Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Genetické algoritmy a genetické programování

Hledání řešení bludiště pomocí genetického algoritmu

Bakalářská práce

Autor: Josef Bydžovský

Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Barbora Tesařová, Ph.D.

Hradec Králové duben 2016

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

podpis

V Hradci Králové dne

Poděkování:

Poděkování patří mé matce Zdeně Bydžovské a bratrovi Arnoštu Táborskému za psychickou a materiální podporu. Též bych chtěl poděkovat Ing. Barboře Tesařové, Ph.D. za korigování práce, odborné rady a návrhy. V neposlední řadě děkuji Mgr. Martině Vrbické a mé přítelkyni Kateřině Hlaváčové za korekturu textu.

Anotace

Práce je zaměřena na představení krátké historie, základního způsobu myšlení a technik v oblasti genetických algoritmů a genetického programování. Dále následuje definování problému hledání cesty v bludišti, který bude pomocí genetického algoritmu řešen. Poté následuje představení a rozbor řešení tohoto problému, který navrhl Indický doktor Nitin S. Choubey. Na závěr jsou prováděny úpravy tohoto řešení, které jsou testovány a porovnávány s původní verzí.

Klíčová slova

genetika, chromozom, jedinec, populace, fitness, křížení, hybridní, bludiště, a-maze

Title: Searching for solution in a maze with genetic algorithm

Annotation

This work is aimed on presenting short history, basic way of thinking and techniques in area of genetic algorithms and genetic programing. Followed by defining a problem of searching a way in a maze, witch going to be solved with genetic algorithm. After that is presenting and examining solution of this problem, that is designed by Indian doctor Nitin S. Choubey. In the end changes of this solution are made, which are tested and compared with original version.

Key words

genetic, chromosome, individual, population, fitness, crossover, hybrid, maze, a-maze

Obsah

1. Úvod

Genetické algoritmy společně s genetickým programováním naleznou v dnešní době uplatnění zejména ve dvou oblastech. První z nich je strojové učení a umělá inteligence, kdy se snažíme dospět na základě nějakého vstupu k potřebnému výstupu (1). Druhou oblastí, kdy může být genetické algoritmy, popř. genetické programování alternativou, jsou problémy s takovou výpočetní složitostí, kdy by tradiční techniky nenašly řešení v pro člověka přijatelném čase (2). V obou případech se jedná o vyhledávání z velkého množství potencionálních řešení, která jsou více či méně blízko k námi vyhledávanému výsledku. (2)

"It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change"

-Charles Darwin

1.1. Cíl bakalářské práce

Cílem této práce je v první řadě čtenáře uvést do problematiky, terminologie a myšlení v oblasti genetických algoritmů a genetického programování, uvedení důležitých milníků z historie genetických algoritmů a genetického programování, vysvětlení rozdílů mezi genetickými algoritmy a genetickými programy. Poté bude následovat definování problému, který se bude pomocí genetického algoritmu řešit a bude podrobně popsáno řešení pana doktora Nitin S. Choubeyho. Ke konci bude popsáno několik úprav originálního řešení a na závěr budou výsledky jednotlivých verzí porovnávány s původním řešením.

2. Historie

V následujících podkapitolách práce jsou stručně představeni tři nejvýznamnější osobnosti, které se nejvíce zasloužily o podobu genetických algoritmů a genetického programování, jak ho známe dnes.

2.1. Charles Darwin

První a nepřímé základy genetickému programování a genetickým algoritmům postavil britský přírodovědec a zakladatel evoluční biologie Charles Darwin. Ten ve svém díle *O vzniku druhů přírodním výběrem čili zachováním vhodných odrůd v boji o život* (3) poprvé zveřejnil svou evoluční teorii, která byla později upravována a obohacována. Nicméně její základní myšlenka je, že pouze dostatečně silný jedinci jsou schopni v prostředí, ve kterém žijí, přežít, popřípadě se dále rozmnožovat a předat tak část svého genu na další generace.

Mezi další, pro nás podstatnými fakty Darwinovy teorie evoluce, je fakt, že každá populace zůstává zhruba stejně velká, navzdory změnám, které na ni působí, což neplatí v případě, že se daná populace blíží ke svému konci a bude následovat její vyhynutí. Za zmínku jistě stojí i jeden ze závěrů výše zmíněného díla a to ten, že boj o přežití je neustálý a nikdy nekončící proces každého žijícího jedince v populaci. To má za následek neustálý vývoj živočišných druhů, větvení jednoho živočišného druhu do dvou a více nových, ale také slepý vývoj, který má za následek vyhynutí celého živočišného druhu. Dále je podstatný a velice důležitý fakt, že v celé populaci neexistují dva naprosto stejní jedinci. Díky tomu je zaručena rozmanitost v populaci a je menší pravděpodobnost stagnace vývoje. Tyto aspekty budou blíže rozebrány v dalších kapitolách.

2.2. John Henry Holland

Ačkoliv paradigmata řešení problémů založená na Darwinově teorii evoluce se začala objevovat již po necelém století po Darwinovi, trvalo dalších třicet let, než bylo možné tato paradigmata použít v praxi na reálná řešení (2). Tato prodleva byla zapříčiněna tehdejším nedostatečným výkonem výpočetní techniky, což je v současnosti vzhledem k důvodu nasazování genetických algoritmů a genetického programování úsměvné.

Za zakladatele genetických algoritmů je dnes považován americký vědec a profesor v oboru psychologie a počítačové vědy John Henry Holland, který se svým dílem *Adaptation in Natural and Artificial Systems* (4) dal principiální základy genetickým algoritmům, jak je známe dnes. V tomto díle s využitím matematiky, ekonomie a samozřejmě Darwinovy teorie evoluce popisuje, jak vytvářet adaptivní a učící se systémy. Tímto dílem Holland spustil zájem o genetické algoritmy a dal podklady, z kterých se vyvinulo genetické programování.

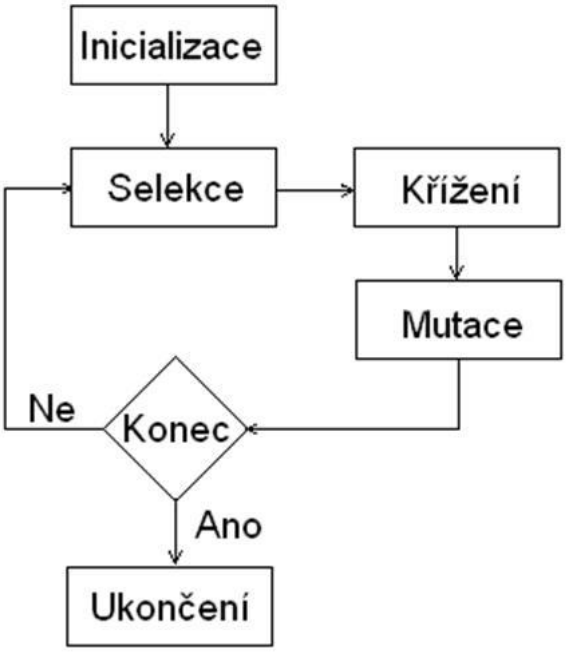
2.3. John R. Koza

Z těchto podkladů vycházel též americký doktor v oboru počítačové vědy John R. Koza a ve svém díle *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems* (1) popisuje metodologickou nadstavbu nad genetickými algoritmy pro tvoření počítačových programů.

3. Úvod do genetických algoritmů a genetického programování

V co možná nejabstraktnější podobě má genetický algoritmu i genetický program takzvanou populaci. Jedná se o skupinu individuálních entit a tu dále dělíme na generace podle pořadí, např.: první generace, druhá generace až poslední generace. Každý jedinec v takové populaci je reprezentován svým chromozomem. Ten je vhodně zakódován a představuje řešení, které ovšem nemusí být ani zdaleka správné. Chromozom každého jedince v populaci je ohodnocen tzv. ohodnocující funkcí, která právě určuje onu vzdálenost od řešení problému nebo od potřebného výsledku. Tomuto ohodnocení se říká fitness a dále také určuje jedincovu šanci na podílení se na tvorbě nové generace. K tvorbě nové generace slouží tzv. operátory tvoření nové generace. První z nich je operátor křížení, který z chromozomu dvou popř. více jedinců vytvoří chromozom popř. více nových chromozomů, které budou reprezentovat potomka popř. potomstvo. Druhým operátorem je operátor mutace. Tento operátor pracuje s určitou pravděpodobností, že se část chromozomu (gen) každého jedince náhodně změní. Tato pravděpodobnost zpravidla není příliš velká.

Samotný cyklus vývoje populace začíná samotným vygenerováním počáteční (první) generace, která může být generována náhodným způsobem. Poté následuje ohodnocení celé populace, již dříve zmíněnou zhodnocovací funkcí. V případě, že se v populaci objeví jedinec, jehož fitness je postačující nebo přímo odpovídá hledanému řešení, je cyklus zastaven. V opačném případě se pokračuje operátory tvoření nové generace. Na závěr je vytvořena nová populace a to zpravidla zvolením všech potomků z nové generace a popř. i některých jedinců předcházející generace. Takto nově vytvořená generace je opět ohodnocena a celý cyklus se opakuje.



Obrázek 1 Schématické znázornění cyklu genetických algoritmů a genetického programování (zdroj: (5))

Z výše popsaného postupu vyplývá, že takto navržený proces prohledává množinu různých a ve většině případů nesprávných řešení a při špatném návrhu může být velice časově náročný anebo nemusí dosáhnout uspokojivých výsledků vůbec. Z toho plyne, že pokud pro daný problém již existuje specializovaný postup, heuristické metody, mezi které patří i výše popsaný postup, mu budou pouze těžko konkurovat, co se týče kvality výsledků a hlavně času potřebného k výpočtům. Proto heuristické metody naleznou uplatnění hlavně tam, kde dosud neexistuje specializovaný algoritmus nebo výsledky takové algoritmu není možné očekávat v reálném čase. To ovšem neznamená, že není možné nasazení heuristických metod i do jiných oblastí. Již v minulosti (5) bylo úspěšně použito genetického programování k objevení nových a znovuobjevení stávajících vynálezů.

4. Rozdíl mezi genetickými algoritmy a genetickými programy

Zjednodušeně řečeno se dá říct, že genetické programování je spíše nadstavba genetických algoritmů, než jako metodologické postupy navzájem si konkurující. Jak již bylo řečeno v kapitolách 4.2. a 4.3. John R. Koza vycházel při tvorbě paradigmatu pro genetické programování z Hollandovy práce. Hlavním rozdílem je struktura chromozomu jedinců. V genetických algoritmech je chromozom chápán spíše jako pevně dlouhá sekvence alel, která jedince definuje, kdežto v genetickém programování je spíše chápán jako hierarchický strom funkcí a jejich parametrů (1). V obou případech chromozom reprezentuje řešení úlohy, na který byl nasazen.

5. Principy genetických algoritmů a genetického programování

V následujících podkapitolách je blíže přiblíženo a podrobněji popsáno fungování genetických algoritmů a genetických programů.

5.1. Chromozom

Jak již bylo dříve řečeno, ze všeho nejdřív potřebujeme populaci jedinců, kde každý jedinec definuje jedno potencionální řešení ať už dostatečné nebo nedostatečné. Každý jedinec a tedy i každé řešení je reprezentováno chromozomem, který může mít několik podob. Např. (1,0,0,1,1,0,0,1;0,0,0,1,1,0,1,1), což můžou být binárně zakódované dvě čísla, v tomto případě to jsou čísla 153 a 27 jakožto hodnoty x a y globálního maxima funkce dvou proměnných. Další příklad chromozomu může být (pohniPravouVezi, H5, pohniKralovnou, D2, pohniPrvnimPescem, A5, pohniPravymKonem, D7). V tomto případě se jedná o chromozom skládající se z funkcí (pohniPravouVezi, pohniKralovnou,…) a parametrů pro danou funkci (H5, D2,…). Genetický algoritmus by tedy vždy vzal dvě alely (funkci a parametr) a volal by da patřičnou funkci s patřičným parametrem. Způsobů, jakým je možné řešení problému zakódovat, je takřka neomezeně mnoho. Na druhou stranu se musíme držet jistých pravidel, pokud chceme, aby náš program měl šanci na úspěch.

