VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

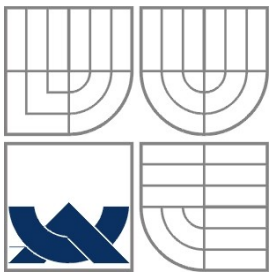
# Evoluce struktury a parametrů agenta pohybujícího se v terénu

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

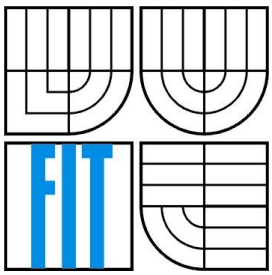
AUTOR PRÁCE
AUTHOR

ALEŠ TOMEČEK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# Evoluce struktury a parametrů agenta pohybujícího se v terénu

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
ALEŠ TOMEČEK

VEDOUCÍ PRÁCE
SUPERVISOR
ING. JIŘÍ ZUZAŇÁK

BRNO 2010

**SEM PROSÍM VLOŽIT ZADÁNÍ**

V JEDNOM VÝTISKU NECHAT PRÁZDNOU STRÁNKU

# Abstrakt

Tato práce stručně pojednává o historii a současnosti genetických algoritmů v informatice. Dále nabízí krátký přehled nejpoužívanějších metod používaných evolučními algoritmy. Jejich použití je demonstrováno na aplikaci evoluce agenta pro průjezd jednoduchým terénem.

# Abstract

This paper briefly discusses the history and present genetic algorithms in computer science. offers a brief overview of most common methods used by evolutionary algorithms. Their use is demonstrated in the application for evaluating agent for the crawl of a simple terrain.

# Klíčová slova

Evoluční algoritmy, genetické algoritmy, evoluce, 2D terén.

# Keywords

Evolutionary algorithms, genetic algorithm, evolution, 2D terrain.

# Citace

Aleš Tomeček, Evoluce struktury a parametrů agenta pohybujícího se v terénu, bakalářská práce, Brno, FIT VUT v Brně, 2010

Evoluce struktury a parametrů agenta pohybujícího se v terénu

# Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jiřího Zuzaňáka.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Aleš Tomeček
4.5.2010

# Poděkování

Za odbornou pomoc bych chtěl poděkovat vedoucímu práce Ing. Jiřímu Zuzaňákovi. Mým rodičům za podporu ve studiu, a v neposlední řadě přítelkyni za obětovaný čas ve prospěch práce.

# Obsah

# 1 Úvod

S moderní dobou rostou stále více požadavky na výpočetní a paměťovou kapacitu. Tyto trendy ale nejsou srovnatelné, nároky na data rostou mnohem rychleji než výpočetní možnosti. Nejen díky tomu se více obrací pozornost na nové způsoby práce s daty a řešení rozsáhlých problémů, na kterých jsou dnešní konvenční přístupy pomalé, či selhávají úplně.

Jednou z oblastí, která je využívána jako zdroj inspirace jsou i přírodní zákonitosti. Ty efektivně řeší problémy už po mnoho miliónů let způsoby, které se začínají v informačních technologiích teprve objevovat. Můžeme jmenovat pár příkladů za mnohé, například spolupráce mravenců, jako výborná ukázka paralelismu či evoluci jako vývoj nejvhodnějšího řešení.

Právě druhému tématu je věnována tato práce. Navzdory problémům s výpočetní kapacitou existovala snaha napodobení evoluce v přírodě již před 35 lety. Z této poměrně dlouhé doby jsme vybrali pár příkladů, které spolu se stručnou historií uvádíme v kapitole 2.

Představení metod pro prohledávání rozsáhlých stavových prostorů se nachází v kapitole 3. Hlubší rozbor fungování evolučních algoritmů a představení nejčastějších metod používaných při jejich implementaci lze naleznout v kapitole 4. Na tuto část navazuje kapitola 5 popisující problémy, které mohou nastat při jejich použití.

Pro demonstraci evolučních algoritmů byla naprogramována aplikace pro simulaci jedince pro pohyb v terénu. Implementační detaily z hlediska evoluce a ovládání programu jsou popsány v kapitolách 6 a 7.

V závěrečné kapitole se nachází shrnutí dosažených výsledků a návrhy na případné další rozšíření.

# 2 Historie a současnost

## 2.1 Původ myšlenky

Současné aplikace evolučních přístupů v informatice mají předobraz v díle Charlese Darwina O původu druhů z roku 1859. I přes rozporuplné přijetí položila tato kniha základ odvětví evoluční biologie. Po více jak sto letech publikoval svou práci průkopník na poli evolučních algoritmů, a podle některých za jejich zakladatel, John Henry Holland. V jeho knize *Adaptation in Natural and Artificial Systems* z roku 1975 byla poprvé publikována myšlenka využití Darwinovi teorie v informačních systémech. Od té doby je tato oblast cílem výzkumů, skutečný rozvoj ale zažila až s příchodem modernějších technologií.

## 2.2 Současnost

V moderní historii se použití evolučních principů v informatice poměrně rozšířilo. Je to hlavně díky většímu rozmachu výpočetního výkonu a zlepšení metodologie jednotlivých přístupů, což umožňuje využití evolučních algoritmů na problémy, na které to v minulosti nebylo možné. Níže uvádíme pár příkladů jejich použití, které demonstrují širokou oblast jejich možného nasazení.

### Evoluce oka

V roce 1994 publikovali Nilsson Dan a Suzanne Pelger studii „Pesimistický odhad potřebný pro evoluci oka“. Pomocí matematického modelu ukázali, co se stane se světločivným povrchem pokožky, pokud mu jsou dovoleny změny, které zlepší optické vlastnosti oka. Během pár stovek tisíc generací dokázali vyvinout model oka podobný našemu. Včetně nerovnoměrně zakřivující čočky a rohovky. Matematický model byl ve své podstatě aplikací evolučních přístupů. Tato práce bývá často předkládána jako jeden z důkazů proti „Inteligentnímu návrhu“ (opak Darwinovy evoluční teorie, tvrdí že vše bylo navrženo vyšší mocí). Původní teorie čistě biologického zaměření tedy po svém přerodu také v teorii informatickou potvrzuje sama sebe.

Je ale nutné dodat, že tento výzkum je částečně zpochybňován. [4][5]

### Vývoj nového hardwaru

Díky blízkosti informatiky a elektrotechniky se dočkala Darwinova teorie využití také v hardwarové oblasti. Inženýr Adrian Thompson z univerzity v Sussexu experimentoval s vyvíjením obvodů od roku 1993. O tři roky později vešel ve známost jeho experiment s rozpoznáváním signálu pomocí programovatelných hradlových polí (Field Programmable Gate Array – FPGA.) Pro normální funkčnost je potřeba, aby byl přiváděn hodinový signál. Thompsonův cíl byl ale zjistit, zda dokáže evoluce využít fyzické vlastnosti křemíku a tak v jeho experimentech nebyly hodiny použity. Úkolem jedinců bylo rozlišit mezi dvěma elektrickými signály o kmitočtu 1kHz a 10kHz. Rozhodovací funkce, která vybírala nejvhodnější jedince ke křížení, byla určena

průměrným rozdílem na výstupu mezi testovacími signály. Thompsonovy populace pracovali s 50 jedinci. Po 3 500 generacích byla vygenerována konfigurace, která pro 1kHz dávala 0V pro 10kHz 5V. Zajímavostí je, že ze 100 propojek je jich potřeba pouze 35, přičemž 5 z nich nemá žádný zdánlivý vliv na zbytek obvodu. Po jejich odstranění ale přestane celý obvod fungovat. Funkce obvodu nebyla dodnes uspokojivě vysvětlena.

