ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA STROJNÍ

BAKALÁŘSKÁ PRÁCE

2016 Jan Šulc

ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA STROJNÍ

Studijní program: B2301 Strojní inženýrství

Studijní zaměření: 2301R016 Průmyslové inženýrství a management

BAKALÁŘSKÁ PRÁCE

Optimalizační metody využitelné v rámci diskrétní simulační optimalizace

Autor: **Jan ŠULC**

Vedoucí práce: **Ing. Pavel Raška, Ph.D.**

Akademický rok 2015/2016

Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

V Plzni dne: ……. **…….**

podpis autora

Poděkování

Rád bych tímto poděkoval vedoucímu bakalářské práce Ing. Pavlu Raškovi, Ph.D. za pomoc a odborné vedení při vypracování této práce. V neposlední řadě děkuji své rodině a nejbližším za jejich podporu.

ANOTAČNÍ LIST BAKALÁŘSKÉ PRÁCE

AUTOR	Šulc	Jan
STUDIJNÍ OBOR	Průmyslové inženýrství a management	
VEDOUCÍ PRÁCE	Ing. Raška Ph.D.	Pavel
PRACOVIŠTĚ	ZČU - FST - KPV	
DRUH PRÁCE	~~**DIPLOMOVÁ**~~	**BAKALÁŘSKÁ**
NÁZEV PRÁCE	Optimalizační metody využitelné v rámci diskrétní simulační optimalizace	

FAKULTA	strojní		**KATEDRA**	KPV		**ROK ODEVZD.**	2016

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	42		**TEXTOVÁ ČÁST**	30		**GRAFICKÁ ČÁST**	5

STRUČNÝ POPIS	Tato bakalářská práce je zaměřena na genetické algoritmy, které patří do skupiny optimalizačních metod využitelných v rámci diskrétní simulační optimalizace. V hlavní části této práce se nachází popis jednotlivých částí genetického algoritmu. V předposlední kapitole je ukázka řešení problému obchodního cestujícího pomocí genetického algoritmu, který lze aplikovat přímo na problémy průmyslového inženýrství.
KLÍČOVÁ SLOVA	Simulační optimalizace, genetické algoritmy, chromozom, funkce fitness, selekce, křížení, mutace

SUMMARY OF BACHELOR SHEET

AUTHOR	Šulc	Jan
FIELD OF STUDY	Industrial Engineering and Management	
SUPERVISOR	Ing. Raška Ph.D.	Pavel
INSTITUTION	ZČU - FST - KPV	
TYPE OF WORK	~~**DIPLOMA**~~	**BACHELOR**
TITLE OF THE WORK	Optimization Method Used for Discrete Event Simulation Optimization	

FACULTY	Mechanical Engineering		**DEPARTMENT**	KPV		**SUBMITTED IN**	2016

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	42		**TEXT PART**	30		**GRAPHICAL PART**	5

BRIEF DESCRIPTION	This thesis is focused on genetic algorithms. Genetic algorithms belong among optimization methods used for discrete event simulation optimization. The main part of this thesis is focused on the description of all parts of the genetic algorithm. At the end of this thesis is shown a solution of traveling salesman problem by genetic algorithm. This problem can be used in the field of industrial engineering.
KEY WORDS	Simulation optimization, genetic algorithms, fitness function, selection, crossover, mutation

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

GA – genetický algoritmus, genetické algoritmy

NP – nedeterministicky polynomiální problémy

TSP – problém obchodního cestujícího (traveling salesman problem)

OX – křížení se zachováním pořadí (order crossover)

PMX – křížení s částečným zobrazením (partially mapped crossover)

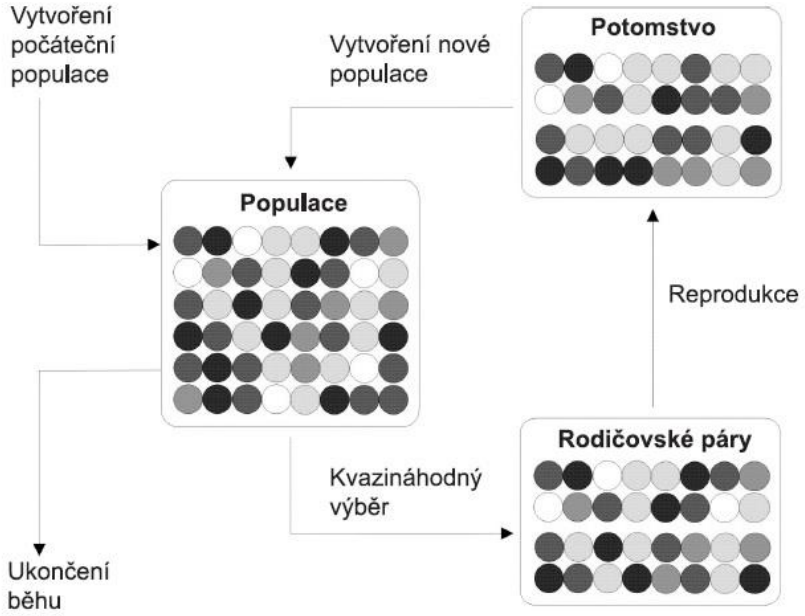
ERX – křížení s rekombinací hran (edge recombination crossover)

1 Úvod

Základním stavebním kamenem této bakalářské práce jsou genetické algoritmy. Jedná se o optimalizační metodu, která patří pod skupinu tzv. evolučních algoritmů vycházejících z teorie napsané Charlesem Darwinem [1]. Samotná myšlenka této teorie je založena na principu evoluce, tedy přímého vývoje živočichů a rostlin po mnoha generacích, kde přežívali jen ti nejschopnější a nejsilnější jedinci. Podle tohoto přímého zákona samozřejmě má větší příležitost k nalezení partnera, k reprodukci, k většímu počtu potomků jedinec silnější a naopak slabší jedinci, jsou utlačování, mají horší podmínky a mohou skončit i bez potomstva, tudíž dospět k vymření.

Genetický algoritmus pracuje s populací jedinců, kde každý jedinec zastupuje řešení daného problému. Avšak toto řešení nemusí být to nejlepší a proto je třeba identifikovat jedince, tedy přiřadit mu určité hodnocení, které slouží k porovnání s ostatními a také k nalezení optimálního řešení. Pomocí náhodného výběru, který je ovlivněn tímto hodnocením, dochází k selekci rodičů, kteří vytvoří novou populaci potomstva, a jeden generační proces je ukončen. K reprodukci dochází pomocí operátoru křížení, mutace a inverze [2].