Hlavním omezením by měl být problém, který se snažíme vyřešit jako takový. Nebylo by moudré například zakódovat ideální velikost cestovního zavazadla v podobě chromozomu („velmi velký“) (4). Prvním nedostatkem, který je zde patrný na první pohled, je nepřesná vypovídající hodnota takového chromozomu. Každý z nás by si mohl pod takovou hodnotou představit zavazadlo o jiných rozměrech než někdo jiný. Další problém, který se zde vyskytuje, je chromozom o pouhé délce jedna. Takový problém nemusí nutně znamenat okamžitou zkázu našeho programu. Pokud je například takovýto potomek členem jedné z prvních generací a víme, že díky operátorům tvorby nové generace se velikost chromozomů následujících potomků bude zvětšovat, můžeme použít i chromozom o délce jedna, pokud k tomu máme opodstatněný důvod.

Jedním z nejstarších a dnes stále oblíbeným kódováním je binární kódování. V binárním kódování má každá část chromozomu (gen) přípustné pouze dva stavy a to 1 nebo 0. Např. chromozom (1,0,1,0,0,0,1,1,1,0,1,0,0,0,1,1,0,0,1,0,1,1,0,1,0,0,0,1,0,1,0,1) může reprezentovat celočíselnou hodnotu v třiceti dvou bitové hloubce, nebo může být dekódován do jiného významu. Dalším důvodem oblíbenosti binárního kódování je fakt, že například v chromozomech člověka je zapotřebí kombinace několika genů k přesnému určení jisté vlastnosti člověka, což má za následek že malá změna jednoho genu z hodnoty 1 na 0 nebo obráceně, bude mít i malý následek na patřičnou vlastnost. Tato vlastnost nabízí jemnou manipulaci s chromozomem a je důležitá pro operátor mutace, který bude rozebrán v jedné z následujících kapitol.

Bohužel v některých případech se může stát, že ona malá změna může mít velice velký významový dopad na dekódovanou hodnotu. Vezmeme-li v potaz osmi bitovou celočíselnou reprezentaci se znaménkem, například (1,0,1,0,0,1,1,0), kde první bit značí znaménko (1 -> - a 0 -> +) a zbytek hodnotu čísla. V tomto případě by se jednalo o číslo -38, kdybychom provedli z hlediska chromozomu nepatrnou změnu a to prvního bitu, dostali bychom chromozom (0,0,1,0,0,1,1,0), což je po dekódování hodnota 38. Zde vidíme, že malá změna v jednom genu způsobila, že se význam hodnoty po celočíselné ose posunul o 76 jednotek (-38 + 76 = 38). Tento negativní efekt, lze odstranit například použitím Grayova kódování (7). Grayovo kódování je pojmenované po svém tvůrci americkém fyzikovi Franku Grayovi, který si ho roku 1953 dal patentovat. Princip Grayova kódování je, že dvě hodnoty, které spolu na číselné ose sousedí, se v binárním kódu liší pouze v jednom genu. Bohužel tento způsob kódování není určen pro záporné hodnoty. Tento nedostatek můžeme odstranit tím, že se při dekódování určí pravidlo, které bude říkat, že např. hodnota 0 (minimální hodnota) po převedení binárního chromozomu (0,0,0,0,0,0,0,0) nebude znamenat hodnotu 0 ale hodnotu -128, analogicky poté hodnota 255 (maximální hodnota) bude reprezentovat ve skutečnosti hodnotu 127. V programu bychom tohoto chování dosáhli jednoduchým odečtením hodnoty 128 po každém dekódování.

Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

Tabulka 1 Ukázka kódování čísel pomocí binárního a Grayova kódování (zdroj: (8))

5.2. První generace

První generace je velice důležitá, protože z ní budou vycházet následující generace a pokud již na samotném začátku nebudeme mít dostatečnou genovou rozmanitost, správného výsledku se spíše nedočkáme a náš evoluční vývoj začne rychle stagnovat. Tohoto nechtěného efektu, bychom se mohli zbavit použitím mutačních operátů, které jsou ovšem podle Hollanda (4) relativně nedůležité, a daný problém řeší jinak. Nejjednodušším způsobem, jakým zle vygenerovat novou generaci je náhodným způsobem. Každá hodnota z rozsahu přípustných hodnot pro daný gen, má stejnou pravděpodobnost, že bude do libovolného chromozomu na libovolné místo dosazena. Například máme-li chromozomy v binární podobě (x,x,x,x,x,x,x,x), potom při náhodném generování takového chromozomu mají čísla 0 i 1 šanci 50%, že bude umístěna místo x. Podobně pokud je náš chromozom tvořen posloupností funkcí, tak i zde má každá funkce stejnou šanci, že se stane hodnotou genu, stejně jako její přípustné parametry. Stejně jako v podobě chromozomu a způsobu jeho kódování se ani v generování první populace nekladou meze, ale opět by zde mělo být použito pravidlo využití takové techniky generování nové populace, která je smysluplná a opodstatněná.

5.3. Fitness a ohodnocující funkce

Stejně jako v přírodě, tak i v evolučním cyklu potřebujeme vědět, jak je každý jedinec silný, jak je schopný přežít a jaké jsou jeho šance na zplození potomstva. K tomu nám zde slouží ohodnocení každého jedince, kterému se říká fitness. Fitness je také zpravidla jedena z podmínek pro ukončení evolučního cyklu a to ve chvíli, kdy jedinec splňuje minimální kritéria, která v populaci vyhledáváme. Jako další ukončovací podmínka by bylo možné hodnotu fitness nebo lépe řečeno průměrnou hodnotu fitness každé generace použít k ověřování, že se naše populace z dlouhodobého hlediska vyvíjí a že nestagnuje nebo dokonce nedegeneruje. V takovém případě je pravděpodobně náš program špatně navržen a pokračování takového programu nemá smysl.

Hodnota fitness může mít několik podob, například to může být celočíselná proměnná, která nejsilnějšímu jedinci přiřadí nejvyšší hodnotu v populaci. Dále to může být hodnota převrácená s desetinou čárkou od 0 do 1, kde hodnota 1 určí jedince, který je nejlepší pro řešení daného problému a všechny nižší hodnoty nejsou tak optimální (1). Ani zde genetické paradigma neklade meze, ale opět nesmíme zapomenout na smysluplnost a opodstatněnost našeho rozhodnutí. Například hodnota fitness v podobě jedné hodnoty typu boolean by byla velice nedostačující kvůli svému omezenému rozsahu.

Ohodnocující funkce jako taková, může mít několik podob v závislosti na konkrétní problematice. Jedna z nejjednodušších podob ohodnocující funkce, kterou bych zde rád uvedl je převzata z knihy (2) Profesora Univerzity Hradec Králové, Josefa Hynka. Tato ohodnocující funkce sčítá počet výskytů čísla jedna v chromozomech jedinců.

Tato ohodnocující funkce je v tuto chvíli čistě demonstrační, na druhou stranu by taková ohodnocující funkce našla uplatnění i v reálné aplikaci. Například kdyby daná čísla v chromozomech jedinců představovala pravdivostní hodnoty vrácené funkcemi složené např. z logických operátorů na základě jiných, v chromozomu nezjistitelných vlastností jedinců. V případě, že tedy aplikujeme naši ohodnocující funkci na čtyřčlennou populaci znázorněnou v tabulce 2, získáme ohodnocení znázorněné v tabulce 3.

číslo jedince	chromozom jedince
1	(1,0,1,0,1,1,0,0)
2	(0,1,1,1,1,0,1,1)
3	(0,0,0,1,0,0,0,1)
4	(1,1,0,0,1,1,0,0)

Tabulka 2 Chromozomy ukázkové populace (zdroj: vlastní tvorba)

číslo jedince	chromozom jedince	ohodnocení
1	(1,0,1,0,1,1,0,0)	4
2	(0,1,1,1,1,0,1,1)	6
3	(0,0,0,1,0,0,0,1)	2
4	(1,1,0,0,1,1,0,0)	4

Tabulka 3 Chromozomy a fitness ukázkové populace (zdroj: vlastní tvorba)

Z tabulky 3 již nyní vyplývá, že nejzdatnějším jedincem v naší populaci je jedinec číslo 2 a nejméně zdatný je jedinec číslo 3, zbylí dva jedinci jsou průměrní.

5.4. Operátor křížení a výběr jedinců

Předtím než začneme jedince v naší populaci křížit, musíme nejdříve vybrat, kteří jedinci se budou křížit. Na tomto výběru velice záleží, protože pokud by docházelo ke špatným výběrům, mohlo by se stát, že bychom začali ztrácet zdatné jedince s bohatými chromozomy. Jedním ze špatných a nejjednodušších způsobů, jakým vybírat jedince ke křížení, je náhodný výběr. Kdybychom zvolili naprosto náhodný výběr, kde každý jedinec má stejnou šanci na to být vybrán ke křížení, porušili bychom jeden z podstatných závěrů Darwinovy práce (3) a to ten, že větší šanci na křížení mají zdatnější jedinci. To zaručí, že do další generace se přenesou a zkombinují silné geny a další populace by měla být v průměru silnější než předešlá. Na druhou stranu nemůžeme různými způsoby křížit pouze dva nejsilnější jedince z celé populace. Tím bychom velice rychle ztratili genovou rozmanitost a náš program by rychle přešel ke stagnaci a nejspíše by k výsledku vůbec nedošel. Potřebujeme tedy do dalších generací dostat i ne příliš silné jedince, kteří z hlediska hledání řešení nepředstavují ideálního jedince, ale z pohledu křížení a zachování bohatého genofondu ve smyslu různorodosti jsou velice důležití.

Jedním z možných způsobů tzv. selekce, jakým lze vybírat dvojce jedinců ke křížení, je přiřazení každému jedinci procentuální šanci na základě jeho fitness. V tomto případě má celá populace 100%, které se rozdělují mezi jedince. Například mějme tabulku 4, ve které jsou čtyři jedinci s jejich hodnotami fitness.

Číslo jedince	fitness jedince
1.	0,2
2.	0,4
3.	0,8
4.	0,6

Tabulka 4 Fitness ukázkové populace (zdroj: vlastní tvorba)

V tomto případě předpokládejme, že čím je fitness menší, tím je jedinec zdatnější a rozsah hodnot pro fitness je od 0 do 1. Když sečteme všechna fitness jedinců, získáme číslo 2, což je celých 100%, které musíme rozdělit mezi všechny jedince. Když použijeme rovnici:

$$S_i = \frac{100}{2} * (1 - F_i),$$

Rovnice 1 Rovnice pro výpočet procentuální šance výběru

kde $i \in \{1,2,3,4\}$, S_i je procentuální šance výběru jedince i ke křížení, F_i je fitness jedince i, poté můžeme vytvořit tabulku 5 s pravděpodobností výběru.

Číslo jedince	fitness jedince	pravděpodobnostní šance
1.	0,2	40%
2.	0,4	30%
3.	0,8	10%
4.	0,6	20%

Tabulka 5 Fitness a procentuální šance ukázkové populace (zdroj: vlastní tvorba)

Tímto způsobem výběru jedinců ke křížení mají i méně zdatní jedinci šanci na předání svých genů další populaci. Zároveň jsou upřednostňováni silnější jedinci, aby se zaručil populační vývoj a nedocházelo ke stagnaci. Způsobů, jakým lze výběr jedinců provádět, je opět jako ve většině částí genetických algoritmů a genetického programování mnoho a stejně i zde platí, že pro konkrétní postup by měl být logický důvod.