Použití těchto postupů na reálných modelech je ale náročnější, než jejich pouhá simulace. Většina vědecké komunity zastává názor, že simulace jsou dostatečnou náhradou. [6]

Evoluční vývojem hardwaru se mimo jiné zabývá i výzkumná skupina na fakultě informačních technologií VUT v Brně *Evolvable Hardware* ve složení Ph.D Bidlo Michal, Ing. Gajda Zbyšek, Ph.D. Růžička Richard, doc. Sekanina Lukáš, Ing. Slaný Karel, Ing. Stareček Lukáš, Ing. Vašíček Zdeněk a Ing. Žaloudek Luděk. [8]

## Návrh antén

Výzkumná skupina NASA *Evolvable systems group* vedená Jasonem Lohnem vyvinula již několik návrhů antén. Výsledky jejich práce jsou například použity v projektech *Mars Odyssey* či *Space Technology 5*. Pro vývoj je použito základních genetických algoritmů. Ačkoli byly výsledné antény obtížnější na výrobu, prokazovaly lepší vlastnosti než jejich konvenčně navržené protějšky. Navíc bylo k jejich výrobě, od návrhu po finální produkt, potřeba téměř o polovinu méně času. [7]

## Portrétování podezřelých

Genetické algoritmy nacházejí využití i mimo technické obory. Příkladem je například software FacePrint patentovaný v roce 1994 Victorem Johnstene, vystudovaným psychologem. Při své práci narazil na zajímavý fenomén, člověk bezpečně pozná známou tvář v davu, ale má obtíže jí popsat. Tento problém je nejvíce markantní při policejním výslechu svědků. Po porovnání současných systémů navrhnul systém identifikace, který zobrazí 30 náhodně vygenerovaných obličejů. Uživatel je pak vyzván k jejich ohodnocení na základě podobnosti s kýženým výsledkem. Obličeje jsou pak zakódovány jako sekvence 35 bitů, a následně je z nich za pomoci evolučních algoritmů vygenerována nová sada tváří k posouzení. Po průměrně 20 generacích jsou dosažené výsledky srovnatelné či lepší v porovnání s klasickými metodami. Tento přístup využití lidského faktoru jako fitness funkce je také znám pod pojmem interaktivní evoluce. Výhodou těchto metod je například možnost použít je tam, kde nelze použít standardní fitness funkce například z důvodu výpočetní náročnosti, nebo obtížné formalizovatelnosti funkce. Mezi stinné stránky patří dramatické zpomalení díky zdlouhavým reakcím uživatele. [1][3]

# 3 Metody prohledávání

Díky stálému nárůstu komplexnosti problémů, které pro jejich složitost nelze řešit analyticky,a pro které jsou klasické algoritmy nedostačující, se začíná rozvíjet oblast heuristických metod. Tyto techniky jsou ze své podstaty stochastické. S úspěchem se používají na NP úplné problémy. Mezi jejich hlavní nevýhody patří rizika uváznutí na lokáním maximu s tím spojený problém nejistoty optimálního řešení a obtížnost replikovat výsledky.

## *3.1 Simulované žíhání*

Simulované žíhání není samo o sobě evolučním algoritmem ve smyslu inspirace Darwinovou metodou. Její použití ale také umožňuje efektivnější prohledávání komplexních stavových prostorů a demonstruje základní problém algoritmů – uváznutí na lokálním maximu (viz kapitola 5.)

Metoda využívá pouze jedno kandidátní řešení. Zkoumá jeho okolí, a pokud najde úspěšnější stav, přesune se do něj. Obranou proti uváznutí na lokálním maximu je navíc funkce „teploty“, udávající s jakou pravděpodobností bude přijato i horší kandidátní řešení. Platí, že čím větší teplota, tím větší šance, že bude vybrán méně úspěšný kandidát. Během prohledávání se teplota postupně snižuje, na základě rychlosti konvergence algoritmu. V prvních krocích, při vysoké teplotě, je pak výběr téměř náhodný.

Díky své podstatě zkoumání nejbližšího okolí, se nejčastěji využívá v diskrétních prostorech – např. průchod grafem.

Za metodu blízkou způsobu, kterým prozkoumává stavový prostor simulované žíhání, může být považován zvláštní případ evoluce populace o jednom agentovi s návratem rodiče.

## *3.2 Evoluční algoritmy*

Evoluční algoritmy patří mezi heuristické metody inspirované Darwinovými evolučními principy. Jako celek se rozdělují na několik typů, které představujeme níže. Společným rysem všech metod práce s populací jedinců, výběr nejvhodnějších kandidátů a na jejich základě tvorba nové generace.

### Genetické algoritmy

Zřejmě nejrozšířenější typ evolučních algoritmů. Řešená úloha je zredukována na parametry, klíčové pro řešení. Ty jsou dále zakódovány do podoby genů, ze kterých se složí genom reprezentující jedince.

### Genetické programování

Genetické programování je blízké genetickým algoritmům. Je zde ovšem zásadní rozdíl v délce

genomu. Zatímco u genetických algoritmů je tato délka většinou neměnná, u programování se může měnit. Jedince nejčastěji reprezentuje stromová struktura, podobná syntaktickým stromům. Jsou ale používány i metody lineární reprezentace [11]. Úspěch v této oblasti dokazuje i to, že programy vyvinuté touto metodou jsou patentovány, díky své nadřazenosti programům psané klasickou metodou. Zatím se ovšem nejedná komplexní programy, ale spíše o optimalizační úlohy řazení, detekce a syntézy. [12]

## Evoluční programování

Tento přístup byl navržen Lawrencem Fogelem okolo r. 1960 [2]. V původní verzi byl použit pro simulaci učícího procesu a umělé inteligence. V praxi je používán se stejnou úspěšností jako evoluční strategie, jemuž je velmi podobný. Původ těchto metod je ale zcela nezávislý. Některé jeho verze jsou podobné genetickému programování, avšak s větším důrazem na vztah rodič-potomek. Dále evoluční programování více využívá strukturu samotného problému, aniž by vyžadovala nutnost jeho zakódování do genomu a nevyužívají se klasické operátory křížení, ale pouze mutace [13].

## Evoluční strategie

Na rozdíl od genetických algoritmů pracují evoluční strategie přímo s reálnými hodnotami, které problém definují. Jejich přístup je velice podobný evolučnímu programování a stoupenci obou teorií přiznávají, že jsou si hodně podobné.

# 4 Prostředky evoluce

## 4.1 Funkce ohodnocení

V některých zdrojích nazývána jako *fitness funkce*. Funkce hodnocení musí umět vrátit ohodnocení libovolného řešení z množiny všech existujících. Ohodnocení by mělo vyjadřovat vhodnost jedince pro řešení zkoumaného problému. Jedná se o analogii Darwinově teorii nejvhodnějšího, kdy schopní jedinci mají vyšší předpoklad k přežití než ostatní.

V praxi se používají 2 přístupy. První používaná metoda nahlíží na ohodnocení jako na vzdálenost od ideálního řešení. Platí tedy, že čím menší hodnota funkce, tím je řešení blíže ideálnímu a tudíž je lepší. Používá se například při průchodu grafem, kdy chceme najít co nejmenší počet přechodů. Pokud je možné určit nejideálnější dosažitelné hodnocení ještě před samotným řešením, bývá toto použito jako kritérium pro zastavení evoluce. Protikladem je přístup hodnotící jedince přímou úměrou. Tedy čím větší ohodnocení, tím vhodnější. Tento přístup se používá převážně tam, kde není možné určit nejideálnější podmínky. Takovým příkladem je například námi řešený problém. Snažíme se najít jedince, který se dostane v terénu co nejdále, a nemáme žádný ideální bod, kterého bychom chtěli dosáhnout.

Na řešeném problému pak záleží typ navracené hodnoty. Z praktických důvodů se nejčastěji se používá o jednorozměrné spojité (popřípadě vhodně diskretizované) číslo. Funkce může ale vracet například i vektor. Důležité je aby se tyto hodnoty daly porovnávat. Více o problémech s fitness funkcí je uvedeno v kapitole 5.

Správný návrh této funkce je jednou z klíčových oblastí návrhu. Při správně navržené funkci je snazší upravit další parametry evoluce, jako například selekci. Špatná nebo nevhodně implementované ohodnocení vede ke komplikacím další částí evoluce. [1]

## 4.2 Selekce

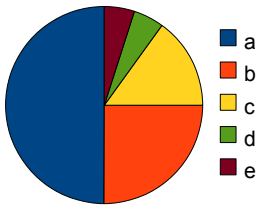
Algoritmus selekce má za úkol imitovat přirozený výběr. Tedy schopnost jedince v přírodě odolat predátorům, nemocem a sehnat dostatek potravy, aby se dožil do reprodukce schopného věku. Vyjádřením této schopnosti by měla být hodnota fitness funkce každého agenta. Nicméně ani v přírodě nedochází ke křížení pouze nejzdatnějších jedinců, a použité algoritmy by toto měly respektovat.

Podle [1] existují studie, které poukazují na to, že všechny metody jsou při vhodném nastavení srovnatelné.

### Výběr pomocí rulety

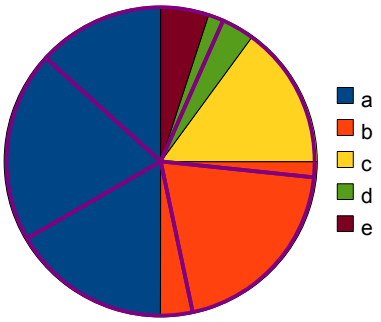
Díky své jednoduchosti implementace při zachování požadavků uvedených výše, se nejčastěji používá ruletový mechanismus selekce. Každý jedinec dostane na pomyslné ruletě přiřazenou část, jejíž velikost je přímo úměrná jeho hodnocení. To dává šanci i řešením s nižším ohodnocením a zároveň zajišťuje, že výhodnější jedinci mají větší šanci na přežití.

| jedinec | hodnocení |
|---|---|
| a | 10 |
| b | 5 |
| c | 3 |
| d | 1 |
| e | 1 |



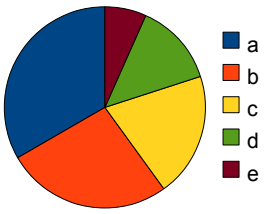

*Ilustrace 1:Klasická ruleta*

Ruleta je zobrazena do rozsahu *0..1,* ze kterého se poté náhodným výběrem určí jedinci pro křížení. Pro každého jedince je přitom proveden tento výběr znovu. Vylepšením této metody je pak *stochastic universal sampling,* která byla navrhnuta J. E. Bakerem v roce 1987. Namísto opakovaných průchodů při výběrech jedince, se vyberou všichni jedinci najednou. Počáteční výběr prvního řešení je vybrán stejně, další se ale vybírají v pravidelných fixních vzdálenostech. Tento výběr je znázorněn fialovými paprsky na grafu níže.




