UNIVERZITA PARDUBICE

FAKULTA ELEKROTECHNIKY A INFORMATIKY

Genetické algoritmy

(vliv parametrů na jejich chování)

Pavel Rypien

Bakalářská práce

2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Pavel RYPIEN**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Genetické algoritmy - vliv parametrů na jejich chování**

Zásady pro vypracování:

Cílem práce je zhodnotit vliv parametrů genetických algoritmů na rychlost a výsledky při řešení optimalizačních úloh. Teoretická část se bude zabývat teorií kolem genetických algoritmů (GA), praktické části budou naprogramovány různé varianty GA, u kterých se bude posuzovat vliv volby struktury a parametrů na výkonnost a spolehlivost GA.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

MAŘÍK V., ŠTĚPÁNKOVÁ O., LAŽANSKÝ J., Umělá inteligence (3), 1. vydání - dotisk, Praha : Academia, 2004. s. 117- 160. ISBN 80-200-0472-6.

Vedoucí bakalářské práce: **Ing. Pavel Škrabánek**
Katedra řízení procesů

Datum zadání bakalářské práce: **15. ledna 2009**

Termín odevzdání bakalářské práce: **15. května 2009**

Univerzita Pardubice Česká republika

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan
vedoucí katedry

V Pardubicích dne 31. března 2009

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 14.8.09

Pavel Rypien

PODĚKOVÁNÍ

Na tomto místě bych chtěl poděkovat panu Ing. Pavlu Škrabánkovi za jeho shovívavost a pomoc při zpracování práce.

Také bych chtěl poděkovat své rodině a přátelům, kteří mě podporovali po celou dobu studia.

Anotace

Bakalářská práce se zabývá genetickými algoritmy a vlivy parametru na jejich chování. Algoritmus je implementován do prostředí Matlab, kde jsou ukázány jednotlivé vlivy parametrů.

Klíčová slova

Genetické algoritmy, Matlab, Problém Obchodního Cestujícího

Title

Genetic algorithms - parameters effects on their behavior

Anotation

The main objective is to describe genetic algorithms and the effects of parameters on their behavior. Algorithm is implemented on to Matlab where the individual effects of parameters are shown.

Key words

Genetic algorithms, Matlab, Traveling Salesman Problem

Obsah

Seznam obrázků

Seznam grafů

Seznam Tabulek

Seznam kódů

Úvod

Při hledání skutečných kořenů, z nichž vyrůstá celá oblast evolučních algoritmů, je třeba se vrátit zpět až do roku 1859, kdy Charles Darwin poprvé publikoval svoji proslulou knihu *O vzniku druhů přirozeným výběrem čili zachováním vhodných odrůd v boji o život*. Darwinova myšlenka, že populace živočichů a rostlin se vyvíjela po mnoho generací podle principu přirozeného výběru a přežití těch nejschopnějších, inspirovala po více než jednom století velice slibně se rozvíjející oblast výzkumu, jehož cílem je tyto přírodní procesy napodobit.[2]

Tato bakalářská práce se věnuje genetickým algoritmům a vlivu parametrů na jejich chování. V první kapitole jsou popisovány základy generických algoritmů, jejich vznik a popis jednotlivých parametrů jako jsou například operátory křížení, mutace či selekce, vysvětlení základních pojmů jako jsou populace, reprezentace, fitness funkce apod.

V následující kapitole je pak uveden příklad implementace GA do prostředí Matlabu. Pomocí toolboxu je implementována Rastringinova funkce, na které jsou ukázány základní možnosti nastavení algoritmu a jeho možnosti grafického znázornění.

V kapitole poslední je pak použit známý kombinatoristický problém řešený i pomocí GA, a to Problém Obchodního Cestujícího, na kterém jsou pak testovány vlivy jednotlivých parametrů.

1. Princip fungování genetického algoritmu

1.1. Biologie versus informatika

Matematika a informatika většinou obohacují ostatní obory svými výsledky. U genetických algoritmů je tomu přesně naopak. Darwinova teorie o vývoji druhů aplikovaná na informatiku nám dala možnost řešit problémy, kde matematický aparát selhával. Dala vzniknout „Evolučním výpočetním technikám“, jejichž odnoží jsou právě „Genetické algoritmy“ (GA).

Jejich vznik můžeme položit do 60. let 20. století, kdy se skupina vědců snažila sestavit evoluční algoritmus natolik univerzální, že by jeho prostřednictvím bylo

možné řešit řadu složitých optimalizačních úloh. Jednou z hlavních osobností vývoje GA je John Holland[1] známý jako „otec genetických algoritmů“, jež sestavil dostatečně univerzální algoritmus. Dnes se algoritmus používá často v různých modifikacích, princip však zůstává nezměněn.

1.2. Matrice života v počítačích

Již podle názvu můžeme soudit, že genetické algoritmy jsou inspirovány myšlenkami z biologie. GA řeší problém tak, že použije jeho vyjádření pomocí chromozomu, jež je zpravidla realizován posloupností bitů či vektoru reálných čísel, a snaží se získat optimální řešení pomocí vyvíjející se populace. Existují problémy, které se dají pomocí GA řešit velmi dobře. Stejně tak je ovšem plno těch, které je lepší řešit jinými způsoby. Asi nejčastější oblastí aplikací GA je optimalizace a strojové učení. [4]

V teorii umělé inteligence je genetický algoritmus proces postupného vylepšování populace jedinců opakovanou aplikací genetických operátorů, který vede k evoluci takových jedinců, kteří lépe vyhovují stanoveným podmínkám než jedinci, ze kterých vznikli. Proces konverguje k situaci, kdy je populace tvořena jen těmi nejlepšími jedinci. Hlavním principem je kopírování a vyměňování řetězců - chromozómů. Chromozóm má pevnou délku, jednotlivé pozice tvoří geny. Geny reprezentuje často binární 0 nebo 1, ale obecně mohou mít libovolnou hodnotu. Množina chromozómů tvoří populaci. Každý chromozóm v populaci má definovánu hodnotící funkci, nazývanou fitness funkce, která charakterizuje vhodnost chromozómu. [5]

1.3. Základní pojmy

Před přiblížením problematiky genetického algoritmu je vhodné ujasnit pojmy, které jsou v práci použity. Některé jsou pak dále přiblíženy v následujících kapitolách.

Populace - je obecné vyjádření pro sled generací

[1] John Holland – nar. 2. Února 1929 ve Fort Wayne, Indiana. [3]

Generace - skupina jedinců, na kterou je aplikována teorie o vývoji druhů

Jedinec - nositel genetické informace

Genotyp - obecná genetická informace

Fenotyp - zobrazení genetické informace do konkrétního jedince

Chromozóm - obecná genetická informace ve tvaru řetězce, je to sekvence symbolů z nějaké abecedy (mohou to být čísla, znaky nebo jejich kombinace)

Alela - určité místo (pozice) v chromozómu

Gen - konkrétní symbol v chromozómu

Fitness - síla jedince v generaci, odvozuje se od ní jeho přežití

Rodič - jedinec vstupující do rekombinace

Potomek (Dítě) - jedinec, jenž je výsledkem rekombinace

Rekombinace - křížení a mutace

Křížení - konstrukce nových jedinců (dětí) dle vybraných jedinců generace (rodičů)

Mutace - změna hodnoty v chromozómu (změna genu v chromozómu)

Selekce - výběr jedinců, kteří přežívají v generaci

1.4. Jednoduchý genetický algoritmus

Vylepšením GA je genetické programování, díky čemuž neexistuje přesná definice, co GA ještě je a co již ne, která by byla akceptována všemi. Lze říci, že genetický algoritmus je evoluční algoritmus, který prostřednictvím svých operátorů získává další potencionální řešení.

Jak již bylo řečeno výše genetický algoritmus napodobuje proces evoluce, kdy následující populace má vlastnosti, které jsou kombinaci vlastností lepších jedinců v předchozí generaci. Populace jednotlivých jedinců druhů v přírodě se postupně

vyvíjí a každá následující generace se zlepšuje. Vývoj kupředu je zajištěn tak že, nevhodní jedinci, kteří se nejsou schopní adaptovat, vymírají, aniž by předali svůj genetický materiál následující generaci. Při páření jsou upřednostňováni jedinci lepší, a tak mají větší šanci předat svůj genetický materiál následující generaci. Jedinci příští generace tak budou mít ve výsledku lepší vlastnosti než generace předešlá. Každou další generací se populace adaptuje na okolní podmínky a její jedinci se blíží k tomu, aby jejich vlastnosti byly optimální.

Pro účely simulace Holland navrhl reprezentovat vlastnosti jedince chromozomy a proces adaptace je řešen pomocí 4 operátorů, které jsou použity pro tvorbu následující generace. Jsou to tyto operátory:

- operátor selekce;
- operátor křížení;
- operátor mutace;
- operátor inverze.

Poslední operátor inverze se příliš neuplatňuje, protože se zjistilo, že na kvalitu výsledné populace a průběh algoritmu nemá výrazný vliv. [2]

1.5. Jak fungují genetické algoritmy

1.5.1. Jedinci a jejich reprezentace

Jedinec je v GA reprezentován prostřednictvím genů. Každý gen vyjadřuje určitou vlastnost jedince. Vlastnosti nabývají omezeného počtů stavů. Geny jsou uspořádány do posloupnosti a tu nazýváme **chromozom**. Chromozom pak zastupuje **genotyp** jedince. Genotyp pak reprezentuje **fenotyp** jedince, což je množina všech výsledných vlastností jedince.[2]

Chromozomy jsou pak nejčastěji reprezentovány binárním řetězcem. Kde hodnota í-tého genu je určena í-tou hodnotou v řetězci. Je možno vytvořit prakticky libovolnou reprezentaci jedince, tvořenou například vektory, maticemi, nebo křivkami.

1.5.2. Průběh genetického algoritmu

Na začátku je vygenerovaná populace z náhodných členů. Každému jedinci je přiřazeno jeho ohodnocení (fitness). Poté jsou pomocí operátoru selekce vybráni vhodní jedinci, kteří jsou modifikováni operátory křížení a mutace, díky čemuž vzniká nová generace. Postupným opakováním tohoto postupu se kvalita populace postupně zlepšuje. A pokračuje, dokud kvalita není dostačující.

Samotný genetický algoritmus se skládá z 6 základních kroků, v kterých jsou genetické operátory uplatněny [2]:

- Vygeneruj počáteční populaci o N jedincích.
- Vypočti vhodnost každého jedince.
- Opakuj, dokud nevytvoříš N nových jedinců:
  Použij operátor selekce a vyber dva jedince.
  S určitou pravděpodobností p_c aplikuj na vybraný pár operátor křížení a získej tak nové dva jedince.
  Pokud operátor není aplikován, vytvoř kopie jedinců a ty považuj za nové dva jedince.
  S určitou pravděpodobností p_m aplikuj na každého jedince z dvojice nových jedinců operátor mutace.
  Výsledné dva jedince vlož do nové populace.
- Současnou populaci nahraď nově získanými jedinci (novou generací).
- Vypočti vhodnost každého jedince v populaci.
- Pokud není splněna ukončovací podmínka, vrať se ke kroku 3. Pokud podmínka je splněna, zjisti nejlepšího jedince a oznam konec.

1.5.3. Funkce vhodnosti

Hodnota funkce vhodnosti neboli fitness vyjadřuje, do jaké míry jedinec splňuje požadovaná kritéria, která požaduje optimální řešení. Musí být sestavena, aby zajišťovala pro nejlepšího jedince nejlepší hodnotu a pro opačný konec populace zase tu nejhorší. Většinou je to řešeno takovým způsobem, že nabývá hodnoty v intervalu <0,1>, kdy hodnota nejlepšího jedince je 1 a nejhoršího 0.

