Využívání genetických algoritmů v počítačových hrách

Petr Voborník
UHK – FIM – dEVOL
petr.vobornik@uhk.cz

Abstrakt

Využívání genetických algoritmů v počítačových hrách se může zdát jako celkem logická alternativa, jak v nich programově docílit umělé inteligence. Zaměříme se tedy na teoreticky možné způsoby těchto využití a především pak jejich existující realizaci v praxi, což bude vždy demonstrováno na vybraných příkladech.

Klíčová slova

Genetické algoritmy, umělá inteligence, počítačové hry, deskové hry, generování scény.

Úvod

Genetické algoritmy jsou populární a především účinnou metodou pro hledání dostatečně vyhovujícího řešení ve velkých stavových prostorech, které nelze definovat snadno řešitelnou funkcí, a kde z výpočetních a časových důvodů selhává klasické prohledávání obyčejnou „hrubou silou".

S využitím vhodných heuristických metod pak lze samotné genetické algoritmy „vylepšit", například při generování výchozí populace, v operátorech mutace či křížení a především u sestavování fitness funkce. Problémy vhodné pro řešení genetickými algoritmy by měly mít jasně definované podmínky řešení a proto je třeba tyto heuristiky volit s rozvahou individuálně pro každý řešený problém. Tímto způsobem vznikají takzvané hybridní genetické algoritmy.

Jednou z oblastí, kde se využití genetických algoritmů přímo nabízí, jsou počítačové hry. V nich je snaha nasazovat stále vyšší stupeň umělé inteligence, která je podstatným faktorem ovlivňujícím oblíbenost, věhlas a tedy i prodejnost těchto her. Vzhledem k prostředí, v němž jsou tyto hry provozovány – na počítačích – mají ideální možnost využít právě genetických algoritmů toto prostředí vyžadujících, a ty tak mohou při vhodné aplikaci představovat umělou inteligenci protihráče člověka.

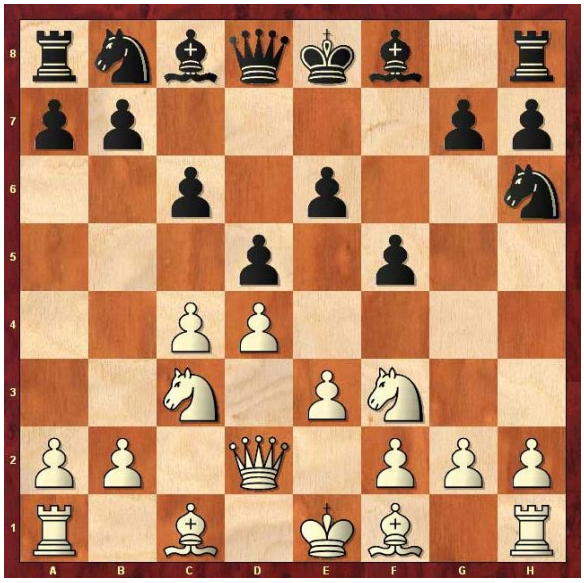
[14]

Zaměříme se zde na několik oblastí, ve kterých jsou genetické algoritmy v počítačových hrách nejčastěji nasazovány a na konkrétních případech si demonstrujeme, jak tomu je ve skutečnosti v reálné praxi. Vybíráno bude především z deskových, přesněji řečeno tahových her, kdy má každý hráč nějaký čas na promyšlení dalšího tahu v závislosti na aktuálním stavu hry, změněném posledním tahem protihráče. Genetické algoritmy totiž vyžadují určitý výpočetní čas na „prohledání" stavového prostoru a nalezení vhodného řešení, tudíž jejich nasazení například v real-time hrách není v této formě zcela vhodné. Tyto hry totiž taktéž spotřebují podstatnou část výpočetního času procesoru pro zobrazení a ozvučení scény, a především na požadavek reagovat v co nejkratší možné době, čímž by se při současném probíhání genetického algoritmu vzájemně zpomalovaly [11].