*Kresba 1: stochastic universal sampling*

Jiný problém se snaží řešit metoda rulety podle pořadí. Problém u původního řešení nastává ve chvíli, kdy je jedeno řešení výrazně úspěšnější než ostatní. Tento jedinec se pak stává nejvhodnějším kandidátem a další generace obsahuje téměř výhradně jeho potomky. Tato nepříjemná vlastnost se potlačuj použitím pořadí v populaci, namísto hodnotící funkce. Ta je použita pouze pro seřazení jedinců.




*Ilustrace 2: Ruleta podle pořadí*

Upravená ruleta s výběrem podle pořadí má své slabiny při podobném ohodnocení většiny jedinců v populaci nejsou v další generaci zastoupeni rovnoměrně.

### Turnajová selekce

Metoda turnajové selekce (*tournament selection*) je, stejně jako EA, inspirována přírodními ději. Stojí na myšlence vzájemného zápolení jedinců mezi sebou o přežití. Díky tomu mají i níže ohodnocená řešení šanci, že se díky dobrému výběru protivníků budou moci zúčastnit vytváření další generace.

Velikost skupiny, která je vybrána pro vzájemné porovnání, bývá nejčastěji 2. Vyberou se tedy dva jedinci a porovnají se na základě jejich ohodnocení. Úspěšnější jedinec je vybrán k další reprodukci.

Golderg a Deb podrobili tuto metodu výběru a přišli s vylepšeným algoritmem. Tato změna umožňuje s určitou pravděpodobností v turnaji vyhrát i horším jedincům. V přírodě se dá tato možnost přirovnat šťastné náhodě, která hraje ve prospěch méně zdatného jedince. Hodnota pravděpodobnosti muže být například 0.05, což by mělo v praxi dopad, že pouze v 95% případů vyhraje lepší jedinec. [1]

## 4.3 Mutace

Pod pojmem mutace se rozumí náhodná změna v chromozomu, která je dána určitou pravděpodobností.

Hodnota pravděpodobnosti je různá podle typu použité mutace (operátoru). Pokud je příliš nízká, nemusí populace dostávat dostatečné množství nových podnětů. Pokud je naopak příliš vysoká, působí nestabilně na vývoj jedinců. Oba dva případy prodlužují, až znemožňují vznik optimálního řešení.

Co přesně se při mutaci stane, určují *operátory*. Například při nejrozšířenějším zápisu chromozomu jako binární řetězec se jako jeden z operátorů nejčastěji používá bitová inverze. Ta invertuje všechny bity mezi dvěma náhodně zvolenými body v chromozomu. Volba operátoru se odvíjí od způsobu reprezentace chromozomu. Například u grafů lze použít změnu hodnoty uzlu, nebo prohození uzlů.

Nalezení vhodné pravděpodobnosti a operátorů mutace je tedy jednou z klíčových oblastí návrhu.

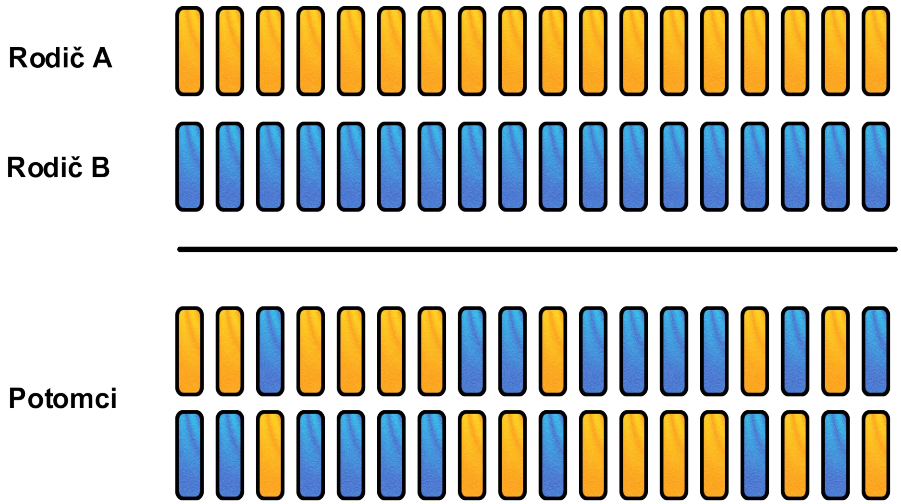
## 4.4 Křížení

Operátor křížení je základním stavebním kamenem pro GA. Na rozdíl od mutace, potřebuje ke své činnosti 2 jedince. Umožňuje potencionálně velkou výměnu chromozomů mezi úspěšnými jedinci: To by mělo vést k rychlejší konvergenci, za předpokladu že se výměny zúčastní souvislý blok úspěšných genů (tzv stavební blok) Pro některé problémy je ale vhodné zachovat řešení nedotčená křížením a umožnit jim tak přímou propagaci do další generace. Tento požadavek se

řeší podobně jako u mutace, křížení je aplikováno pouze s určitou pravděpodobností. Pokud je zvolen tento přístup, pak pravděpodobnost bývá mezi 75-95%. [1][2] Neexistuje žádná nejlepší obecná metoda pro křížení jedinců, každá aplikace potřebuje individuální posouzení. V praxi se začíná zkoušet nejprve uniformní křížení a dále se pokračuje bodovými kříženími.

## Uniformní křížení

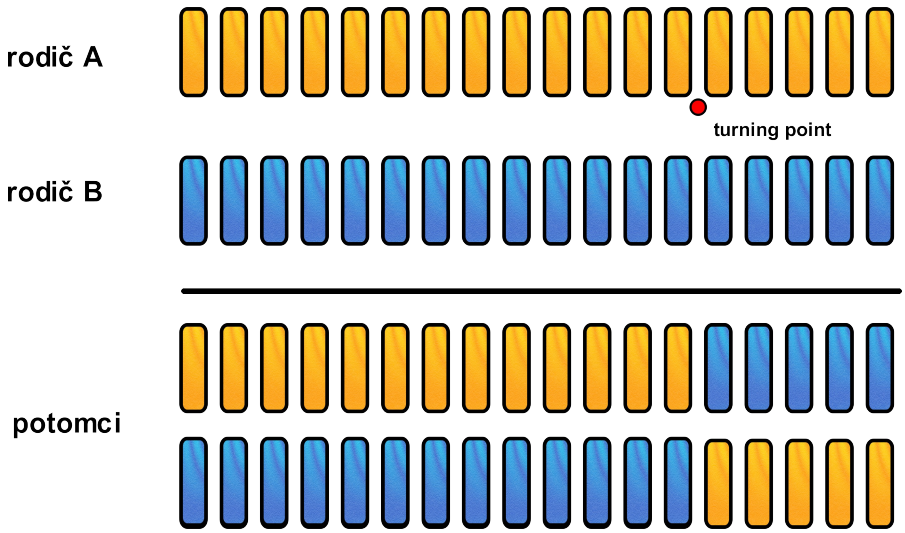
Uniformní křížení jeden nejčastěji používaných rekombinačních operátorů. Při aplikaci prochází genomy rodičů a s určitou pravděpodobností náhodně vybírá rodiče, jehož gen bude použit v potomkovi. Nevýhodou tohoto přístupu je poměrně velké nebezpečí rozvracení souvislých funkčních bloků. Naopak výhodou je, že aplikace toho algoritmu pomáhá při problémech s předčasnou konvergencí populace k lokálním extrémům. Dále je stavový prostor prohledáván mnohem intenzivněji, než u ostatních metod.




*Ilustrace 3: uniformní křížení*

## Jedno a více bodové křížení

Patří mezi nejjednodušší způsoby křížení. U jednobodové varianty je nejprve náhodně vybrán bod v rámci jednoho rodiče. Potomci pak zdědí část před bodem od jednoho rodiče, část po něm od rodiče druhého. Díky zachovávání souvislých celků genů, může být tato metoda výhodná u určitých typů problémů.