Pokud se tento proces opakuje i po více generacích, dochází takřka k vymizení slabších jedinců (tedy i horších řešení, které zastupují) protože vlastnosti silnějších se začínají prolínat skrze celou populaci. Nelze říci, že tyto algoritmy naleznou nejlepší řešení daného problému, ale můžeme podle stanovených počátečních podmínek dostat řešení dostačující.



Obrázek 1.1 Schéma genetického algoritmu [2]

Výraznou vlastností, kterou se genetické algoritmy liší od ostatních optimalizačních metod, je práce s celou populací počátečních řešení prohledávaného prostoru namísto jen jediného. Právě proto jsou genetické algoritmy (déle už jen GA) výrazně mohutnější a také univerzální oproti ostatním optimalizačním metodám. Univerzální použití může být výhodou, ale s tím přichází i nevýhoda delšího výpočetního času. GA se také nemohou porovnávat se speciálními algoritmy, které jsou vytvořeny pro určitý problém, počítají s parametry daného problému a jsou na něj nastavené. Ve speciálních algoritmech je možno dostat kvalitnější řešení v kratším čase. Nejúčinnějším prostorem uplatnění GA jsou zatím problémy, pro které nebyla nalezena žádná speciální metoda, kde bohužel nic lepšího neexistuje.

2 Současný stav

V dnešní době je snaha dospět k nejlepšímu řešení daného problému v jakémkoliv technickém či jiném oboru. Ať se už vychází ze zkušeností, pokusem nebo například odhadem, člověk dospěje k nějakému řešení. Nyní nastává otázka, zda toto řešení je dostačující, a pokud ano, už není třeba se dále ničím zabývat. Častějším případem v praxi je však stav, při kterém se dospěje k něčemu, co není tím pravým a je třeba daný výsledek dále upravovat.

Právě k tomu slouží optimalizace, tedy proces, při kterém dochází k zlepšování, dokud řešení nesplňuje určité výstupní požadavky. Samozřejmě tento proces stojí peníze, které se ve výsledku mnohonásobně vrátí ušetřením za zbytečné opravy chyb nedokonalých návrhů a podobně. Jak si ale toto všechno ověřit? Bylo by velmi nelogické a neefektivní provádět vše reálně, vyrábět tisíce a tisíce prototypů. Naštěstí pomocí výkonných počítačů se může reálná věc simulovat, vše se vyzkoušet teoreticky a až poté něco začít vyrábět. To je největší výhoda těchto simulačních optimalizací.

Na začátku je vytvořen simulační model, který odpovídá danému reálnému problému, za pomoci vstupních parametrů. Následuje samotná výpočetní část, které je model vystaven, a ze které jsou následně generovány výsledky [3].

2.1 Řešené problémy pomocí GA

Jak bylo zmíněno v úvodu, genetické algoritmy jsou velmi vhodnou metodou řešení problémů, kde by nebylo možno dosažení výsledku v přípustném čase. Tyto problémy lze zařadit do kategorie NP (nedeterministicky polynomiálních problémů), u kterých se množina řešení exponenciálně narůstá s každým novým vstupním parametrem simulačního modelu [3].

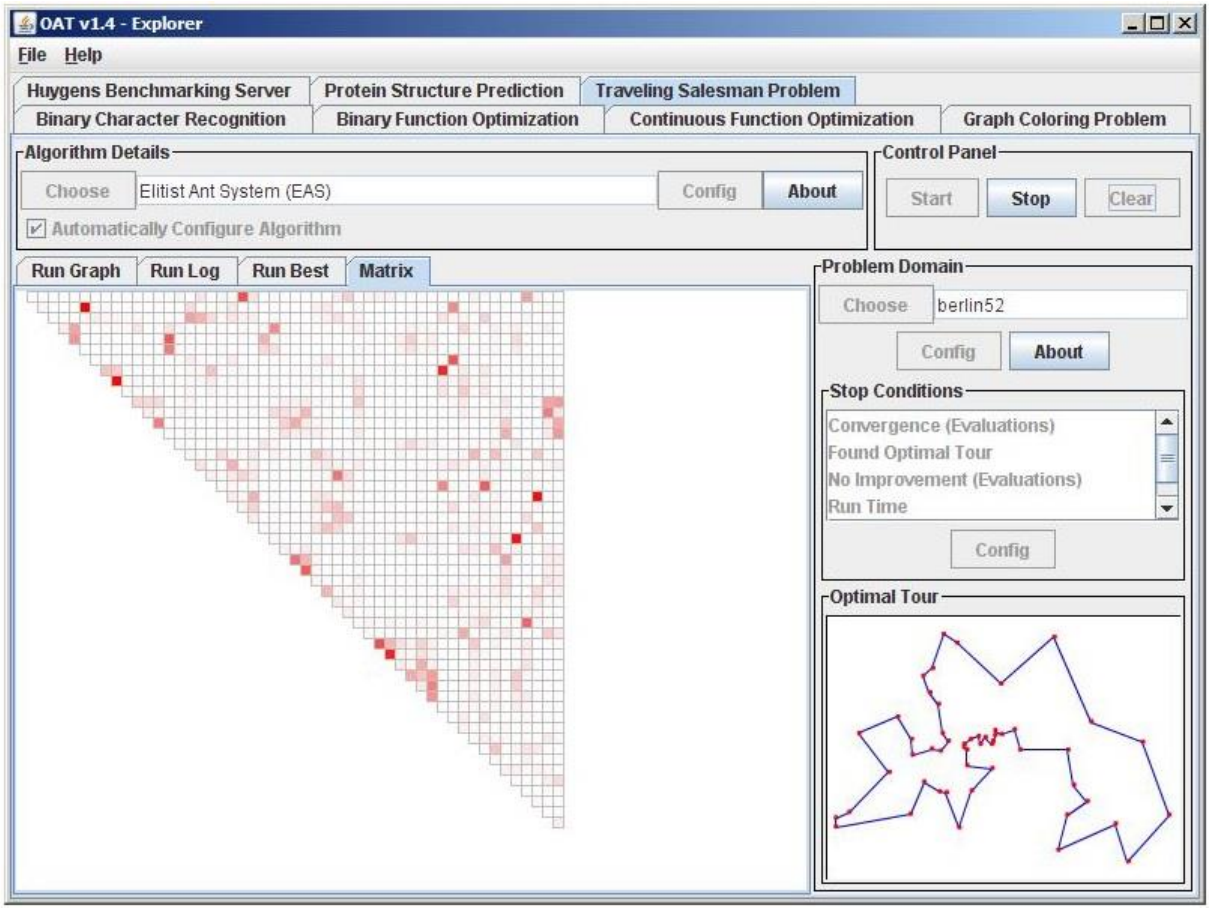
Mezi tyto problémy patří například problém obchodního cestujícího [4], který přímo souvisí s jakýmkoliv plánováním trasy v rámci podniku, nebo plánováním trasy zavážek poboček určitého podniku. Dále např. problémy spojené s rozvrhováním výrobních procesů [5]. Také problémy spojené s časovým rozvrhováním jednotlivých výrobních úkolů ať už z pohledu pracovníků, výrobních strojů či pracovišť [6].