Umělá inteligence protihráče

Šachy

Asi nejznámější tahovou hrou pro dva hráče jsou šachy. Mají jasně určená pravidla, daný cíl, pevně vymezenou hrací plochu a definované vlastnosti hracích kamenů. Šachy se dočkaly již stovek zpracování jako hra pro počítače, s různě vydařeně naprogramovanou umělou inteligencí pro počítač, jako protihráče člověka. Mnohé z nich pak dají zabrat i šachovým velmistrům a ty nejlepší dokáží důstojně soupeřit i se světovými špičkami v šachu.



Obr. 1 – Ukázka rozehrané partie šachu [1]

Ovšem ač se šachy mohou na první pohled zdát ideálním kandidátem pro aplikaci genetických algoritmů, v praxi tomu tak vůbec není a je jich využíváno jen poskromnu. Zde totiž tento způsob přístupu stále kvantitativně (co do času nezbytného pro nalezení vhodného řešení) i kvalitativně (vhodnosti nalezeného řešení - tahu) zaostává před klasickými postupy. Užití přímých heuristických metod spolu s hledáním hrubou silou a dále pak vyhledávání v databázi již odehraných partií totiž poskytuje kvalitnější výsledky v kratším čase. Programy implementující genetický přístup tedy zůstávají spíše pomůckou pro studijní účely, nežli komerčně úspěšným software [2].

I když tedy pro samotnou hru v šach není v praxi genetických algoritmů příliš využíváno, jsou hry odvozené z šachu poměrně často předmětem úloh pro řešení právě s jejich pomocí. Patří mezi ně třeba dohrání různých speciálních koncovek šachových partií (např. KRK[2], KRKN[3], ...) [13], řešení problému N dam[4] [3], apod.. Ovšem i zde stále vítězí odlišné přístupy. Postupy pro dohrání klasických šachových koncovek jsou již obecně známy, a standardní prohledávací strom tyto postupy dodržující nebývá až zas tak nepřekonatelně rozsáhlý, aby bylo třeba uchylovat se ke genetickým algoritmům. Případně se od jistého postavení hry stačí jen držet výherní strategie bez ověřování, vede-li skutečně k cíli [13]. Stejně tak problém N dam je výhodnější řešit vhodnými heuristickými algoritmy. Nicméně vzhledem k jednoduššímu stupni obtížnosti definování a vyhodnocování fitness funkce (u koncovek dát mat na co nejmenší počet tahů, u N dam počet konfliktů) bývá k těmto úlohám ve sféře genetických algoritmů dosti často saháno jakožto cvičebním či prezentačním.

Některé další tahové hry

A jak je tomu s využíváním genetických algoritmů u ostatních známých tahových her? O některých z nich se tu tedy letmo zmíníme.

Othello (Reversi)

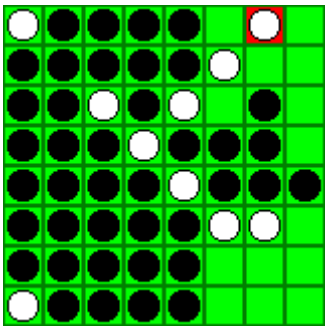
V této hře se rozhodně nelze rozhodovat pouze podle počtu zabraných protihráčových kamenů pro každou alternativu aktuálního tahu, jak by se na první pohled třeba mohlo zdát, ale je zapotřebí strategii plánovat o mnoho kroků dopředu.

[1] http://www.excalibur-publishing.co.uk/fritzscr.htm

[2] KRK – Šachová koncovka, kdy na šachovnici zbývá pouze bílý král, bílá věž a černý král v několika možných specifických postaveních.

[3] KRKN – Šachová koncovka, kdy na šachovnici zbývá pouze bílý král, bílá věž a černý král a černý jezdec.

[4] Problém N dam – Úkolem je rozmístit N dam na šachovnici rozměrů NxN tak, aby se vzájemně neohrožovaly.

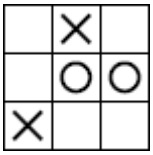


Obr. 2 – Ukázka hry Othello [4]

Hra Othello je taktéž velmi často využívána jako studijní příklad využití genetických algoritmů [5], [4]. Tyto projekty pak bývají více či méně zdařilé co do umělé inteligence protihráče. Ovšem například testovaná verze [4] založená na genetickém algoritmu příliš neobstála a daleko lepších výsledků dosahovaly verze hry využívající heuristické strategie než-li slepého, byť geneticky urychleného prohledávání.