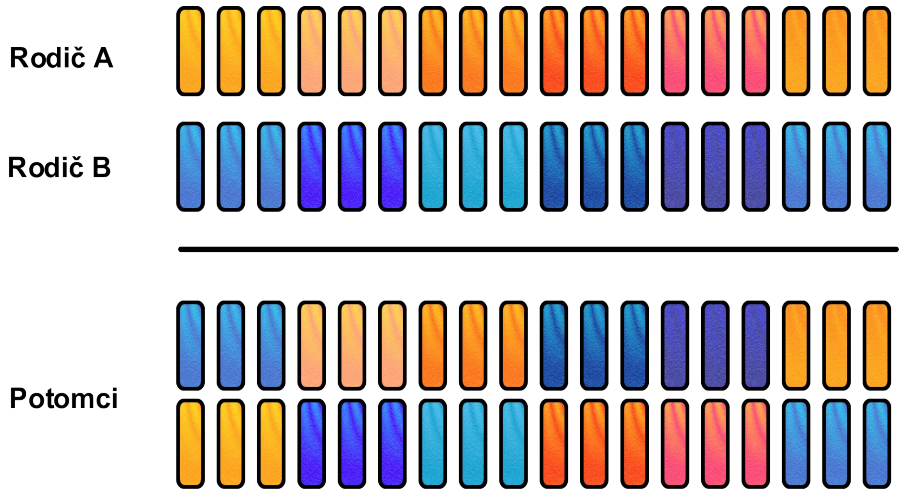


*Ilustrace 4: ukázka jednobodového křížení*

## Respektování stavebních bloků

Nevýhodou výš uvedených metod je jejich slepé chování ke stavebním blokům, tedy blokům genů, které úspěšně přispívají k řešení. Jednou z možností, která toto nabízí, jsou tzv. *hybridní genetické algoritmy.* [1] Ty využívají znalosti zkoumaného problému a jeho vnitřní reprezentace. V našem případě by se například jednalo o bloky genů reprezentujících součástky (viz reprezentace jedince v kapitole 6.2.)

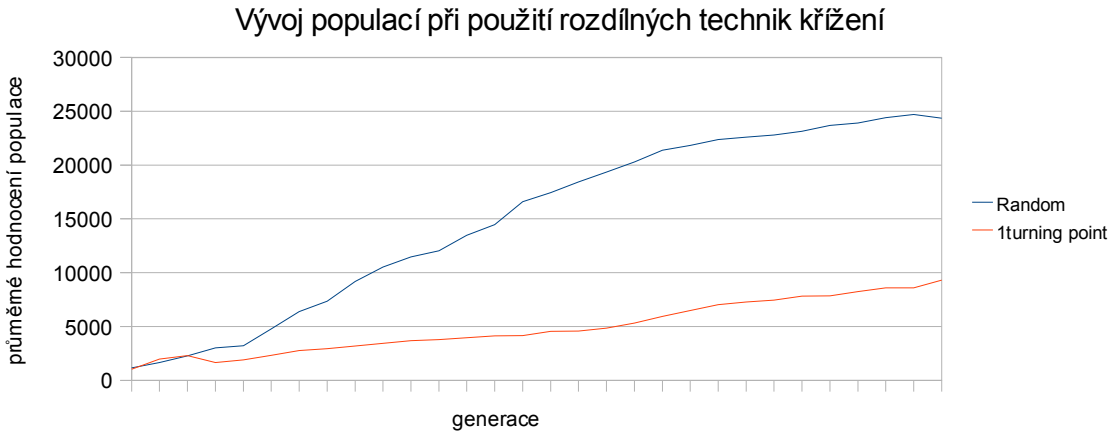
V takovém případě algoritmus zachovává jednotlivé logické celky genů, a podobně jako by při návrhu postupoval člověk, skládá nové jedince ze součástek původních. Nevýhodou tohoto přístupu je potřeba znalosti zkoumaného problému, a také možná nutnost zvýšení pravděpodobnosti mutace. Jinak by se nové části nevyvíjely, ale pouze by se kombinovali neexistující.




*Ilustrace 5: uniformní křížení s respektováním struktury*

### Srovnání

Pro potřeby srovnání algoritmů jsme implementovali dvě metody: uniformní (random) a jednobodové křížení (1turning point). Každá křivka znázorňuje zprůměrované průměrné hodnocení 30 populací. Každá populace obsahovala 20 jedinců a vyvíjela se po 30 generací. Parametry simulace byly pro všechny testy stejné, lišila se jenom použitá metoda křížení.




*Ilustrace 6: srovnání metod křížení*

## 4.5 Obnova populace

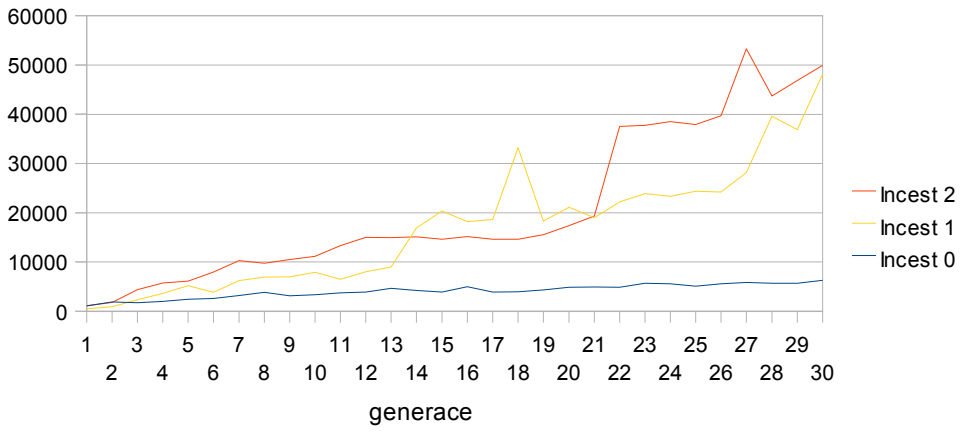
### Úplná obnova

Nejzákladnější a nejjednodušší metodou pro vytvoření nové generace jedinců je její úplná obnova. Všichni jedinci v populaci jsou nahrazeni novými potomky. Tato metoda neumožňuje jedinci ovlivňovat vývoj déle než jednu generaci. Zároveň hrozí reálné riziko, že dobře ohodnocené řešení se při přerodu do nové generace ztratí, ať už díky náhodě při výběru, nebo později při mutaci.

## Elitismus

Jednou z metod, řešících neduhy jako je ztráta úspěšných jedinců, je tzv. elitismus. Pokud je použit, tak nedochází k úplnému přegenerování nové populace. Místo toho se zachová předem stanovený počet nejlepších jedinců z aktuální generace. Díky tomu, že se pak v populaci nacházejí jak původní rodičové, tak jejich potomci, je někdy tento postup označován jako incest. Počet zachovaných jedinců je většinou malý, v porovnání s velikostí populace.

Na této ilustraci je vidět, jak zachování nejlepších jedinců přispívá ke zlepšení hodnocení celé populace.




*Ilustrace 7: Srovnání populací při incestu*

## Setrvalý stav

Podobnou metodou je i tzv *steady-state reproduction* (setrvalý stav.) Liší se pouze v počtu přenášených jedinců. U metody setrvalého stavu se používá větší množství zachovaných jedinců a střídá se pouze zlomek populace. Díky tomu se zachová velká část předcházející populace a noví jedinci jí pouze pomalu vylepšují.

## Další vylepšení

V případě větší potřeby zachovávat aktuální výhodnou populaci se používá následující úprava. Místo toho, aby noví jedinci měli vždy zajištěné místo v populaci, musí o něj soupeřit s nejhorším jedincem v současné populaci. Teprve pokud mají hodnocení lepší, jsou do populace zahrnuti a původní jedinec odstraněn.

V některých případech je vhodné zachovávat rozmanitost populace a tím mít více typů kandidátních řešení. Z tohoto hlediska je kontraproduktivní zachovávat jedince s identickým genotypem. Jejich objevení v populaci se zabraňuje při selekci. Pokud má nový jedinec stejný (nebo podobný) chromozom jako jedinec současný, zachová se pouze ten s lepším hodnocením. Horší jedinec uvolní místo pro jiné řešení.

# 5 Problémy evolučních algoritmů

## 5.1 Reprezentace genu

Při reprezentaci genu je třeba mít na paměti, jaké genetické operátory budou použity a zda je zvoleno vhodné kódování. Příkladem nevhodného kódování může být například binární zápis.

| binární zápis | dekadická hodnota | rozdíl hodnot | bitový rozdíl |
|---|---|---|---|
| 0 1 1 1 | 7 | | |
| 1 0 0 0 | 8 | 1 | 4 |
| 0 0 0 0 | 0 | 8 | 1 |

Na této tabulce je vidět, že vzdálenost binárního zápisu nereflektuje dekódovanou dekadickou hodnotu. Pokud by tedy byl pro mutaci použit operátor bitové inverze, pak nové hodnoty můžou být v závislosti na ovlivněném bitu téměř stejné, či zásadně odlišené.

Jedním z možných řešení tohoto problému je například použití Grayova kódu. Ten zakóduje čísla tak, aby mezi sousedícími hodnotami byl změněn vždy pouze jeden bit. Ukázka kódování na 3 bitech je uvedena níže.