Mezi konkrétní selekční mechanizmy patří například ruletová selekce (2), nebo třeba turnamentová selekce (8)

Způsobů, jakým lze chromozomy vybraných jedinců kombinovat, je mnoho. Jedním ze způsobů je tzv. jednobodové křížení (2). To spočívá v rozdělení chromozomu obou rodičů v jednom náhodném místě, čímž získáme čtyři chromozomy. Ty poté spojíme vždy první část od prvního rodiče s druhou částí druhého rodiče a druhou část prvního rodiče s první částí druhého rodiče. Tyto nové dva chromozomy budou představovat dva potomky. Tento postup je znázorněn v tabulce 6.

Číslo rodiče	Chromozomy	rozdělení na čtyři	chromozomy dvou nových potomků
1.	(1,1,1,1,0,0,0,0)	(1,1,1,1) a (0,0,0,0)	(1,1,1,1,1,1,1,1)
2.	(0,0,0,0,1,1,1,1)	(0,0,0,0) a (1,1,1,1)	(0,0,0,0,0,0,0,0)

Tabulka 6 Znázornění operátoru křížení (zdroj: vlastní tvorba)

V tomto případě dva předci zplodili dva nové potomky. Za předpokladu že v populaci o N jednotlivcích dojde k $\frac{N}{2}$ křížení výše popsaným způsobem, poté máme zajištěnou konstantní velikost populace a konstantní délku chromozomu jedinců po celou dobu evoluce. Oba výše popsané důsledky lze úpravami takového křížení změnit a i zde platí pravidlo, že by každá taková změna měla mít důvod. V případě že se rozhodneme pro nekonstantní velikost populace, musíme si dávat pozor, aby naše populace nevymřela, nebo donekonečna rostla. Musíme si uvědomit, že pokud naše populace bude lineárně nebo dokonce exponenciálně růst, velice rychle může dojít k zahlcení paměti počítače nebo bude celá evoluce trvat déle, než je přípustné. Podobné nástrahy nás čekají v případě, kdy se rozhodneme pro nekonstantní délku chromozomů.

5.5. Operátor mutace

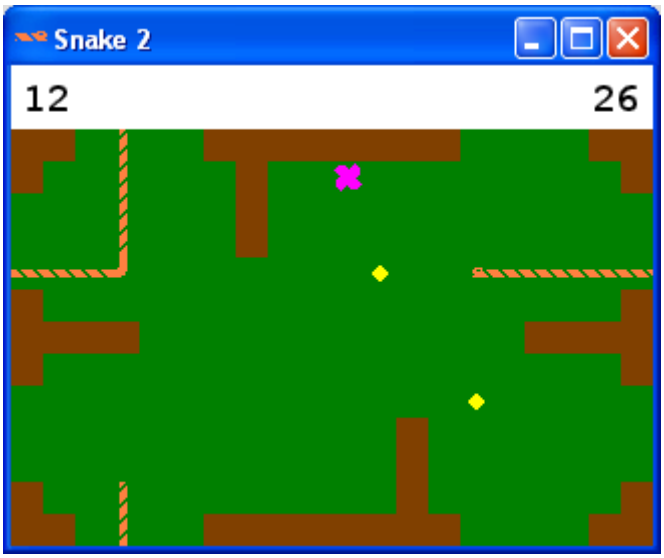
Na rozdíl od operátoru křížení v operátoru mutace již nedochází k výběru jedinců, na které se operátor aplikuje, ale všichni jedinci, bez ohledu na fitness mají stejnou šanci. Jak již bylo řečeno, operátor mutace pracuje s jistou pravděpodobností, že se každý gen (část chromozomu) každého jedince náhodně změní. Tento operátor zajišťuje rozmanitost genů v populaci v závislosti na své pravděpodobnosti. Tato pravděpodobnost je zpravidla malá, v opačném případě by totiž docházelo k velkým změnám v chromozomech nové generace a spíše by šlo o náhodné změny v chromozomech, než o cílené křížení zdatných jedinců.

6. Úvod do řešené problematiky

Jako řešená problematika pomocí genetického algoritmu byla zvolena problematika hledání nejlepší nebo alespoň optimální cesty v bludišti.

6.1. Definování problému

Bludiště je definované jako dvourozměrná diskrétní oblast obsahující volná pole umožňující pohyb a překážky, které v pohybu brání. Bludiště má dále okraje, tedy není možné bludiště opustit ani nemá propojené konce ve smyslu nekonečného obíhání mapy dokola. Na obrázku 2 vidíme příklad bludiště, které nemá okraje a je tedy možný nekonečný pohyb v jednom směru.

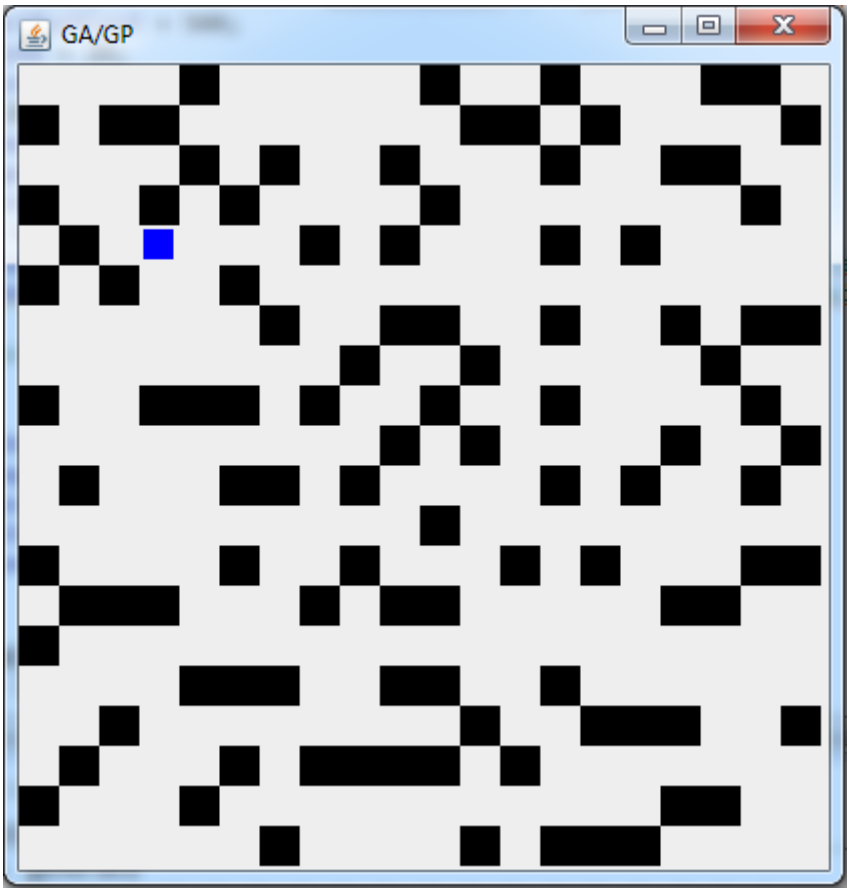


Obrázek 2 Ukázka z počítačové hry had (zdroj: (10))

Dále definujeme jedince jakožto entitu, která se v bludišti bude pohybovat pouze po volných polích. Na rozdíl od hada se jedinec bude nacházet právě v jednom poli bludiště. Úkolem jedince je dostat se v rámci bludišti z bodu A do bodu B. Tyto body jsou díky diskrétnímu pojetí bludiště specifikovány celočíselnými hodnotami X a Y kdy bod A je počáteční umístění jedince v bludišti a bod B je cíl.

6.2. Popis prvního prototypu

Problém je řešen za použití genetického algoritmu, který je napsán v jazyce Java verze 8 update 60, popř. v jeho dalších aktualizacích kde dni 27. 4. 2016. Program se v této fázi skládá z hlavního vlákna a časovače, které slouží jako druhé vlákno. Časovač slouží k nezávislému vykreslování postupu nejlepšího jedince bludištěm a hlavní vlákno slouží k nastavení potřebných parametrů, vytvoření první generace a provádění samotné evoluce. Vykreslování probíhá do jednoduchého GUI, které je znázorněno na obrázku 3.



Obrázek 3 Ukázka mapy (zdroj: vlastní tvorba)

Strukturálně je aplikace rozdělena do následujících tříd: Pozice, Populace, Jedinec, MujJedinec, GUI, Bludiste a APP.

Třída Pozice slouží k uchovávání a manipulaci s hodnotami x a y jak pro aktuální pozici jedince, tak pro jeho směr.

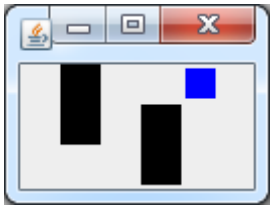
Třída Populace obsahuje funkce potřebné k samotné evoluci, mezi tyto funkce patří.: mutace s určitou pravděpodobností, náhodné jednobodové křížení, ruletová selekce přímo úměrná fitness jedinci (2), náhodné vygenerování první generace a zastřešuje tak celou populaci jedinců.

Třída Jedinec je abstraktní třída, která slouží jako předloha pro konkrétní specifikace tříd jedinců. Je také používána třídou Populace, díky čemuž je třída Populace vysoce znovupoužitelná. Dále obsahuje nezbytné vlastnosti jako ID, fitness a chromozom v podobě spojového seznamu objektů, díky čemuž získáváme nezávislost na konkrétním kódování chromozomu a opět získáváme znovu použitelnost.

Třída MujJedinec, která dědí od třídy Jedinec, obsahuje pouze atributy a funkce nezbytné pro konkrétní úlohu, mezi něž patří: pozice, směr, ohodnocující funkci a funkce umožňující pohyb (vpřed, otočení vpravo a otočení vlevo).

Třída GUI pak obstarává záležitosti ohledně časovače (instance třídy Timer) a samotného uživatelského rozhraní, které slouží jako kreslící plátno. Po prvotní inicializaci GUI je zahájeno cyklické vykreslování postupu nejlepšího jedince. První takovýto jedinec je získán z první náhodně vygenerované generace a poté co jeho cesta v bludišti vykreslena se časovač dotazuje k získání nejlepšího jedince z aktuální generace a cyklus se opakuje dokud probíhá evoluce.

Třída Bludiste má za úkol generovat a uchovávat si informace o různých variantách bludišť v dvourozměrném poli, mezi které patří náhodně vygenerovaná bludiště s 50% a 25% šancí na vytvoření překážky na každé pozici v poli, tak také přesně daná bludiště jako je bludiště bez překážek nebo jednoduchá bludiště. Příklad jednoduchého bludiště je na obrázku 4.



Obrázek 4 Ukázka jednoduchého bludiště (zdroj: vlastní tvorba)

Informace o bludištích uchovává v podobě dvourozměrného pole typu boolean, kdy hodnota 1 značí překážku a hodnota 0 volnou cestu. V neposlední řadě je účel funkce Bludiště umožňovat, popř. zabraňovat jedincům v pohybu v bludišti.

Poslední třída APP pak vytváří a nastavuje instance tříd Bludiste, Populace a GUI, vytvoření startu a cíle a zahajuje celou evoluci v hlavním vlákně aplikace.

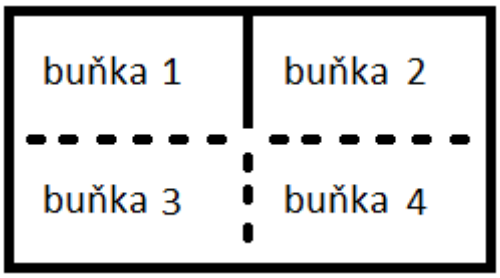
V této podobě je program spíše základní kámen, který bude dále obohacován a upravován do finální podoby tak, jak bude popsáno v následujících kapitolách.

7. Představení práce doktora Nitin S. Choubey

V následujících podkapitolách bude představen článek pana profesora Nitin S. Choubeyho z indické univerzity v Mumbai, který byl publikován ve vědeckém časopise International Journal of Computer Applications (6), který slouží jako předloha a odrazový můstek.