Někdy je velice obtížné stanovit funkci vhodnosti v případech, kdy o vhodnosti řešení má informace pouze jeden člověk, který je není schopen sdělit vhodným způsobem, nebo není možné funkci vhodnosti sestavit z důvodu, že se hledá řešení problému, které je založeno na citovém vjemu člověka. V tom případě algoritmus musí být v interakci s daným člověkem, který po každém cyklu algoritmu musí každého jedince sám ohodnotit. Pochopitelně takto realizovaný algoritmus je pomalý, nicméně i tímto způsobem je možné při omezeném počtu jedinců a generací dojít k výsledku. Příkladem takové aplikace je v USA v praxi využívaný systém Faceprints sloužící k identifikaci pachatelů trestné činnosti. Svědek trestné činnosti je vyzván k určení podobnosti jednotlivých obličejů s hledanou osobou, k čemuž využívá desetibodové stupnice. Na základě takto získaného ohodnocení genetický algoritmus provede selekci a pomocí genetických operátorů se vytvoří nová sada obličejů. [2]

1.5.4. Operátor selekce

Operátor selekce vybírá jedince, na kterých budou následně aplikovány genetické operátory, přičemž při výběru musí upřednostňovat lepší jedince. Operátor napodobuje proces přirozeného výběru vhodných jedinců v přírodě, kdy lepší jedinci mají větší šanci na přežití a jsou v případném páření také svými protějšky při páření upřednostňováni před méně úspěšnými samci, kteří pravděpodobně opustí svět bez potomků a nebudou moct své nekvalitní geny šířit dále. Existuje několik způsobů, jak může být operátor selekce realizován. Vždy však platí, že pravděpodobnost výběru jedince musí být úměrná jeho vhodnosti. Pravděpodobnost, že bude vybrán vhodnější jedinec, musí být vždy vyšší, než že bude vybrán jedinec horší. Při návrhu selekčního mechanismu je nutné mít na zřeteli dvě věci. Pokud selekční mechanismus příliš upřednostňuje nejlepší jedince, dojde rychle k tomu, že se ztratí rozmanitost populace a následná evoluce bude silně omezená, protože jedinci brzy budou složeni ze stejných vlastností, které budou moci být změněny jenom velice obtížně, a hrozí, že genetický algoritmus uvízne v blízkosti suboptimálního řešení. Na druhou stranu pokud je však selekční mechanismus natolik benevolentní, že příliš toleruje méně vhodné jedince, algoritmus se bude k optimálnímu řešení přibližovat velice pomalu. [1]

Nejčastější způsob selekce představuje selekce založená na přímé úměře k vhodnosti jedince, selekce založená na přímé úměře k pořadí jedince a turnajová selekce. Každá ze selekcí má své výhody i nedostatky. Často jsou selekce doplněny některými mechanismy, které zvyšují efektivnost algoritmu, tím že tyto nedostatky potlačují. Příkladem takových mechanismů může být škálování nebo elitismus.

Ruletová selekce přímo úměrná vhodnosti jedince

Ruletový výběr jedinců je zřejmě nejběžnější typ selekčního mechanizmu. Můžeme si ho představit jako ruletové kolo. Reálné ruletové kolo je rozděleno na stejně velké díly. Když se jedná o selekci ruletového mechanismu, každá jeho výseč představuje jednoho jedince. Velikost výsečí záleží na hodnotě vhodnosti konkrétního jedince. Tím je dosaženo toho, že vhodnější jedinec bude mít větší velikost výseče a tím i větší pravděpodobnost výběru. Poté bude vygenerováno náhodné číslo, reprezentující místo, na kterém se zastaví kulička ruletového kola, tím selekční mechanizmus vybere jedince, na kterého poté aplikuje genetické operátory, a výsledek předá následující generaci.[2]

Je několik způsobů pro určení velikosti výseče. Nejjednodušší způsob je přímá úměra mezi velikosti výseče a vhodností jedince. Sečteme hodnoty všech jedinců v populaci a jedinci pak přiřadíme výseč proporcionálně odpovídající jeho hodnosti. Předpokládáme, že funkce nabývá nezáporných hodnot a je maximalistická.

Vyjádření toho vztahu vypadá následovně. Máme populaci o velikosti N a ohodnocení každého jedince je nezáporné, pak pravděpodobnost, že bude jedinec vybrán je [2]:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}, i \in \{1,..,N\}, \qquad (1)$$

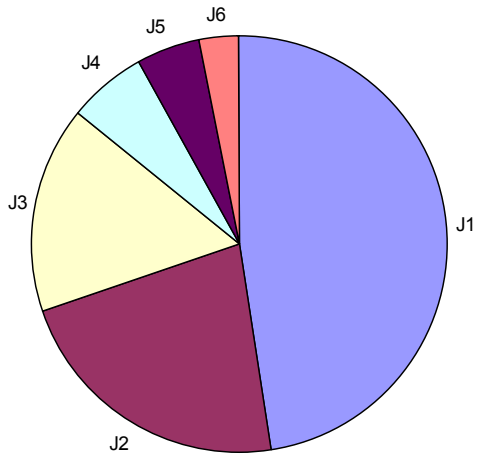
kde f_i představuje vhodnost í-tého jedince, $\sum_{j=1}^{N} f_j$ součet vhodnosti všech jedinců a p_i je pravděpodobnost výběru í-tého jedince.

Tento mechanismus však trpí svými nedostatky. Pokud se v populaci vyskytne jedinec s ohodnocením velmi vysokým oproti ostatním, bude mu přiřazena velká část plochy ruletového kola. Tento jedinec bude pak v selekci natolik dominantní, že většina následující generace bude tvořena z jeho genů, čímž populace ztrácí variabilitu.

Dalším nedostatkem je, když jsou rozdíly v populaci příliš malé. Každému jedinci je přiřazena přibližně stejná plocha. Tento stav většinou nastává, když se GA blíží k nalezení optimální hodnoty. Lepší jedinci nejsou dostatečně zvýhodněni oproti ostatním. Dochází k nedostatku selekčního tlaku.[2]

Tabulka 1 - Pravděpodobnosti jedinců při selekci přímo úměrné vhodnosti [zdroj vlastní]

jedinec	J1	J2	J3	J4	J5	J6
vhodnost	30	14	10	4	3	2
pravděpodobnost	0,48	0,22	0,16	0,06	0,05	0,03



Graf 1 - Rozložení pravděpodobnosti při selekci přímo úměrné vhodnosti [zdroj vlastní]

Ruletová selekce přímo úměrná pořadí jedince

Předpokládáme, že jedinci jsou seřazení vzestupně podle jejich hodnosti.

Velikost í-tého jedince na ruletě se spočítá podle vztahu [2]:

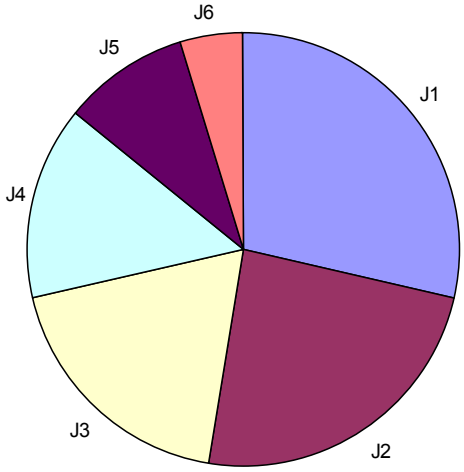
$$p_i = \frac{i_i}{\sum_{j=1}^{N} j} = \frac{2.i}{N.(N+1)}, i \in \{1,..,N\}, \tag{2}$$

kde N je velikost populace, $\sum_{j=1}^{N} j_j$ součet pořadí všech jedinců a p_i je pravděpodobnost výběru í-tého jedince. Index i zároveň určuje pořadí jedince v populaci.

Výhodou tohoto selekčního mechanizmu je, že díky výpočtu velikosti výseče založené na pořadí odpadá nutnost reprezentace vhodnosti nezápornou hodnotou. Tato selekce také potlačuje roli nadprůměrných jedinců, kteří by mohli negativně ovlivnit algoritmus tím že bude prohledávat pouze okolí nadprůměrného jedince. Zajišťuje také selekční tlak na konci algoritmu, kdy je populace složena z jedinců, mezi kterými nejsou velké rozdíly a jejichž výseče by byly prakticky stejné. [2]

Tabulka 2 - Pravděpodobnosti jedinců při selekci přímo úměrné pořadí [zdroj vlastní]

jedinec	J1	J2	J3	J4	J5	J6
vhodnost	20	14	10	4	3	2
pořadí	1	2	3	4	5	6
pravděpodobnost	0,29	0,24	0,19	0,14	0,10	0,05



Graf 2 - Rozložení pravděpodobnosti při selekci přímo úměrné pořadí [zdroj vlastní]

Turnajová selekce

Zcela jinak přistupuje k mechanizmu selekce. Opouští náhodný či polonáhodný výběr s pravděpodobnostmi souvisejícími s kvalitou jedinců a nahrazuje ho soutěží jedinců zápasících o přežití. Algoritmicky se postupuje tak, že z původní populace se zcela náhodně (nezávisle na ohodnocení) vyberou skupinky jedinců (převážně dvojčlenné), mezi nimiž se uspořádá turnaj. Vítěz se stane prvkem selektované populace. Tento postup se opakuje, dokud selektovaná populace nemá potřebných N prvků. Experimenty ukazují, že turnajová selekce spojená se samotným uchováváním dosud nejlepšího řešení dává nejpříznivější výsledky.[1]

Elitismus

Předešlé techniky obecně nezaručují postup nejlepšího jedince do nové generace. Zvláště v malých populacích je tato ztráta vnímána velmi negativně. Experimenty

prokázaly, že ztráta nejlepších jedinců může být opakována a znovuvytvoření jedince není automatické.

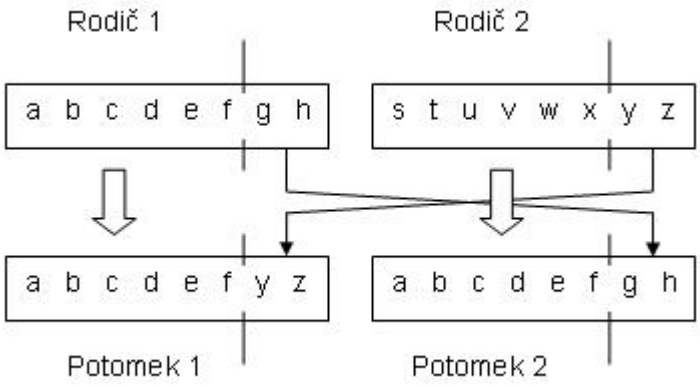
Elitismus je jednoduchá technika, kterou vybíráme určitý (velmi malý, např. je vybrán pouze jeden) počet nejlepších jedinců, a ti se nepodrobují selekčnímu tlaku, ale postupují do nové generace přímo.[1]

1.5.5. Operátor křížení

Křížení je operace kombinující dva jedince – rodiče, z nichž generuje dva potomky. Celý proces křížení je realizován tak, že nad každým selektovaným jedincem se provede jednoduchý náhodný pokus a příslušný jedinec se s pravděpodobností Px (pravděpodobnost Px 0,6 -1) stane kandidátem na páření. Vznikne tak množina jedinců, z nichž se vybírají dvojice buď náhodně, nebo se za pár prohlásí po sobě jdoucí jedinci tak, jak byli do množiny zařazeni. Je-li počet prvků této množiny lichý, jeden kandidát se vyloučí. [1]

GA reprezentující jedince bitovými řetězci používají jednobodové křížení, které pracuje tak, že zvolíme křížící bod jako náhodné číslo v rozsahu 1 až (L-1), kde L je délka řetězce. V tomto místě vedeme řez oběma rodičovskými řetězci. Potomky pak vytvoříme tak, že první z nich je tvořen levou částí prvého rodiče a pravou částí druhého rodiče, u druhého potomka je tomu naopak. [1]

GA reprezentující jedince bitovými řetězci používají jednobodové křížení, které pracuje tak, že zvolíme křížící bod jako náhodné číslo v rozsahu 1 až (L-1), kde L je délka řetězce. V tomto místě vedeme řez oběma rodičovskými řetězci. Potomky pak vytvoříme tak, že první z nich je tvořen levou částí prvého rodiče a pravou částí druhého rodiče, u druhého potomka je tomu naopak. [1]

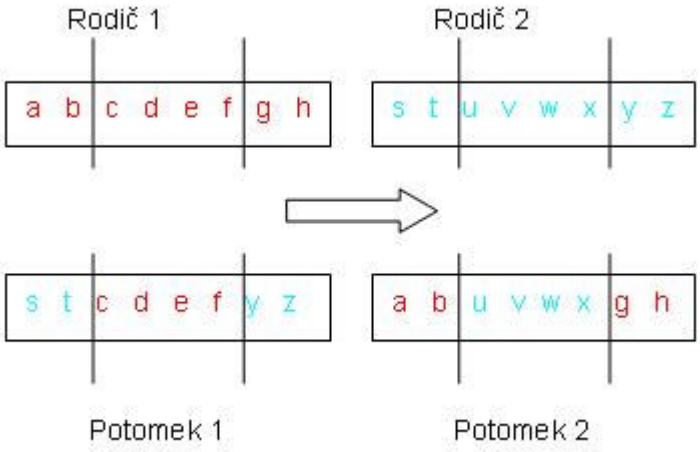


Obrázek 1 - Křížení jednobodové převzato z [1]

0010|011 = 19 X 1111|001 = 121 ➔ 0010|001 = 17 X 1111|011 = 123

Obrázek 2 - Příklad jednobodového křížení zdroj [1]

Tento příklad ukazuje křížení jednobodové. Pokud potřebujeme zaměňovat více částí, hovoříme pak o křížení vícebodovém, kdy se rodiče dělí ve více bodech.