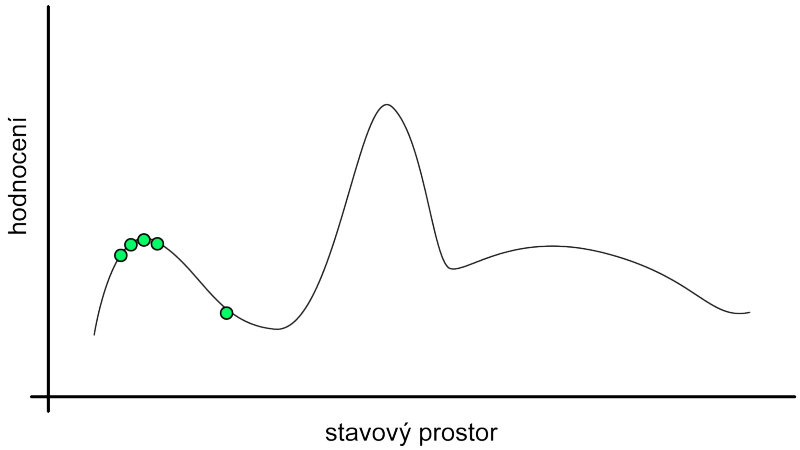
| Hodnota | Binární kód | Grayův kód |
|---|---|---|
| 0 | 000 | 000 |
| 1 | 001 | 001 |
| 2 | 010 | 011 |
| 3 | 011 | 010 |
| 4 | 100 | 110 |
| 5 | 101 | 111 |
| 6 | 110 | 101 |
| 7 | 111 | 100 |

## 5.2 Uváznutí na lokálním maximu

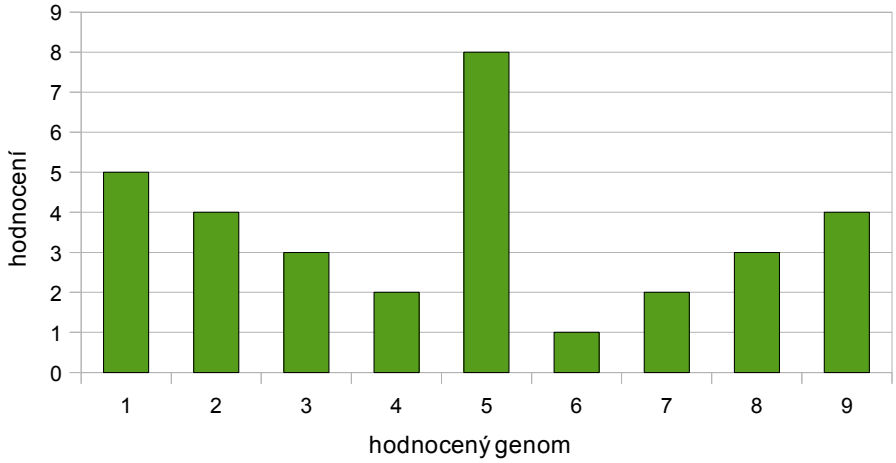
Při prohledávání stavového prostoru, vykazuje většinou fitness funkce nelineární chování. Díky tomu vznikají lokální maxima, jejichž příklad je znázorněn na ilustraci 8. Tomuto chování se dá předejít například zachováváním rozmanitosti populace metodami popsanými v kapitole 4.5. Další možností je paralelní vývoj několik nezávislých populací a jejich následné sloučení po určitém počtu generací.

## 5.3 Klamná fitness funkce

Fitness funkce je jediným ukazatelem pro výběr nových jedinců. Pokud zkoumaný stavový prostor obsahuj izolované lokální maximum, obklopené podprůměrnými řešeními. V tom případě algoritmus s největší pravděpodobností skončí na lokálních maximech, než aby dosáhl maxima globálního. Existují práce, které se věnují detekci klamných funkcí, včetně metod předcházení jejich vytvoření. [1] Ilustrace 9 znázorňuje případ klamné funkce.




*Ilustrace 8: Lokální maximum*




*Ilustrace 9: klamná funkce*

## 5.4 Problém implementace

Posledním problémem, který zde zmíníme, přímo nesouvisí s evolučními algoritmy. Při testování je třeba brát zřetel na to, že evoluční algoritmy jsou stejně úspěšné v hledání řešení, jako v hledání chyb, které jim k tomuto řešení pomohou. Před samotnou evolucí finálního řešení je vhodné spustit větší množství menších testů pro ověření zamýšleného chování.

# 6 Implementace

## 6.1 Analýza problému

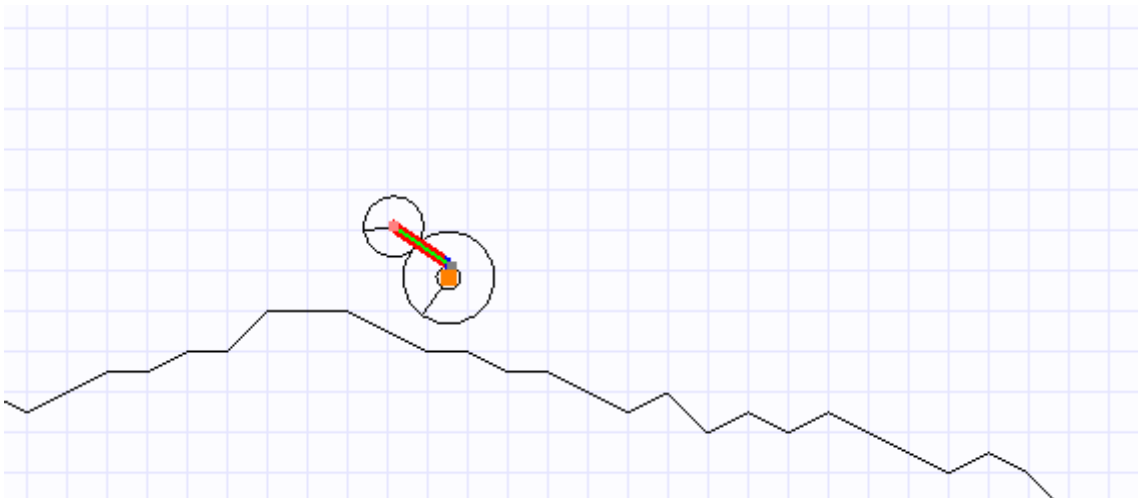
Úkolem této práce bylo navrhnout a naprogramovat agenta pro pohyb ve 2D či 3D terénu. Vzhledem k jednodušší implementaci 2D terénu a vlastním zkušenostem s fyzikální knihovnou, bylo rozhodnuto soustředit se tímto směrem.

Příklady 2d primitiv jsou například kruh, úsečka, čtverec, trojúhelník. Pro reprezentaci agenta bylo rozhodnuto použít pouze první tři, přímo podporované fyzikální knihovnou. Kruh a čtverec (dále jen součástky), se dají definovat následujícími vlastnostmi: výška, šířka, pozice v prostoru, úhel otočení. Pro potřeby fyziky dále svou hmotností a vlastnostmi povrchu. Úhel otočení a vlastnosti povrchu jsme se rozhodli pro zjednodušení zanedbat. Hmotnost je dále vypočítávána z rozměrů součástky. Úsečka je použita jako znázornění spojů, mezi jednotlivými součástkami. Pro její určení je potřeba znát kam a na které součástky se připojit, typ a parametry spoje. Jednotlivé typy spojů jsou uvedeny v kapitole 6.3.

Po dohodě s vedoucím, byla do práce zahrnuta i evoluce inteligence. Po prototypech inteligencí se byla vybrána podmnožina možných podmínek a způsob jejich reprezentace. O způsobu řízení jedince je více zmíněno v kapitole 6.4.

Pro zajištění správné funkčnosti je potřeba zajistit stejné podmínky vyhodnocení pro každého jedince. Aby toho bylo dosaženo, je třeba zajistit stejný terén. Ten ale není možné definovat dopředu, protože není dán horní limit, kam až se může jedinec dostat. Krajina je tedy funkcionálně generována. Funkce je zvolena tak, aby se postupně ztěžovala obtížnost terénu. Pro zajištění shodného terénu se krajina generuje do mřížky, která je znázorněna na ilustraci 10.

Knihovna použité jako základ pro genetickou část aplikaci je rozebrána ve vlastní kapitole. Knihovna pyMunk použitá pro simulaci fyziky není z pohledu problému zajímavá a není dále rozebírána.



*Ilustrace 10: mřížka pro generování terénu s jedincem po 24. generaci*

## 6.2 Knihovna pyGene

Při implementaci byla použita knihovna pyGene od Davida McNaba. Jedná se o knihovnu určenou pro skriptovací jazyk Python a usnadňující psaní skriptů využívajících evoluční metody. Knihovna byla pro naše potřeby rozšířena o další metody selekce a křížení.