7.1. A-Maze

Choubey ve svém článku využívá tzv. A-maze techniku pro tvorbu bludiště, která zaručuje, že mezi startem a cílem existuje alespoň jedna cesta. Výstup této techniky je možné si představit jako dvourozměrné diskrétní pole obsahující buňky. Každá buňka je charakterizována pěti vlastnostmi, které nabývají jedné ze dvou hodnot a to 0, 1 nebo 2. První čtyřmi vlastnostmi jsou nahoře, dole, vpravo a vlevo a hodnota v nich značí, zda se daným směrem nachází dveře (hodnota 1), popř. zeď (hodnota 0). Hodnotu 2 využívá pouze pátá vlastnost buňky (viz. dále). Zeď neumožňuje pohyb ze současné buňky do buňky sousední a obráceně. Mějme příklad čtyř buněk tak, jak je znázorněno na obrázku 5.

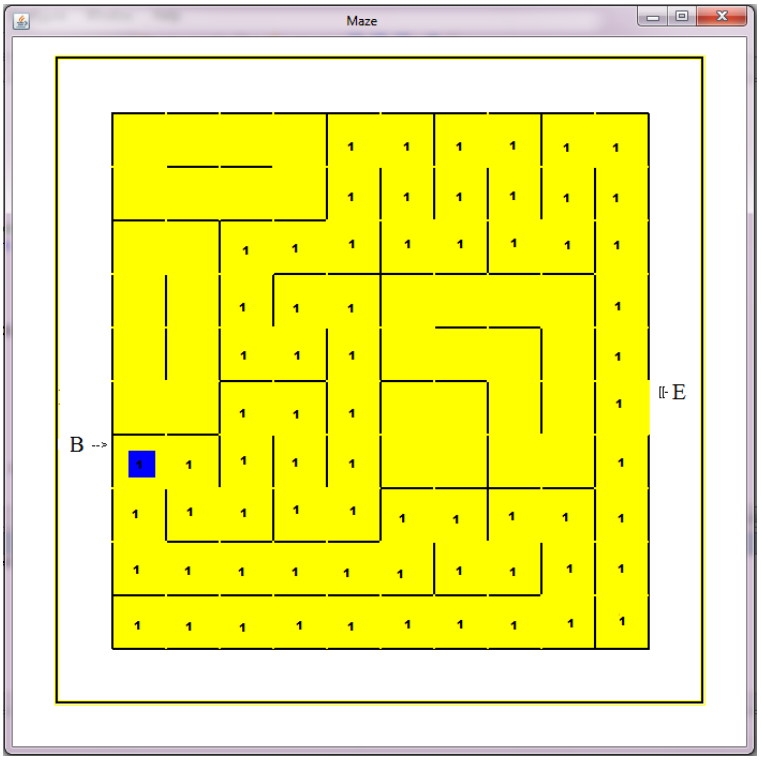


Obrázek 5 Ukázka A-Maze techniky (zdroj: vlastní tvorba)

Z obrázku 5 je patrné, že není možné přímý pohyb doprava z buňky 1 do buňky 2. Tento fakt může být zaznamenán způsobem bludiste[0][0].vpravo = 0, kdy bludiště je dvourozměrné pole buněk odpovídající bludišti z obrázku 5. Analogicky poté můžeme z obrázku vydedukovat, že jediný možný pohyb z buňky 1 je do buňky 3, který zle zaznamenat bludiste[0][0].dole = 1.

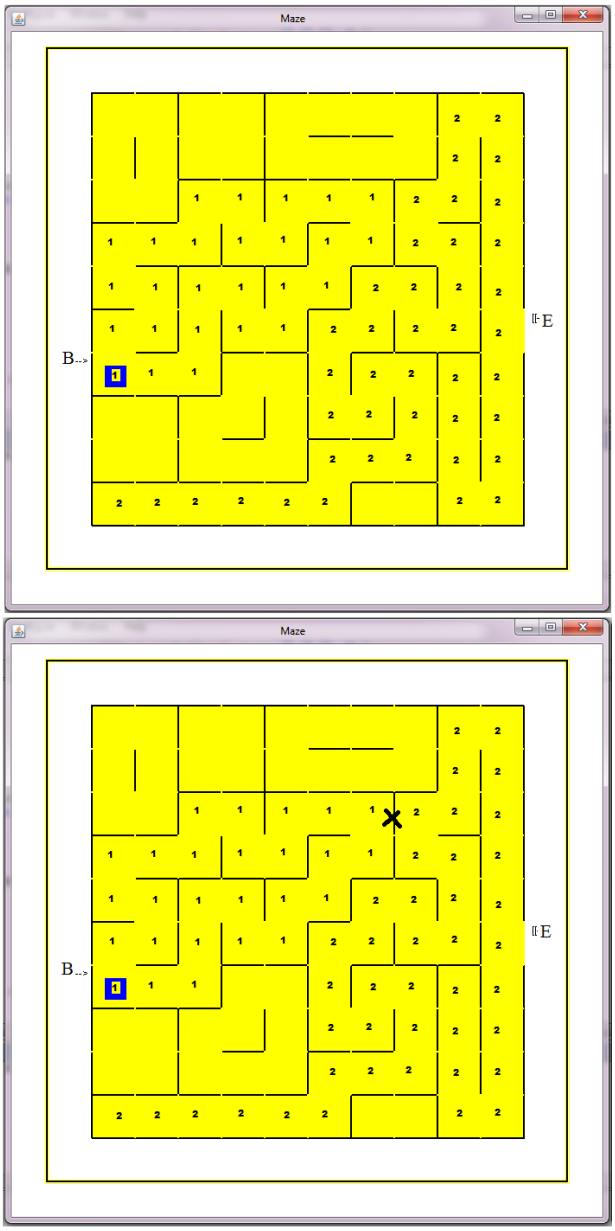
Tyto čtyři vlastnosti jsou vygenerovány pro všechny buňky z bludiště náhodně s pravděpodobností 50% jak pro hodnotu 0, tak pro hodnotu 1. Výjimku tvoří pouze hraniční buňky, které mají minimálně jednu zeď (okrajové buňky), popř. dvě zdi (rohové buňky) a to takovým způsobem, aby nebylo možné bludiště opustit. Tento způsob tvoření bludiště ovšem ne vždy obsahuje cestu od startu k cíli.

K určení, zda existuje v takto vygenerovaném bludišti řešení, se využívá tzv. záplavový algoritmus a pátá vlastnost každé buňky, kterou je tzv. štítek. Nejdříve se náhodným způsobem určí buňky reprezentující start a cíl v bludišti. Buňce reprezentující start se vlastnost štítek nastaví na hodnotu 1, buňce reprezentující cíl na hodnotu 2 a všem zbylým buňkám se nepřiřazuje žádná hodnota. Poté je spuštěn záplavový algoritmus, který začíná ve startu a mění vlastnost štítek na hodnotu 1 v každé buňce, do které se dostane. Pro záplavový algoritmus platí stejná pravidla, co se týče pohybu v bludišti jako u jedince hledajícího řešení a to ta, že se může pohybovat pouze mezi sousedními buňkami, mezi kterými není zeď. Záplavový algoritmus může narazit na tři různé případy u buněk, které zaplavuje. První z nich je, že vlastnost štítek v dané buňce nemá žádnou přiřazenou hodnotu. Té záplavový algoritmus přiřadí hodnotu 1 a snaží se pokračovat v zaplavování sousedních buněk. Druhý případ, který může nastat, je že vlastnost štítek má přiřazenou hodnotu 1. V tomto případě algoritmus ví, že tuto buňku již zaplavil a dále nepokračuje. Posledním případem, je hodnota 2, která značí, že tato buňka je cílová a také to, že v daném bludišti existuje alespoň jedno řešení a záplavový algoritmus může úspěšně skončit.



Obrázek 6 Ukázka nalezení cíle pomocí záplavového algoritmu (B značí začátek a E cíl) (zdroj: (6))

Může ovšem nastat situace, kdy záplavový algoritmu již nemá další místo, které by mohl dále zaplavovat a k cílové buňce se nedostal. To znamená, že v daném bludišti neexistuje řešení a je potřeba využití tzv. techniky bourání zdí. V této technice se nejprve provede stejný záplavový algoritmu, který je popsán výše, pouze s tím rozdílem, že začíná v buňce reprezentující cíl a přiřazuje hodnotu 2 do vlastnosti štítek u každé buňky, které zaplavuje. Následně je hledána taková dvojice buněk, které spolu sousedí a jedna buňka z této dvojice má ve vlastnosti štítek hodnotu 1 a druhá 2. Díky fungování záplavového algoritmu se dá vyvodit, že mezi těmito buňkami se musí nacházet zeď, která je odstraněna a tím je dosaženo stavu, ve kterém má dané bludiště alespoň jedno řešení. Tento případ je znázorněn na obrázku 7.



Obrázek 7 Ukázka bourání zdi mezi ideálními sousedy (zdroj: (6))

Opačnou situací k situaci v předchozím odstavci je případ, kdy se nepodařilo nalézt dvě přímé sousedící buňky, kde by vlastnost štítek jedné z nich měla hodnotu 1 a u druhé buňky hodnotu 2. V tomto případě se náhodně zvolí jedna buňka, jejíž vlastnost štítek má hodnotu 1 a zároveň sousedí s takovou buňkou, jejíž hodnota ve vlastnosti štítek nemá žádnou hodnotu. Poté se mezi takovýmito buňkami zeď zruší a spustí se záplavový algoritmus měnící hodnotu vlastnosti štítek na 1 a začíná v oné sousední buňce, jež nemá přiřazenou hodnotu ve vlastnosti štítek. Poté se opět hledá dvojce buněk, ve které má první buňka ve vlastnosti štítek hodnotu 1 a druhá buňka hodnotu 2. V případě nalezení takové dvojice se zeď mezi těmito buňkami odstraní. V opačném případě se cyklus opakuje, dokud není taková dvojce nalezena.

Pomocí tohoto postupu je možné vytvořit náhodné bludiště o rozměrech N x N, popř. M x N, které bude mít alespoň jedno řešení.

7.2. Genetický algoritmus

Choubey ve své práci využívá číselného kódování, konkrétně hodnot od 0 do 3, které mají přiděleny následující významy: 0 – krok doleva, 1 – krok nahoru, 2 – krok doprava a 3 krok dolů.

Ohodnocující funkce má podobu:

$$f = \frac{(sb - brs)}{(brc - sb)} * 100,$$

Rovnice 2 Ohodnocující funkce (zdroj: (9))

kde f je hodnota fitness, sb je současná buňka, brs je buňka reprezentující start a brc buňka reprezentující cíl.

Dále je využit operátor křížení, který ze dvou zvolených rodičů vytvoří dva nové potomky dvěma různými způsoby. První z nich je sčítací metoda a druhým odčítací metoda. Obě metody pracují s chromozomy stejné délky a sčítají, popř. odčítají hodnotu genu z chromozomu rodičů na stejných pozicích. Tyto hodnoty jsou poté celočíselně vyděleny číslem 4 a zbytek po tomto dělení je novou hodnotou do chromozomu potomka na stejnou pozici, jako u rodičů. Díky tomuto procesu zůstávají noví potomci konzistentní vůči kódování chromozomu. Příklad celého postupu je zobrazen v tabulce 7.

	chromozom								
rodič 1	0	1	1	3	2	1	0	0	
rodič 2	3	3	2	2	1	0	0	2	
potomek 1	3	0	3	1	3	1	0	2	sčítací metoda
potomek 2	3	2	1	1	1	1	0	2	odčítací metoda

Tabulka 7 Ukázka operátoru křížení (zdroj: vlastní tvorba)

Poslední z operátorů, který má vliv na novou generaci jedinců, je operátor mutace s určitou pravděpodobností změny každého genu v chromozomu.