Piškvorky 3x3

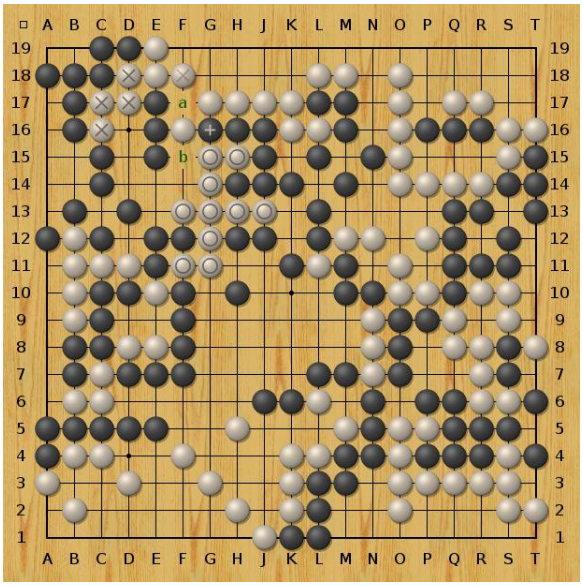
U piškvorek s hrací plochou o rozměru 3x3 (Tic-Tac-Toe) je stavový prostor docela malý. Celkem je možno odehrát „pouze" 9! (362 880) různých her, přičemž na hrací ploše se může vyskytovat jen 3^9 (19 683) různých stavů. Takovýto stavový prostor již není neproveditelné prohledat celý a nalézt tak nejlepší možný tah, ovšem i zde lze využitím genetického algoritmu počet prohledávaných možností velmi významně redukovat [7]. Nicméně v této hře je celkem jasná strategie jak neprohrát, tedy minimálně remízovat, kterou většina hráčů odhalí již po několika málo odehraných partiích. Implementace této strategie přímo do uměle inteligence počítačového protihráče poté již není nikterak složitá [6].



Obr. 3 – Ukázka hry Tic-Tac-Toe [7]

GO

Klasická hra GO pro dva hráče se hraje na hrací ploše o rozměru 19x19, kde každý z nich postupně pokládá své hrací kameny. Pomineme-li ostatní pravidla, je počet možných stavů hry $3^{(19*19)}$, tedy více než $1{,}74*10^{172}$. Počet potenciálních tahů v každém kole je průměrně 250-300, čili 10x více než je tomu například v případě šachů. Celkový počet všech možných partií pak vede ještě k daleko větším hodnotám. Ač tedy v této hře vystupují pouze dva hrací kameny (černý a bílý), stavový prostor je i tak neskutečně obrovský, že zde jakékoli prohledávání hrubou silou, i při použití nejrůznějších heuristik selhává. Hybridní genetické algoritmy sice znamenají značné urychlení, ovšem i tak mohou většinou soupeřit maximálně s lidskými začátečníky v této hře.



Obr. 4 – Ukázka hry GO [5]

Jistou nadějí pro úspěšnější umělou inteligenci hry GO je propojení hybridního genetického algoritmu a neuronovou sítí. [9]

Další tahové hry

Mezi další tahové hry, kde lze aplikovat genetické algoritmy patří například Dáma, Mastermind, Backgamon, Scrabble, Bridge, Poker atd.. [14]

[5] http://home.gna.org/quarry/screenshots.html

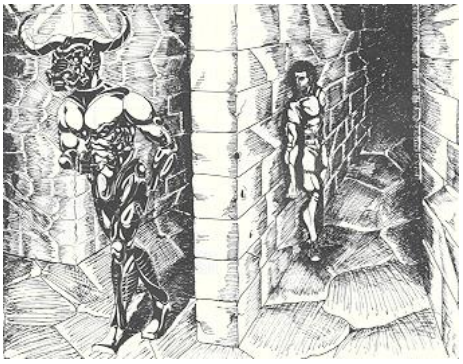
Generování scény

Jednou ze žádoucích vlastností většiny her je vedle kvalitní umělé inteligence protihráče také unikátnost každé hry, tedy co největší počet kol (levelů). Ať jich však již tvůrci hry připraví jakýkoli počet, bude vždy konečný. Jistou možností je hráčům vytvořit editor, kde si mohou jednotlivé levely vytvářet sami, nicméně i v tomto případě je velmi užitečnou funkcí možnost nechat level automaticky vygenerovat (a pak až jej třeba případně upravit).