### Populace

Populace je reprezentována vlastní třídou. Stará se o selekci řazení jedinců a veškeré další operace nad všemi jedinci v populaci. Selekce je implementována s podporou obnovy populace, která je reprezentována parametrem *incest*. Tento parametr určuje, kolik nejlepších jedinců z populace zůstane zachováno. Pokud se jeho hodnota blíží velikosti populace, jedná se o stav *steady-state,* pokud se naopak blíží k nule, jedná se o *elitismus*. Další z možných parametrů umožňuje volitelně vložit do populace nového jedince získaného stejným postupem jako při inicializaci populace. Další parametry umožňují nastavit počáteční velikost populace, počet nově vygenerovaných potomků a finální oříznutí. Význam těchto hodnot je uveden níže.

Při inicializaci třída vygeneruje náhodné nové jedince, tak, aby dosáhla stanoveného počtu jedinců v populaci. Každý jedinec je ohodnocen. Následně se jedinci seřadí podle jejich ohodnocení a provede se křížení. Způsob výběrů může být programátorem libovolně upraven. Způsob křížení nechává na třídě organismu. Křížení probíhá, dokud není vytvořen počet potomků daný parametrem. Za předpokladu, že se nejedná o první generaci, je k těmto nově vygenerovaným jedincům přidán nastavený počet nejlepších jedinců z generace předchozí. Počet těchto jedinců určuje hodnota parametru *incest*. Volitelně je také přidán nově vygenerovaný jedinec. Následně je celá nová populace seřazena podle přepočítaného hodnocení a oříznuta na počet daný parametrem oříznutí. Tato výsledná množina jedinců je pak prohlášena za novou generaci.

### Organismus

Jednotlivý jedinci jsou reprezentováni třídou organismus. Organismus se skládá z genů a zpřístupňuje metody pro ohodnocení jedince a jeho uložení. Způsob ohodnocení je rozepsán ve vlastní kapitole.

### Geny

Knihovna pyGene obsahuje základní geny pro práci s celočíselnými i reálnými typy čísel. Dále pro práci s ASCII znaky a logické hodnoty. Pro každý gen je možno nastavit jeho vlastní pravděpodobnost mutace. Problému se zakódováním informace popsaným v kapitole 5 se vyhýbá mutací dekódovaných hodnot. Gen je nejprve dekódován, pak je zjištěno v jaké míře se může změnit a z tohoto rozsahu je vybrána konečná hodnota. S tím souvisí parametry genu, lze mu nastavit pravděpodobnost mutace, minimální a maximální meze a největší povolenou míru mutace.

## 6.3 Struktura jedince

Struktura jedince byla původně navržena pro reprezentaci pomocí grafu, kde by uzly

představovaly součástky a hrany jednotlivé spoje. Kvůli implementačním problémům byl tento návrh přehodnocen a struktura se celá kóduje do lineárního řetězce.

Každá součástka se skládá z pěti genů. Tyto označují následující vlastnosti: typ součástky (kruh, obdélník), šířku, výšku a relativní posun na obou osách. Pro potřeby simulace se z rozměrů součástky vypočítává ještě hmotnost. Pokud se jedná o kruh, je použit jenom parametr pro výšku. Na základě rozměrů součástky se také vypočítává maximální síla, pokud bude součástka aktivována jako motor.

Z praktických důvodů může součástka také nabývat typu *žádná*. Díky tomu mohou existovat pasivní (nepoužité) geny, a získá se tak větší variabilita jedinců.

Pro spojení součástek slouží spoje. Ty jsou zakódovány obdobně jako součástky, jsou ale definovány osmi parametry. První určuje typ spoje, dále označení spojovaných součástek a jejich bodů připojení a nakonec tři parametry pro spoj. Implementovali jsme větší množství typů spojů:

1. *pevný bod* - spojí přípojné body napevno k sobě a snaží se mezi nimi udržet nulovou vzdálenost)
2. *příčka* - udržuje přípojené ve stejné vzdálenosti jako na začátku simulace
3. *s vůli* - ve své podstatě příčka, která udržuje vzdálenost součástek mezi maximální a minimální hodnotou
4. *tlumený* - tlumený spoj funguje jako pružina, která se, podobně jako příčka, snaží udržet součástky ve stejné vzdálenosti

Nevýhoda těchto spojů nastává v okamžiku, kdy příčky vytvoří kolizní stav. Například se pokusí vytvořit 2 pevné body na součástkách, které to fyzicky neumožňují. To vede k nestabilnímu stavu simulace. Řešení tohoto problému je znázorněno v pseudokódu níže:

```
vlož všechny součástky
vlož všechny spoje
opakuj:
   proveď krok simulace
   smazanSpoj := NEPRAVDA
   pro každý spoj z množiny všech spojů:
      zkontroluj zda je spoj namáhán nadlimitní silou:
           odstraň ho ze simulace
           odstraň spoj z množiny spojů
           smazanSpoj := PRAVDA
dokud velikost( spoje v simulaci ) > 1 A smazanSpoj = PRAVDA
```

Vhodná limitní hodnota síly byla určena experimentálně.

Jednotlivé typy spojů jsou barevně rozlišeni pro snazší identifikaci. Zajímavým zjištěním je, že jedinci konvergují k používání tlumených spojů i přesto, že v prvotní náhodné populaci je

rozdělení rovnoměrné.

Rozmezí velikosti jedince, stejně jako maximální počet spojů je programově nastavitelné. Podle horních mezí se pak odvíjí velikost genomu.

## 6.4 Řídící funkce jedince

Řízení jedince bylo původně zamýšleno implementovat jako rozhodovací strom. Pro sjednocení metody kódování bylo od grafu upuštěno a používá se lineární zápis podmínek. Jeden rozhodovací uzel potřebuje následujících šest genů pro svojí reprezentaci:

1. *negace* - určuje zda bude výsledek podmínky negován, či ne
2. *typ* - typ podmínky
3. *parametr A* - celočíselná hodnota, označuje buď součástku,
4. *parametr B* nebo hodnotu pro porovnání
5. *akce* - označení akce která se má vykonat, implementováno pouze zapnutí motoru
6. *parametr akce*

Poslední gen, parametr akce není využit a poskytuje pouze místo pro rozšíření. Počet možných vygenerovaných podmínek je díky možnostem přidávat logické operace nad nimi značný.

Podmínky:

1. *AND* - *podmínka je pravdivá, pokud jsou podmínky A a B pravdivé*
2. *OR* - *podmínka je pravdivá, pokud je alespoň jedna z podmínek A a B pravdivá*
3. *věštec* - *jedinec porovná rozdíl výšky terénu A pixelů před sebou se svojí aktuální výškou a výsledek se porovná s hodnotou B. Pokud je menší je podmínka pravdivá*
4. *vlevo* - *pokud je součástka A vlevo od součástky B, je podmínka pravdivá*
5. *nahoře* - *pokud je součástka A výš než součástka B, je podmínka pravdivá*
6. *pravda* - *podmín*ka je vždy pravdivá

Podmínky jsou vyhodnocovány při každém kroku simulace. Ze získaných výsledků není patrné, že by rozvinutější struktury řízení měly nutně lepší výsledky.

## 6.5 Použité prostředky

Pro demonstrační účely byly implementováno celkem tři metody selekce. Ruletový mechanismus, dále jeho modifikaci s pořadím a nakonec mechanismus implementovaný knihovnou pyGene.

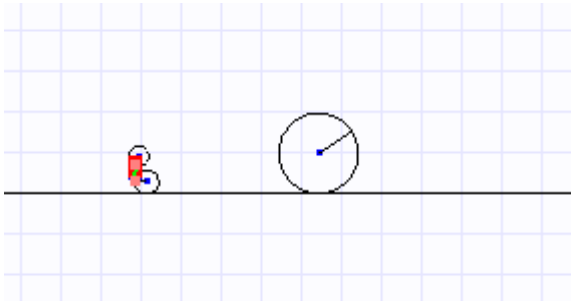
Pro křížení bylo použito křížení uniformního, dále pak turning point a metody s respektováním stavebních bloků.

Díky problémům s výpočetní náročností jsme nebyli schopni získat relevantní data pro demonstraci úspěšnosti jednotlivých metod. Jedinou vyjímkou je porovnání uniformního a 1turning point křížení.

## 6.6 Hodnocení

Finální fitness funkce odráží původní myšlenku, vygenerovat složitého agenta, který si na terénu povede co nejlépe. Ohodnocení je získáno sečtením ujeté vzdálenosti všech součástek. Tímto je dosaženo obou cílů, největší hodnocení mají jedinci, kteří dopraví co největší počet součástek co nejdále.

