppers/cs (2002-28-10)

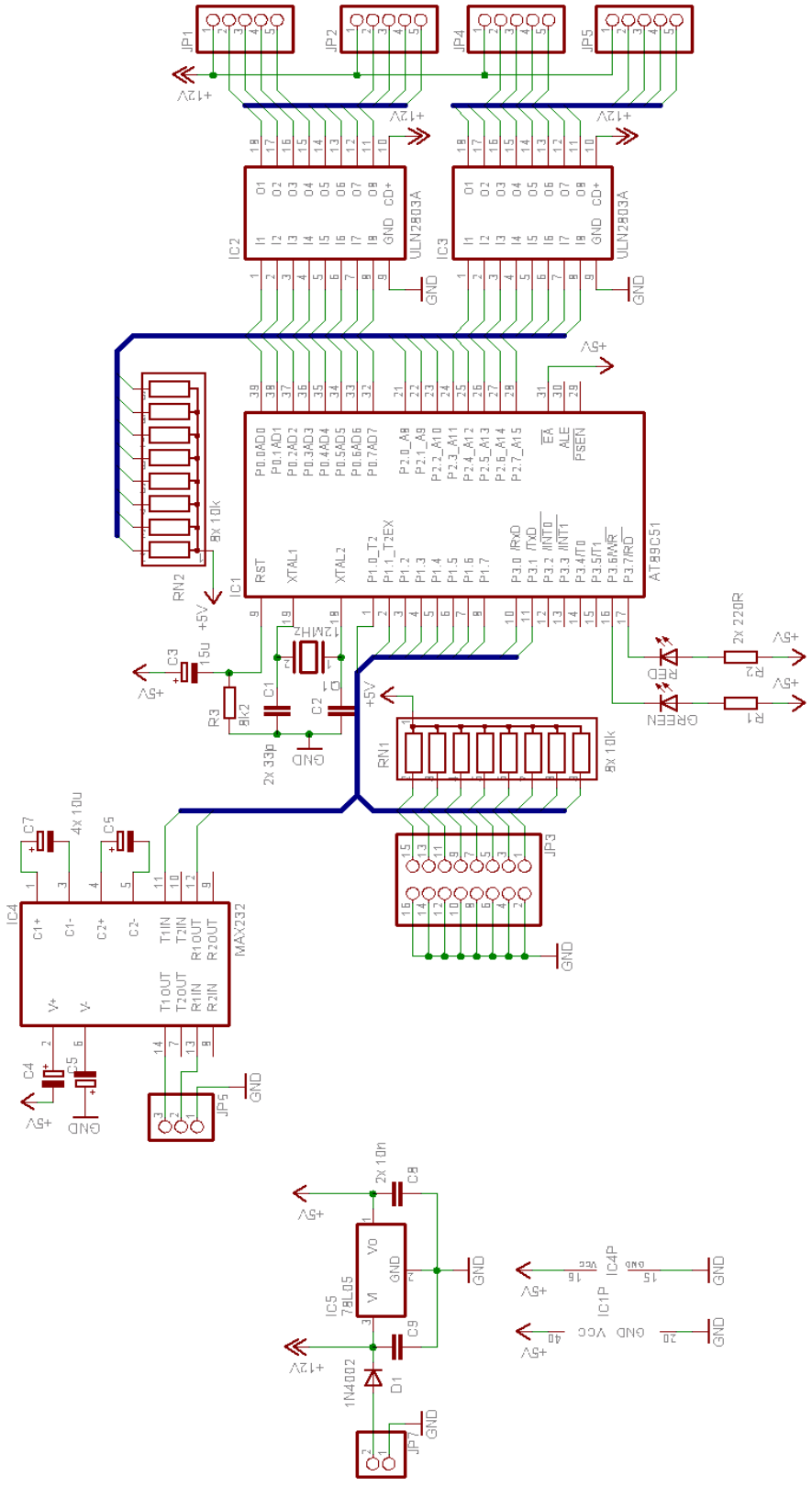
[Wik05] Wikipedia – The Free Encyclopedia
http://www.wikipedia.org heslá: artificial intelligence, cybernetics, expert systems, robot, robotics, (2005)