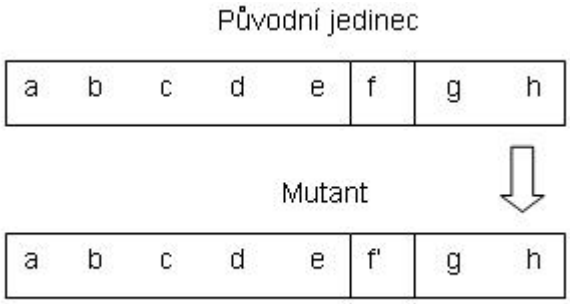


Obrázek 3 - Křížení vícebodové převzato a upraveno z [6]

Experimenty prokázaly, že co do výsledného chování algoritmu jsou oba případy ekvivalentní a žádný z nich nemá smysl upřednostňovat. A pokud ano, pak jednobodové křížení, neboť je výpočetně jednodušší. [1]

1.5.6. Operátor mutace

Mutace má velmi jednoduchou základní podobu. Nad každým bitem v celé populaci provede náhodný experiment s pravděpodobností úspěchu rovnou Pm (pravděpodobnost Pm 0,05-0,15). Vyjde-li pozitivní výsledek, pak příslušný bit invertujeme. Alternativní interpretací je volba jedince s pravděpodobností L*Pm a inverze jednoho náhodně zvoleného bitu tohoto jedince. [1]



Obrázek 4 - Mutace nad binární reprezentací zdroj [1]

1111011 = 123 ➔ 1111010 = 122

Obrázek 5 - Příklad mutace zdroj [1]

Mutace má na algoritmus pozitivní vliv, protože u jedinců vytváří nové vlastnosti. Jakmile se algoritmus dostává do blízkosti suboptimálního řešení, mutace udržuje variabilitu populace. Díky změnám vlastností jedinců se prohledává celý prostor potenciálních řešení. [6]

Tím, že mutace vytváří u jedinců nové vlastnosti, obohacuje prostor prohledávaných řešení, kterých se nedá dosáhnout křížením nebo selekcí, protože není obsažena v předchozí generaci.

Operátor mutace ale narušuje genetickou informaci jedinců, jeho příliš velké uplatňování by mělo za následek, že se algoritmus bude v prostoru pohybovat náhodně a k optimálnímu výsledu se dostane velice pomalu.[2]

1.5.7. Operátor inverze

Princip spočívá v tom, že v souvislé části genů přehodí jejich pořadí. Dnes již minimálně používaný operátor. Ukázalo se, že nezvyšuje účinnost a na nalezení optimálního řešení má zanedbatelný vliv. [6]

11|0111|01 ➔ 11|1110|01

Obrázek 6 - Operátor inverze zdroj [6]

1.5.8. Teorie o schématech

Představuje doposud jediný široce uznávaný přístup k objasnění chování genetických algoritmů a evolučních technik vůbec, i když s postupem času byl tento aparát podrobován mnohým kritikám a je předmětem mnoha výhrad. Chromozomy tvořené binárními řetězci představují jednotlivé vzorky prohledávaného prostoru. Jednotlivé komponenty (bity) mohou příznivě či nepříznivě ovlivňovat řešení. Otázka pak může například znít: Jaký má vliv jedničkový bit na třetí pozici? Ke studiu tohoto vlivu se zavádí pojem schéma, které je opět řetězcem vytvořeným nad abecedou symbolů rozšířený o nový symbol *. Abeceda řetězců ve schématech tedy je V*={0,1,*}. Symbol * má význam náhradního znaku zastupujícího libovolný ze symbolů základní abecedy.

Snadno nahlédneme, že binárních řetězců délky L je 2^L zatímco schémat je 3^L. V každém schématu může být 0 až L „hvězd“. Schéma bez těchto symbolů je vlastně reprezentantem jediného řetězce, schéma s jedním zástupným znakem koresponduje se dvěma řetězci atd. Schéma H s j zástupnými znaky má k=(L-j) základních symbolů. Toto číslo k se nazývá řád schématu a značí se o(H). Každé schéma řádu o(H) pokrývá $2^{L-o(H)}$ řetězců.[1]

L=7, H=(0,1,*,*,0,1,*), o(H)=4, L-o(H)=3

Pokrývá 2^3 = 8 řetězců

(0,1,0,0,0,1,0), (0,1,0,0,0,1,1), (0,1,1,0,0,1,0), (0,1,1,0,0,1,1),
(0,1,0,1,0,1,0), (0,1,0,1,0,1,1), (0,1,1,1,0,1,0), (0,1,1,1,0,1,1),

Obrázek 7 - Příklad schéma zdroj [1]

Pro každé schéma H se vedle řádu o(H) definuje další parametr, kterým je definiční délka označovaná δ(H). Je to celé číslo v rozsahu 0 až (L-1) vyjadřující největší vzájemnou vzdálenost dvou základních symbolů ve schématu. Schémata řádu 0 a 1 mají konvencí stanovenou definiční délku δ(H)=0. Například schéma H = (0,1,*,*,0,1,*) má definiční délku δ(H)=5.

V dostatečně velké populaci G(t) můžeme kvalitu schéma definovat následovně

$$f(H) = \frac{\sum_{x_i = H} f(x_i)}{\left| x_i \propto H \right|}, \qquad (3)$$

kde jmenovatel značí počet jedinců v populaci, kteří korespondují se schématem H. Kvalita schématu řádu 0, které pokrývá celou populaci $f(H_0)$, je průměrnou kvalitou f‘ celé populace.

Položme si nyní otázku, jak se bude měnit počet výskytů daného jedince v čase t. Protože však jedinci mohou byt rozbiti rekombinačními operátory, bude lepší zabývat se schématy, neboť ta pokrývají celé množiny jedinců. Označme tedy m(H,t) počet jedinců pokrytých schématem H v populaci G(t) a ptejme se na počet m(H,t+1).

Analyzujme napřed vliv selekce a uvažujme, že GA používají mechanizmus ruletového kola. Bude-li m(i,t) počet výskytů jedince $x_{i,}$ v populaci G(t), pak selekce způsobí, že počet m(i,t+1) bude při dostatečně velké populaci přibližně rovno

$$m(i,t+1) = (i,t)\Pr(i). \qquad (4)$$

Tento vztah vyplývá z experimentu s ruletovým kolem, kdy se zajímáme pouze o m(i,t) „zatočení“ ruletou pro výběr námi uvažovaných výskytů daného jedince. Podobně pro schémata bude platit

$$m(H,t+1) = m(H,t)\frac{f(H)}{f'}. \qquad (5)$$

Jestliže zapíšeme $f(H) = f' + cf'$, kde pro číslo c platí, že c < 0 pro průměrná schémata a c > 0 pro schémata nadprůměrná, můžeme vztah (5) přepsat na

$$m(H,t+1) = m(H,t)\frac{f'+cf'}{f'} = (1+c)m(H,t). \qquad (6)$$

Budeme-li navíc o čísle c předpokládat, že se v čase nemění, můžeme (6) zapsat do tvaru

$$m(H,t) = (1+c)'\, m(H,0). \qquad (7)$$

Ze vztahu (7) vyplývá, že počet jedinců pokrytých nadprůměrným schématem poroste přibližně geometrickou řadou, zatímco jedinci podprůměrní budou geometrickou řadou vymírat. V praxi ovšem c není konstantní, neboť selekce v populacích konečného rozsahu množinu jedinců homogenizuje okolo průměru, takže číslo c konverguje k nule. Vztah (7) je tak sice jen přibližným odhadem chování, avšak určitou vypovídající schopnost jistě má.

Všimněte si nyní vlivu křížení na schémata. Uvažujme schémata H1=(*,1,*,*,*,*,0), δ(H1)=5 a H2=(0,1,*,*,*,*,*), δ(H2)=1, viz Obrázek 8. Dále vezměme pár rodičů A a B. Vidíme, že $A \propto H_1$ a $A \propto H_2$, zatímco B s těmito schématy nekoresponduje. Bude-li při páření rodiči A a B zvolen křížící bod mezi třetí a čtvrtou pozicí, dostaneme potomky A‘ a B‘.

H1	=	*	1	*	|	*	*	*	0	H1	=	*	1	*	|	*	*	*	0
H2	=	0	1	*	|	*	*	*	*	H2	=	0	1	*	|	*	*	*	*
A	=	0	1	1	|	1	0	0	0	B	=	0	0	0	|	0	0	0	1
A'	=	0	1	1	|	0	0	0	1	B'	=	1	1	1	|	1	0	0	0

Obrázek 8 - Vliv křížení na schémata zdroj [1]

Potomek A‘ nadále koresponduje se schématem H2, zatímco korespondence se schématem H1 byla rozbita. Schéma H2 tedy křížení přežilo, zatímco H1 nikoliv.

Z mechanizmu křížení vidíme, že pravděpodobnost PD(H) rozbití schématu H závisí na jeho definiční délce δ(H), takto křížící bod volíme náhodně s pravděpodobnostmi stejnými pro všechny pozice

$$p_{xD}(H) = \frac{\delta(H)}{L-1}. \tag{8}$$

Pravděpodobnost, že schéma H přežije křížení tedy je

$$p_{xS}(H) = 1 - \frac{\delta(H)}{L-1}. \tag{9}$$

Vzhledem k tomu, že jedinci podstupují křížení jen s pravděpodobností Px, je pravděpodobnost přežití schématu H

$$p_{xS}(H) = 1 - P_x \frac{\delta(H)}{L-1}. \tag{10}$$

Navíc i když křížící místo padne mezi fixované pozice schématu, schéma přesto může přežít. Přesnější zápis tedy je

$$p_{xS}(H) \geq 1 - P_x \frac{\delta(H)}{L-1}. \tag{11}$$

Pokud vezmeme v potaz mutaci, tak její vliv na schémata je následující. Každý bit je invertován s pravděpodobností Pm<<1 a má tedy pravděpodobnost přežití (1-Pm). Schéma H mající o(H) specifikovaných bitů má tedy pravděpodobnost přežití

$$P_{mS} = (1 - P_m)^{o(H)}, \tag{12}$$

což lze vzhledem k Pm << 1 aproximovat jako

$$P_{mS} = 1 - P_m o(H). \tag{13}$$

Celkový vliv genetických operátorů na počet výskytů jedinců korespondujících s daným schématem H lze pak vyjádřit jako

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{f'}\left(1 - P_x \frac{\delta(H)}{L-1} - P_m o(H)\right). \quad (14)$$

Vztah (13) představuje klíčovou formuli obecně považovanou za vysvětlení funkce GA. Slovy se vyjadřuje jako teorém o schématech:

„Počet krátkých nadprůměrných schémat nízkého řádu v jednotlivých generacích exponenciálně roste.“

Bezprostředním důsledkem teorému je hypotéza o stavebních blocích:

„Genetický algoritmus hledá svoje chování blízké optimálnímu tím, že upřednostňuje a přeskupuje krátká nadprůměrná schémata nízkého řádu, nazývaná stavební bloky.“

Teorém o schématech i hypotéza o stavebních blocích představují spíše pokus než skutečně reálné funkce GA. Především tato teorie vychází z předpokladu nekonečně velkých populací. Při rozumně velkém rozsahu populace přestávají závěry velmi rychle platit. Navíc lze v celku snadno najít řadu příkladů, kdy tyto teoretické výsledky neplatí. Jde zejména o tzv. klamné problémy.

2. Implementace genetických algoritmů

2.1. Matlab

MATLAB je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích komunikačních systémů. Je to nástroj pro vývoj širokého spektra aplikací.

Jedním z nich je Genetic Algorithm Toolbox, který rozšiřuje optimalizační možnosti Matlabu o nástroje pracující s genetickými algoritmy. Je využitelný zejména v úlohách, které jsou obtížně řešitelné tradičními optimalizačními metodami, včetně úloh, které nejsou dobře definovatelné nebo je lze obtížně modelovat matematicky. Jsou použitelné také v případech, kdy nejsou počítané funkce spojité, či jsou vysoce nelineární, stochastické nebo mají nejisté nebo neznámé derivace.[7]

2.2. Implementace GA v Matlabu

Genetické algoritmy spadají do oblasti metod kombinatorické optimalizace. Při hledání například globálního extrému lze zakódovat body na funkci tak, že k nim lze přistupovat jako ke kusům dědičného materiálu. Poté lze s výhodami použít různých způsobů modifikace dědičného materiálu jako v přírodě (křížení, mutace). Řízení těchto rekombinačních kroků je možné parametrizací nebo (což je přirozenější) selekcí. Zde se dostáváme k parametrizaci, neboli řízení heuristiky.[8]

Možnosti nastavení GA v toolboxu jsou velmi rozsáhlé:

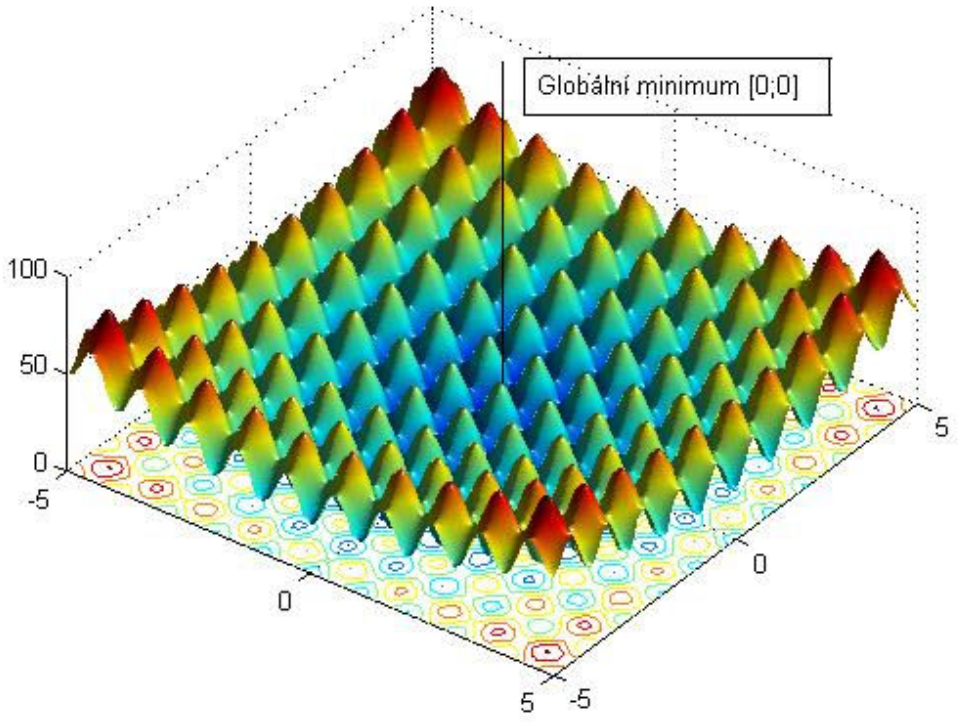
- Mutace a křížení lze řídit elitářstvím, řízenou mutací (určení, jak silně bude populace mutována a jak bude mutace v průběhů generací ubývat), řízením křížení (zde je možno určit, kolik procent populace z neelitního podílu generace se bude křížit).
- Nastavení maximálního počtu generací.
- Řízení výběru rodiče na základě jeho fitness.
- Poměření fitness, což umožňuje snazší nalezení vhodných rodičů pro budoucí generaci.