Pro zlepšení výběru a urychlení algoritmu byly dále přidány doplňující podmínky. Pokud se jedinec dostane pod úroveň terénu, je mu přiděleno nejmenší možné ohodnocení a tím je efektivně vyřazen z další generace. Obdobně, pokud se součástky dostanou příliš daleko od sebe, což značí, že s největší pravděpodobností zůstaly ležet a nejsou připojeny, je jedinec ohodnocen stejně. Původním cílem bylo zamezit oddělování součástek od jedinců. Díky možnostem řídících funkcí, se ale začal objevovat vzor chování, kdy je jedinec rozdělen na 2 části (viz ilustrace 11), které na sebe navzájem čekají. Tím se nikdy nesplní podmínka velké vzdálenosti a jedinec přežije. Jako poslední je přidána kontrola pohybu vpřed. Pokud se jedince déle než dvě vteřiny nepohne kupředu, je jeho vyhodnocování ukončeno s aktuálně dosaženým hodnocením



*Ilustrace 11: jedinec obcházející pravidla*

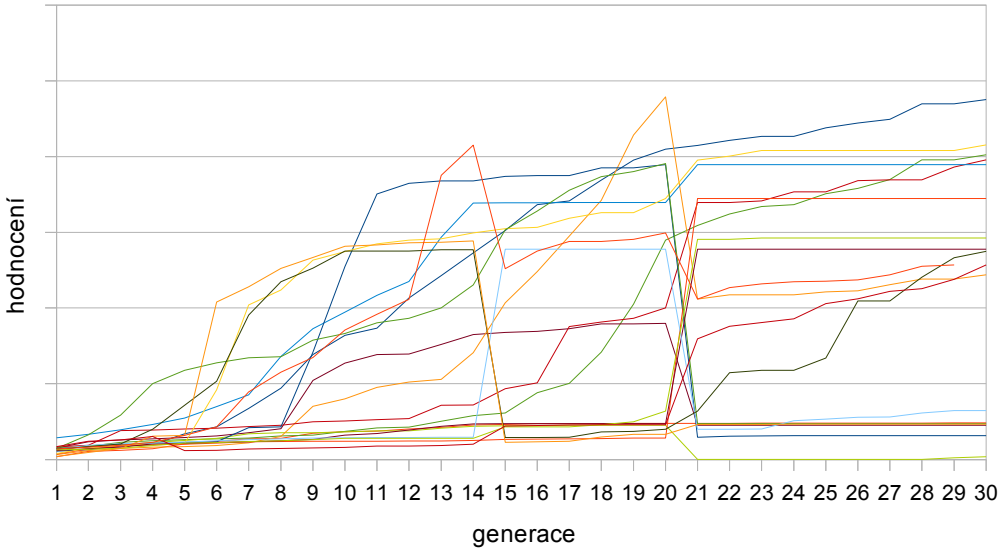
## 6.7 Problémy při implementaci

Při vytváření programu jsem narazil na několik problémů, které ale nesouvisely přímo s genetikou. Jedním z problémů je nedeterministické chování použité fyzikální knihovny zapříčiněné zřejmě používáním čísel s plovoucí desetinou čárkou. Další problémy souvisí s agenty, kteří našli skuliny ve vyřazovacích pravidlech a podařilo se jim využít chyby a aproximací (např. u kolizí) v simulaci. Tento problém je obzvlášť citelný, protože jeho řešení si vyžádalo zvýšení výpočetních nároků. Po každé změně se zároveň zneplatnila data nasbíraná za předchozí verze programu.

## 6.8 Vyhodnocení výsledků

Za pomoci časové komprese a možnosti spouštět simulaci v dávkovém režimu jsme získali množství dat. Z důvodů popsaných v předchozí kapitole ale nemáme možnost jak je porovnávat.

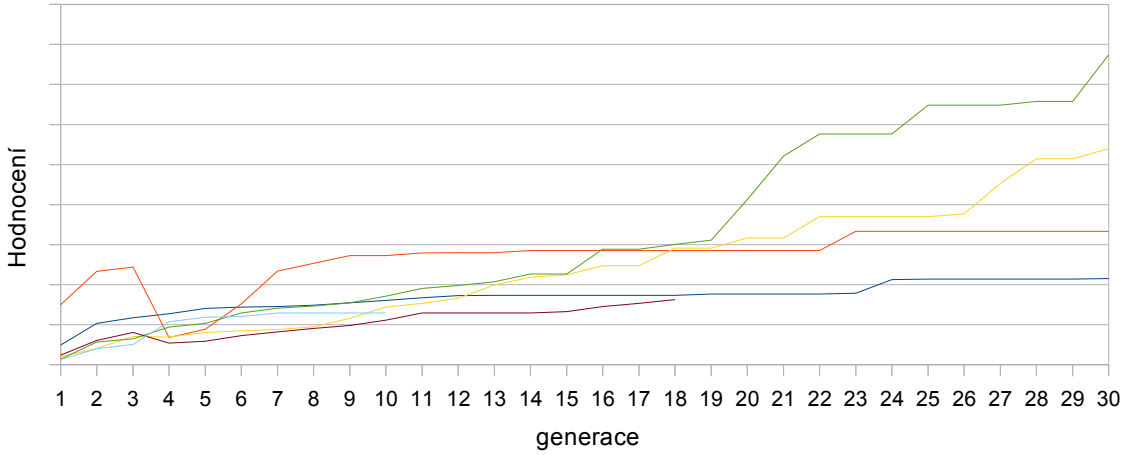
Navzdory tomu dokážou alespoň tato data demonstrovat úspěšnost použitých postupů.




*Ilustrace 12: maximální hodnocení – vzorek z testování uniformního křížení*

Křivky na dolní straně grafu nestagnují, pouze jsou jejich změny příliš malé na vynesení.

Následující graf ukazuje, že algoritmus 1 turning point konverguje pomaleji, ale pořád si zachovává stoupavou tendenci. To znamená, že křížení a výběr se chovají korektně.




*Ilustrace 13: maximální hodnocení - turning point*

# 7 Možnosti programu

## 7.1 Ovládání

Program byl během vývoje rozšiřován o parametry umožňující ovlivnění chování evoluce. Význam jednotlivých parametrů je popsán v předchozích kapitolách. Zde se nachází pouze seznam spolu s krátkým popisem jejich chování.

Simulaci lze spustit ve dvou režimech. První slouží pro evoluci populace, druhý pak pro zobrazení uložených jedinců.

Níže následuje seznam parametrů, které je možné použít jako argumenty programu. Pokud není uvedeno jinak, označuje `bool` číselnou hodnotu, která je převedena na `TRUE`, pokud je nenulová, v opačném případě na `FALSE`.

`<soubor_s_agentem>`

pokud je simulace spuštěna takto, nesmí obsahovat žádné přídavné přepínače, místo spuštění evoluce je zobrazen jedinec, který je načten ze souboru *soubor_s_agentem*, při tomto režimu nefungují vyřazovací podmínky popsané v kapitole 6.6.

`-c` *`číslo`* `;` `--cas` *`číslo`*

*`Číslo`* může nabývat reálných hodnot, kde *1.0* značí normální běh času, čísla větší pak odpovídající zrychlení, menší zpomalení, není doporučeno používat příliš vysoká čísla s ohledem na možnou nestabilitu fyzikálního systému

`-g` *`bool`* `;` `--grafika` *`bool`*

Kde *`bool`* označuje, zda se má použít okno pro zobrazení jedince, či zda má aplikace běžet v bezokenním režimu, pravdivá hodnota okno povolí,nepravdivá zakáže

`--ukladejNejlepsi` *`bool`*

dafaultně zapnuto, nastavuje zda se mají ukládat nejlepší agenti do souborů, Číslo může reálných hodnot, kde *1.0* značí normální běh času, čísla větší pak odpovídající zrychlení, menší zpomalení, není doporučeno používat příliš vysoká čísla s ohledem na možnou nestabilitu fyzikálního systému

`--typagenta` *`trida`*

umožňuje použít vlastní třídu pro typ jedince, vhodné pro jinou implementaci algoritmů křžení

`--incest` *`číslo`*

nastaví hodnotu incest na příslušnou hodnotu, viz kapitola 4.5 Obnova populace

`--maxGeneraci` *`číslo`*

zastaví evoluci po daném počtu generací

`-l` *soubor* ; `--log` *soubor*

nastaví cestu k logovacímu souboru, soubor obsahuje středníkem oddělené číslo generace, hodnocení nejlepšího jedince a průměrné hodnocení populace

`--strukturaMax` *číslo*; `--strukturaMin` *číslo*

nastaví odpovídající konstantu určující velikost struktury na příslušnou hodnotu

`--casSpani` *číslo*

pokud je použit, bude aplikace pasivně čekat počet milisekund udaný číslem při každém kroku simulace, vhodné pro snížení zátěže generované simulací

