MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

HERNÍ BOTI S UMĚLÝM VĚDOMÍM

UMĚLÁ INTELIGENCE (PB016)

RICHARD VĚŽNÍK
383561

14.12.2014

Obsah

1. Úvod

Umělá inteligence jako obor je velice rozsáhlá a také poněkud nedefinovatelná. Už sama otázka co je inteligence zatím nemá obecně uznávanou odpověď, ale většinou si pod tímto pojmem představujeme něco, co se rozhoduje podobně jako člověk – inteligentně.

Většinou je to jednostranně zaměřená inteligence, kdy se systém naučí (nejčastěji s pomocí příkladů) řešit jednu situaci, ať už s pomocí člověka nebo v informacích nalezne systém sám. Umělá inteligence tak úzce souvisí s rozličnými obory lidské činnosti, jejichž znalosti využívá.

Herní průmysl je v současnosti jedno z nejrychleji se rozvíjejících odvětví informatiky. Za minimálně posledních pětadvacet let lze sledovat tento rozvoj nejen na vizuální stránce hry (přechod z textového režimu na pixelovou grafiku, přechod od 2D ke 3D řešením apod.), ale i na kvalitě oponentů, proti kterým hráč soupeří. Samotná kvalita protivníků se vyvíjela také postupně – od jednoduchých algoritmů, přes algoritmy odpovídajících sekvenci postupů ze seznamu, až po algoritmy založené na naučených údajích, evoluci, s možností predikce postavené na neuronových sítích…

Cílem této práce je tedy seznámit čtenáře s problematikou umělé inteligence nejen v počítačových hrách.

2. Hledání cesty

Hledání cesty (pathfinding) je název jednoho z nejčastějších problémů a aplikace umělé inteligence ve hrách. Jedná se o hledání nejvhodnější cesty mezi dvěma body uloženými v rovině (2D hry: strategie apod.), anebo v prostoru (3D hry: letecké simulátory, First Person Shooter…) podle určitých pravidel tak, aby vyhovovali určitým kritériím. Jako příklad můžeme uvést hledání cesty pro zaútočení na nepřítele ve hře typu strategie: mezi pravidla například patří podmínka, že cesta nesmí vést přes vodu, anebo jinak neprůchozí terén, výstupním kritériem může být kritický čas, za který musí být schopný daný oddíl cíl dosáhnout.

2.1. McFly

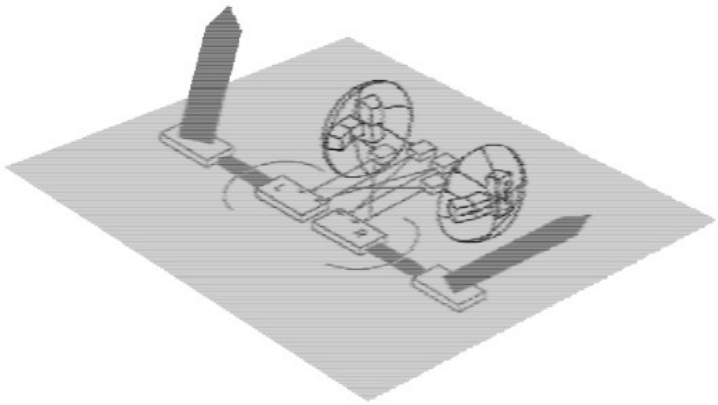
Jedním z možných způsobů, jak takové hledání cesty realizovat, je *genetický algoritmus na základě simulované evoluce*. Tento algoritmus byl použit i v pokusu, který měl demonstrovat učícího se letce v simulátoru přeletu dlouhou chodbou s horizontálními a vertikálními překážkami, jenž byl pojmenovaný Minimalistické chování pro vyhýbání se překážkám v prostoru.

Základ tohoto pokusu byl založený na poznatcích z experimentů na mouchách (zkoumali je vědci zabývající se biologií hmyzu). V simulovaném pokuse bylo možné vidět i takové chování, jako u samotných much – při pokusu o přelet z jednoho prostředí do druhého si hledali cestu tak, že opakovaně zkoušeli přeletět jedním místem, dokud nenašli vhodný průchod (například jak mouchy narážejí o sklo v okně, když hledají cestu ven).

Jednotliví simulovaní piloti (minimalističtí létající agenti) byli kombinací dvou motorů a čtyř snímačů pohybu (dva na každém boku – směr vertikální a horizontální). Navíc na každého pilota působila síla směrem dolů, kterou simulovali působení gravitace, aby se pokus co nejvíce blížil podmínkám v realitě. Celé řízení pilota bylo zabezpečené jednoduchou neuronovou sítí.