Automatické generování, zvláště není-li přítomen editor, však sebou nese jeden specifický požadavek – hratelnost vygenerovaného levelu – tedy, že je vůbec možné tento level úspěšně zakončit. Jeho obtížnost je pak záležitostí další. Postup takovéhoto generování si ukážeme na následujícím příkladu.

Theseus a Minotaurus

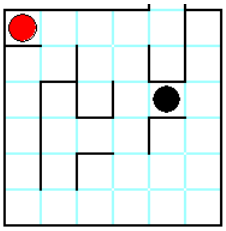
Na této jednoduché hře pro jednoho hráče bude prezentováno generování scény (levelu) s využitím genetického algoritmu.



Obr. 5 – Ilustrace ke hře Theseus a Minotaurus [1]

Pravidla

Pravidla této hry jsou následující: hrací plocha je ohraničena a rozdělena na čtverce o počtu NxN. Některá hrací pole ohraničuje po některých jejich stranách zeď. Z hrací plochy existuje právě jeden východ beze zdi. Na hracím poli jsou dva hráči: Theseus a Minotaurus. Jakmile Theseus vykoná jeden krok (tah – pohyb o jedno hrací pole horizontálně či vertikálně, nikoli však přes zeď), následují dva tahy Minotaura. Ty se řídí těmito pravidly: Nejprve otestuje, může-li se posunout horizontálně směrem k Theseovi. Pokud ano, posune se tam. Poté otestuje, může-li se posunout vertikálně blíže k Theseovi a pokud ano posune se tam. Jestliže to předtím nešlo horizontálně, je tento test opakován znovu a pokud opět nevyjde, je ještě jednou vyzkoušen test pro pohyb vertikální. Pokud by se posunutím horizontálně ani vertikálně více nepřiblížil Theseovi, zůstane na místě. Vynechat tah může samozřejmě i Theseus. Jeho úkolem je tedy dostat se labyrintem z hrací plochy ven, aniž by ho Minotaurus dostihl (posunul se na stejné hrací pole jako je on). [1]



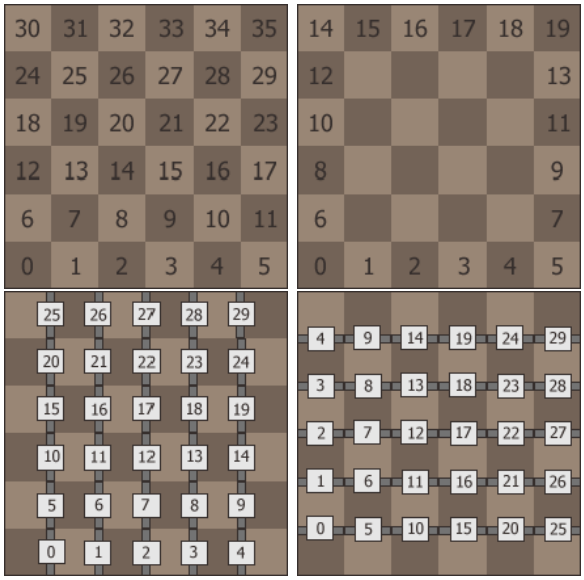
Obr. 6 – Hra Theseus a Minotaurus, 6x6 [6]

Zakódování rozložení scény

Vše bude prezentováno na příkladu hrací plochy o rozměrech 6x6 (N=6).