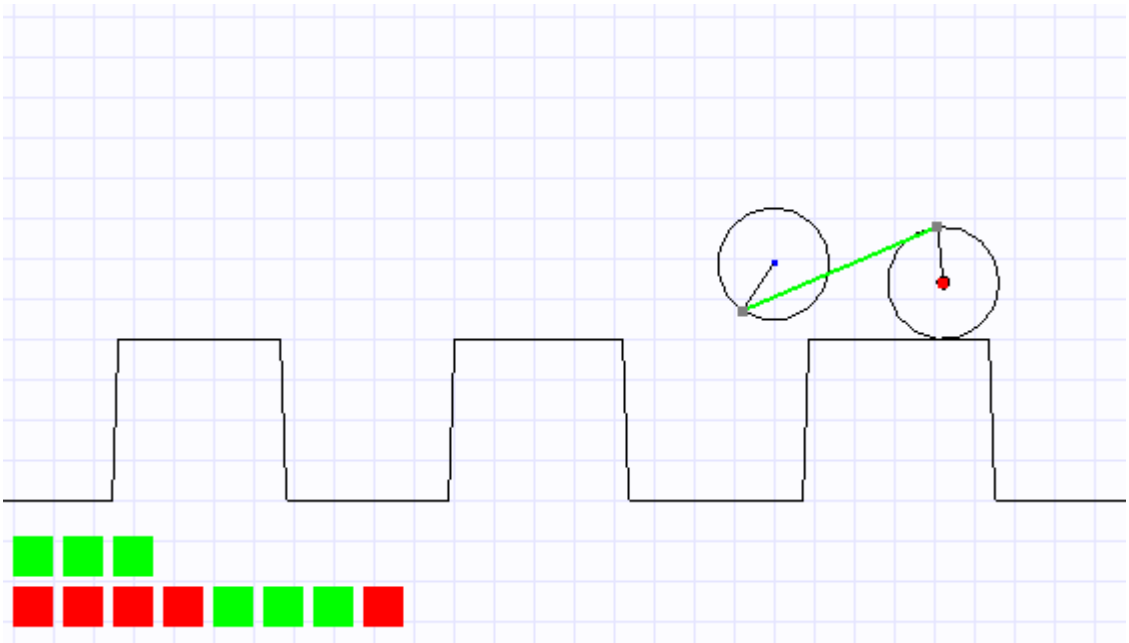
`--populace` *soubor*

umožňuje uložit/znovunačíst stav populace, pokud je použit, pak se stav aktuální generace ukládá do souboru, při opětovném spuštění je tak pokračováno v předchozí generaci

Pro ukončení běhu aplikace lze v okenním režimu použít křížek okna, či jakoukoli ekvivalentní metodu pro jeho zavření. V módu bez grafiky pak přerušení pomocí break signálu (na Win `ctrl+c`), který je aplikací ošetřen.

## 7.2 Vzhled

Pokud je program spuštěn v grafickém režimu (defaultní chování), zobrazuje vždy aktuálně testovaného jedince. V levém dolním rohu je informační lišta, které informuje o počtu a stavu součástek (horní ukazatel) a lišta se stavy řídících uzlů . Zelená znamená aktivovaný motor/sepnutý uzel, červená značí opak. Pokud je grafika spuštěná v evolučním režimu, zobrazuje se navíc nad agentem modrý čtvereček, který znázorňuje hranici, kterou musí překonat do určitého času, než bude vyřazen (viz kapitola 6.6 Hodnocení).



*Ilustrace 14: ukazatele motorů a uzlů řízení*

# 8 Závěr

První bod zadání se mi podařilo splnit a poznatky pocházející z prostudování uvedených materiálů, dostupných především ve fakultní knihovně a za pomoci internetu, jsou shrnuty v kapitolách 2 až 5.

Návrh reprezentace terénu je je detailněji popsán v kapitole 6.1. V kapitolách navazujících je dále rozepsána jak struktura agenta tak popis jeho chování. Oproti semestrální práci zde došlo k přechodu od popisu agenta grafem na lineární zápis. Důvody jsou zmíněny v příslušných kapitolách.

Detailní popis samotné implementace se nachází v kapitole 6. aplikace. Zhodnocení jejích výstupů je pak uvedeno v závěru stejné kapitoly.

Všechny body zadání se podařilo úspěšně vyřešit. Aplikace je schopna vyvinout agenty, kteří schopni pohybu po vygenerovaných krajinách. Žádný z vygenerovaných agentů nebyl lepší, než agent navržený člověkem. Tento jev lze přičítat dvěma faktorům. Za prvé, nebyl k dispozici dostatečný výpočetní výkon pro vývoj populace déle než po 50 generací. Za druhé, zkoumaný prostor je příliš velikým, než aby hledání rychle konvergovalo k použitelným výsledkům. Návrh řešení tohoto problému je uveden v druhé části této kapitoly. Generovaní agenti obsahují většinou konvenční stavební postupy. Vyjímky jsou přiloženy spolu s programem na CD.

Program svým konceptem a zvoleným skriptovacím jazykem poskytuje velký prostor pro případná vylepšení. Je tak například i díky tomu, že je snadno přístupná funkce pro generování terénu nebo pro ohodnocování agentů. Z hlediska implementačních vylepšení by bylo dobré použít jinou fyzikální knihovnu. Současně použitá vykazuje v řídkých případech nedeterministické chování, což je velký problém pro ohodnocování jedinců. Dalším z možných vylepšení je podpora paralelismu, která by efektivně umožnila řádově zrychlit ohodnocování jedné populace. Krokem, který by výrazně ovlivnil vliv rozhodovacích uzlů na ohodnocení agenta, by bylo rozšíření efektorů místo jediného – motoru.

Během vývoje programu bylo díky své implementační složitosti, nebo pro zjednodušení výsledné implementace vypuštěno několik ideí, které jsou spolu s dalšími, které vyplynuly až z používání, uvedeny dále.

Stavový prostor prohledávaný algoritmem je příliš veliký pro rychlý vývoj agentů. Pokud by se uživatelům umožnilo zamknout určité části agentů, které by podle nich měly být výhodné. Tyto by pak nepodléhaly dělení při křížení ani mutaci. V krajních případech by se tak dala zamknout celá struktura jedince a nechat vyvíjet jednom inteligence, či naopak.

Vylepšení, které by mohlo za určitých podmínek přinést zajímavé výsledky, by bylo rozšíření typů součástek, například n-úhelníky, elipsa.

Jako poslední ze seznamu možných vylepšení je pak možnost uživatelského definování agentů. Tato změna by šla využít například pro vytvoření lepší počáteční populace, vytvoření lepších stavebních bloků a v důsledku zrychlení konvergence algoritmu.

# 9 Zdroje


[1] HYNEK, Josef. *Genetické algoritmy a genetické programování*. Praha : Grada Publishing, 2008. 168 s.

[2] SCHWARZ, Josef; SEKANINA, Lukáš. *Aplikované evoluční algoritmy : Studijní opora* [online]. 1. 2006 [cit. 2010-05-14]. Dostupné z WWW: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/EVO-IT/texts/EVO_opora_2006_ESF.pdf>.

[3] JOHNSTON, Victor. *U.S. patents* [online]. 1994 [cit. 2010-05-04]. Method and apparatus for generating composites of human faces. Dostupné z WWW: <http://www.patentstorm.us/patents/5375195/description.html>.

[4] YOUNG, Matt; STRODE, Paul. *Why Evolution Works : (and Creationism Fails)*. New Brunswick, N.J. : Rutgers University Press, 2009. 224 s.

[5] EDWARDS, Mark. *A Critique of PBS's Evolution* [online]. 2001 [cit. 2010-04-20]. "Evolution" Series Claim on Eye Evolution More Fiction than Fact. Dostupné z WWW: <http://www.reviewevolution.com/press/pressRelease_EyeEvolution.php>

[6] PFEIFER, Rolf; SCHEIER, Christian. *Understanding Intelligence*. Cambridge, Mass. [u.a.] : MIT Press, 2001. 697 s.

[7] *Evolvable systems group* [online]. 2006 [cit. 2010-04-21]. Automated Antenna Design. Dostupné z WWW: <http://ti.arc.nasa.gov/projects/esg/research/antenna.htm>.

[8] *Fakulta informačních technologií VUT Brno* [online]. 2010 [cit. 2010-05-10]. Výzkumná skupina Evolvable Hardware. Dostupné z WWW: <http://www.fit.vutbr.cz/research/groups/ehw/index.php>.

[9] *PyGene* [online]. 2010 [cit. 2010-05-11]. PyGene. Dostupné z WWW: <http://www.freenet.org.nz/python/pygene/>.

[10] *PyMunk* [online]. 2010 [cit. 2010-05-11]. PyMunk. Dostupné z WWW: <http://code.google.com/p/pymunk/>.

[11] BANZHAF, Wolfgang. *Genetic programming : an introduction ; on the automatic evolution of computer programs and its applications*. San Francisco, Calif : Kaufmann, 1999. 470 s. ]

[12] *Genetic programming* [online]. 2003 [cit. 2010-05-14]. 36 Human-Competitive Results Produced by Genetic Programming. Dostupné z WWW: <http://www.genetic-programming.com/humancompetitive.html>

[13] HEITKÖTTER, Jörg. *The Hitch-Hiker's Guide to Evolutionary Computation* [online]. 2001 [cit. 2010-05-14]. What's Evolutionary Programming (EP)?. Dostupné z WWW: <http://www.aip.de/~ast/EvolCompFAQ/Q1_2.htm>.