- Hendikepování známé oblasti hledání, kde již víme, že zde pravděpodobně globální extrém neleží.
- Užití vyhledávacího algoritmu na dohledání optima poté, co GA ukončily svou činnost.
- Možnost nastavit hranice prostoru, jež bude prohledáván. [8]

Můžeme si také zvolit, zda budeme volit funkci GA pomocí příkazového řádku, užití toolboxu grafického prostředí pro genetické algoritmy, anebo kombinací obou uvedených metod.

2.3. Realizace GA pomocí toolboxu

Jako ukázku použijeme Rastriginovu funkci, která je často používána k testování genetického algoritmu, protože obsahuje spoustu lokálních minim, ale pouze jedno globální minimum. [7]



Obrázek 9 - Rastriginova funkce [zdroj vlastní]

Pro dvě nezávislé proměnné je tato funkce definována

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)\,. \qquad (15)$$

2.3.1. Nastavení parametrů GA

Nejprve musíme toolbox nastavit:

Nastavení funkce a počtu proměnných:

```
Fitness function: @rastiginsfcn
Number of variables: 2
```

Kód 1 - Fitness funkce a počet proměnných [7]

Velikost populace:

```
Population size: 20
```

Kód 2 - Velikost populace [7]

Prohledávaný prostor:

```
Initial Range: [0;2]
```

Kód 3 - Prohledávaný prostor [7]

Počet nejlepších jedinců, kteří postoupí do další generace, na zbytek budou aplikovány genetické operátory:

```
Elite count: 2
```

Kód 4 - Počet elitních jedinců [7]

Poměr mutace a křížení:

```
Crossover fraction: 0.8
```

Kód 5 - Crossover fraction [7]

Limit počtu generací:

```
Generations: 150
```

Kód 6 - Počet generací [7]

Časový a fitness limit - pokud by byly nastaveny tyto možnosti, algoritmus by se zastavil při jejich dosažení:

```
Time limit: inf
Fitness limit: -inf
```

Kód 7 - Časový a fitness limit [7]

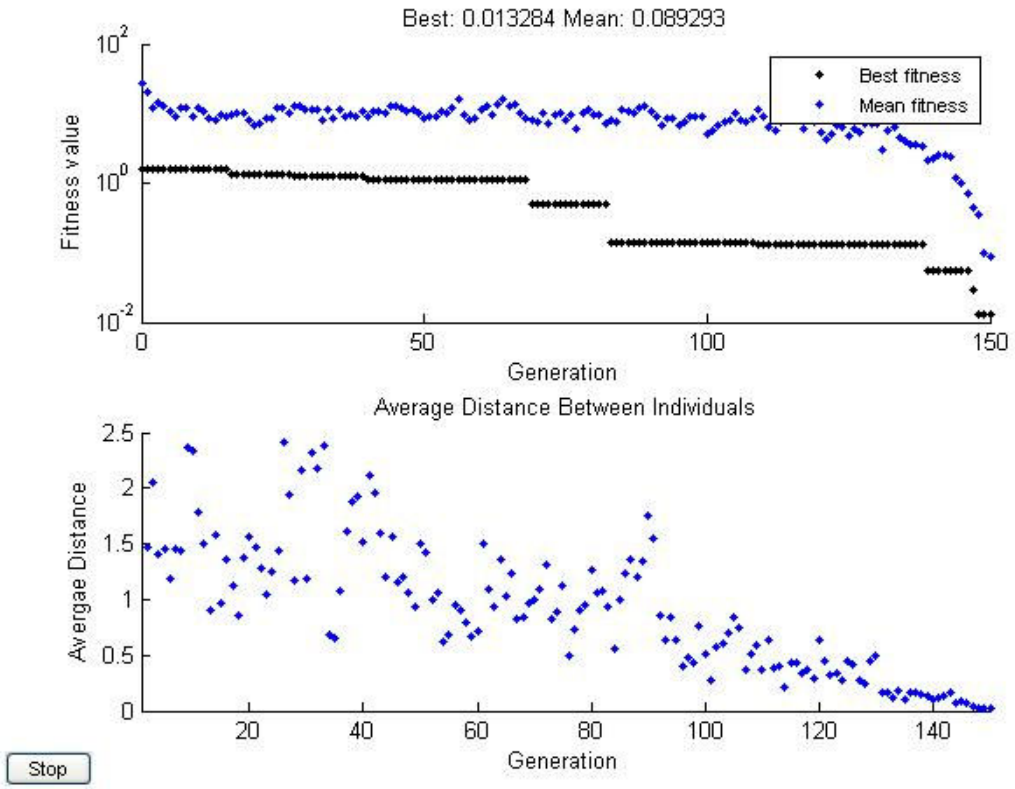
Dále nastavení funkce pro mutaci, křížení, selekci a fitness, které jsou prvotně nastaveny na:

```
Fitness: Rank
Selection: Stochastic uniform
Mutation: Gaussian
Crossover: Scattered
```

Kód 8 - Fitness, selekční, mutační a křížící funkce [7]

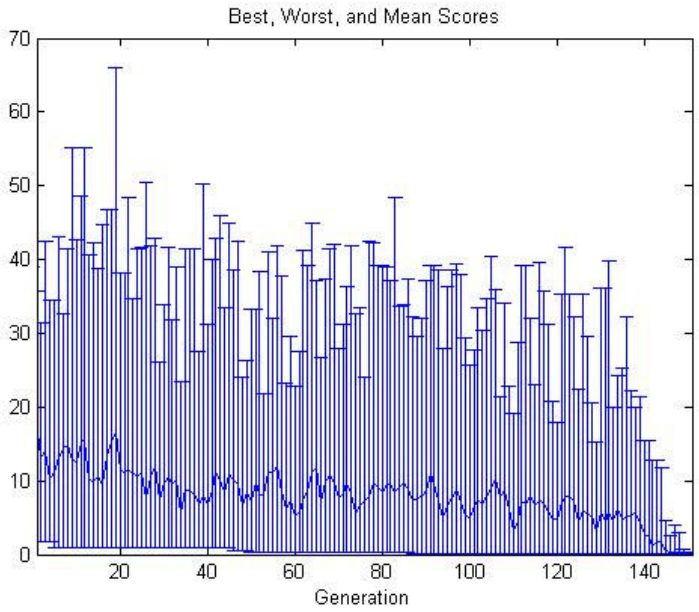
2.3.2. Vizualizace výsledků

Pro vizualizaci algoritmu nám toolbox nabízí několik možností. Na grafu č. 3 vidíme 2 grafy - první z nich ukazuje nejlepší a mezní fitness jedinců, na druhém je zobrazena vzdálenost mezi jedinci.



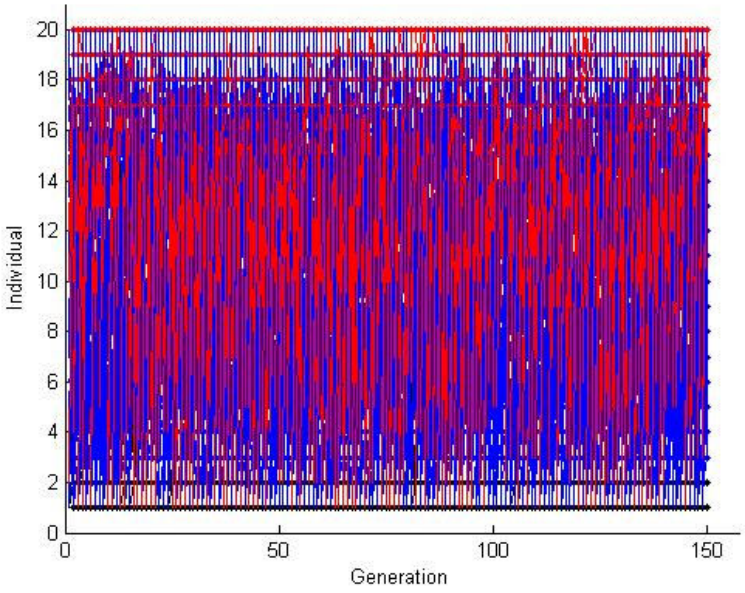
Graf 3 - Nejlepší a mezní fitness, vzdálenost [zdroj vlastní]

Na grafu 4 můžeme vidět výsledky nejhorších, nejlepších a mezních jedinců. Můžeme z něho soudit, že jsme měli jediného vážného kandidáta na globální minimum.



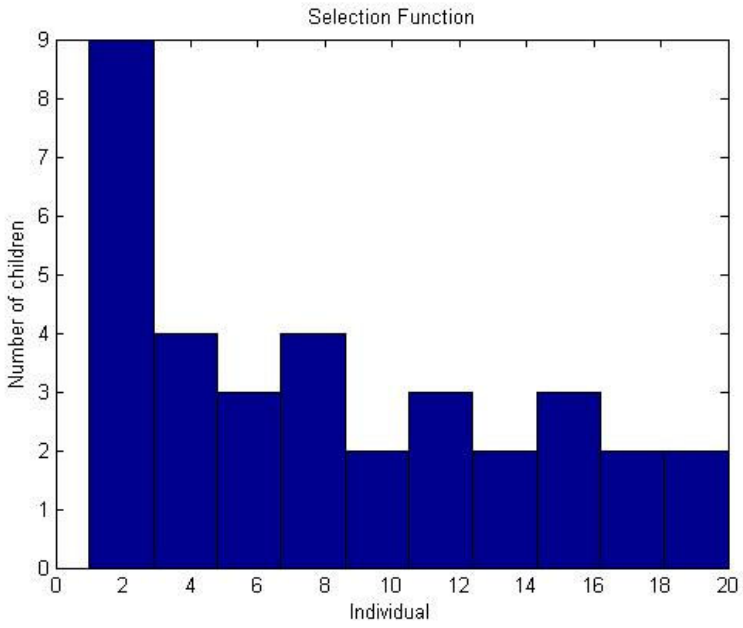
Graf 4 - Nejlepší, nejhorší a mezní výsledky [zdroj vlastní]

Grafem číslo 5 je genealogie, jež zobrazuje kombinace a míšení genů v jednotlivých krocích algoritmu.