2.2 Software využívající GA

Pro řešení GA je navrženo mnoho programů a doplňujících platforem pro většinu programovacích jazyků (Java, C++, nebo Python). Pro programovací jazyk Java je například vyvinut program JGAP (Java Genetic Algorithms Package) [7]. Tento program poskytuje základní mechanismus GA se základními genetickými operátory s možností implementace vlastních přizpůsobených operátorů.

Podobným programem je nástroj opět podporovaný v jazyce Java: Optimization Algorithm Toolkit slouží k vývoji, hodnocení a experimentování na klasických optimalizačních metodách, mezi kterými lze najít i GA [8]. Výhodou je i možná vizualizace řešeného problému.

Na obrázku - Obrázek 2.1 Optimization Algorithm Toolkit – ukázka řešení TSP - je vidět příklad řešení problému obchodního cestujícího s vizualizací nejkratší obchodní cesty.



Obrázek 2.1 Optimization Algorithm Toolkit – ukázka řešení TSP [8]

Pro programovací jazyk C++ byl vytvořen GAlib soubor genetických objektů pro jakýkoliv program podporující tento jazyk [9]. Tato knihovna obsahuje přehled implementací a programovacích technik, příklady řešených problémů (např. TSP), popis jednotlivých používaných operátorů atd. [9].

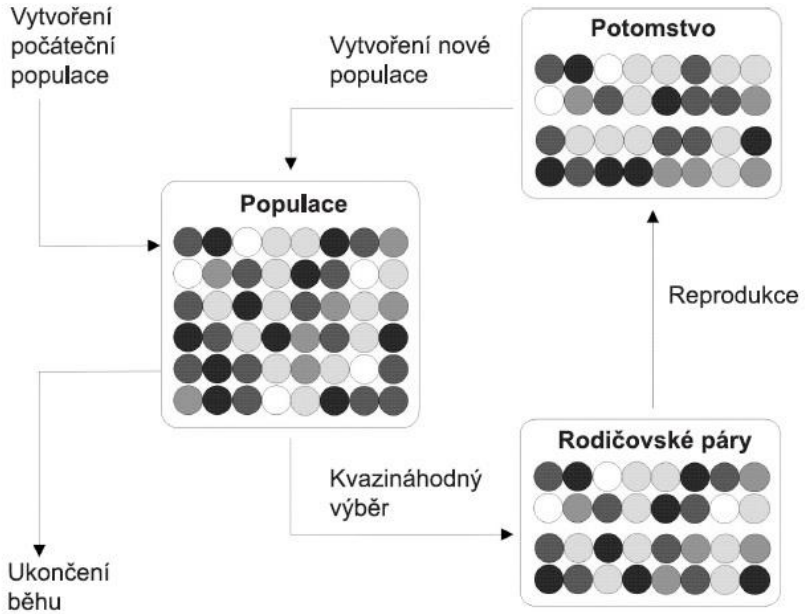
Existují také programy spolupracující s Microsoft Office Excel jako např. SolveXL [10] nebo program z lit. [11]: GA Optimization for Excel. Tyto programy spolupracují s daty vytvořenými daty v MS Excel, a následná výpočetní operace probíhá v samotném programu.

Toto byl jen stručný přehled programů, které lze použít pro řešení GA, nicméně existuje nepřeberné množství používaných softwarů. Seznam těchto programů lze nalézt v lit. [12], kde lze specifikovat výběr softwaru z pohledu, v jakém programovacím jazyce má být zaprogramován, či pro jaký operační systém je určený.

3 Genetický algoritmus

Základním stavebním kamenem této bakalářské práce jsou genetické algoritmy. Genetické algoritmy jsou jen podskupina evolučních algoritmů vycházejících z teorie napsané Charlesem Darwinem [1]. Samotná myšlenka této teorie je založena na principu evoluce, tedy přímého vývoje živočichů a rostlin po mnoha generacích, kde přežívali jen ti nejschopnější a nejsilnější jedinci. Podle tohoto přímého zákona samozřejmě má větší příležitost k nalezení partnera, k reprodukci, k většímu počtu potomků jedinec silnější a naopak slabší jedinci, jsou utlačování, mají horší podmínky a mohou skončit i bez potomstva, tudíž dospět k vymření.

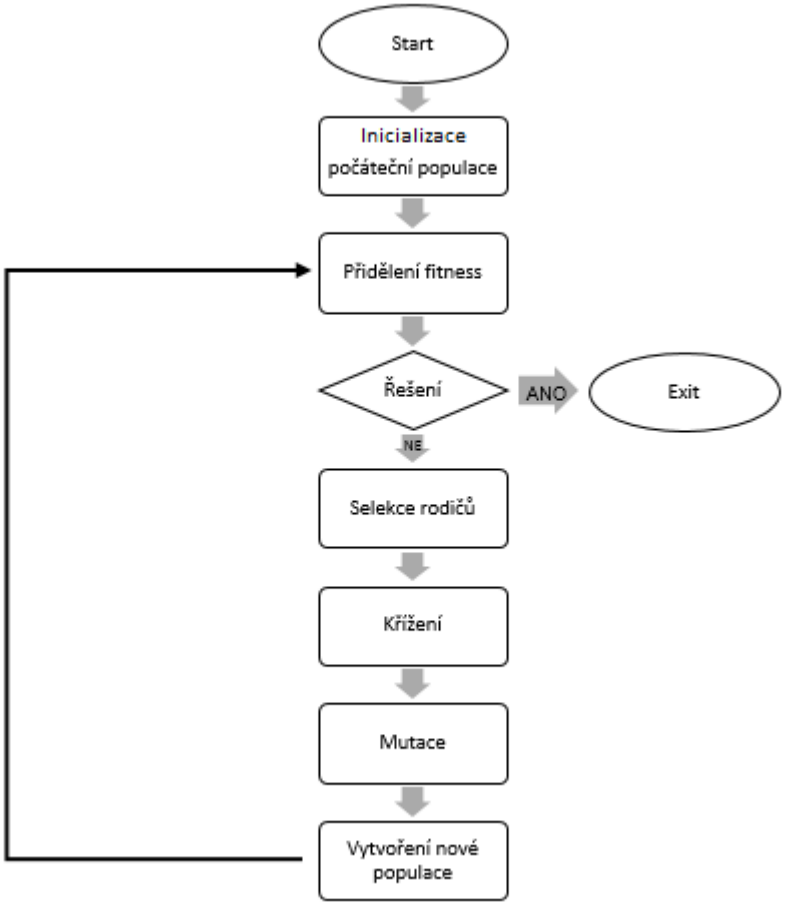
Genetický algoritmus pracuje s populací jedinců, kde každý jedinec zastupuje řešení daného problému. Avšak toto řešení nemusí být to nejlepší a proto je třeba identifikovat jedince, tedy přiřadit mu určité hodnocení, které slouží k porovnání s ostatními a také k nalezení optimálního řešení.