ZOZNAM POUŽITÝCH SKRATIEK

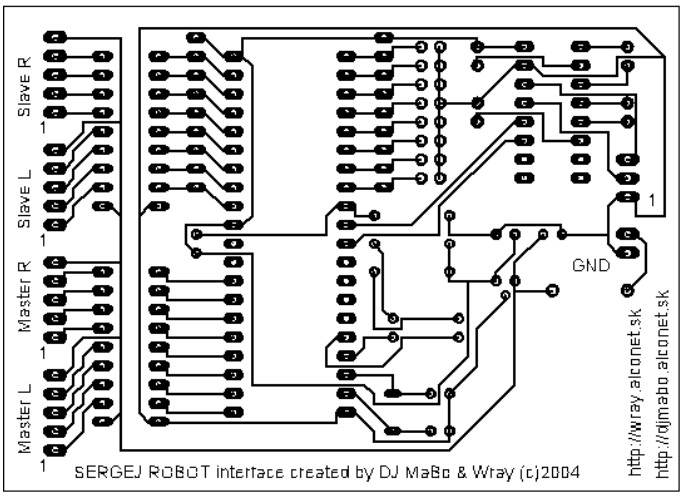
AI	Artificial Intelligence	Umelá inteligencia
BU	Báza údajov	
BZ	Báza znalostí	
ES	Expert System	Expertný systém
GV	Generátor výsledkov	
IM	Inference Mechanism	Inferenčný mechanizmus
KM	Komunikačný modul	
ĽS	Ľavá strana (súbor podmienok)	
PS	Pravá strana (súbor akcií)	
VM	Vysvetľovací modul	

ZOZNAM PRÍLOH

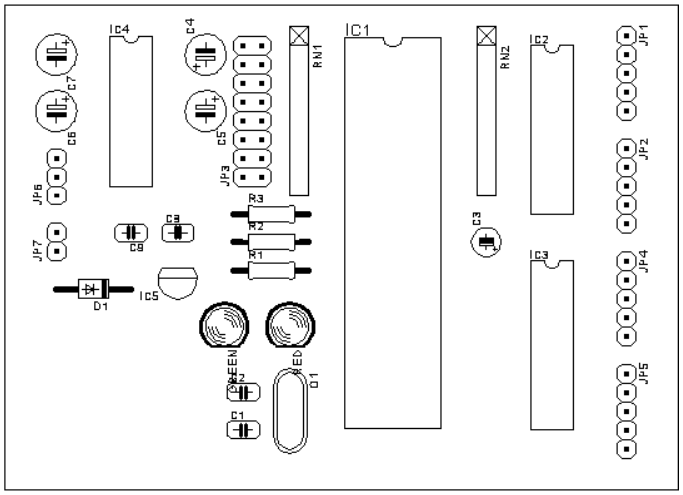
PRÍLOHA A: Konštrukcia robota



Obr. A.1 Schéma zapojenia



Obr. A.2 Pohľad na dosku plošného spoja zo strany spojov



Obr. A.3 Pohľad na dosku plošného spoja zo strany súčiastok

Rozpis súčiastok

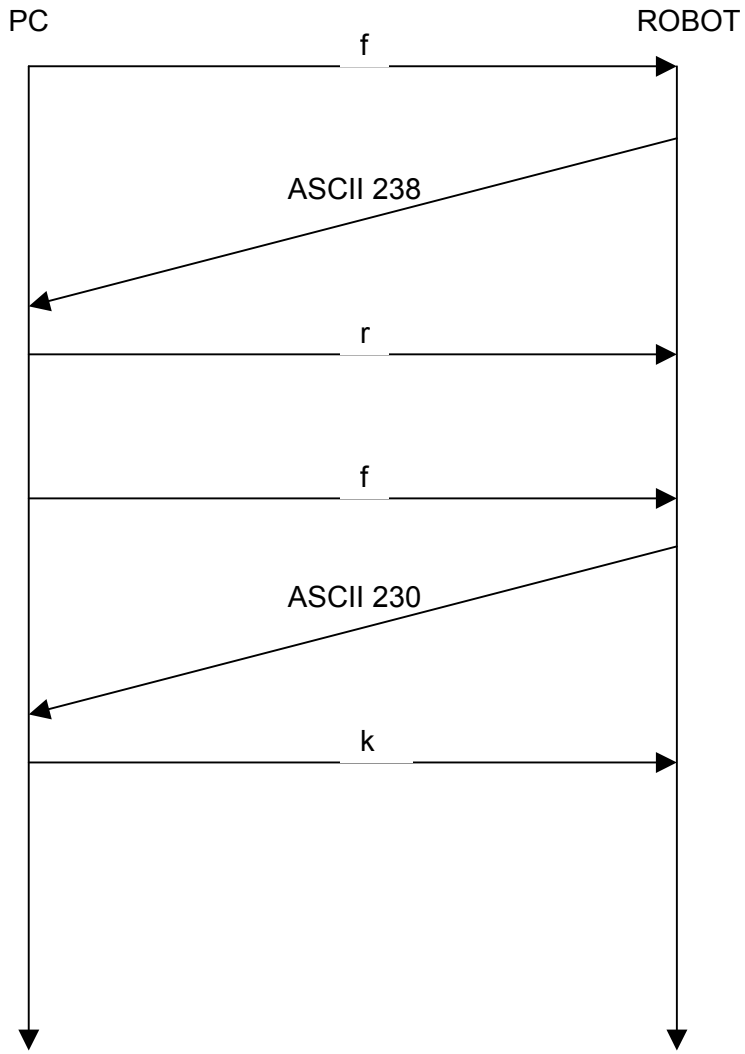
IC1	89C51	C4, C5, C6, C7	10uF/15V
IC2, IC3	ULN 2803	R1, R2	220R
IC4	MAX232	R3	8k2
IC5	78L05	RN1, RN2	8x10k
D1	1N4002	JP1, JP2, JP4, JP5	PINHD-1X5
D2, D3	LED 5mm	JP3	PINHD-2X8
C1,C2	33pF	JP6	PINHD-1X3
C3	15uF/15V	JP7	PINHD-1X2
Q1	12MHZ		