Nejprve je tedy třeba zvolit vhodné zakódování rozložení scény do „chromozomu". Do něho se musí vměstnat výchozí poloha Thesea a Minotaura, poloha východu a rozložení zdí. Theseus a Minotaurus mohou teoreticky začínat na libovolném hracím poli, což je v tomto případě 36 možností (N^2) a na zakódování této hodnoty do binárního řetězce je třeba 2∗6 bitů (viz Obr. 7a). Poloha východu může být pouze v okrajových polích, kterých je 20 (4∗N-4), tedy 5 bitů (viz Obr. 7b). V rozích sice může být východ umístěn ve dvou směrech, to už ale hru nijak neovlivní. Zdi pak mohou být celkem na 60ti různých místech (2∗N∗(N-1)), na 30ti vertikální a na 30ti horizontální (viz Obr. 7c,d). Zde bude nejpraktičtější jejich polohy zakódovat do dvou řetězců o délce 2∗30 bitů, kde 0 bude znamenat nepřítomnost zdi a 1 naopak její výskyt.

[6] http://www.logicmazes.com/theseus.html



Obr. 7 – Systém kódování rozložení scény: a) možnosti výchozích pozic Thesea a Minotaura, b) možnosti pozice východu z bludiště, c) možnosti umístění vertikálních zdí, d) možnosti umístění horizontálních zdí [8]

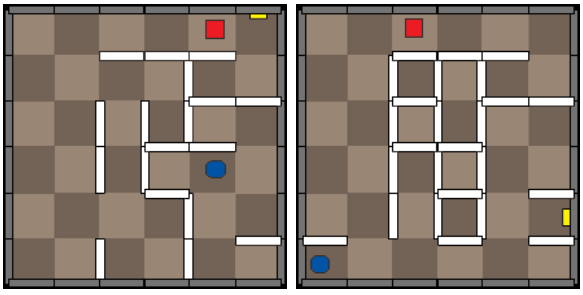
Dále je třeba ošetřit alternativy, že určení pozice hráčů či východu přesáhne možný rozsah (0-35, 0-19), což délka bitů (6, 5) umožňuje (0-63, 0-32). V tom případě bude pozice hráče či východu upravena funkcí modulo, tedy *hodnota mod 35* (případně 19), což ji vrátí do příslušných mezí.

Celkem tedy bude mít jeden chromosom délku 77 bitů (2∗6 + 5 + 2∗30). Je tedy zřejmé že možných kombinací rozmístění je 2^{77}, po úpravě pozic $36*36*20*2^{30}*2^{30}$ = 29 883 725 399 409 473 617 920, tedy zhruba necelých $3*10^{22}$ možností rozmístění scény. Pro hrací plochu 8∗8 by to bylo již téměř $6*10^{38}$. Stavový prostor je tedy skutečně veliký a prioritním úkolem genetického algoritmu bude nalézt přípustná řešení. [8]

Neřešitelné varianty

Při generování mohou nastat tyto neřešitelné varianty výchozího postavení a rozložení:

- Theseus se nachází na stejném hracím poli jako Minotaurus.
- Zdi zcela ohraničují Thesea a neumožňují mu přístup k východu (viz Obr. 8a).
- Neexistuje žádná možnost výhry, Theseus je vždy cestou k východu dostižen Minoteurem (viz Obr. 8b). [8]



Obr. 8 – Příklady neřešitelných variant: a) Theseus ● nemá přístup k východu z bludiště ▭, b) Theseus je vždy dostižen Minotaurem ■ [8]

Generování scény

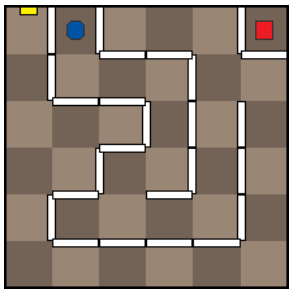
Pro celý proces je použit klasický postup genetického algoritmu: vygenerování náhodné výchozí populace, ohodnocení jednotlivých jedinců populace vhodnou fitness funkcí, výběr jedinců pro křížení metodou rulety, křížení vybraných jedinců, mutace, vznik nové populace atd., dokud není vygenerováno dostatečně vhodné řešení či více různých řešení. Úspěchu či neúspěchu celého algoritmu tedy záleží především na vhodně zvolené fitness funkci.