Obrázek 3.1 Schéma genetického algoritmu [2]

Výraznou vlastností, kterou se genetické algoritmy liší od ostatních optimalizačních metod, je práce s celou populací počátečních řešení prohledávaného prostoru namísto jen jediného. Právě proto jsou genetické algoritmy (déle už jen GA) výrazně větší a také univerzální oproti ostatním optimalizačním metodám. Univerzální použití může být výhodou, ale s tím přichází i nevýhoda delšího výpočetního času. GA se také nemohou porovnávat se speciálními algoritmy, které jsou vytvořeny pro určitý problém, počítají s parametry daného problému a jsou na něj nastavené. Ve speciálních algoritmech je možno dostat kvalitnější řešení v kratším čase. Nejúčinnějším prostorem uplatnění GA jsou zatím problémy, pro které nebyla nalezena žádná speciální metoda, kde bohužel nic lepšího neexistuje.

Jak bylo zmíněno v úvodu této kapitoly, GA jsou založeny na Darwinovském principu evoluce. Lepší řešení problému (vyhovující podmínkám) je nalezeno opakováním cyklu, při kterém postupně mizí slabší jedinci, tedy horší řešení. Aby bylo možno rozeznat jednotlivce od sebe, musí se každému přiřadit určité hodnocení (fitness) [13]. Na základě tohoto hodnocení budou vybírány potenciální rodičovské páry, ze kterých vzejde reprodukcí nová populace. Po několika takto provedených evolučních cyklech může GA dojít k nalezení optimálního řešení. Někdy se ovšem stane, že populace zmutuje, tedy algoritmus se zacyklí u jednoho řešení, které však nemusí být nejlepší v celém prohledávaném prostoru, nýbrž jen lokálním extrémem, což často bývá odlišné od toho globálního.



Obrázek 3.2 Vývojový diagram obecného GA [14]

Na obrázku - Obrázek 3.2 Vývojový diagram obecného GA [14] - je vidět vývojový diagram obecného GA. Na počátku Je nutno definovat vstupní parametry řešeného problému. Na tomto základě je vytvořena počáteční populace jedinců. Jedinci jsou reprezentováni pomocí chromozomů neboli řetězců. Tento řetězec v sobě ukrývá veškerou informaci o daném jedinci, je tedy tvořen jednotlivými geny, uspořádanými lineárně za sebou. Geny jsou ve standardních genetických algoritmech reprezentovány pouze dvouhodnotovými symboly, např pomocí binárního kódování nebo Grayova kódu [2].

V dalším kroku je nutno přiřadit jednotlivcům hodnocení (fitness). Účelem hodnotící funkce je vytvoření spektra, na kterém lze jednotlivé chromozomy srovnat od nejhoršího k nejlepšímu a tím získat přehled o celé populaci.

Po ohodnocení všech jedinců se přechází k prvnímu zásadnímu okamžiku, ve kterém hraje GA roli, výběru rodičovských párů (tzv. selekce) [2]. Výběr rodičů probíhá s ohledem na hodnotící funkci. Silnější a lepší jedinci budou mít větší pravděpodobnost, že budou vybráni.

Selekce může probíhat mnoha mechanismy, jedním z nejčastějších bývají turnajová selekce anebo ruletový mechanismus [15].

Následujícím krokem je samotná reprodukce, která je prováděna dvěma základními operátory, a to mutací a křížením [16]. Existují také další jiné operátory, jádrem všech je vzájemná výměna částí chromozomů. U křížení dochází k výměně částí řetězců mezi jednotlivými chromozomy, zatímco při mutaci se mění geny v rámci jednoho chromozomu.

Dokončením reprodukce vzniká generace jedinců a je ukončen jeden cyklus celého GA. Nyní je třeba této nové populaci opět přidělit hodnocení fitness, a pokud není nalezeno vhodné řešení, opakují se stejné kroky, jako v prvním cyklu.

U GA je přijat efekt přímé evoluce. To znamená, že jedinec ovlivňuje celou populaci pouze v období jedné generace. Tedy od svého vzniku k přidělení hodnocení, následné selekci a reprodukci, až po samotný zánik.

V následujících kapitolách budou podrobně rozebrány jednotlivé kroky GA.

3.1 Vznik počáteční populace a reprezentace

Základní reprezentací jedince je chromozom neboli řetězec. Tento řetězec v sobě ukrývá veškerou informaci o daném jedinci, je tedy tvořen jednotlivými geny, uspořádanými lineárně za sebou. Pro jednoduchost jsou geny ve standardních genetických algoritmech reprezentovány pouze dvouhodnotovými symboly, zpravidla 0 a 1. Lineární řazení genů v chromozomu má svůj důvod, protože ve stejném pořadí jsou řazeny i geny u ostatních jedinců (například druhý gen chromozomu reprezentuje barvu očí). Obecně i-tý gen vyjadřuje stejnou vlastnost u každého jedince. Proti přírodě dochází k výraznému zjednodušení, jelikož jedním chromozomem je popisován celý jedinec, jedním genem popsána celá vlastnost. U živočichů a rostlin jsou jednotlivé geny pouze střípky, které spojením vytvoří určitý rys a společným seřazením chromozomů vzniká individuum. Jako příklad může posloužit člověk sám. Naše genetická informace je vypsána do DNA, která je složena právě sekvencí chromozomů. Jeden chromozom by na popsání člověka zdaleka nestačil [2].

Z předchozího odstavce je důležité si odnést poznatek, že chromozomy jsou reprezentací jedinců, obsahují geny, které jsou lineárně seřazené, nabývajících pouze dvou hodnot zpravidla 0 a 1 a délka chromozomů pro daný problém je jednotná. Existuje několik způsobů, kterými lze informace do chromozomu zakódovat. Vždy záleží na problému, který má GA řešit. Podle toho se vybírá způsob kódování.