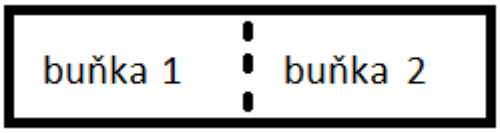
Parametry experimentu jsou nastaveny následovně: velikost populace na velikost 100 jedinců, maximální počet generací 500, pravděpodobnost operátoru křížení 80%, pravděpodobnost operátoru mutace 10%, rozměry bludiště 20 x 20 a délku chromozomu jedinců na dvojnásobek délky bludiště, tedy hodnotu 40. Choubey také ve své práci uvádí, že pro optimálnější průběh evoluce je délka chromozomu postupně zvětšována na maximální hodnotu délky bludiště na druhou.

8. Vlastní implementace práce profesora Nitin S. Choubey

V následujících podkapitolách bude práce profesora Nitin S. Choubey analyzována, upravena, implementována, otestována a tento celý cyklus se bude několikrát opakovat.

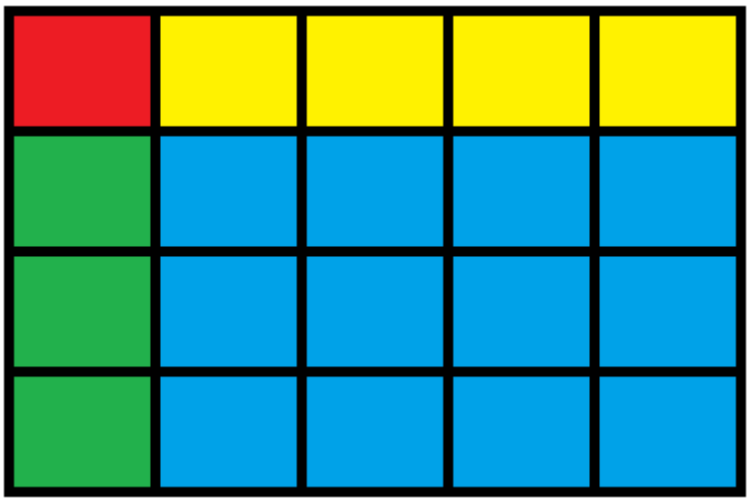
8.1. Tvorba a záznam bludiště

A-maze, jakožto způsob tvorby bludiště, je velice elegantní způsob, jak vždy vytvořit bludiště, které má alespoň jedno řešení. Nicméně má jeden nedostatek v podobě duplicity dat při zaznamenávání informací o podobě bludiště. Jak již bylo řečeno, každá buňka nese informaci, zda se do všech čtyř směrů k jeho sousedním buňkám nachází, popř. nenachází zeď, plus informaci, zda může daná buňka být součástí řešení. Na základě toho víme, že buňka 1 z obrázku 8 obsahuje informaci o tom, že směrem vpravo se nacházejí dveře (buňka1.vpravo = 1). Tuto stejnou vypovídající hodnotu ovšem nese i buňka 2, pouze s tím rozdílem, že z jejího úhlu pohledu se jedná o pozici vlevo (buňka2.vlevo = 1).



Obrázek 8 Ukázka duplicit dat v sousedních buňkách (zdroj: vlastní tvorba)

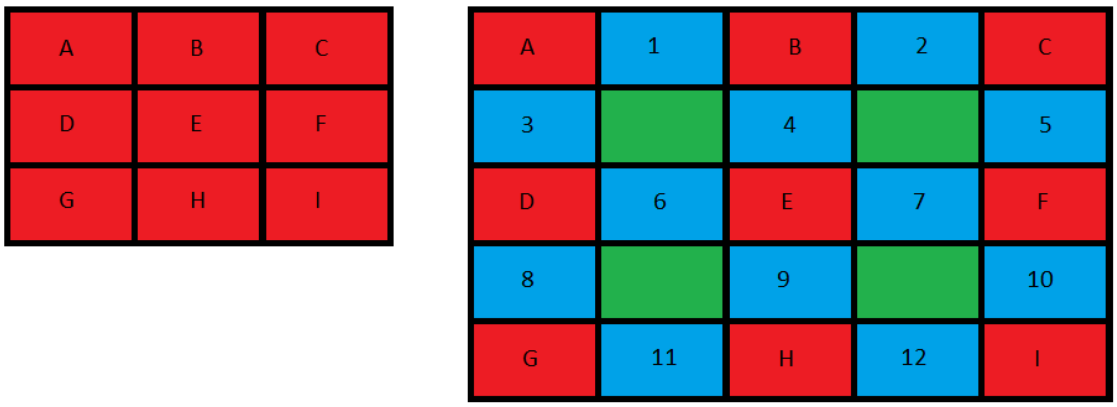
Jedním ze způsobu jak se duplicitních informací zbavit, je vytvoření více druhů buněk, které podle umístění v bludišti budou obsahovat pouze určitá data. Tento způsob je znázorněn na obrázku 8.



Tabulka 8 Ukázka více druhů buněk v bludišti (zdroj: vlastní tvorba)

Na obrázku 8 jsou celkem 4 druhy různých druhů buněk. Červená buňka uchovává informace o překážkách do všech směrů, stejně jako původní podoba buňky. Žluté buňky uchovávají informace pouze o potencionálních překážkách do směrů dolů, vpravo a nahoru. Zelené buňky informace ve směrech vlevo, dolů a vpravo a modré pouze dolů a vpravo. Díky tomuto způsobu záznamu je zaznamenáváno o $n-1+m-1+[(m-1)*(n-1)]*2$ méně informací, kde n je šířka bludiště a m je výška bludiště (bližší porovnání dále). Tento způsob záznamu bludiště ovšem bude vyžadovat sofistikovanější způsob vyhodnocování, zda je možné pohyb z jedné buňky do druhé, což může mít za následek celkové zpomalení celého evolučního cyklu. Zvážíme-li, že takovéto vyhodnocování by bylo prováděno $n*m$ při každém evolučním cyklu, kde n je délka chromozomu jedince a m počet jedinců v generaci. Proto tedy rozhodnutí výběru jednoho z těchto dvou způsobu záznamu záleží na několika parametrech, mezi které patří například: výpočetní výkon, velikost operační paměti, velikost bludiště, délka chromozomu jedinců, celková velikost populace, atd. Obecně by se dalo říci, že při menším výpočetním výkonu, popř. na bludištích o menších rozměrech, by bylo vhodnější použít původní strukturu buňky, za cenu vyšších nároků na paměť a při menší operační paměti, popř. u menších populací s relativně krátkými chromozomy by bylo vhodnější použití metody popsané výše.

Další možnosti se nabízí zkombinováním obou předchozích způsobu záznamu do jednoho. V tomto případě by se jednalo o rozšíření bludiště ve smyslu změny původní jedné buňky nesoucí pět informací do tří, dvou nebo jedné buňky, kde každá nese informaci pouze jednu. Příklad takové transformace je ukázán v tabulce 9.



Tabulka 9 Příklad transformace (zdroj: vlastní tvorba)

V této transformaci byla buňka A z levého bludiště transformována na buňku A, 1 a 3 z pravého bludiště a to následujícím způsobem. Informace nahoru a vlevo z původní buňky A byly vynechány a mechanizmus, jakým je zaručeno, že není možné opustit bludiště byl změněn (viz. níže). Informace dolů buňky A a nahoru buňky D, tedy ona duplicitní informace, kterou jsme tímto způsobem odstranili, je teď uchovávána pouze v buňce 3 stejným způsobem, jako tomu bylo u původní pěti-složkové buňky, tedy hodnota 1 pro dveře a hodnota 0 pro zeď. Obdobný způsobem toto platí pro všechny modré buňky z bludiště vpravo, tedy že původně duplicitní informaci uložena u každé mezi sebou sousedních buněk je nyní uložena pouze na jednom místě. Díky tomuto zůstala v červených buňkách uchovávána pouze jedna informace a tou je informace štítek, která funguje stejným způsobem, jako tomu bylo u původních buněk, tedy, že značí, zda může být buňka potencionálního řešení nebo nikoliv. Posledním druhem buněk jsou buňky zelené, které nenesou žádnou informaci a jedná se tedy o buňky bez jakékoliv vypovídající hodnoty nutné pro postačující záznam bludiště.

Jak již bylo naznačeno, hraniční buňky nenesou žádnou informaci o tom, že ve směru mimo bludiště již není možný pohyb. Aby se ovšem zmiňovaná situace nemohla nastat, je potřeba vytvořit jiné opatření. Toto opatření je spíše na úrovní programátorské, nežli na úrovni logické. Příklad z obrázku 17 by se dal zaznamenat jako dvourozměrné pole buněk, např. bludiste[n*2-1][m*2-1], kde n je šířka bludiště a m jeho výška. V případě, že bychom se pokusili například z buňky D o pohyb doleva, tedy dotazovali bychom se, zda se vlevo nacházejí dveře nebo zeď, vypadal by tento dotaz například takto: je bludiststе[-1][2] pravda/nepravda? Zde je patrné, že pokud bychom se snažili podobný dotaz naprogramovat, obdrželi bychom chybovou hlášku ve smyslu, že zadaný index je mimo rozsah. Stejnou chybovou hlášku bychom dostali i v případě pokusu o překročení hranic bludiště do všech zbylých tří směrů. Proto je tedy nutné tuto část kódu patřičně ošetřit a tím i zabezpečit, že není možné bludiště opustit.

V tabulce 10 vidíme porovnání všech tří způsobu záznamu bludiště, co se týká množství informací, které o bludišti uchovávají.

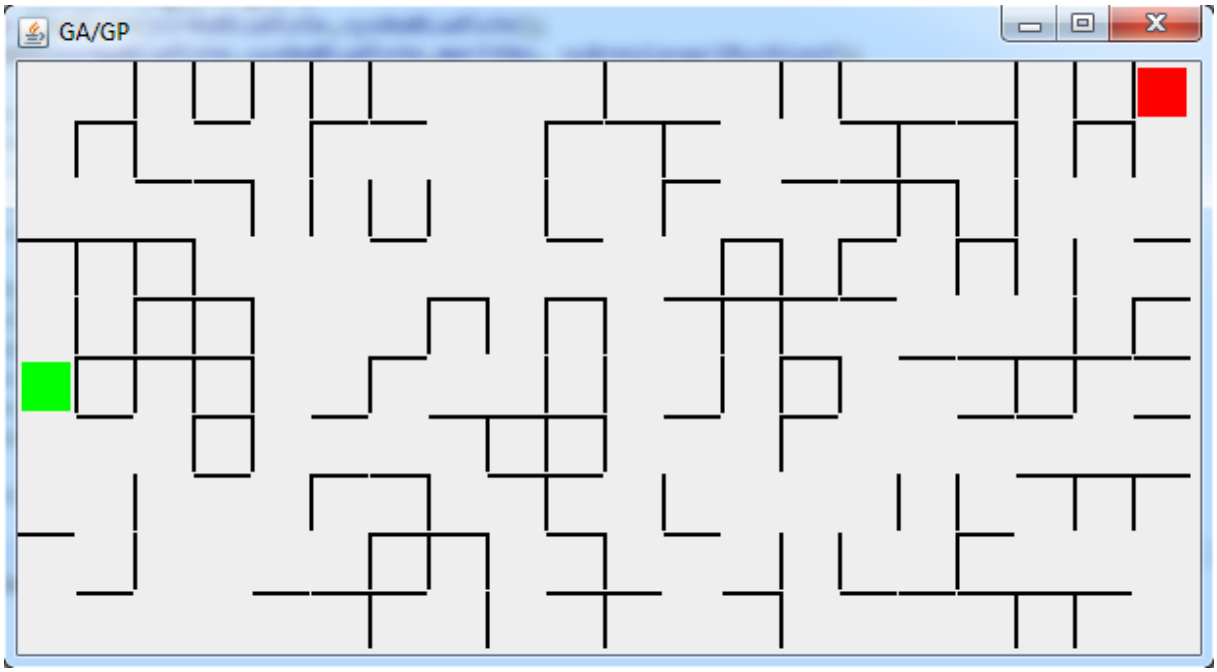
		celkové množství uchovávaných informací		
šířka	výška	původní struktura buněk	1. nová struktura buněk	2. nová struktura buněk
10	10	500	320	261
100	100	50000	30200	29601
1000	1000	5000000	3002000	2996001
10000	10000	500000000	300020000	299960001
100000	100000	50000000000	30000200000	29999600001
1000000	1000000	5000000000000	3000002000000	2999996000001

Tabulka 10 Množství nesených informací (zdroj: vlastní tvorba)

Z tabulky je tedy patrné, že poslední způsob záznamu nese nejmenší množství informací o samotném bludišti a též nevyžaduje žádné sofistikovaný způsob dotazování se na možnosti pohybu a může se tak jevit jako nejlepší technikou. Na druhou stranu je si třeba uvědomit, že třetí způsob má potencionálně velkou nevýhodu v podobě hluchých míst, která jsou zobrazena zelenou barvou v tabulce 9. I když tato místa nenesou žádnou informaci, je pravděpodobné, že většina programovacích jazyků i těmto hluchým místům vymezí potřebnou paměť, která i přesto, že nebude využívána, bude stále rezervována a tedy i po celou dobu evoluce nepoužitelná.