Fitness funkce má za úkol směřovat genetický algoritmus k eliminování neřešitelných variant a penalizovat nekvalitní (snadno řešitelné) varianty.

Pro určení, je-li level řešitelný či nikoli a zároveň získání dat pro fitness funkci je třeba jej vyřešit. Nejedná se však o nic jiného než klasické prohledávání stavového stromu, nejlépe metodou do šířky. Prohledávaný strom může nabývat pouze 1 296 stavů (36 možných pozic Thesea * 36 možných pozic Minotaura). Rozložení zdí a východu zůstává vždy stejné. Samozřejmostí je detekce a eliminace tahů vedoucích k zacyklení.

Prohledáváním do šířky se tedy nalezne první nejsnadnější řešení (s nejmenším počtem tahů), případně po prohledání celého stromu je určeno, že level nemá řešení. Při úspěšném řešení jsou na cestě k němu vedoucí spočteny tyto hodnoty:

- Suma počtu možných akcí v každém tahu, tedy kolika směry se Theseus mohl v každém svém tahu vydat. Tato hodnota odlišuje levely kde je mnoho možností pohybu a hráč tak musí přemýšlet nad tím, kterou zvolí od levelů, kde má jednu jasně zdmi vyznačenou cestu.
- Délka cesty, čili počet tahů (navštívených polí). Tím se odlišuje krátké (snadné) řešení od delšího (těžšího).
- Počet tahů Minotaura. Tento faktor má také podstatný vliv na obtížnost levelu, protože kdyby například byl Minotaurus od Thesea odříznut zdmi, či od začátku „zaklesnutý" v nějakém uskupení zdí do tvaru písmene U, nebyla by hra příliš obtížná (viz Obr. 9).



Obr. 9 – Level, kde je zcela eliminována hrozba Minotaura [8]

Z těchto hodnot pak lze již snadno ohodnotit fitness každého zjištěného levelu, jak například ukazuje Vzorec 1.

$$Fitness = \frac{pocet\_akcí}{6 \cdot velikost\_mapy} \cdot 200 + \\ + \frac{pocet\_tahů}{24 \cdot velikost\_mapy} \cdot 100 + \\ + \frac{pocet\_tahů\_Minotaura}{5 \cdot velikost\_mapy} \cdot 200$$

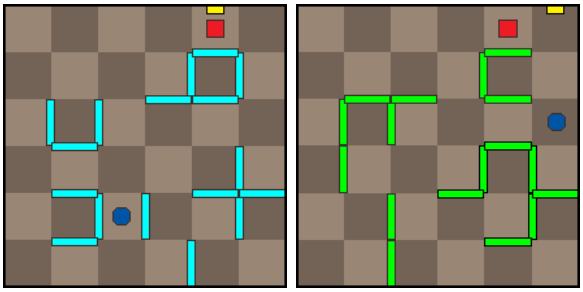
Vzorec 1 – Fitness funkce hodnotící parametry nalezeného řešení vygenerovaného levelu hry Theseus a Minotaurus.

Vzorec 1 počítá rovnou s možností různě velikých map a dle její velikosti upravuje příslušnou hodnotu parametru. Míra úprav parametrů pak vychází ze zkušeností s kvalitními mapami. Funkce dále váhami (2:1:2) přikládá důležitost takto upraveným hodnotám parametrů.

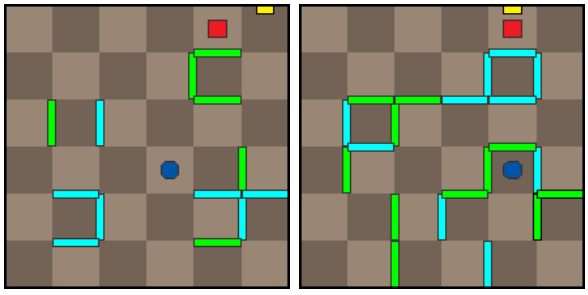
Co se týče neřešitelných variant, je možné jim přiřadit nějakou hodně nízkou leč nenulovou hodnotu fitness třeba dle dalších kritérií, nebo je z následujícího výběru zcela vyloučit (nastavit jim hodnotu fitness = 0) a místo nich náhodně vygenerovat levely nové.