3.1.1 Binární kódování

V tomto případě gen může nabývat pouze dvou hodnot (0 a 1). Pokud vlastnost může nabývat více než jen 2 hodnot, řeší se to tak, že více genů popisuje jednu vlastnost, přičemž platí, že n genů ukrývá 2^n možností. Tato metoda skrývá velké riziko, protože v binárním kódování se může přihodit situace, při které i se dva zcela vzdálené prvky mohou lišit pouze v hodnotě jednoho genu. Pokud náhodou na tomto genu bude provedena mutace a dojde ke změně hodnoty, celková změna jedince je radikální, a to nemusí být vůbec posun kupředu, ba naopak krok zpět. Pracuje se s předpokladem, že malá změna v chromozomu způsobí malé změny vlastností jedince. Tímto je použití binárního kódování velmi limitováno.

Pro názornost je ukázáno binární kódování na chromozomu o délce 8 genů. Náhodně je vygenerována populace např. takto (GA, délka jednotlivých chromozomů, tabulky ze zdroje [2]):

jedinec č.	chromozom
1	(0, 0, 1, 1, 1, 0, 0, 1)
2	(1, 0, 0, 1, 0, 0, 1, 1)
3	(0, 1, 1, 0, 0, 1, 0, 0)
4	(1, 1, 1, 0, 0, 1, 0, 1)

Tabulka 3.1 Počáteční populace

3.1.2 Grayův kód

Zásadním rozdílem oproti binárnímu kódování je odlišnost dvou sousedních prvků pouze v jednom genu. Proto pokud dojde k mutaci chromozomu, pravděpodobnost výrazné změny vlastností se zmenší. Jednoduchý příklad pro ukázání rozdílu obou kódů je vidět v tabulce pod tímto odstavcem, ve které je zakódováno 8 číslic pomocí tří genů (bitů).

číslo	binární kód	Grayův kód
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

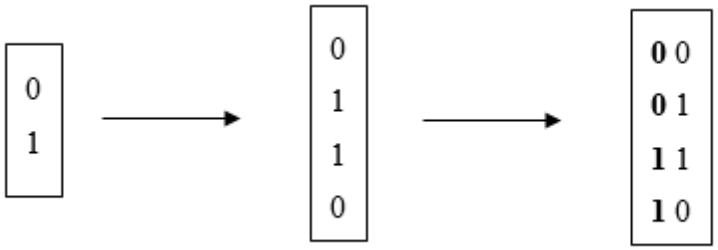
Tabulka 3.2 Srovnání binárního a Grayova kódu [17]

Zatímco v binárním kódu se čísla 3 a 4 neshodují ani v jednom genu, v druhém případě se liší právě v jednom. Právě kvůli této vlastnosti, že Grayův kód bere v potaz blízkost dvou jedinců, bývá používán častěji než binární kódování.

Způsob, jakým se Grayův kód vytváří, je následující [16]:

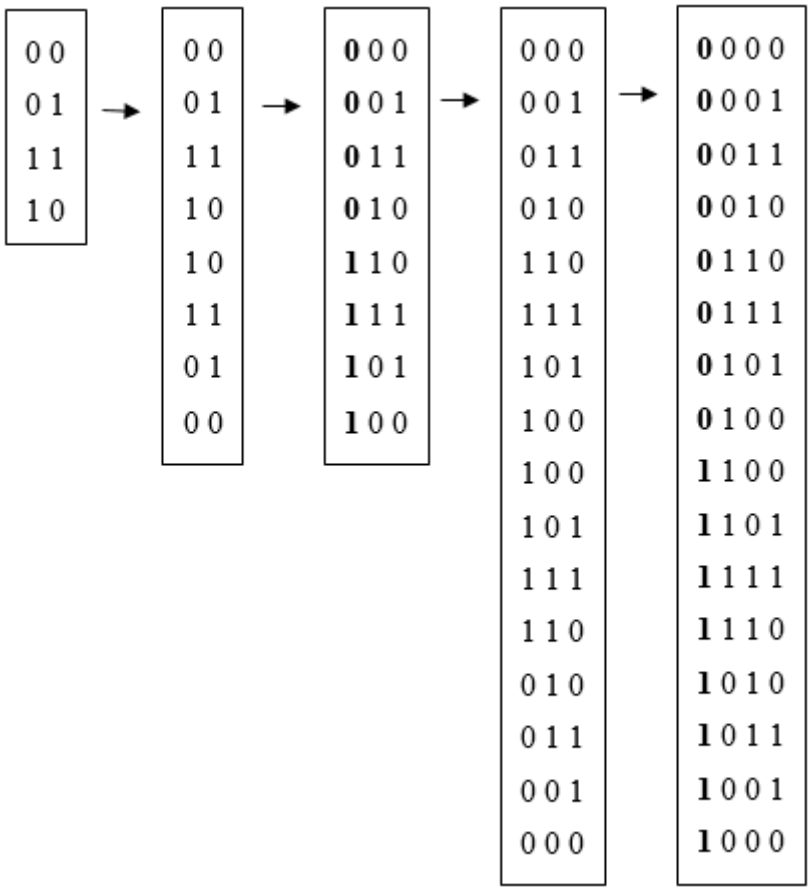
Na začátku se vytvoří dva odlišné řetězce délky 1. Jeden nese znak 0 a druhý 1. Nově vzniklých řetězců bude dvakrát tolik. Vzniknou tak, že se první dva řetězce napíšou pod sebe a poté se pod ně napíšou ty samé řetězce v opačném pořadí (zrcadlově překlopené). Nyní musí být splněna podmínka toho, že dva sousední řetězce se smí lišit pouze v jednom znaku a také, že žádné dva řetězce nemohou být shodné. Toho se docílí tak, že v horní polovině se přidá před řetězec 0 a v dolní polovině se přidá před řetězec 1. Tím jsou vytvořeny celkem 4 nové řetězce o délce 2.

Pro názornost je vše vidět graficky na Obrázek 3.3 Grayův kód, vytvoření řetězce:



Obrázek 3.3 Grayův kód, vytvoření řetězce [16]

Takto dochází i k vytvoření delších řetězců, pomocí stejného postupu.



Obrázek 3.4 Grayův kód, delší řetězce [16]

Na obrázku - Obrázek 3.4 Grayův kód, delší řetězce [16] - je ukázán princip na vytvoření řetězců o délce 4. Postup by se stále takto opakoval i pro větší délky, bylo-li by jich třeba. Nyní stačí pouze přiřadit jednotlivým řetězcům čísla, jako to je tomu v tabulce (Tabulka 3.2 Srovnání binárního a Grayova kódu).

3.1.3 Kódování pomocí reálných čísel

Dalším způsobem, jak lze informace do chromozomu ukládat, je pomocí reálných čísel [2]. V některých případech je tento způsob mnohem výhodnější. Pokud výsledné řešení má být přesnější, právě použitím reálných čísel je možno toho docílit. Pokud by bylo používáno pouze dvouznakových genů, chromozomy by musely být podstatně delší a výpočtový čas celé optimalizace by narůstal.