Nakonec byl zvolen třetí způsob záznamu a tomu byly také upraveny třídy Bludiste a MujJedinec z původního základu popsaného v kapitole 6.2. Třída Bludiste bylo upraveno na odlišný způsob záznamu a byla přidáno opatření zabraňující opuštění bludiště a třída MujJedinec byla upravena ze způsobu pohybu pomocí směrového vektoru na čtyř-směrový pohyb. Dále byly přidány mechanizmy pro zajištění, že náhodně vygenerované bludiště bude mít vždy řešení. Jedná se tedy o mechanizmy hledání sousedů ideálních ke sloučení, odstranění zdi mezi nimi a v neposlední řadě záplavový algoritmus, který byl mírně upraven ve smyslu možných hodnot. Na rozdíl od původního návrhu, kdy vlastnost štítek u buněk nabývala hodnot 1, když patřila do oblasti přístupné ze startu a hodnotu 2 v případě oblasti cíle, jsou nyní tyto hodnoty true pro oblast startu a false pro oblast cíle.

Dále bylo implementováno jedno pravidlo, které lze vyvodit ze všech ilustrativních obrázků náhodně vygenerovaných bludišť v Choubeyho práci týkající se umístění startu a cíle v bludišti. Tímto pravidlem je, že buňka reprezentující start je vždy náhodně zvolena z levých krajních buněk a buňka cíle je zvolena z buněk pravých krajních buněk. Příklad je ukázán na obrázku 9, na kterém je start vyznačen zeleným čtvercem a cíl červeným.



Obrázek 9 Ukázka pravidla pro umístění startu a cíle (zdroj: vlastní tvorba)

8.2. Samotný genetický algoritmus

Na rozdíl od způsobu záznamu bludiště, který nemá vliv na výsledky samotné evoluce, ale pouze na dobu potřebnou k výpočtům a množství potřebné paměti, by úpravy v samotném logickém fungování evoluce znamenaly odklon od původní předlohy, kterou se budeme snažit upravovat až v pozdějších kapitolách a také bychom ztratili možnost porovnávat tyto modifikace s původní předlohou. Z tohoto důvodu nebude v této fázi probíhat žádné úpravy samotné evoluce na úrovni logického fungování a spíše budou doplněny o blíže nespecifikované části.

Choubey využívá ve svém experimentu mnoho standardních technik, které jsou dále stručně popsané. Generování první generace náhodným způsobem, kdy každá přípustná alela (hodnota genu) má stejnou pravděpodobnost výběru pro daný gen. Jednoduché kódování v podobě celočíselných hodnot od 0 do 3 (včetně obou těchto hodnot), kdy hodnota 0 značí pohyb doleva, hodnota 1 nahoru, hodnota 2 doprava a hodnota tři dolů. Jednobodová mutace s pravděpodobností 10%.

Další technikou, která ovšem již není tak běžná, je technika křížení, která používá sčítání a odčítání a je blíže popsána v kapitole 7.2. Z pohledu autora této práce se jedná o techniku, která je velice nestandardní a ne vždy se gen z rodičovského chromozomu přenese do genu alespoň jednoho potomka, což je jeden ze základních principů jak evoluce v přírodě, tak evoluce v genetickém programování. Tabulka 11 znázorňuje všechny možné kombinace.

A	B	sčítání	odčítání
0	0	0	0
0	1	1	3
0	2	2	2
0	3	3	1
1	0	1	1
1	1	2	0
1	2	3	3
1	3	0	2
2	0	2	2
2	1	3	1
2	2	0	0
2	3	1	3
3	0	3	3
3	1	0	2
3	2	1	1
3	3	2	0

Tabulka 11 Kombinace křížení (zdroj: vlastní tvorba)

Na obrázku vidíme čtyři sloupce, kde první dva sloupce reprezentují chromozomy dvou rodičů a další dva sloupce poté jejich potomky vytvořené pomocí sčítání a odčítání, u nich jsou barevně vyznačeny dva možné případy. Prvním případ je vyznačen zeleně a je to případ, kdy daný gen byl převzat od jednoho z rodičů. Druhý případ je vyznačen červeně a označuje gen, který neměl ani jeden z rodičů. Nicméně je zde patrný i fakt, že každá přípustná hodnota má v potomkových chromozomech četnost výskytu stejnou jako všechny ostatní, díky čemuž není žádná hodnota upřednostňována před ostatními a další populace nemají sklon k degeneraci a zachovává se potřebná genová variabilita.

Dále pravděpodobnost takovéhoto křížení je 80%, což nechává 20% šanci, že rodičovská dvojice se bez jakýkoliv změn stává součástí další generace, což jednak mírně zachovává genovou variabilitu a také jsou zachováni zdatní jedinci v nezměněné formě do další generace a mají šanci se křížit až s další generací.

Jedním z nejlogičtějších způsobů, jak uchopit Choubeyho fitness funkci (Rovnice 2) a blíže ji dodefinovat do použitelné podoby, je sčítání rozdílů x a y hodnot daných buněk, tedy přepsání rovnici do podoby:

$$f' = \frac{[(sb.x - brs.x) + (sb.y - brs.y)]}{[(brc.x - sb.x) + (brc.y - sb.y)]} * 100,$$

Rovnice 3 Druhý tvar funkce fitness

V tabulce 12 jsou znázorněny buňky startu a cíle a hodnoty fitness vypočítané pomocí rovnice 4 pro každou buňku v bludišti o rozměrech 5x5.

	-33,33	-20,00	0,00	33,33	100,00
	-20,00	0,00	33,33	100,00	300,00
start ->	0,00	33,33	100,00	300,00	dělení nulou <- cíl
	33,33	100,00	300,00	dělení nulou	-500,00
	100,00	300,00	dělení nulou	-500,00	-300,00

Tabulka 12 Ukázka prvního návrhu fitness funkce (zdroj: vlastní tvorba)

Z tabulky 12 jsou na první pohled patrné tři nedostatky. První z nich jsou hodnoty fitness, které nelze vypočítat kvůli dělení nulou. K tomu dochází ve chvílích, kdy součet souřadnic x a y dané buňky se rovná součtu souřadnic cílové buňky. Dalšími nechtěnými hodnotami jsou hodnoty záporné, které jsou jako hodnoty fitness naprosto nevyhovující a porušují základy genetických algoritmů i genetického programování. Třetím velkým nedostatkem jen odlišné hodnocení, než které bychom očekávali. Z výše napsané funkce bychom očekávali, že lépe budou ohodnocena individua blíže k cíli a dále o startu nicméně z obrázku 20 vidíme na vedlejší diagonále stejné hodnoty fitness. Všechny tyto problémy lze eliminovat pomocí absolutní hodnoty a přepsání rovnice do následujícího tvaru:

$$f'' = \frac{[|sb.x - brs.x| + |sb.y - brs.y|]}{[|brc.x - sb.x| + |brc.y - sb.y|]} * 100,$$

Rovnice 4 Třetí tvar funkce fitness

Pomocí rovnice 5 může vytvořit obdobnou tabulku 13.

	33,33	60,00	100,00	166,67	300,00	
	20,00	50,00	100,00	200,00	500,00	
start ->	0,00	33,33	100,00	300,00	dělení nulou	<- cíl
	20,00	50,00	100,00	200,00	500,00	
	33,33	60,00	100,00	166,67	300,00	

Tabulka 13 Ukázka druhého návrhu fitness funkce (zdroj: vlastní tvorba)

Zde již vidíme požadované chování funkce, a hodnoty fitness, které dávají z logického pohledu odpovídající hodnoty a tedy, že lépe budou ohodnocení jedinci, kteří jsou blíže cíli a dále od startu. Nicméně i zde přetrval jeden problém, na který jsme již narazili dříve a tím je dělení nulou. Nicméně se tento problém nachází pouze na buňce reprezentující cíl a tedy se dá považovat za signál, že jsme nalezli jedince, který se úspěšně dostal do cíle. Tato podoba fitness funkce je již vyhovující, a proto byla v této podobě i implementována.

Poslední netradiční technikou v Choubeyho experimentu využívanou je technika postupné zvětšování délky chromozomu. Jako počáteční délka je podle Choubeyho postačující dvojnásobek délky bludiště (n * 2, kde n je délka bludiště) a postupně je zvětšována na délku bludiště na druhou (n^2, kde n je délka bludiště), kterého dosahuje pouze ve výjimečných případech, jelikož je řešení nalezeno daleko dříve. Tyto hodnoty platí pro bludiště ve čtvercovém tvaru, pro obdélníkové tvary už Choubey tyto výpočty neupravuje. Nicméně by jako ideální mohly být rovnice upraveny, aby nejdříve vypočítaly průměrnou hodnotu obou stran a poté by tato hodnota byla roznásobena 2, popř. se vypočítala její druhá mocnina. Tyto rovnice by tedy měly tvar:

$$dch = \frac{(n+m)}{2} * 2,$$

Rovnice 5 Rovnice pro výpočet počáteční délky chromozomu

který se dá přepsat do tvaru:

$$dch = n + m,$$

Rovnice 6 Přepis rovnice 6

kde n je šířka bludiště, m je výška bludiště a dch je délka chromozomu pro výpočet počáteční délky chromozomu. Pro výpočet maximální délky chromozomu by se pak tato rovnice upravila do tvaru:

$$dch = \frac{(n+m)^2}{2},$$

Rovnice 7 Rovnice pro výpočet maximální délky chromozomu

kterou bychom mohli přepsat do jednoduší podoby, která by vracela skoro stejné hodnoty a to:

$$dch = n * m,$$

Rovnice 8 Přepis rovnice 8

kde n je šířka bludiště, m je výška bludiště a dch je délka chromozomu.

Toto postupné zvětšování délky chromozomu se odvíjí od počtu generací a samotná délka chromozomu závisí na pořadí dané generace. První náhodně vygenerovaná generace bude začínat s minimální délkou chromozomu tedy s dvojnásobkem délky bludiště a postupně se tato velikost bude zvětšovat teoreticky až k poslední generaci s maximální délkou, tedy délka bludiště na druhou.

Poslední nezbytnou technikou, o které se Choubey ve své práci nezmiňuje je selekční mechanizmus, a proto byl použit již implementovaný mechanizmu ruletové selekce s přímou úměrností vzhledem k fitness jedinci (2).

Parametry simulace byly nastaveny podle kapitoly 7. 2. v odstavci 5.