Graf 5 – Genealogie [zdroj vlastní]

Zobrazení jedinců a počtu jejich potomků:



Graf 6 – Selekce [zdroj vlastní]

Numerické vyjádření výsledků:

Fitness function value: 0.00189910705936569

Final point: Souřadnice odpovídající globálnímu minimu Rastriginovy funkce: [0;-0]

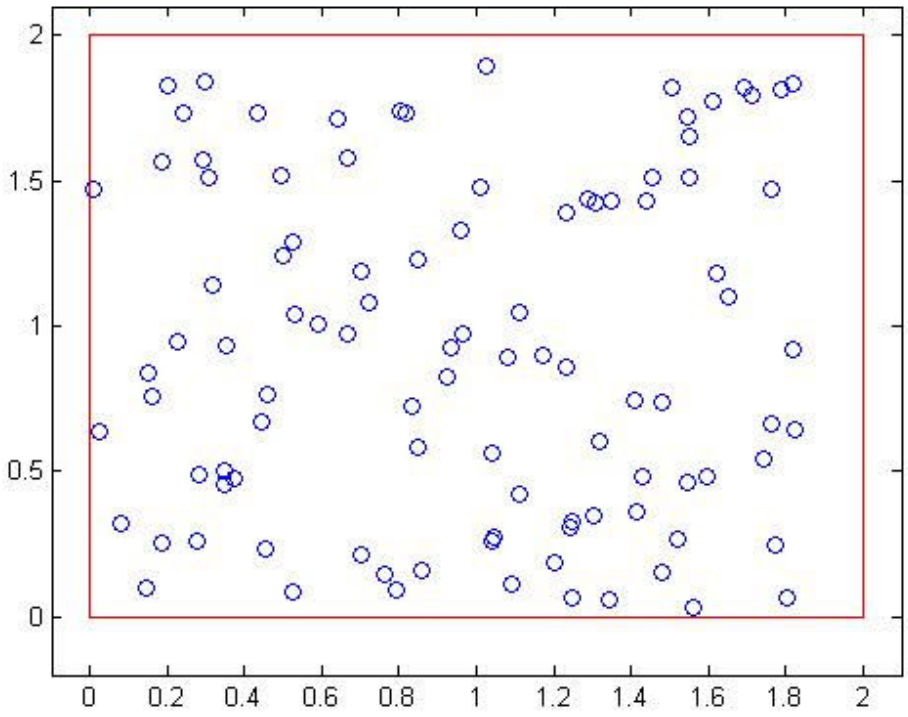
3. Vliv parametrů GA na jejich výkonnost a spolehlivost

Pro ukázku vlivu parametrů GA na jejich výsledky jsem si vybral Problém obchodního cestujícího (dále jen TSP – Traveling Salesman Problem). Je to jeden z nejznámějších kombinatoristických problémů. Slovní definice problému je taková, že obchodní cestující musí při své cestě navštívit každé město v přidělené oblasti právě jedenkrát a vrátit se do výchozího bodu. Kromě toho je třeba minimalizovat jeho cestovní náklady, které jsou definovány jako celková cena postupného přesunu mezi jednotlivými městy a závisí na vzdálenosti mezi městy. Cílem je tedy navrhnout takovou okružní cestu, kde součet nákladů nutných pro uskutečnění této cesty bude minimální. [2]

3.1. Popis nastavení prostředí a implementace TSP

3.1.1. Generování měst a vzdáleností

Před testováním algoritmu je nutné nastavit výchozí hodnoty. Na následujícím grafu (Graf 7) je znázorněna mapa měst ohraničená červenou hranicí:



Graf 7 - Znázornění mapy měst v TSP [upraven 7]

Po vygenerování souřadnic měst je potřeba zjistit jejich vzájemné vzdálenosti. Souřadnice měst jsou generovány do matice locations[počet mest x 2], z těchto souřadnic jsou dále vygenerovány vzdálenosti do matice Distances[počet mest x počet měst].

```
distances = zeros(cities);
for count1=1:cities,
    for count2=1:count1,
        x1 = locations(count1,1);
        y1 = locations(count1,2);
        x2 = locations(count2,1);
        y2 = locations(count2,2);
        distances(count1,count2)=sqrt((x1-x2)^2+(y1-
y2)^2);

distances(count2,count1)=distances(count1,count2);
    end;
end;
```

Kód 9 - Vytváření vzdálenostní matice [7]

3.1.2. Implementace operátorů

Funkce pro vytvoření

Vytváří matici propojení měst (jedinců), kde každý jedinec specifikuje pořadí měst, jak jimi bude obchodník projíždět.

```
function pop = TSP_create(NVARS,FitnessFcn,options)

totalPopulationSize = sum(options.PopulationSize);
n = NVARS;
pop = cell(totalPopulationSize,1);
for i = 1:totalPopulationSize
    pop{i} = randperm(n);
end
```

Kód 10 - Funkce vytvářející spojení cest [7]

kde NVARS je počet proměnných (měst), FitnessFcn fitness funkce a options jsou možnosti nastavení struktury pro GA.

Funkce pro křížení

Vyvolává funkci křížení, kdy potomek je kombinací rodičů.

```
function xoverKids  =
TSP_crossover(parents,options,NVARS, ...
    FitnessFcn,thisScore,thisPopulation)
nKids = length(parents)/2;
xoverKids = cell(nKids,1) ;
index = 1;
for i=1:nKids
    parent = thisPopulation{parents(index)};
    index = index + 2;
    p1 = ceil((length(parent) -1) * rand);
    p2 = p1 + ceil((length(parent) - p1- 1) * rand);
    child = parent;
    child(p1:p2) = fliplr(child(p1:p2));
    xoverKids{i} = child;
end
```

Kód 11 - Funkce křížení [7]

kde PARENTS jsou rodiče , kteří byli vybráni na základě selekce, OPTIONS možnosti nastavení, THISSCORE vektor výsledků současné populace, THISPOPULATION matice jednotlivých jedinců současné populace.

Funkce mutace

Mutační funkce mění pořadí měst jednotlivých jedinců.

```
function mutationChildren = TSP_mutate(parents
,options,NVARS, ...
    FitnessFcn, state,
thisScore,thisPopulation,mutationRate)

mutationChildren = cell(length(parents),1);
for i=1:length(parents)
    parent = thisPopulation{parents(i)};
    p = ceil(length(parent) * rand(1,2));
    child = parent;
    child(p(1)) = parent(p(2));
    child(p(2)) = parent(p(1));
    mutationChildren{i} = child;
end
```

Kód 12 - Funkce mutace [7]

kde MUTATIONRATE je míra mutace.

Fitness funkce

Udává celkovou vzdálenost pro danou kombinaci měst.

```
function scores = TSP_fitness(x,distances)

scores = zeros(size(x,1),1);
for j = 1:size(x,1)

    p = x{j};
    f = distances(p(end),p(1));
    for i = 2:length(p)
        f = f + distances(p(i-1),p(i));
    end
    scores(j) = f;
end
```

Kód 13 - Fitness funkce [7]

GA volá fitness funkci s jediným argumentem a to je „x“. My v tuto chvíli máme dva, tj. x a distances. Proto vytvoříme funkci FitnessFcn, kde nám matici distance pak bude zachycovat anonymní funkce.

```
FitnessFcn = @(x) TSP fitness(x,distances);
```

Kód 14 - Dokončení fitness funkce [7]

Funkce pro vytvoření grafu

Abychom mohli řádně zobrazit průběh algoritmu, je potřeba vytvořit si vlastní graf.

```
function state =
TSP_plot(options,state,flag,locations)

persistent xx yy
if strcmpi(flag,'init')
  load('plocha.mat','xx','yy');
end
plot(xx,yy,'Color','red');
axis([-0.1 2.1 -0.2 2.1]);

hold on;
[unused,i] = min(state.Score);
genotype = state.Population{i};
plot(locations(:,1),locations(:,2),'bo');
plot(locations(genotype,1),locations(genotype,2));
title(sprintf('Total Distance = %1.4f, Generace = %d',
min(state.Score), state.Generation));
hold off
```

Kód 15 - Funkce grafu [7]

Opět musíme použít anonymní funkci pro správné volání genetického algoritmu.

```
my_plot = @(options,state,flag) TSP_plot(options, ...
    state,flag,locations);
```

Kód 16 - Dokončení funkce pro graf [7]

3.1.3. Nastavení GA

Nyní již přejdeme k nastavování konkrétních možností GA. Vytvoříme si strukturu možností, kde definujeme reprezentaci populace a její velikost.

```
options = gaoptimset('PopulationType',
,custom','PopInitRange', ...
    [1;cities]);
```

Kód 17 - Nastavení možností GA 1 [7]

Vzhledem k tomu, že máme vlastní reprezentaci jedinců, je potřeba GA nadefinovat vlastní vytvářecí, křížící, mutační a fitness funkci a také je třeba nastavit, kdy má algoritmus skončit.

```
options =
gaoptimset(options,'CreationFcn',@TSP_create, ...
    'FitnessScalingFcn',@fitscalingrank, ...
    'SelectionFcn',@selectiontournament, ...
    'CrossoverFcn',@TSP_crossover, ...
    'MutationFcn',@TSP_mutate, ...
    'PlotFcn', my_plot, ...
    'Generations',500,'PopulationSize',2000, ...
    'StallGenLimit',200,'Vectorized','on');
```

Kód 18 - Nastavení GA 2 [7]

Poté již jen nadefinujeme počet proměnných.

```
numberOfVariables = cities;
```

Kód 19 - Počet proměnných [7]

a můžeme spustit samotný algoritmus.

```
tic;
[x,fval,reason,output] =
ga(FitnessFcn,numberOfVariables,options)
toc
```

Kód 20 - Spuštění algoritmu [7]

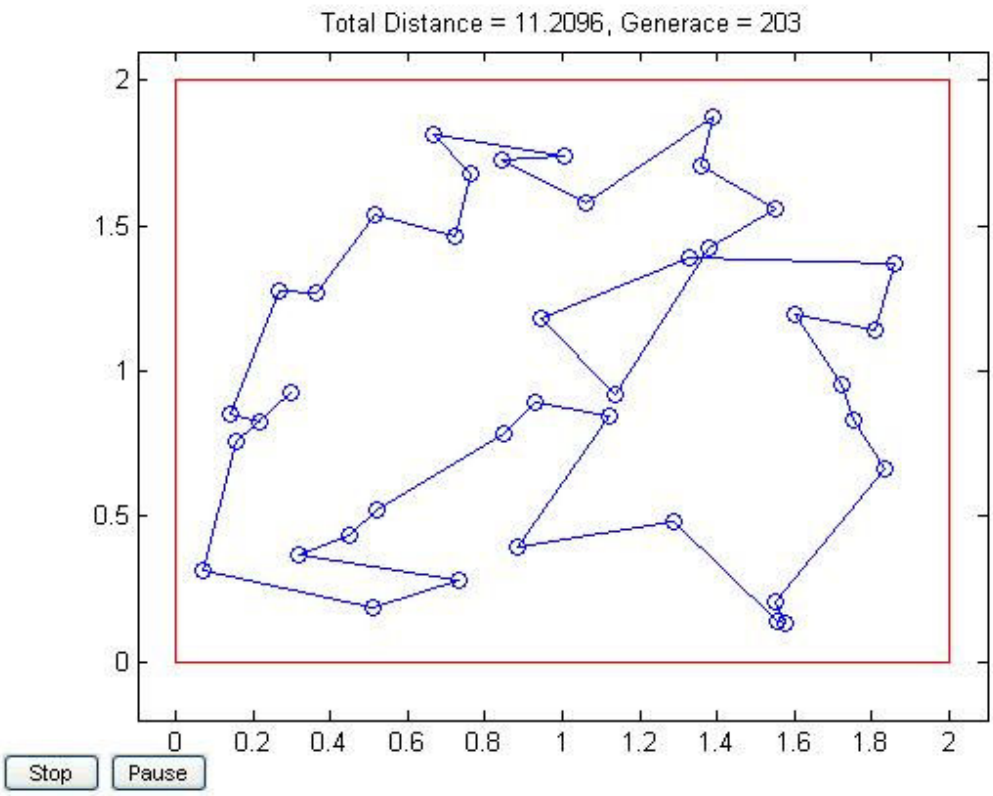
3.2. Výsledky při základním nastavení

Při základním nastavení je počet měst 40 a velikost populace 60, algoritmus se nejdříve zastaví po 200 generaci a maximálně při 500 generaci.

```
x =     [1x40 double]

fval =   11.7408

reason =     1

output =    problemtype: 'unconstrained'

                rngstate: [1x1 struct]

                generations: 205

                funccount: 12360

                message: [1x86 char]
```

Kód 21 - Výsledek TSP při základním nastavení [zdroj vlastní]

Grafické zobrazení výsledků vypadá následovně:



Graf 8 - Grafické zobrazení výsledku při základním nastavení [zdroj vlastní]

3.3. Změny parametrů GA

Nyní přejděme k samotnému testování algoritmu. Nejprve budeme sledovat vliv počtu měst na algoritmus, kde při každé změně počtu měst budeme vytvářet novou matici souřadnic. Souřadnice pro hodnotu 100 měst si uložíme pro pokračování testování dalších parametrů. Dále se pak budeme zabývat vlivem velikostí populace, počtu generací a nakonec možností nastavení GA v toolboxu. S každým nastavením budeme provádět 5 pokusů při stejném rozložení měst.

Vzdálenost je hodnota nejkratší cesty mezi městy, kterou algoritmus dosáhl. Čas uvádí dobu provádění algoritmu a hodnota generace udává generaci, u které se algoritmus zastavil, protože změny nejlepší hodnoty fitness byly již tak malé, že nemělo smysl pokračovat a nebo dosáhl maximálního počtu generací. Z těchto 10 pokusů je pak vybírán nejhorší a nejlepší výsledek, z nichž je dále počítán rozdíl v procentech. Průměrný rozdíl hodnot je počítán jako průměrná procentuální odchylka od nejlepšího výsledku zbylých 9 hodnot. Kompletní tabulky jsou kvůli přehlednosti uvedeny v příloze. Ke každé proměnné je uvedena tabulka obsahující nejlepší a nejhorší výsledek, jejich rozdíl a relativní chybu.