3.1.4 Permutační kódování

V neposlední řadě se používá také permutační kódování [2]. Typickým případem pro toto kódování jsou příklady, ve kterých jde o plánování trasy, při které je nutno navštívit určitá místa. Výsledkem by měla být nejkratší ujetá vzdálenost. V jedné z dalších kapitol, kde se práce bude zabývat právě tímto problémem, bude permutační kódování vysvětleno podrobně, ale ve zkratce je každému místu přiděleno číslo a jedinec je reprezentován právě čísly jednotlivých míst právě tak, jak je projel za sebou. Např. jedinec reprezentovaný chromozomem (7, 6, 1, 3, 5, 2, 4), jako první projel místo 7, poté jel k místu 6 a tak dále, až se dostal do poslední zastávky na jeho trase. Nutno podotknout, že jednotlivá místa má za úkol projet jen jednou, a nezáleží, kde začne, ani kde končí. Výhoda permutačního kódování spočívá v tom, že žádné číslo objevující se v chromozomu nemůže být použito vícekrát.

3.1.5 Sousedská reprezentace

Pro řešení problému obchodního cestujícího, který se zabývá plánováním trasy, bylo navrženo více způsobů kódování, mezi které patří i tzv. sousedská reprezentace. Tato metoda funguje tak, že jednotlivá města jsou očíslována přirozenými čísly a celková cesta je charakterizována vektorem, jehož délka je rovna počtu měst [2]. Číslo *i* se v chromozomu na *j*-té pozici nachází právě tehdy, když se cestující přesunuje z místa *j* do místa *i*.

3.1.6 Ordinální reprezentace

Tento způsob kódování je také vhodný pro problémy spojené s plánováním trasy. Cesta je zakódována do chromozomu na základě tzv. referenčního seznamu R (viz lit. [15]), ve kterém jsou všechna města seřazena v určitém pořadí.

Obecný princip dekódování spočívá v tom, že z chromozomu se vezme postupně zleva doprava číslo *k* na *i*-té pozici, které představuje *k*-tý prvek referenčního seznamu *R*. Následně se tento prvek vyjme z referenčního seznamu a umístí se do cesty [15]. Princip dekódování bude ukázán u řešení problému obchodního cestujícího v kapitole 4.3 Kódování.

Výhodou této metody je i fakt, že pomocí genetického operátoru jednobodového křížení vzniknou vždy chromozomy představující přípustná řešení a nenastane např. předčasné ukončení cesty, jako k tomu mohlo nastat u sousedské reprezentace [2]. Nevýhodou této metody je nespolehlivá funkce genetických operátorů a proto se od tohoto způsobu kódování upouští [15].

3.2 Ohodnocující funkce fitness

V dalším kroku je nutno přiřadit jednotlivcům hodnocení (fitness). Pro jednoduchost může být nastavena, jako součet hodnot v chromozomu. Za použití binárního kódování by tak nejlepším jedincem byl ten se samými 1, naopak nejhorším byl ten se samými 0, Nicméně tato funkce je pouze ukázková a funkce používané při řešení reálných problémů jsou nastavovány přesně na míru problému. Pro základní popis, je ale tato funkce dostačující a je přiřazeno jednotlivcům hodnocení takto (tabulka viz lit. [2]):

jedinec č.	chromozom	hodnocení
1	(0, 0, 1, 1, 1, 0, 0, 1)	4
2	(1, 0, 0, 1, 0, 0, 1, 1)	4
3	(0, 1, 1, 0, 0, 1, 0, 0)	3
4	(1, 1, 1, 0, 0, 1, 0, 1)	5

Tabulka 3.3 Populace s přiřazeným hodnocením

V tomto případě se jedná o přímou úměru. Tento způsob funguje pouze na papíře, pokud si člověk představí, co jednotlivé geny v chromozomu zastupují, dospěje k tomu, že samé jedničky vůbec nejsou optimálním neboli nejlepším řešením daného problému.

Proto je třeba volit ohodnocující funkci s ohledem na právě řešený problém. Přímá úměra bude kolísat v případě, kdy například budou tři řešení daného problému na stejné úrovni. V tomto případě nelze stanovit, které řešení je vhodné a musí se vybrat jiný způsob řešení.

Tento problém se řeší rozdělením populace jedinců na dvě poloviny a jedinci z jedné skupiny jsou porovnáváni pouze s jedinci z druhé poloviny. Nebo jedinci soutěží v rámci turnaje. Tedy z každých dvou porovnávaných jedinců je vybrán jeden postupující do dalšího kola. Vítěz turnaje získává nejlepší hodnocení a ostatní získají hodnocení podle postupu v turnaji. Metodou turnaje se hodnocení nevyvaruje zjednodušení, jelikož proběhne pouze N-1 porovnání jedinců [2].

Posledním typem hodnotící funkce je ta, při které uživatel přímo za běhu zasahuje do procesu. Využívá se např. v oblasti kriminalistiky k identifikaci pachatele. Svědek trestného činu pracuje s počítačovým programem založeným na GA, ve kterém jsou generovány obličeje pachatele. Svědek přiřazuje hodnocení podle podobnosti jednotlivých obličejů k pachateli. Obličejů s vyšším hodnocením budu využito k tvorbě nových jedinců pomocí vhodných genetických operátorů [2].

3.3 Selekce

Po ohodnocení všech jedinců se přechází k prvnímu zásadnímu okamžiku, ve kterém hraje GA roli, výběru rodičovských párů. Výběr je prováděn s ohledem na hodnotící funkci fitness, kdyby tomu tak nebylo, nemělo by ani smysl jednotlivce hodnotit. Díky zohlednění lepších jednotlivců dochází k přenosu informací do nové generace, která může poskytnout přirozené zlepšení celé populace. Vývoj populace se dá také ovlivnit regulací důrazu na lepší jedince. Čím větší bude kladen důraz, tím rychleji z populace vymizí slabá řešení a naopak [18].