9. Testy, úpravy a porovnávání

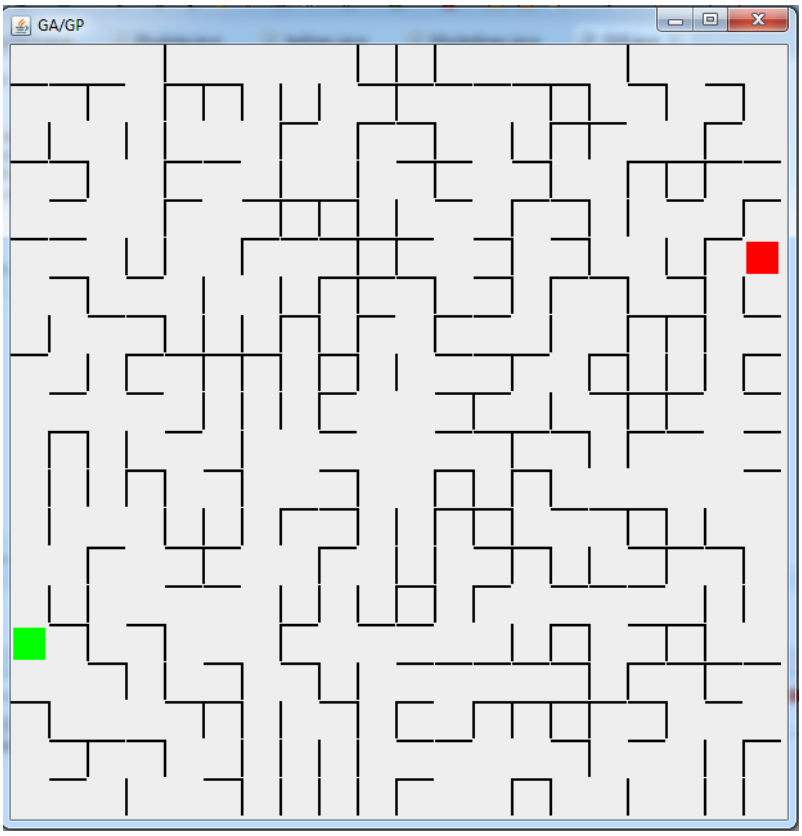
V této kapitole bude nejprve uvedeno několik druhů bludišť, na kterých bude testováno nejprve originální řešení, jehož implementace je popsána v kapitole 8. Poté budou prováděny různě změny ať už na úrovni technik, které byly v evoluci použity, nebo parametrů, jako třeba velikost populace, pravděpodobnost mutace, atd. Tyto různé modifikace budou poté testovány a porovnávány mezi sebou a původním řešením. Cílem těchto modifikací bude vylepšit původní řešení v několika ohledech. Mezi tyto aspekty patří snížení celkového počtu volání ohodnocující funkce, snížení času potřebného k nalezení řešení a zvýšení míry úspěšnosti.

Testy budou prováděny na notebooku Lenovo G500 s dvoujádrovým procesorem Intel Core i5-3230M o výkonu 2.6 GHz, operační pamětí 4 GB a 64bitovým operačním systémem Windows 7 Profesional. Všechny testy budou prováděny v době, kdy nebude notebook vytížen žádnou jinou aplikací nežli testovací.

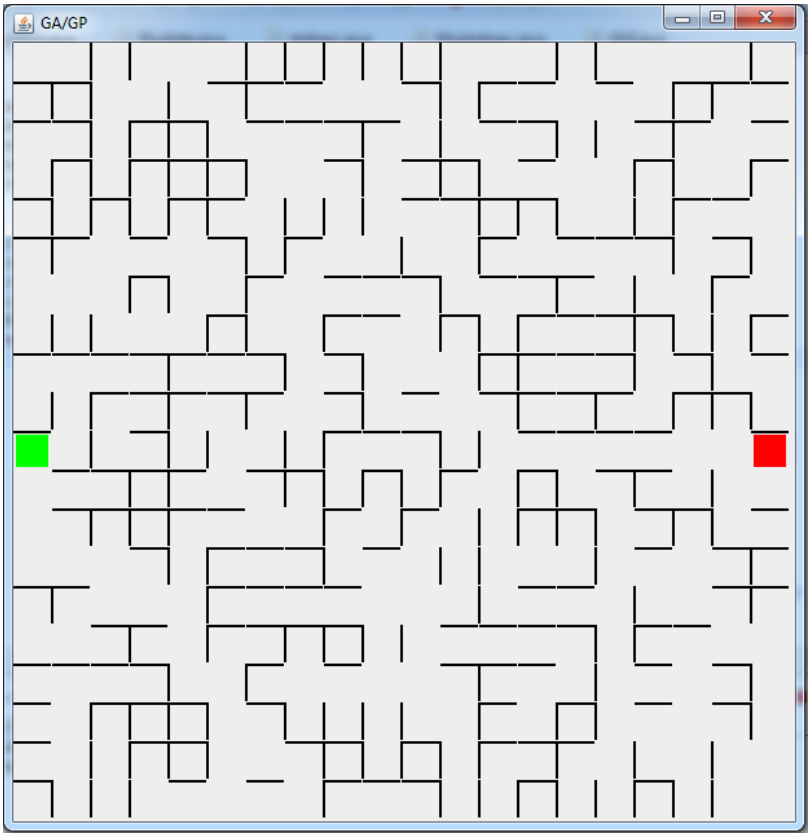
Originální řešení a všechny jeho další modifikace budou na každém bludišti testovány celkem tisíckrát (pokaždé nový evoluční cyklus), přičemž budou měřeny dva údaje a to celkový počet volání funkce fitness a celkový čas potřebný pro běh celého evolučního cyklu a to jak pro případy, kdy bylo nalezeno řešení, tak pro případy, kdy se nalézt řešení nepodařilo. Z těchto hodnot budou poté vypočítávány aritmetické průměry, mody, mediány, rozptyly a míry úspěšnosti.

9.1. Jednotlivá bludiště

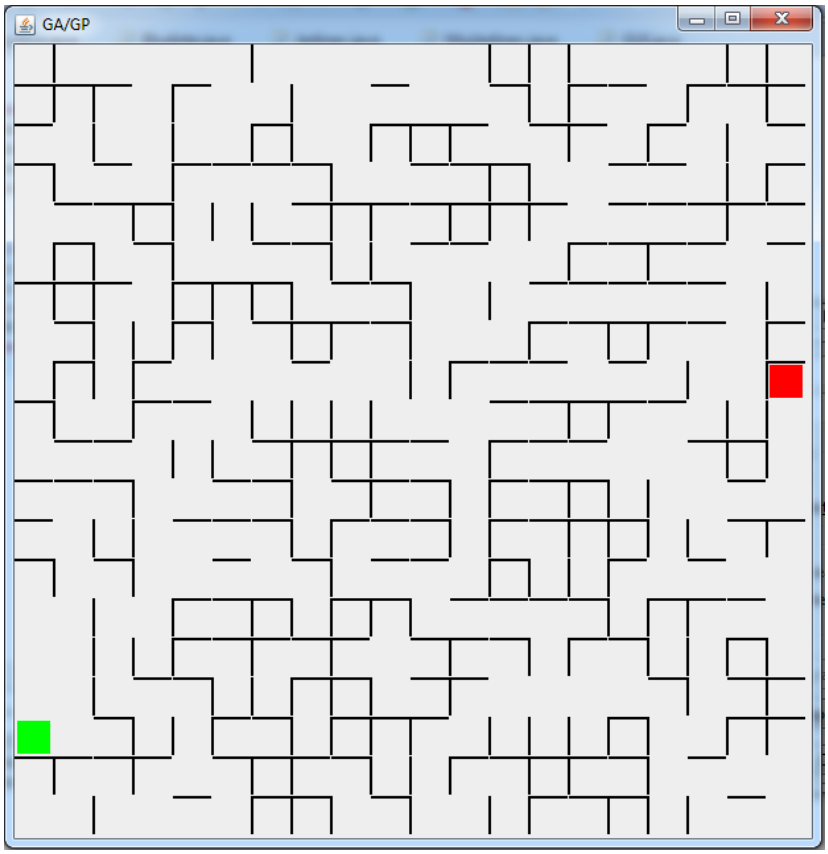
Na obrázcích 10 až 14 jsou vidět všechna bludiště, na kterých budou prováděny všechny následující testy. Tato bludiště byla vygenerována pomocí upravené metody A-maze, která je popsána v kapitole 8.1.



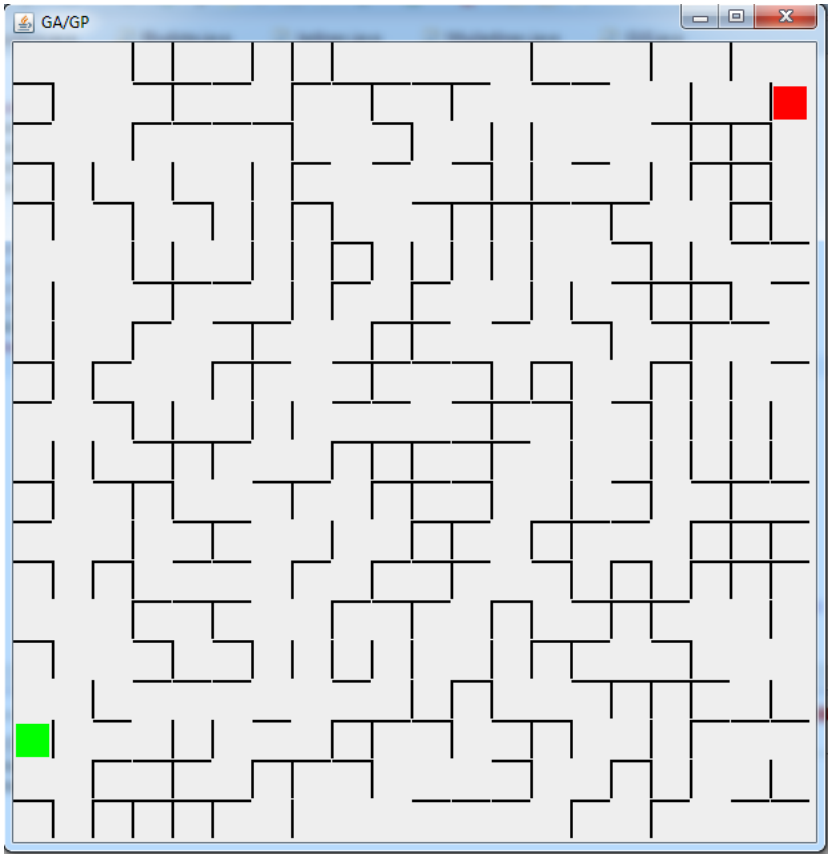
Obrázek 10 Bludiště č. 1 (zdroj: vlastní tvorba)



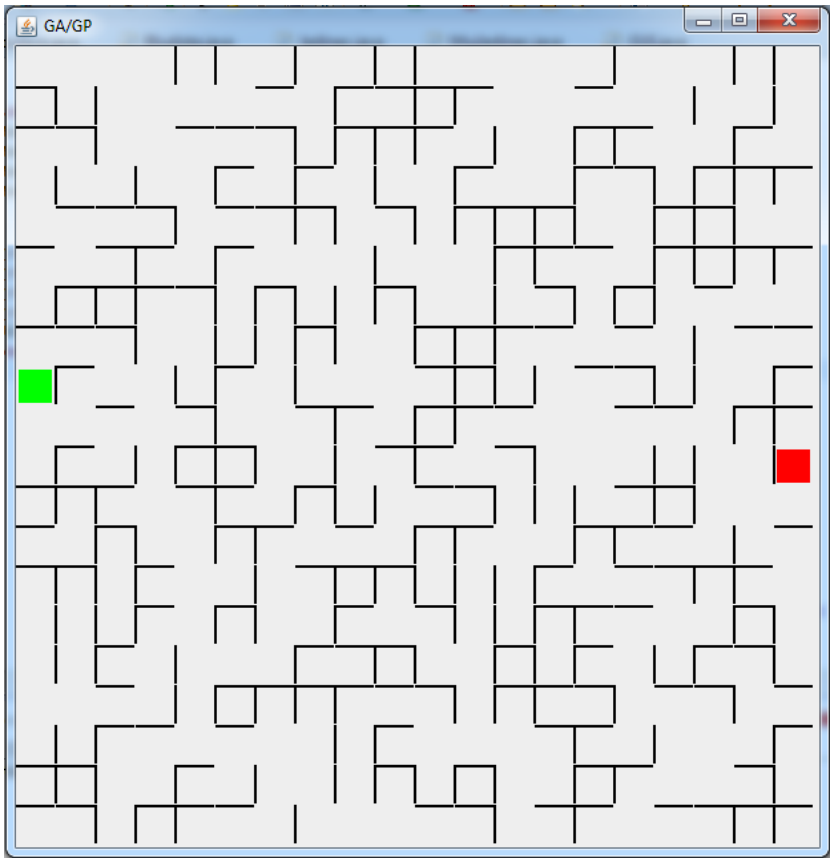
Obrázek 11 Bludiště č. 2 (zdroj: vlastní tvorba)



Obrázek 12 Bludiště č. 3 (zdroj: vlastní tvorba)



Obrázek 13 Bludiště č. 4 (zdroj: vlastní tvorba)



Obrázek 14 Bludiště č. 5 (zdroj: vlastní tvorba)

9.2. Výsledky testu Nitin S. Choubeyho řešení

V tabulkách 14 až 16 vidíme výsledky testů originálního řešení, jehož implementace je popsána v kapitole 8.