Byl tedy použit genetický algoritmus na vytvoření nejlepšího letce prostorem. Genetický algoritmus spočívá ve: vytvoření jedinců s nějakými parametry, jejich aplikace na daný pokus, výběr nejvhodnějších (nejšikovnějších) jedinců podle vhodných pravidel (např. Nejkratší čas přeletu, nejnižší počet kolizí s pře-

kážkami…) a následně jejich křížení, mutace a reprodukce. Tak vznikají nové generace jedinců, na kterých je tento postup znovu a znovu provedený a to až do chvíle, dokud nevznikne dvacátá osmá generace pilotů o dvou stech členech, která má vhodné výsledky.



McFly: létající agent (motory, detektory pohybu, propojení motorů a snímačů).

2.2. A star

Jedná se o heuristickou metodu hledání cesty. Heuristická metoda se od algoritmu liší, i když její výsledek může být stejný. Zatímco algoritmus je přesný postup, jak dosáhnout určitého výsledku - pokud zadáme platné hodnoty (vstup má vhodné hodnoty pro daný algoritmus), tak heuristika nezaručuje, že v hledání výsledku uspěje. Ale může k výsledku dospět i v případech, kdy konkrétní algoritmus na daný problém neexistuje.

Takovéto heuristické hledání cesty je použité například ve hrách typu Pac-Man a funguje následovně: známe startovní a cílovou pozici. Ve startovní pozici se nachází objekt, který chceme přesunout do cílové pozice konečným počtem kroků, a jestliže existuje více řešení, tak se použije to nejefektivnější. Z aktuální pozice, na které se přesouvaný objekt nachází, určíme další možné kroky (příklad možností: jeden krok vlevo, jeden krok vpravo, výstřel…) a tady se cesta algoritmu dělí na takový počet, kolik je možností na další krok, tudíž se jedná o rekurzivní aplikaci hledací funkce. V každé další cestě algoritmu se potom tento postup opakuje. Problémem tohoto algoritmu je však to, že kdybychom se v každém kroku vydali všemi směry z daného kroku, za velmi krátkou dobu by došlo k za-

plnění celé paměti počítače a také k velké spotřebě procesorového času; získání výsledku by bylo zdlouhavé a náročné (jedná se o exponenciální nárůst počtu prohledávaných směrů). Proto se do tohoto postupu aplikují ještě malé úpravy, respektive podmínky, které ho do značné míry kvalitativně vylepšují. Jedná se například o označování cesty, po které už prohledávací rutina procházela (aby nedošlo k vracení se po stejné cestě zpátky, popř. k zacyklení rutiny) a také upřednostňování některých směrů, jenž se podle nějakých pravidel zdají být lepší než ty ostatní (např. blížení se k cíli – zmenšování vzdálenosti mezi objektem a cílem).

Další nabízející se možnosti jak řešit hledání cesty jsou více matematické postupy, např. konverze mapy terénu do hranově orientovaného ohodnoceného stromu, na který je možné dále aplikovat známé algoritmy pro hledání minimální kostry, kritické cesty apod. (např. Dijkstrův a Bellman-Fordův algoritmus…)

Ve hrách není vždy žádoucí, aby cesta, kterou najdeme, byla perfektní a nejkratší. To proto, aby nebyla „moc počítačová“ a spíše vypadala lidštěji (menší obcházky apod.); tohoto se dá dosáhnout například přidáním šumu – připočítáme malé náhodné číslo do ohodnocené hrany grafu.

3. Příklady z her

3.1. Unreal Tournament

Boti v této hře jsou řešeni trochu netradičním způsobem oproti většině ostatních her. Je totiž možné, aby bot běžel na místním počítači nebo na síťovém. Při síťovém provozu si posílá tzv. senzorické zprávy (pozice v prostoru, vzdálenost, viditelnost nepřátelů apod.) Bot následně na tyto podněty reaguje a posílá příkazy (pohyb, střílení, hlášky…) nazpět do hry, která pak tyto příkazy vykonává.

Výhodou takovýchto botů je jednak možnost použití jakéhokoliv programovacího jazyka a druh umělé inteligence na jeho realizování (odpadá nutnost učit se specifický skriptovací jazyk pro boty v dané hře), tak nezávislost na procesorovém čase – na počítači, na kterém hra běží hra samotná, nemusí běžet i boti (ti můžou běžet na ostatních nevyužitých strojích v síti).

Nevýhodou je slabý přístup (pomalý – kvůli síti) k proměnným ve hře charakterizující prostředí hry, tedy nutnost odkládat si jejich hodnoty lokálně v

botovi; na straně druhé to má malou výhodu v tom, že bot nemůže podvádět tradičním způsobem (např. zeptat se hry, kde je nepřítel, kde je lékárnička…)

3.2. U.N. Bot

Výsledek výzkumu umělé inteligence na univerzitě v Edinburghu je projekt jménem U. N. Bot. Používá směs algoritmů a zajímavé algoritmy z oblasti počítačů a agentové architektury. Snahou jeho vývojářského týmu bylo dosáhnout co nejlidštějšího chování. Náplní práce tohoto bota je objevování a hlídání terénu. Výhodou jeho průzkumného algoritmu je to, že na rozdíl od standardních botů do her nepotřebuje žádnou naplánovanou trasu. Dokáže se sám bez jakéhokoliv externího zásahu projít po naprosto neznámém terénu a orientovat se v něm. Také sám dynamicky obchází překážky, takže se dokáže vyhnout kolizím za pomoci změny svého kurzu.