PRÍLOHA B: Tabuľka použitých riadiacich kódov

Tab. B.1 Tabuľka použitých riadiacich kódov

Riadiaci znak	Smer	Popis činnosti
ASCII 230	ROBOT -> PC	Zopnutý snímač na P1.0
ASCII 231	ROBOT -> PC	Zopnutý snímač na P1.1
ASCII 232	ROBOT -> PC	Zopnutý snímač na P1.2
ASCII 233	ROBOT -> PC	Zopnutý snímač na P1.3
ASCII 234	ROBOT -> PC	Zopnutý snímač na P1.4
ASCII 235	ROBOT -> PC	Zopnutý snímač na P1.5
ASCII 236	ROBOT -> PC	Rezerva
ASCII 237	ROBOT -> PC	Rezerva
ASCII 238	ROBOT -> PC	Krok dopredu OK
ASCII 239	ROBOT -> PC	Krok dopredu CHYBA
ASCII 240	ROBOT -> PC	Krok dozadu OK
ASCII 241	ROBOT -> PC	Krok dozadu CHYBA
ASCII 242	ROBOT -> PC	Otočenie doľava OK
ASCII 243	ROBOT -> PC	Otočenie doľava CHYBA
ASCII 244	ROBOT -> PC	Otočenie doprava OK
ASCII 245	ROBOT -> PC	Otočenie doprava CHYBA
f	PC -> ROBOT	Krok dopredu (F odozva)
b	PC -> ROBOT	Krok dozadu (B odozva)
l	PC -> ROBOT	Otočenie doľava
r	PC -> ROBOT	Otočenie doprava
t	PC -> ROBOT	Otočenie o 180 stupňov
s	PC -> ROBOT	Reštart robota
u	PC -> ROBOT	Vypnutie motorov
g	PC -> ROBOT	L odozva
h	PC -> ROBOT	R odozva
i	PC -> ROBOT	FL odozva
j	PC -> ROBOT	FR odozva
k	PC -> ROBOT	LF odozva
m	PC -> ROBOT	RF odozva
n	PC -> ROBOT	LB odozva
o	PC -> ROBOT	RB odozva
p	PC -> ROBOT	BL odozva
q	PC -> ROBOT	BR odozva
+	PC -> ROBOT	Zvýšenie rýchlosti
-	PC -> ROBOT	Zníženie rýchlosti

PRÍLOHA C: Príklad komunikácie robota s nadradeným počítačom



Obr. C.1 Príklad komunikácie robota s nadradeným počítačom

Na obr. C.1 je príklad komunikácie medzi robotom a nadradeným počítačom. Počítač najprv poslal po sériovom kanáli znak „f“, čo značí pohyb vpred. Robot sa teda pohol vpred. Keďže nenarazil do žiadnej prekážky, poslal po sériovom kanáli počítaču naspäť ASCII znak 238, ktorý značí, že krok vpred prebehol v poriadku. Počítač poslal po sériovom kanáli znak „r“, čo značí otočenie o 90 stupňov doprava. Keďže otočenie doprava prebieha v jednom štvorci, nemohlo dôjsť k narazeniu a robot teda nemusel hlásiť nadradenému počítaču, či sa krok vykonal v poriadku. Počítač iba počkal dostatočne dlhý čas, kým sa robot skutočne otočil a poslal mu znak „f“ aby sa robot pohol dopredu. Robot však pri pohybe dopredu narazil spínačom P1.0 do prekážky a preto poslal počítaču ASCII znak 230. Zároveň sa robot vrátil na pozíciu, v ktorej sa nachádzal predtým, ako narazil do prekážky. Počítač zanalyzoval situáciu a poslal robotovi znak „k“, ktorý znamená požiadavku aby sa robot otočil doľava, vykonal krok vpred, otočil sa doprava a vykonal krok vpred.

PRÍLOHA D: Tabuľka povolených génov

Tab. D.1 Tabuľka povolených génov

Poradie génu (situácia)	Dovolené hodnoty
0	1 2 3 5 6 7 8
1	1 2 5 6 7
2	1 2 3 5 7
3	1 2 5 7
4	2 3 7 8
5	2 7
6	2 3 7
7	2 7
8	1 2 3 6 8
9	1 2 6
10	1 2 3
11	1 2
12	2 8
13	2
14	2 3
15	2
16	1 3 5 6
17	1 6 5
18	1 3 5
19	1 5
20	3 8
21	4 11 12
22	3
23	4 11 12
24	1 3 6 8
25	1 6
26	1 3
27	1
28	3 8
29	4 11 12
30	3
31	4 11 12