3.3.1. Počet proměnných

Jak je již uvedeno výše, prvním z parametrů, u kterého jsem se rozhodl sledovat jeho vliv na chování GA, je počet vstupních proměnných (měst). Sledovat budeme odchylku mezi nejlepším a nejhorším výsledkem a také vliv počtu měst na časovou náročnost algoritmu. Velikost populace byla v základním nastavení tj. 60, minimální počet generací byl 200 a maximální 500. Vybral jsem si hodnoty 40, 100, 1 000 a 5000 měst (při vetší hodnotě jsem se potýkal s nedostatkem paměti).

Když se podíváme na výsledky (Příloha - Tabulka 1, 2), tak u počtu měst 40 je rozdíl mezi nejlepší a nejhorší vzdáleností 1,3280, u 100 měst je to 5,8651, u 1000 měst 110,5393 a nakonec u 5 000 měst 215,3. Z toho tedy vyplývá, že s počtem měst absolutní chyba měření roste, ale u hodnoty 5 000 měst rapidně klesá chyba relativní. Co se týče časové náročnosti, v souladu s očekáváním, s počtem měst roste i doba potřebná k provedení algoritmu. Také větší počet měst při tomto nastavení potřebuje více generací k nalezení optima (nejlepší fitness hodnoty).

Tabulka 3 - Vliv počtu proměnných detail na nejlepší a nejhorší výsledek [zdroj vlastní]

Počet proměnných	40	100	500	1000
nejlepší výsledek	10,7731	32,8720	239,4245	610,6246
nejhorší výsledek	12,1011	38,7371	259,6350	721,1939
Vzdálenostní rozdíl	1,328	5,8651	20,2105	110,569
relativní chyba	12,33%	17,84%	8,44%	18,11%
Počet proměnných	2000	3000	4000	5000
nejlepší výsledek	1496,4000	2416,4000	3335,1000	4388,3000
nejhorší výsledek	1582,2000	2557,1000	3550,1000	4603,6000
Vzdálenostní rozdíl	85,8	140,7	215	215,3
relativní chyba	5,73%	5,82%	6,45%	4,91%

3.3.2. Velikost populace

Dále byl zkoumán vliv velikosti populace, se kterou pracuje GA. Počet měst v níže uvedených testech bude 100. Minimální a maximální počet generací zůstává nezměněn. Je pozorováno, jaký vliv má velikost populace na odchylku mezi nejlepším a nejhorším výsledkem.

Z tabulky vlivu populace (Příloha - Tabulka 3, 4) je na první pohled je zřejmé, že velikost populace má výrazný vliv na kvalitu výsledku. Ovšem zcela logicky se zvětšující se velikostí populace se zvyšuje i časová náročnost výpočtu. Tento parametr má velký vliv na kvalitu dosažených výsledků. Například rozdíl mezi nejlepším a nejhorším výsledkem všech 20 měření je více než dvojnásobný. Pro větší přehled jsem hodnotu odchylky uvedl do následující tabulky, kde je jasně vidět, že odchylka při populaci 10 000 je skoro 6x menší než odchylka při 60. Rovněž je zde i vidět vliv tohoto parametru na kvalitu dosažených výsledků.

Tabulka 4 - Vliv velikosti populace detail na nejlepší a nejhorší výsledek [zdroj vlastní]

Velikost populace	60	100	500	1000
nejlepší výsledek	31,8342	27,0856	20,7245	19,0117
nejhorší výsledek	38,4226	33,6083	25,5690	22,3953
Vzdálenostní rozdíl	6,5884	6,5227	4,8445	3,3836
relativní chyba	20,70%	24,08%	23,38%	17,80%
Velikost populace	2000	5000	7000	10000
nejlepší výsledek	17,8035	16,5848	15,9296	15,5164
nejhorší výsledek	19,9287	18,1330	17,8713	16,6164
Vzdálenostní rozdíl	2,1252	1,5482	1,9417	1,1
relativní chyba	11,94%	9,34%	12,19%	7,09%

3.3.3. Počet generací

Dalším sledovaným parametrem je vliv počtu generací. Z předchozích dvou parametrů zůstává počet měst na hodnotě 100 a pro velikost populace jsem zvolil hodnotu 1000. V tomto případě je sledováno, jaký vliv má počet generací na odchylku mezi nejlepším a nejhorším výsledkem.

Při pohledu na tabulku vlivu generací na vlastnosti GA (Příloha – Tabulka 5, 6) je zřejmé, že s počtem generací roste i časová náročnost. Počet generací má pozitivní vliv na kvalitu výsledku, ale rozdíl mezi hodnotou 1 000 a 2 000 generací je již minimální. Při počtu generací 1 700 je získán nejlepší výsledek s tím, že odchylka je vetší než u předchozích nastavení.

Tabulka 5 - Vliv počtu generací detail na nejlepší a nehorší výsledek [zdroj vlastní]

Počet generací	200	500	700	1000
nejlepší výsledek	20,6755	16,0294	15,6877	15,5287
nejhorší výsledek	24,0813	17,5843	17,2506	16,2120
Vzdálenostní rozdíl	3,4058	1,5549	1,5629	0,6833
relativní chyba	16,47%	9,70%	9,96%	4,40%
Počet generací	1200	1500	1700	2000
nejlepší výsledek	15,6833	15,0135	15,6698	15,0753
nejhorší výsledek	16,5075	16,6239	16,2066	16,5261
Vzdálenostní rozdíl	0,8242	1,6104	0,5368	1,4508
relativní chyba	5,26%	10,73%	3,43%	9,62%

Nyní budeme využívat vestavěných možností toolboxu, jako je počet přenesených elitních jedinců do následující generace, druh selekčního mechanismu a podíl křížení a mutace. Na závěr se pak pokusíme kombinací všech testovaných parametrů docílit co nejkvalitnějšího výsledku a nejmenší odchylky. Jako kompromis mezi časovou náročností a spolehlivostí algoritmu jsem zvolil následující parametry.

- Počet Měst: 100
- Velikost populace: 1 000
- Minimální počet generací: 1 000
- Maximální počet generací: 2 000

3.3.4. Selekční mechanismus

Použité selekční mechanismy:

Stochastic Uniform - Je to základní selekční funkce GA v Matlabu, podobný ruletovému mechanismu.

Remainder - Zbytková selekce.

Roulete - Klasický ruletový mechanismus.

Uniform - Mechanismus vhodný spíše pro testování funkčnosti GA a případné úpravy kódu pro praktické využití nevhodný..

Tournament - Turnajový selekční mechanismus.

Kvůli přehlednosti jsem selekční mechanismy rozdělil do dvou tabulek. V druhé z nich jsou pouze varianty turnajového mechanismu, kdy do turnaje vstupují 2, 4, 16 a nebo 32 jedinců.

V následujících tabulkách (Příloha – Tabulka 7 a 8) je sledován vliv selekčního mechanismu na chování GA. Z první tabulky lze vyčíst, že o první místo se dělí mechanismus Remainder a Roulete, přičemž první jmenovaný je nepatrně lepší. V tabulce druhé jsou uvedeny 4 varianty posledního mechanismu a to turnajového, kdy do turnaje vstupují 2, 4, 16 a nebo 32 jedinců. Počet jedinců vstupujících do turnaje výrazně ovlivňuje kvalitu výsledku a časovou náročnost. Dobrý vliv má i na odchylku mezi jedinci.

Tabulka 6 - Vliv selekčního mechanismu detail na nejlepší a nejhorší výsledek 1 [zdroj vlastní]

Selekční mechanismus	Stochastic Uniform	Remainder	Roulete	Uniform
nejlepší výsledek	15,0817	15,3705	15,3974	28,7202
nejhorší výsledek	16,9352	16,5993	16,6925	33,3667
vzdálenostní odchylka	1,8535	1,2288	1,2951	4,6465
relativní chyba	12,29%	7,99%	8,41%	16,18%

Tabulka 7 - Vliv selekčního mechanismu detail na nejlepší a nejhorší výsledek 2 [zdroj vlastní]

Selekční mechanismus	Tournament 2	Tournament 4	Tournament 16	Tournament 32
nejlepší výsledek	22,3266	17,7356	15,0634	14,7875
nejhorší výsledek	24,4745	19,4314	16,5747	16,1061
vzdálenostní odchylka	2,1479	1,6958	1,5113	1,3186
relativní chyba	9,62%	9,56%	10,03%	8,92%

3.3.5. Počet elitních jedinců

V následující tabulce (Příloha - Tabulka 9) je sledován vliv elitních jedinců, kteří postupují rovnou do další generace, aniž by na ně byly uplatněny operátory mutace nebo křížení. Selekční mechanismus je v základním nastavení.

Při velikosti populace 1 000 se jeví jako nejlepší počet jedinců 8, protože podává o 4 desetiny lepší výsledky než při počtu 4. Při porovnání časové náročnosti lze konstatovat, že jednotlivá nastavení na ní nemají vliv.

Tabulka 8 - Vliv elitních jedinců detail na nejlepší a nejhorší výsledek [zdroj vlastní]

Elitní jedinci	2	4	8	16
nejlepší výsledek	15,6266	15,4244	15,0317	15,2073
nejhorší výsledek	17,1114	16,4665	16,0515	16,7389
Vzdálenostní odchylka	1,4848	1,0421	1,0198	1,5316
relativní chyba	9,50%	6,76%	6,78%	10,07%

3.3.6. Crossover fraction

Dále následuje možnost crossover fraction (Příloha – Tabulka 10), což je poměr mezi mutací a křížením. Je to číslo v intervalu <0;1>, kde 0 znamená žádné křížení a naopak 1 žádnou mutaci. Toto se vztahuje na každého jedince vyjma elitních.

Pokud je pro nás podstatná hlavně kvalita výsledku, potom je nejlepší možností nastavení hodnoty na 0,8, kdy bylo dosaženo nejlepších výsledků. Vzdálenostní odchylka však stoupla.

Tabulka 9 - Vliv crossover fraction detail na nejlepší a nejhorší výsledek [zdroj vlastní]

Crossover fraction	0,2	0,4	0,6	0,8
nejlepší výsledek	16,0529	15,9109	15,7716	15,3377
nejhorší výsledek	18,1719	17,5671	16,8677	16,6306
Vzdálenostní odchylka	2,119	1,6562	1,0961	1,2929
relativní chyba	13,20%	10,41%	6,95%	8,43%

3.3.7. Kombinace parametrů

Jak jsem předesílal na začátku kapitoly, pokusím se zvolit kombinaci parametrů takovou, abych získal nejkvalitnější výsledek s minimální odchylkou v přiměřeném čase. V kódu (Kód 22) je uvedena finální konfigurace algoritmu s tím, že verze 1 používá turnajový selekční mechanismus s 32 jedinci a verze 2 se 64 jedinci. U verze 3 jsem použil nastavení Crossover fraction 0,6 a turnajový mechanismus s 32 jedinci.

```
PopulationSize: 1000

EliteCount: 8

CrossoverFraction: (0.8000 / 0.6000)

Generations: 2000

StallGenLimit: 1000

SelectionFcn: @selectiontournament,  (32/64)
```

Kód 22 - Finální konfigurace algoritmu [zdroj vlastní]

Tabulka 10 - Vliv kombinace parametru detail na nejlepší a nejhorší výsledek [zdroj vlastní]

Verze	1	2	3
nejlepší výsledek	14,8914	14,7208	14,8886
nejhorší výsledek	15,6980	15,9239	15,8101
Vzdálenostní odchylka	0,8066	1,2031	0,9215
relativní chyba	5,42%	8,17%	6,19%

Při kombinaci všech parametrů nastaveni GA jsou rozdíly minimální (Příloha – Tabulka 11), těžko vybrat nejlepší z nich. Verze 1 má nejmenší odchylku, verze 2 nejlepší hodnotu a verze 3 nejlepší průměrnou hodnotu všech výsledků, ale jelikož má verze 1 nejmenší nejhorší výsledek považuji toto nastavení za nejkvalitnější z testovaných.

4. Závěr

Tato bakalářská práce se věnovala problému vlivu parametrů genetického algoritmu na jeho chování.

V první části práce byl popsán genetický algoritmus, jeho vlastnosti a parametry. Byly probrány základní pojmy používané v této problematice a možnosti zlepšení jeho výsledků.

V druhé části byla ukázána jeho implementace v Matlabu, kde byla pomocí toolboxu realizována Rastingerova funkce, na které jsou ukázány základní přednosti genetického algoritmu při vyhledávání globálního minima v přítomnosti mnoha lokálních minim. Byla také ukázána možnost grafického znázornění průběhu algoritmu.

Třetí část se nejprve věnovala implementaci Problému Obchodního cestujícího do prostředí Matlabu. Vzhledem k tomu, že nelze použít ani jednu přednastavenou možnost reprezentace jedinců implementovanou přímo programem, bylo nutné vytvořit vlastní reprezentaci a k tomu také funkci vytvářející populaci, křížící a mutační funkci. Po ukázkách jednotlivých funkcí a předvedení běhu algoritmu byl v následující časti řešen vliv vstupních parametrů algoritmu na jeho chování, kde se postupně testovaly parametry jako jsou: počet vstupních dat, velikost populace, množství generací. Dále pak možností nastavení v toolboxu a to: selekční mechanismus, počet postupujících elitních jedinců do následující generace a poměr mezi křížením a mutací, kde u každého parametru byly vybrány 4 možnosti nastavení a s každou z nich bylo provedeno 5 testovacích průběhů, u kterých byla sledována doba provedení, kvalita výsledku a rozdíl mezi nejlepším a nejhorším výsledkem. Na konci kapitoly jsem se pak pokusil kombinací jednotlivých parametrů dosáhnout co nejlepšího výsledku v přiměřeném čase.