3.3. Chat-Boti, Alice

Chat-boti sice nepatří přímo do herního průmyslu, ale také obsahují různé aplikace umělé inteligence a též slouží více méně k zábavě (ať už formou přímé konverzace, anebo různými hláškami z her).

Dá se říct, že fungují ve dvou fázích: učení a komunikace. Ve fázi učení sledují rozhovory ostatních a sbírají věty do databáze typu otázka-odpověď. Ve fázi komunikace se snaží najít na položenou otázku buď přímou odpověď, anebo se snaží najít nějaké klíčové slovo v otázce a podle toho vytvořit odpověď (např. v otázce je slovo *počasí* a chat-bot, který si neví rady, odepíše podle klíčového slova například „U nás je hezky. Co u vás?“ V případě, že netuší, co má odpovědět, generuje náhodné univerzální odpovědi typu „Dobře, jak myslíš, hm-hmm…“ Asi nejznámějším chat-botem je Alice.

4. Prostředky a prog. jazyky pro UI ve hrách

4.1. Lisp

Považuje se za jazyk „jako stvořený pro programování umělé inteligence“. Základy mu položil John McCarthy. Některé hry se vytvářejí takovým způsobem, že

grafické (vykreslování, animace…) a jiné části (obsluha klávesnice, síťová zapojení…) se vytvoří v nižším programovacím jazyce (např. C++) a samotná logika hry a její řízení se píše ve vyšším skriptovacím jazyce. Takovým vyšším skriptovacím jazykem může být např. i implementace Lispu, protože Lisp jako takový je lehký, ale výkonný, skvělý na popis dat a je standardizovaný.

Příkaz jdi v imaginární hře bludiště pomocí funkce v Lispu:

```
(let (a (member (roomsymbol mistnost) (mistnost-cesta you)) (
  (if (eq a nil) (print "tam jit nemuzes") (setq you (car a))))
(nejsem si zcela jist jestli je spravne..kdyz ne tak me omluvte)
```

4.2. Prolog

Prolog patří mezi deklarativní programovací jazyky, ve kterých programátor popisuje pouze cíl výpočtu, přičemž přesný postup, jakým se k výsledku dostane, je ponechán na systému. Vznikl v roce 1972 a autorství je připisováno Alainovi Colmerauerovi a Philippemu Rousselovi. Jeho přednosti jsou velmi podobné jako u Lispu: lehkost, ale výkonnost…

Příkaz jdi v imaginární hře bludiště pomocí ProLogu:

```
jdi(X) :-       %funkce na chozeni
  you(L),       %do L ulozi vasi pozici
  connect(L,X),%zjisti jestli vede cesta z L do X-zde muze provadeni selhat
  retract(you(L)),      %Zrusi spojeni ze starou mistnosti
  assert(you(X)).       %Zavede do databaze novou asociaci-you a nova mistnost
jdi(X):-        %V pripade ze cesta nevede-prvni jdi selhalo
  write(' Tam to nejde. '),nl.
```

5. Závěr

Zábavní průmysl je v současnosti jedno z nejrychleji, ne-li nejrychleji se rozvíjející odvětví a herní část v něm zastupuje významný podíl. Dříve si prognostikové představovali rapidní vývoj umělé inteligence (Terminátory, Matrix…), ale vytvořit něco tak komplexního a složitého zabere spoustu, opravdu spoustu času. Nelze toho dosáhnout ani navyšováním hrubého výpočetního výkonu. Nejedná se totiž o dílčí problémy, ale je potřeba tyto problémy sjednotit (*problém inteligence*). Existují ale různé projekty, které mají všestranný přístup k problematice (*Artificial general intelligence*). Mezi ně například spadá i AIXI.

Bude zajímavé sledovat vývoj těchto systémů. Nabízí se totiž otázka, zda by tato umělá inteligence dokázala přijít na řešení, na která by člověk sám nepřišel. To se ale pravděpodobně v nadcházejících desetiletích nestane.

6. Bibliografie

[online]. [cit. 2014-12-14]. Dostupné z: http://www.kyb.tuebingen.mpg.de/fileadmin/user_upload/files/publications/pdfs/pdf373.pdf

[online]. [cit. 2014-12-13]. Dostupné: http://nlp.fi.muni.cz/uui/

[online]. [cit. 2014-12-14]. Dostupné z: http://dcgi.felk.cvut.cz/home/felkepet/projekty/Hruskf1_semestr_proj.pdf

[online]. [cit. 2014-12-15]. Dostupné: http://gamebots.sourceforge.net/

[online]. [cit. 2014-12-16]. Dostupné: http://neuralnetworks.ai-depot.com/bots/