Vzhledem k tomu, že v tomto případě může jedinou změnou vysoce kvalitního levelu vzniknout level neřešitelný, je zde vhodné nechat „přežít elitu", tedy určitý počet nejlepších řešitelných levelů propustit do další generace beze změn.

V případě křížení je příhodné použít vícebodové křížení, jelikož dojde k více komplexnímu promísení rozložení zdí, než u křížení jednobodového.



Obr. 10 – Dva jedinci vybraní pro operaci křížení [8]



Obr. 11 – Dva potomci vzniklí křížením jedinců z Obr. 10 [8]

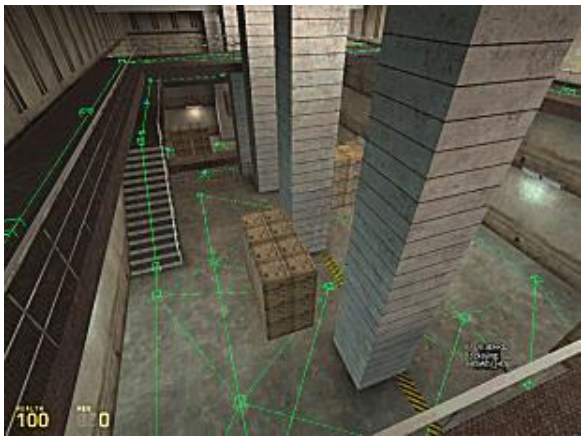
Na Obr. 11 vlevo je nalezené řešení následující: →↑→←← ↓↓↓→→ o←←←← ←↑↑↑↓ ↓↓→→→ ↑↑→↑→ ↑↑. Počet tahů je 32, počet možných akcí 152 a počet pohybů Minotaura 15. Výsledek fitness funkce je tedy 383,333.

Na Obr. 11 vpravo je nalezené řešení následující: ↓↓→o← ↑←↓←↑ ↑←↓←↑ ↑↑→→→ ↑→. Počet tahů je 22, počet možných akcí 97 a počet pohybů Minotaura 10. Výsledek fitness funkce je tedy 256,25.

Operátor mutace pak může zasáhnout do celého procesu změnou (negací) některého bitů chromozomu. To má za efekt, že se v nové generaci někde buď objeví, nebo naopak zmizí zeď, případně se změní výchozí pozice Thesea, Minotaura či východu. [8]

Akční hry a strategie

V akčních hrách a strategiích nacházejí genetické algoritmy také své využití, i když tedy neprobíhají přímo při hře samotné. Používají se především k evoluci chování protivníků, přičemž hráčův styl hry zde funguje jako hodnotící funkce - lepší protivníci jsou ti, kteří dokázali hráče porazit nebo dosáhli lepšího skóre. Další generace jsou proto ve hře proti tomuto hráči úspěšnější. Tento přístup je aplikovatelný hlavně v akčních hrách, protože testování protivníků probíhá rychle díky jejich vysoké úmrtnosti. Pro jiné žánry, např. strategické hry, kdy testování jednoho protivníka trvá dlouho, se nabízí alternativa vyvinout protivníky předem při vývoji hry, a hráči představit hned na začátku pouze nejúspěšnější výsledky. Jinou možností je nalezení kvalitních hráčů metodou kompetice – nechat různě naprogramované umělé inteligentní hráče hrát proti sobě a vybrat ty vyhrávající (zde je výhodou rychlost testování, která je závislá pouze na výkonu počítače místo na lidských schopnostech). [12]



Obr. 12 – Navigační síť ve hře Half-Life [10]

Závěr

Genetické algoritmy (spíše tedy hybridní genetické algoritmy) se v počítačových hrách využívají spíše zřídka. Umělou inteligenci hráče pro většinu her lze totiž daleko efektivněji řešit definováním příslušné strategie vyplývající z podstaty a pravidel každé hry. Heuristikami je pak možné vymezit klasické prohledávání stavového prostoru hry tak, že nalezený výsledek bývá většinou vyšších kvalit, než výsledek nalezený genetickým algoritmem, kde není záruka nalezení globálního maxima. Většina aplikací genetických algoritmů pro umělou inteligenci v počítačových hrách tak spíše zůstává akademického charakteru.