3.3.1 Mechanizmus ruletového kola

Prvním možným způsobem, jak vybrat jedince je pomocí mechanizmu ruletového kola. Silnější a lepší jedinci budou mít určitou výhodu, větší pravděpodobnost, že budou vybráni. Představit si toto lze jako ruletu, na které budou menší a větší díly, představující jedince podle jejich hodnocení [15]. K vytvoření kola stačí pouze vyjádřit procentuální zastoupení jednotlivých hodnocení vůči celkové sumě. Pokud se hodnocení označí f_i, poté pro *i*-tého jedince platí pravděpodobnost podle vzorce:

$$P_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \cdot 100[\%]$$

Takto získané pravděpodobnosti odpovídají přímé úměrnosti. Existují i jiné způsoby, jak plochy ruletového kola zkonstruovat. Nejrozšířenějším je právě tento způsob, kde se využívá procentuální zastoupení jednotlivých hodnocení.

jedinec č.	chromozom	hodnocení (f_i)	pravděpodobnost (P_i)	kumulované ohodnocení
1	(0, 0, 1, 1, 1, 0, 0, 1)	4	25 %	0,2500
2	(1, 0, 0, 1, 0, 0, 1, 1)	4	25 %	0,5000
3	(0, 1, 1, 0, 0, 1, 0, 0)	3	18,75 %	0,6875
4	(1, 1, 1, 0, 0, 1, 0, 1)	5	31,25 %	1,0000
	Průměrné hodnocení populace	4		

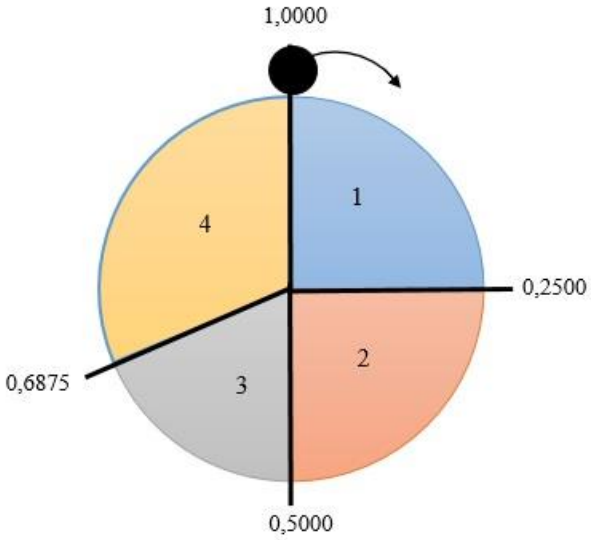
Tabulka 3.4 Populace s přiřazenou pravděpodobností

Z tabulky je vidět, že třetí jedinec má nejmenší šanci na to být vybrán, naopak nejsilnějším je čtvrtý. Nyní stačí pouze náhodně vygenerovat číslo $r \in \langle 0; 1\rangle$ a následně vybrat jedince pro kterého platí tato nerovnice:

$$\sum_{i=1}^{i-1} P_i < r \leq \sum_{i=1}^{i} P_i \quad i \in \{1; 2; \dots; N\}$$

Proto v tabulce - Tabulka 3.4 Populace s přiřazenou pravděpodobností - je sloupec kumulovaného ohodnocení, při kterém dochází k přičítání pravděpodobnost jedince k součtu pravděpodobností prvků předcházejících. Pro názornost, pokud bylo vygenerováno číslo $r = 0{,}67$, bude vybrán třetí jedinec, protože součet prvních dvou je 0,5, a pokud se k tomuto číslu přičte pravděpodobnost třetího jedince, součet je roven 0,6875 a číslo *r* se nachází právě mezi těmito dvěma hodnotami. Matematicky platí nerovnice:

$$0{,}5 < r \leq 0{,}6875$$



Obrázek 3.5 Ruletového kolo s vyznačeným kumulativním ohodnocením [2]

Na obrázku výše je vidět ruletové kolo s naznačenými výsečemi představující jedince. V jakém pořadí byli jedinci generováni, v takovém jsou i srovnáni za sebou ve směru točení ruletového kola. Pro číslo $r = 0{,}67$ se kulička zastaví ve 3. sektoru a je vybrán jedinec č. 3. V ruletě byla provedena celkem 4 zatočení, při kterém bylo náhodně vygenerováno číslo *r*, čímž se určily první dva páry k reprodukci:

	jedinec č.	chromozom
1. pár	3	(0, 1, 1, 0, 0, 1, 0, 0)
	4	(1, 1, 1, 0, 0, 1, 0, 1)
2. pár	4	(1, 1, 1, 0, 0, 1, 0, 1)
	1	(0, 0, 1, 1, 1, 0, 0, 1)

Tabulka 3.5 První rodičovské páry

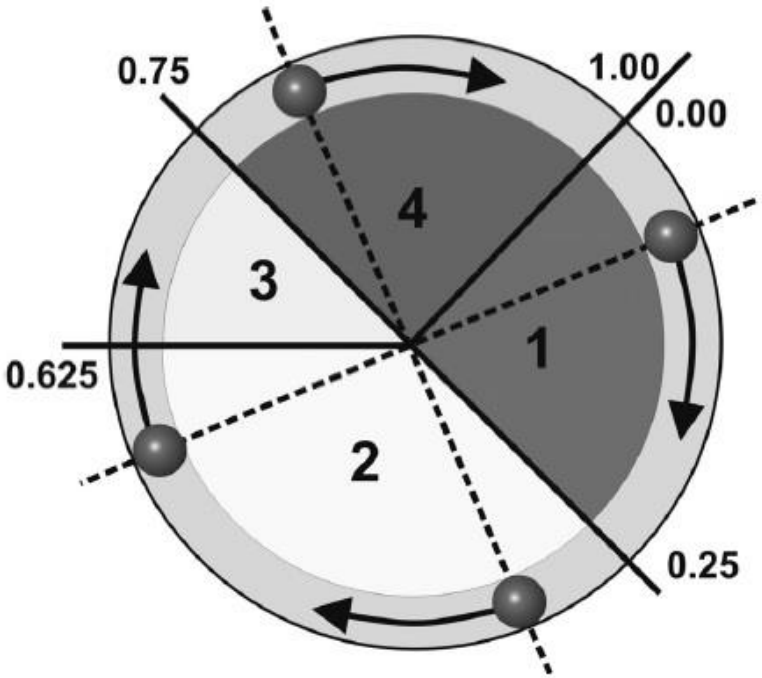
Z tabulky - Tabulka 3.5 První rodičovské páry - je vidět, že i přes nejmenší šanci byl vybrán jedinec č. 3. Také se ukázala síla 4. jedince, který byl vybrán pro oba páry.

3.3.2 Vylepšený ruletový mechanizmus

Jak bylo řečeno v druhé kapitole, k reprodukci dochází pomocí ruletového kola, ve kterém jsou upřednostňováni jedinci s vyšším hodnocením. Na kole tedy zabírají mnohem větší část než slabší jedinci. Pokud by pravděpodobnost silných jedinců byla na tolik velká, že by zabíraly výraznou část kola, docházelo by k předčasnému konvergovaní algoritmu k určitému řešení, které z hlediska celkového pohledu nemusí být optimálním. Celý proces by rychleji ztrácel rozmanitost.