Při hodnocení výsledků jsem vycházel z počtu vstupních měst 100, kde při prvním nastavení algoritmu byl nejlepší výsledek 32,8720 a nejhorší 38,7371, algoritmus byl zpracován průměrně za 5 desetin vteřiny. Když bylo naopak použito nastavení poslední, čili kombinace všech parametrů „Verze 1“, bylo dosaženo 14,8914 jakožto nejlepšího výsledku a 15,6980 jakožto nejhoršího průměrně za 2,5 minuty. Ač

je časový nárůst obrovský, tak kvalita výsledku je na druhou stranu mnohem lepší a odchylka jednotlivých měření se také snížila. Při nastavení ještě vyšších hodnot u parametrů velikosti populace, selekční metody a k tomu správně určeného počtu generací by se přesnost algoritmu jistě ještě navýšila samozřejmě za cenu časovou, ale i ta by se dala eliminovat případným použitím lepšího HW. Určitě je zde i možnost úpravy mutační a křížící funkce, která by taky mohla kladně přispět k výkonnosti a výsledkům algoritmu.

Toto nastavení parametrů může být samozřejmě absolutně nevhodné pro jiné problémy. Každý problém potřebuje svůj osobitý přístup a konkrétní nastavení parametrů pro jeho průběh.

Genetický algoritmus jako takový je zakomponován do celku evolučních výpočetních technik. Jednou z nich je genetické programování, které velmi rozšiřuje jeho možnosti použití.

Použitá literatura

[1] MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J. Umělá inteligence (3). Academia, Praha, 2001. 80-200-0472-6.

[2] HYNEK, Josef. Genetické algoritmy a genetické programování. [s.l.] : Grada Publishing, a.s., 2008. 179 s. ISBN 978-80-247-2695-3.

[3] John Henry Holland [online]. [cit. 2009-08-02]. Dostupný z WWW: <http://krasnow.gmu.edu/images/DOM%20Photos/Vita%2007.doc>.

[4] OBITKO, Marek. *Genetické algoritmy – matrice života v počítačích* [online]. [cit. 2009-08-02]. Dostupný z WWW: <http://www.scienceworld.cz/sw.nsf/ID/9B80774399F1B837C1256E9700489AEC?OpenDocument&cast=1>.

[5] TEDA, Jaroslav. Genetické algoritmy a jejich aplikace v praxi [online]. [cit. 2009-08-02]. Dostupný z WWW: <http://www.programujte.com/view.php?cisloclanku=2005072601>.

[6] WEISE, Thomas. Global Optimization Algorithms : Theory and Application. [s.l.] :[s.n.], c2008. 686 s. 2. [cit. 2009-08-02]. Dostupný z WWW: <http://www.itweise.de/projects/book.pdf>.

[7] Autor neuveden. Stránky skupiny Humusoft. *Matlab, popis produktu* [on-line].9.8.2009. Dostupné z <http://www.humusoft.cz/matlab/matlab.htm>.

[8] BLAŽEK, J., HABIBBALA, V., PAVLISKA, V. *Vybrané heuristiky pro globální optimalizaci a jejich implementace v Matlabu* [on-line]. 9.8.2009. Dostupné z <http://www.volny.cz/habiballa/publ/matlab05.pdf>.

Seznam příloh

Příloha - Tabulka 1- Vliv počtu proměnných 1 [zdroj vlastní]

Počet proměnných	40			100			500			1000		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas (sec)	generace
pokus č.1	11,5193	0,6	201	35,5567	0,7	210	252,2475	1,7	416	615,2168	3,2	500
pokus č.2	11,6375	0,4	203	34,6825	0,4	201	251,0266	1,6	400	625,2629	3,3	500
pokus č.3	11,5056	0,4	201	33,9040	0,6	241	259,6350	1,5	366	704,9576	2,1	339
pokus č.4	11,9941	0,4	201	38,7371	0,5	202	245,1933	1,7	405	610,6246	3,1	500
pokus č.5	11,9974	0,3	202	32,8720	0,5	204	239,4245	1,9	452	688,5966	2,3	365
pokus č.6	12,1011	0,6	204	37,1213	0,7	210	256,6407	1,6	390	721,1939	2,1	297
pokus č.7	11,6604	0,3	204	33,5290	0,5	201	251,5316	1,7	400	627,6659	2,9	485
pokus č.8	11,1523	0,6	204	33,7762	0,4	220	239,7487	1,7	430	612,0728	3,2	500
pokus č.9	11,6173	0,3	201	34,0071	0,7	228	248,2886	1,6	396	622,0173	3,4	500
pokus č.10	10,7731	0,6	210	34,8870	0,5	219	243,8932	2,9	430	634,1960	2,9	459
nejlepší výsledek	10,7731	0,3	201	32,8720	0,4	201	239,4245	1,5	366	610,6246	2,1	297
nejhorší výsledek	12,1011	0,6	210	38,7371	0,7	241	259,6350	2,9	452	721,1939	3,4	500
relativní chyba	12,33%	100,00%	4,48%	17,84%	75,00%	19,90%	8,44%	93,33%	23,50%	18,11%	61,90%	68,35%
průměrný rozdíl	7,64%	50,00%	1,04%	6,19%	37,50%	6,27%	3,90%	19,33%	11,61%	5,82%	35,71%	49,66%

Příloha - Tabulka 2 - Vliv počtu proměnných 2 [zdroj vlastní]

Počet proměnných	2000			3000			4000			5000		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace
pokus č.1	1505,6	5,8	500	2424,7	10,2	500	3361,6	17,8	500	4568,9	14,7	265
pokus č.2	1582,2	4,4	373	2522,6	8,1	400	3505,7	11,7	323	4492,0	18,4	332
pokus č.3	1509,8	5,7	500	2418,7	10,2	500	3550,1	11,0	307	4564,4	13,9	250
pokus č.4	1502,2	5,7	500	2511,8	8,3	405	3369,1	17,9	500	4564,5	13,6	246
pokus č.5	1502,8	5,8	500	2435,1	10,2	500	3348,0	17,9	500	4522,1	17,4	303
pokus č.6	1540,9	5,1	434	2416,4	10,2	500	3446,0	15,3	426	4404,2	22,9	408
pokus č.7	1502,2	5,9	500	2433,5	10,2	500	3393,2	16,7	462	4388,3	23,9	423
pokus č.8	1496,4	5,8	500	2422,8	10,3	500	3353,9	18,0	500	4512,4	17,6	314
pokus č.9	1502,6	6,1	500	2557,1	7,4	358	3405,5	16,5	463	4390,6	23,5	413
pokus č.10	1503,3	5,8	500	2431,9	10,3	500	3335,1	17,9	500	4603,6	13,1	230
nejlepší výsledek	1496,4000	4,4	373	2416,4000	7,4	358	3335,1000	11,0	307	4388,3000	13,1	230
nejhorší výsledek	1582,2000	6,1	500	2557,1000	10,3	500	3550,1000	18,0	500	4603,6000	23,9	423
relativní chyba	5,73%	38,64%	34,05%	5,82%	39,19%	39,66%	6,45%	63,64%	62,87%	4,91%	82,44%	83,91%
průměrný rozdíl	1,23%	27,50%	28,87%	1,70%	28,92%	30,25%	2,15%	46,09%	45,96%	2,57%	36,64%	38,43%

Příloha - Tabulka 3 - Vliv velikosti populace 1 [zdroj vlastní]

Velikost populace	60			100			500			1000		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas (sec)	generace
pokus č.1	36,3517	0,3	216	31,7494	0,7	222	22,2907	2,8	218	19,6234	5,5	222
pokus č.2	37,3809	0,7	201	31,5889	0,7	230	24,0302	2,7	208	22,3503	5,4	206
pokus č.3	36,0759	0,3	237	29,3471	1,0	236	22,5785	2,6	203	20,1724	5,8	230
pokus č.4	31,8342	0,3	237	31,7946	0,7	220	25,5690	2,8	214	19,0117	5,6	222
pokus č.5	33,6316	0,5	228	30,3845	0,7	224	20,7245	3,3	250	19,8528	5,8	232
pokus č.6	38,4226	0,5	215	33,6083	0,7	217	23,6165	3,1	244	20,9815	5,3	214
pokus č.7	35,0147	0,4	201	32,9391	0,7	208	21,2076	3,5	275	22,3953	5,1	206
pokus č.8	35,3795	0,5	219	27,5459	0,6	262	24,9037	2,8	216	20,3366	5,4	225
pokus č.9	33,8771	0,4	207	32,6013	0,7	215	23,5757	2,9	222	20,5452	6,3	241
pokus č.10	35,4625	0,7	207	27,0856	0,8	266	21,2805	3,6	277	20,4838	5,5	221
nejlepší výsledek	31,8342	0,3	201	27,0856	0,6	208	20,7245	2,6	203	19,0117	5,1	206
nejhorší výsledek	38,4226	0,7	237	33,6083	1,0	266	25,5690	3,6	277	22,3953	6,3	241
relativní chyba	20,70%	133,33%	17,91%	24,08%	66,67%	27,88%	23,38%	38,46%	36,45%	17,80%	23,53%	16,99%
průměrný rozdíl	11,02%	53,33%	7,86%	13,95%	21,67%	10,58%	10,87%	15,77%	14,63%	8,22%	9,22%	7,72%

Příloha - Tabulka 4 - Vliv velikosti populace 2 [zdroj vlastní]

Velikost populace	2000			5000			7000			10000		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace
pokus č.1	19,9287	10,1	205	17,2628	26,0	213	15,9296	38,2	223	16,6164	51,3	214
pokus č.2	18,2499	12,1	246	17,4866	25,0	205	16,5209	36,4	212	16,5044	50,6	209
pokus č.3	18,2987	12,0	244	16,7751	25,6	210	17,2304	35,6	207	16,2746	50,4	210
pokus č.4	19,4856	11,1	226	17,3615	25,3	207	16,7864	34,6	201	15,6040	51,3	214
pokus č.5	17,8035	11,2	227	17,0035	31,8	261	17,0974	39,6	232	15,5218	48,7	202
pokus č.6	19,2335	10,2	207	17,7598	27,9	229	17,8713	34,3	201	15,5164	52,2	516
pokus č.7	18,9952	9,9	201	17,3378	25,7	210	17,3618	35,2	206	15,5765	51,8	212
pokus č.8	18,1869	10,4	211	18,1330	24,9	204	16,7861	36,4	213	16,1120	52,9	213
pokus č.9	18,9464	10,6	217	16,5848	25,3	208	16,3042	37,0	217	16,5142	52,3	216
pokus č.10	19,0238	10,1	206	17,1664	25,9	212	16,7078	36,7	215	16,4871	49,3	202
nejlepší výsledek	17,8035	9,9	201	16,5848	24,9	204	15,9296	34,3	201	15,5164	48,7	202
nejhorší výsledek	19,9287	12,1	246	18,1330	31,8	261	17,8713	39,6	232	16,6164	52,9	516
relativní chyba	11,94%	22,22%	22,39%	9,34%	27,71%	27,94%	12,19%	15,45%	15,42%	7,09%	8,62%	155,45%
průměrný rozdíl	5,68%	8,79%	8,96%	4,23%	5,78%	5,83%	5,84%	6,12%	5,82%	3,59%	4,89%	19,21%

Příloha - Tabulka 5 - Vliv počtu generací 1 [zdroj vlastní]

Počet generací	200			500			700			1000		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas (sec)	generace
pokus č.1	22,9323	5,0	200	17,5843	12,6	500	16,3117	17,6	700	15,8734	25,6	1000
pokus č.2	22,4696	4,9	200	16,2614	13,1	500	15,6877	17,6	700	16,0710	25,6	1000
pokus č.3	22,6392	5,0	200	16,2668	12,8	500	15,8284	17,6	700	15,9796	26,1	1000
pokus č.4	21,4713	5,0	200	17,4845	13,1	500	16,3990	17,6	700	15,7117	26,0	1000
pokus č.5	21,9608	5,0	200	16,2088	13,2	500	16,6213	17,6	700	16,0654	30,2	1000
pokus č.6	20,6755	4,9	200	17,1670	13,1	500	16,3046	17,6	700	16,1004	26,1	1000
pokus č.7	22,6320	4,7	200	16,5650	13,0	500	17,2506	17,6	700	15,5287	26,2	1000
pokus č.8	24,0813	5,3	200	16,1728	13,0	500	16,0263	17,6	700	16,1863	25,8	1000
pokus č.9	21,4781	5,0	200	16,8208	12,7	500	17,0991	17,6	700	15,6683	25,7	1000
pokus č.10	22,3855	5,1	200	16,0294	12,7	500	16,0083	17,6	700	16,2120	25,5	1000
nejlepší výsledek	20,6755	4,7	200	16,0294	12,6	500	15,6877	17,6	700	15,5287	25,5	1000
nejhorší výsledek	24,0813	5,3	200	17,5843	13,2	500	17,2506	17,6	700	16,2120	30,2	1000
relativní chyba	16,47%	12,77%	0,00%	9,70%	4,76%	0,00%	9,96%	0,00%	0,00%	4,40%	18,43%	0,00%
průměrný rozdíl	7,72%	6,17%	0,00%	3,91%	2,62%	0,00%	4,25%	0,00%	0,00%	2,65%	3,06%	0,00%