Nicméně genetické algoritmy mohou být velmi nápomocny při učení této „umělé inteligence". Jednak mohou automaticky generovat velkou spoustu partií a z těch úspěšnějších odvozovat výherní strategie. Nebo mohou tyto partie sloužit pro automatické učení neuronových sítí. Případně lze zkoušet různé strategie na lidského protivníka a pomocí genetických operátorů je v každé další hře upravovat a vylepšovat.

Genetických algoritmů se dá také s úspěchem využívat i v jiných aspektech počítačových her, než jen pro simulaci inteligence hráče. Například při generování scén levelů není požadováno nalezení nejlepšího výsledku, ale pouze výsledku hratelného a dostatečně kvalitního. V této oblasti pak genetické algoritmy při správné aplikaci mohou dosahovat velmi kvalitních výsledků a připravovat tak hráčům neustále nová a nová herní kola.

Hlavní sféra pro využívání genetických algoritmů tak zůstává spíše v aplikaci na konkrétní problémy reálného světa. Pro tyto účely lze totiž algoritmus vždy optimalizovat pro každý konkrétní problém, jehož každé lepší řešení většinou přináší i určitý prospěch.

Literatura

[1] Abbott, R.: *Theseus and the Minotaur*, logicmazes.com, 1994.

[2] Cogley, J.: *Designing, implementing and optimising an object-oriented chess system using a genetic algorithm in Java and its critical evaluation*, The Open University's Master of Science Degree in Computing for Commerce and Industry, 2001.

[3] Crawford, K.D.: *Solving the n-queens problem using genetic algorithms*, Kansas City, Missouri, USA, 1992.

[4] Currie, D.: *Othello game player*, St John's College, Oxford, UK, 1997.

[5] Eskin, E.; Siegel, E.: *Genetic Programming Applied to Othello: Introducing Students to Machine Learning Research*, Columbia University, New York, USA, 1999.

[6] Gordon, A.: *A general algorithm for tic-tac-toe board evaluation*, Journal of Computing Sciences in Colleges, USA, 2006.

[7] Hochmuth, G.: *On the Genetic Evolution of a Perfect Tic-Tac-Toe Strategy*, Stanford University, Stanford, California, USA 2003.

[8] Chaisilprungrueng, S.; Moradi, H.: *Designing a Theseus-Minotaur game using Genetic Algorithms*, University of Southern California, Computer Science Department, Los Angeles, USA, 2003.

[9] Churchill, J.; Cant, R.; Al-Dabass, D.: *A hybrid genetic alternative to game tree search in GO*, The Nottingham Trent University, Nottingham, UK, 2004.

[10] Jakob, M.: *Umělá inteligence v počítačových hrách*, scienceworld.cz, ČR, 2004.

[11] James, G.: *Using Genetic Algorithms for Game AI*, GIGnews.com, 2005.

[12] Kukačka, M.: *Umělá inteligence v počítačových hrách*, Univerzita Karlova, Matematicko-fyzikální fakulta, Praha, ČR, 2004.

[13] Lassabe, N.; Sanchez, S.; Luga, H.; Duthen, Y.: *Genetically programmed strategies for chess endgame*, Seattle, Washington, USA, 2006.

[14] Musil, D.: *Možnosti a srovnání freeenginů pro umělou inteligenci herních avatarů*, Masarykova univerzita, Fakulta informatiky, Brno, ČR, 2005.

[15] Repenning, A.: *Excuse me, I need better AI! Employing Collaborative Diffusion to make Game AI Child's Play*, Boston, Massachusetts, USA, 2006.

[16] Shannon, C.E.: *Programming a Computer for Playing Chess*, Philosophical Magazine, Bell Telephone Laboratories, Inc., New York, USA, 1950.