časy v sekundách				
	průměr	modus	medián	rozptyl
Bludiště č. 1	10,4	12	10,2	29,32
Bludiště č. 2	3,1	0,8	2,6	5,28
Bludiště č. 3	3,6	1,3	3,4	5,19
Bludiště č. 4	3,0	1,6	2,7	3,92
Bludiště č. 5	8,2	7,6	7,6	22,36

Tabulka 14 Časové výsledky z testu originálního řešení (zdroj: vlastní tvorba)

volání funkce fitness				
	průměr	modus	medián	rozptyl
Bludiště č. 1	31761,4	35106	33665,0	144694188,6
Bludiště č. 2	22574,6	22771	22053,0	172670093,7
Bludiště č. 3	23952,4	16769	23685,0	178152476,4
Bludiště č. 4	22751,0	12736	21700,0	196134849,4
Bludiště č. 5	28735,9	6621	29721,0	153236310,3

Tabulka 15 Výsledky počtů volání funkce fitness z testu originálního řešení (zdroj: vlastní tvorba)

počty testů			
	neúspěšných	úspěšných	úspěšnost (v %)
Bludiště č. 1	423	577	57,7
Bludiště č. 2	69	931	93,1
Bludiště č. 3	91	909	90,9
Bludiště č. 4	58	942	94,2
Bludiště č. 5	323	677	67,7

Tabulka 16 Výsledky úspěšnosti z testu originálního řešení (zdroj: vlastní tvorba)

9.3. Změna operátoru křížení

Jako první úprava bude provedena změna křížícího operátoru. Jak již bylo řečeno v kapitole 8. 2. operátor křížení, který navrhl doktor Nitin S. Choubey má relativně destruktivní účinky na chromozomy rodičů a předává pouze část informací na potomky. To byl hlavní důvod, proč byla jako první změna zvolena právě změna operátoru křížení. Jako nový operátor křížení bylo zvoleno náhodné jednobodové křížení se 100% pravděpodobností.

9.4. Výsledky testu první upravené verze

V tabulkách 17 až 19 vidíme výsledky testů první upravené verze originálního řešení, jehož úprava je popsána v kapitole 9.3.

časy v sekundách				
	průměr	modus	medián	rozptyl
Bludiště č. 1	5,6	4,3	5,3	7,98
Bludiště č. 2	0,7	0,9	0,7	0,21
Bludiště č. 3	0,9	0,7	0,8	0,31
Bludiště č. 4	1,3	0,8	1,3	0,65
Bludiště č. 5	2,5	1,1	2,4	2,10

Tabulka 17 Časové výsledky z testu první upravené verze (zdroj: vlastní tvorba)

volání funkce fitness				
	průměr	modus	medián	rozptyl
Bludiště č. 1	31795,0	47064	33095,5	141007821,4
Bludiště č. 2	16544,3	15347	15999,5	92873007,7
Bludiště č. 3	18008,9	27849	17144,0	122557933,4
Bludiště č. 4	21309,9	22143	20398,0	175225000,9
Bludiště č. 5	24853,3	28002	24275,0	179992274,3

Tabulka 18 Výsledky počtů volání funkce fitness z testu první upravené verze (zdroj: vlastní tvorba)

počty testů			
	neúspěšných	úspěšných	úspěšnost (v %)
Bludiště č. 1	286	714	71,4
Bludiště č. 2	0	1000	100
Bludiště č. 3	0	1000	100
Bludiště č. 4	12	988	98,8
Bludiště č. 5	121	879	87,9

Tabulka 19 Výsledky úspěšnosti z testu první upravené verze (zdroj: vlastní tvorba)

Jak můžeme porovnat tyto výsledky s výsledky z testu originálního řešení, pouhou změnou operátoru křížení jsme dosáhly až několikanásobně lepších časových výsledků. V některých případech se jedná o 2x až 4x v průměru kratší čas potřebný k nalezení řešení. Co se týče počtů volání funkce fitness, ve většině případů bylo opět dosaženo lepších výsledků a to až v řádech pěti tisíců povolání. Zde by autor práce rád upozornil, že kvůli velkým rozptylům jsou hodnoty modus velmi nepřesné, zpravidla se jedná o dva, popř. tři výskyty daného čísla. Na závěr se podařilo dosáhnout i lepší úspěšnosti nalezení řešení.

9.5. Změna operátoru mutace

První změna, změna operátoru křížení, popsaná v kapitole 9.3 odstraňovala z původního řešení ono velké „rozbíjení“ rodičovských chromozomů. Nicméně i v této úpravě přetrvává ještě další faktor, který má podobný neblahý efekt. Tímto faktorem je operátor mutace s 10% šancí změnit každou alelu každého jedince. Proto pro další úpravu bylo zvoleno snížení této pravděpodobnosti na 5%.

9.6. Výsledky testu druhé upravené verze

V tabulkách 20 až 22 vidíme výsledky testů druhé upravené verze, která navazuje na první upravenou verzi z kapitoly 9.3. Tyto změny jsou uvedeny v kapitole 9.5.

časy v sekundách				
	průměr	modus	medián	rozptyl
Bludiště č. 1	3,3	3,5	3,3	2,87
Bludiště č. 2	0,2	0,1	0,2	0,01
Bludiště č. 3	0,1	0,1	0,1	0,01
Bludiště č. 4	0,2	0,1	0,2	0,02
Bludiště č. 5	0,7	0,5	0,7	0,19

Tabulka 20 Časové výsledky z testu druhé upravené verze (zdroj: vlastní tvorba)

volání funkce fitness				
	průměr	modus	medián	rozptyl
Bludiště č. 1	27267,1	17976	27381,0	162905824,9
Bludiště č. 2	9440,2	8949	9196,0	27596837,72
Bludiště č. 3	7485,6	7983	7162,5	10211171,05
Bludiště č. 4	9931,2	8390	8788,5	31073322,68
Bludiště č. 5	16796,9	11203	15836,5	74586056,93

Tabulka 21 Výsledky počtů volání funkce fitness z testu druhé upravené verze (zdroj: vlastní tvorba)

počty testů			
	neúspěšných	úspěšných	úspěšnost (v %)
Bludiště č. 1	157	843	84,3
Bludiště č. 2	0	1000	100
Bludiště č. 3	0	1000	100
Bludiště č. 4	0	1000	100
Bludiště č. 5	0	1000	100

Tabulka 22 Výsledky úspěšnosti z testu druhé upravené verze (zdroj: vlastní tvorba)

V porovnání s výsledky jak z první upravené verze z kapitoly 9.4, tak s výsledky originálního řešení z kapitoly 9.2 jsme dosáhli opět lepších výsledků ve všech ohledech. Mezi nejvýraznější zlepšení zde patří průměrný počet volání funkce fitness, který byl zmenšen v některých případech až o deset tisíc volání, což je v přepočtu 20 generací.

10. Závěr

Tato práce uvádí čtenáře do krátké historie genetických algoritmů a genetického programování a poté popisuje základy a rozdíly těchto dvou paradigmat. Další sekce je věnována definování problému hledání cesty v bludišti a představení řešení doktora Nitin S. Choubeyho. Dále je toto řešení implementováno a jsou na něm provedeny dvě modifikace. Na závěr jsou výsledky testů jak modifikovaných řešení, tak originálního řešení porovnávány.

Obě navržené modifikace dosáhly lepších výsledků než originální řešení. V obou případech toho bylo dosaženo lepším zachováváním chromozomů zdatných jedinců a jejich jednoduchým kombinováním. Nicméně je zde nutno podotknout, že všechny testy probíhaly pouze na pěti bludištích, které měly stejně rozměry 20x20. Z tohoto důvodu by mohlo být velice zajímavé všechna řešení porovnávat nejenom na bludištích o rozměrech 20x20, ale o rozměrech, kde by klasické metody řešení, jako například záplavový algoritmus, nemohly být aplikovány z důvodu obrovského množství potenciálních řešení, což je právě oblast, kde genetické algoritmy excelují nad konvenčními metodami. Jako další oblast, kde by bylo zajímavé porovnávat řešení popsaná v této práci, je rozšíření bludišť do dalších rozměrů a to ne jenom do třetího, ale do libovolného n rozměru.

V takovýchto bludištích by velice záleželo na výskytu lokálních maxim (cest v bludišti, která vedou velice blízko k cíli, ale jedná se o slepé cesty), V případě nižšího výskytu lokálních maxim, by upravené verze genetického algoritmu měli mít větší výhodu díky lepšímu zachovávání dobrých částí chromozomů. V případě, že lokálních maxim by bylo více, s největší pravděpodobností by bylo originální řešení úspěšnější, díky silnému mutačnímu operátoru a upravenému operátoru křížení (kapitola 8.2), které zachovávají genovou variabilitu.

11. Seznam zdrojů

1. **Koza, John R.** *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems.* Stanford : Stanford University, 1990.

2. **Hynek, Josef.** *Genetické algoritmy a genetické programování.* Praha : Grada Publishing, a.s., 2008.

3. **Darwin, Charles.** *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life.* London : John Murray, 1859.

4. **Holland, John Henry.** *Adaptation in Natural and Artificial Systems.* Michigan : University of Michigan Press, 1975.

5. **Brně, Modelová univerzita v.** is.mendelu.cz. *Modelová univerzita v Brně.* [Online] [Citace: 23. Duben 2016.] http://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=21840.

6. **Koza, John R., a další, a další.** *genetic programming iv routine human-competitive machine intelligence.* Norwell : Kluwer Academic Publishers, 2003.

7. **Gray, Frank.** *Pulse code communication. US 2632058 A* USA, 17. Březen 1953.

8. **team, tutorialspoint.** tutorialspoint.com. *tutorials point simply easy learning.* [Online] [Citace: 23. Duben 2016.] http://www.tutorialspoint.com/computer_logical_organization/binary_codes.htm.

9. **Miller, Brad L. a Goldberg, David E.** complex-systems.com. [Online] 1995. [Citace: 23. Duben 2016.] http://www.complex-systems.com/pdf/09-3-2.pdf.

10. **Gun, Jeroen van der.** http://games.blijbol.nl/. *blijbol.* [Online] 2010. [Citace: 23. Duben 2016.] http://games.blijbol.nl/en/pc/snake.

11. **Choubey, Nitin S.** A-Mazer with Genetic Algorithm. *Journal of Computer Applications.* 2012.

12. Přílohy

12.1. Obrázky

12.2. Rovnice

12.3. Tabulky