Příloha - Tabulka 6 - Vliv počtu generací 2 [zdroj vlastní]

Počet generací	1200			1500			1700			2000		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace
pokus č.1	15,7575	30,1	1200	15,5634	37,7	1500	16,1859	42,7	1700	16,0484	52,7	2000
pokus č.2	15,7365	30,2	1200	15,4692	37,7	1500	15,7184	42,8	1700	16,1552	51,5	2000
pokus č.3	15,9177	30,2	1200	16,6239	37,7	1500	15,7919	42,7	1700	15,2099	52,3	2000
pokus č.4	15,6833	30,2	1200	15,1436	37,7	1500	15,6698	42,7	1700	15,0753	51,6	2000
pokus č.5	15,6976	30,2	1200	16,4551	37,7	1500	16,2066	42,7	1700	16,0065	51,3	2000
pokus č.6	16,2969	30,2	1200	15,6932	37,7	1500	15,7241	42,7	1700	15,4431	51,9	2000
pokus č.7	16,0579	30,2	1200	16,4458	37,7	1500	15,9444	42,7	1700	15,1862	53,0	2000
pokus č.8	16,5075	30,2	1200	15,0135	37,7	1500	15,8348	42,8	1700	15,3498	52,4	2000
pokus č.9	16,0900	30,3	1200	16,0733	37,7	1500	16,1074	42,8	1700	16,5261	51,6	2000
pokus č.10	16,3494	30,2	1200	16,1216	37,7	1500	16,1528	42,7	1700	15,3263	51,7	2000
nejlepší výsledek	15,6833	30,1	1200	15,0135	37,7	1500	15,6698	42,7	1700	15,0753	51,3	2000
nejhorší výsledek	16,5075	30,3	1200	16,6239	37,7	1500	16,2066	42,8	1700	16,5261	53,0	2000
relativní chyba	5,26%	0,66%	0,00%	10,73%	0,00%	0,00%	3,43%	0,23%	0,00%	9,62%	3,31%	0,00%
průměrný rozdíl	2,08%	0,33%	0,00%	5,64%	0,00%	0,00%	1,68%	0,07%	0,00%	3,70%	1,36%	0,00%

Příloha - Tabulka 7 - Vliv selekčního mechanismu 1 [zdroj vlastní]

Selekční mechanismus	Stochastic Uniform			Remainder			Roulete			Uniform		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace
pokus č.1	15,5998	25,5	1001	15,9497	67,3	1001	15,5229	47,2	1001	29,1815	25,0	1001
pokus č.2	16,2750	25,5	1001	15,3705	68,1	1001	15,8338	47,9	1001	33,3667	24,0	1001
pokus č.3	15,5639	24,7	1001	16,1913	67,9	1001	16,6925	47,2	1001	31,2543	25,5	1001
pokus č.4	15,0817	25,1	1001	15,7576	68,1	1001	15,8379	47,6	1001	31,5699	25,0	1001
pokus č.5	16,1778	24,8	1001	15,7352	68,0	1001	16,2756	47,6	1001	29,1756	24,8	1001
pokus č.6	16,9352	24,5	1001	16,3556	68,2	1001	15,3974	47,6	1001	31,0448	25,1	1001
pokus č.7	16,3257	24,8	1001	16,5993	67,9	1001	15,6132	48,3	1001	28,7202	25,4	1001
pokus č.8	15,6057	25,2	1001	16,1037	69,0	1001	16,1574	47,0	1001	30,0984	24,8	1001
pokus č.9	15,7280	24,5	1001	15,8288	70,2	1001	15,8748	47,9	1001	30,0058	25,2	1001
pokus č.10	15,2215	24,8	1001	16,3880	68,7	1001	15,5457	49,1	1001	30,3802	24,8	1001
nejlepší výsledek	15,0817	24,5	1001	15,3705	67,3	1001	15,3974	47,0	1001	28,7202	24,0	1001
nejhorší výsledek	16,9352	25,5	1001	16,5993	70,2	1001	16,6925	49,1	1001	33,3667	25,5	1001
relativní chyba	12,29%	4,08%	0,00%	7,99%	4,31%	0,00%	8,41%	4,47%	0,00%	16,18%	6,25%	0,00%
průměrný rozdíl	5,10%	1,80%	0,00%	4,28%	1,55%	0,00%	3,10%	1,57%	0,00%	6,13%	4,00%	0,00%

Příloha - Tabulka 8 - Vliv selekčního mechanismu 2 [zdroj vlastní]

Selekční mechanismus	Tournament 2			Tournament 4			Tournament 16			Tournament 32		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace
pokus č.1	23,7704	35,1	1001	18,3702	44,4	1001	15,6214	94,3	1001	15,0795	161,1	1001
pokus č.2	24,0149	35,2	1001	18,3974	44,2	1001	16,5747	96,1	1001	15,6958	158,6	1001
pokus č.3	23,8200	34,9	1001	19,1245	44,3	1001	16,2917	94,9	1001	15,2045	158,0	1001
pokus č.4	23,3089	34,9	1001	19,0707	44,0	1001	15,9506	95,3	1001	15,1993	157,8	1001
pokus č.5	22,8614	35,0	1001	17,7356	46,8	1001	15,8861	98,4	1001	14,7875	158,0	1001
pokus č.6	23,6259	34,5	1001	17,8492	44,1	1001	15,5460	93,7	1001	15,4485	158,1	1001
pokus č.7	24,4745	34,6	1001	19,1452	45,3	1001	15,8497	95,0	1001	16,1061	157,5	1001
pokus č.8	22,9785	35,1	1001	19,2061	44,7	1001	15,9272	94,3	1001	15,2846	158,8	1001
pokus č.9	23,9655	34,6	1001	18,3749	43,9	1001	15,0634	93,0	1001	15,5659	158,2	1001
pokus č.10	22,3266	35,3	1001	19,4314	44,0	1001	15,7213	95,4	1001	15,8540	158,3	1001
nejlepší výsledek	22,3266	34,5	1001	17,7356	43,9	1001	15,0634	93,0	1001	14,7875	157,5	1001
nejhorší výsledek	24,4745	35,3	1001	19,4314	46,8	1001	16,5747	98,4	1001	16,1061	161,1	1001
relativní chyba	9,62%	2,32%	0,00%	9,56%	6,61%	0,00%	10,03%	5,81%	0,00%	8,92%	2,29%	0,00%
průměrný rozdíl	5,32%	1,22%	0,00%	5,27%	1,53%	0,00%	5,18%	2,19%	0,00%	4,29%	0,60%	0,00%

Příloha - Tabulka 9 - Vliv elitních jedinců [zdroj vlastní]

Elitní jedinci	2			4			8			16		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace
pokus č.1	17,1114	24,8	1001	15,4244	25,5	1001	15,2877	25,2	1001	15,8992	24,8	1001
pokus č.2	15,6873	25,2	1001	15,8343	25,6	1001	15,8894	26,2	1001	15,2073	24,7	1001
pokus č.3	17,0625	24,9	1001	15,8608	24,9	1001	15,7690	27,2	1001	15,4223	25,2	1001
pokus č.4	15,6266	25,3	1001	15,7701	24,5	1001	15,9342	27,9	1001	16,7389	24,4	1001
pokus č.5	15,9115	24,6	1001	15,6477	24,5	1001	15,0317	26,2	1001	16,0080	24,9	1001
pokus č.6	16,2714	24,7	1001	15,4477	24,6	1001	15,5750	24,5	1001	15,4970	24,9	1001
pokus č.7	16,3064	25,9	1001	15,8971	24,6	1001	16,0515	24,9	1001	15,9057	25,4	1001
pokus č.8	16,4204	24,9	1001	16,1957	24,8	1001	15,3048	25,3	1001	15,7176	24,8	1001
pokus č.9	15,7779	25,6	1001	15,9294	24,3	1001	15,4692	24,4	1001	16,1067	25,3	1001
pokus č.10	16,3597	25,4	1001	16,4665	26,1	1001	15,5208	24,8	1001	15,6410	25,1	1001
nejlepší výsledek	15,6266	24,6	1001	15,4244	24,3	1001	15,0317	24,4	1001	15,2073	24,4	1001
nejhorší výsledek	17,1114	25,9	1001	16,4665	26,1	1001	16,0515	27,9	1001	16,7389	25,4	1001
relativní chyba	9,50%	5,28%	0,00%	6,76%	7,41%	0,00%	6,78%	14,34%	0,00%	10,07%	4,10%	0,00%
průměrný rozdíl	4,01%	2,15%	0,00%	2,74%	2,63%	0,00%	3,67%	5,16%	0,00%	3,99%	2,25%	0,00%

Příloha - Tabulka 10 - Vliv Crossover Fraction [zdroj vlastní]

Crossover fraction	0,2			0,4			0,6			0,8		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace
pokus č.1	17,2641	18,7	1001	17,4392	21,0	1001	15,7735	23,7	1001	16,1964	24,6	1001
pokus č.2	18,1719	19,3	1001	17,5671	21,6	1001	16,8677	22,7	1001	16,0395	24,3	1001
pokus č.3	17,3036	19,4	1001	16,6493	21,1	1001	16,2716	23,2	1001	16,3118	24,2	1001
pokus č.4	16,9333	19,7	1001	16,9769	20,6	1001	16,1006	23,7	1001	15,9360	24,2	1001
pokus č.5	16,4156	18,1	1001	16,2330	20,5	1001	15,8627	22,8	1001	15,9941	24,2	1001
pokus č.6	16,9713	18,6	1001	16,6391	20,6	1001	16,7917	23,3	1001	16,2272	24,1	1001
pokus č.7	17,4571	18,8	1001	15,9109	21,0	1001	16,5786	23,6	1001	16,6306	24,1	1001
pokus č.8	16,0529	18,3	1001	16,7694	21,3	1001	15,7716	23,9	1001	15,6360	24,2	1001
pokus č.9	17,4975	18,5	1001	16,6345	21,4	1001	16,6696	25,8	1001	16,2319	24,2	1001
pokus č.10	17,5215	18,6	1001	17,0509	20,6	1001	16,1989	22,0	1001	15,3377	24,2	1001
nejlepší výsledek	16,0529	18,1	1001	15,9109	20,5	1001	15,7716	22,0	1001	15,3377	24,1	1001
nejhorší výsledek	18,1719	19,7	1001	17,5671	21,6	1001	16,8677	25,8	1001	16,6306	24,6	1001
relativní chyba	13,20%	8,84%	0,00%	10,41%	5,37%	0,00%	6,95%	17,27%	0,00%	8,43%	2,07%	0,00%
průměrný rozdíl	6,89%	3,87%	0,00%	5,51%	2,29%	0,00%	3,28%	6,68%	0,00%	4,67%	0,54%	0,00%

Příloha - Tabulka 11 - Vliv kombinace všech parametrů [zdroj vlastní]

Verze	1 (Tournament 32, CF 0.8)			2 (Tournament 64, CF 0.8)			3 (Tournament 32, CF 0.6)		
	vzdálenost	čas (sec)	generace	vzdálenost	čas	generace	vzdálenost	čas (sec)	generace
pokus č.1	15,6980	158,8	1001	15,0867	292,5	1001	14,9753	142,0	1000
pokus č.2	14,8914	159,6	1001	15,9239	291,4	1001	15,8101	141,9	1000
pokus č.3	15,3366	157,1	1001	14,9178	293,4	1001	14,8886	144,7	1000
pokus č.4	15,1532	156,3	1001	15,8302	293,6	1001	15,4567	143,0	1000
pokus č.5	15,2738	1580	1001	15,8746	293,7	1001	14,9783	142,1	1000
pokus č.6	15,5658	156,7	1001	14,9896	292,6	1001	15,0623	141,4	1000
pokus č.7	14,9015	156,5	1001	15,6158	296,0	1001	15,5451	142,7	1000
pokus č.8	15,6542	156,7	1001	14,7208	293,0	1001	15,2711	143,0	1000
pokus č.9	15,6583	152,1	1001	15,5874	293,9	1001	15,5408	145,5	1000
pokus č.10	15,3103	151,3	1001	15,3665	292,3	1001	15,5114	144,2	1000
nejlepší výsledek	14,8914	151,3	1001	14,7208	291,4	1001	14,8886	141,4	1000
nejhorší výsledek	15,6980	159,6	1001	15,9239	296,0	1001	15,8101	145,5	1000
relativní chyba	5,42%	5,49%	0,00%	8,17%	1,58%	0,00%	6,19%	2,90%	0,00%
průměrný rozdíl	3,04%	3,31%	0,00%	4,55%	0,63%	0,00%	2,79%	1,17%	0,00%