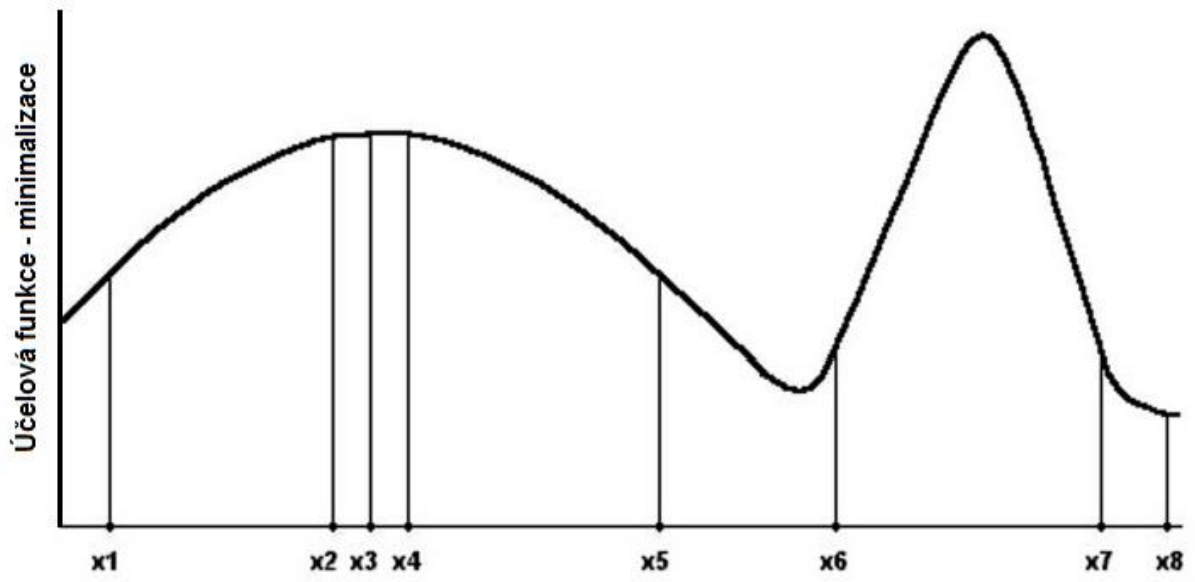
Částečně se dá ztrátě rozmanitosti řešení zabránit vylepšením ruletového mechanismu. Normálně jedním zatočením ruletového kola je vybrán jeden jedinec (na ruletovém kole se pohybuje jedna kulička). Pokud se bude na ruletovém kole vyskytovat po intervalech k kuliček, jedním zatočením získáme *k* vybraných jedinců. Na obrázku - Obrázek 3.6 Vylepšený ruletový mechanismus [2] - je vidět takto upravený ruletový mechanizmus, ve kterém se nachází 4 kuličky rozmístěny na obvodu s posunutím o úhel 90°. Jedním zatočením

jsou vybrání hned 4 jedinci. Tento způsob pomáhá udržení rozmanitosti, nicméně problémem zůstává stále velká pravděpodobnost, s kterou jsou silnější jedinci vybíráni k reprodukci.



Obrázek 3.6 Vylepšený ruletový mechanismus [2]

Pokud se v populaci nachází jeden, či více extrémně lepších jedinců. Jejich síla je na tolik velká, že šance slabších jedinců podílení se na další reprodukci je takřka nulová. Tento efekt je dobře vidět na obrázku (Obrázek 3.7), kde lépe hodnocení jedinci nacházející se za sebou (x_2, x_3, x_4), mají větší možnost ovlivnit další vývoj populace než například jedinec x_6. Když tedy na ruletovém kole jsou jedinci seřazeni za sebou podle jejich pořadí, řešení x_2, x_3, x_4 s velmi podobnými vlastnostmi obsazují mnohem větší výseč a tak roste i pravděpodobnost, že algoritmus sklouzne k lokálnímu extrému.



Obrázek 3.7 Předčasná konvergence k lokálnímu extrému [2]

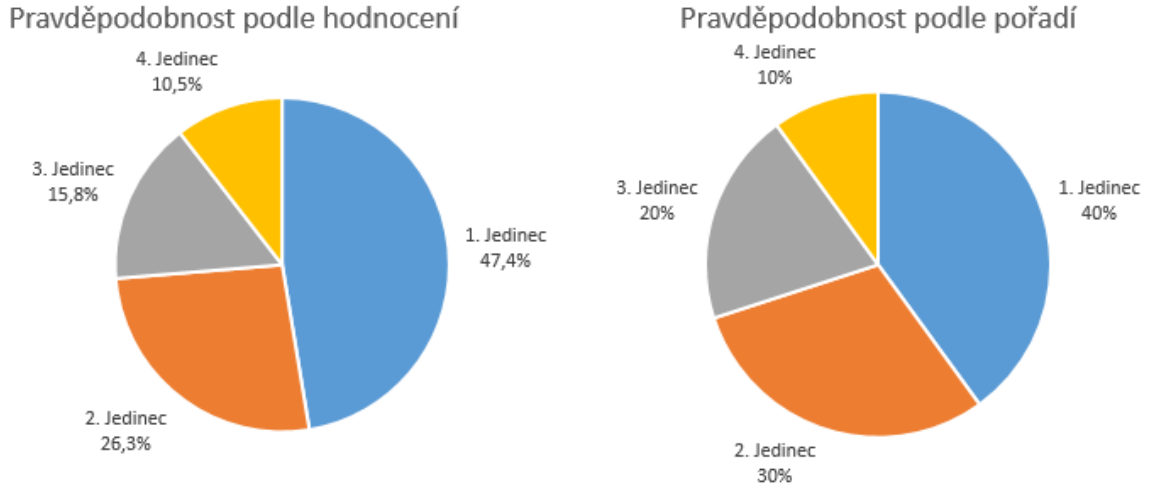
Aby se zamezilo tomuto efektu, každému jedinci je přiděleno pořadové číslo podle jeho hodnocení, tím se velké rozdíly v hodnocení mezi jedinci zmírní. Změna se projeví zrovnoměrněním velikosti výsečí na ruletovém kole.

Pro lepší pochopení je uveden příklad. Populace obsahuje 4 jedince a je jim přiděleno hodnocení podle následující tabulky:

jedinec č.	hodnocení	pravděpodobnost podle hodnocení	Pořadí	pravděpodobnost podle pořadí
1	9	47,4 %	4	40 %
2	5	26,3 %	3	30 %
3	3	15,8 %	2	20 %
4	2	10,5 %	1	10 %

Tabulka 3.6 Srovnání pravděpodobnosti podle hodnocení a podle pořadí

Z tabulky je vidět, pokud by bylo kolo zkonstruováno pomocí velikosti hodnocení, rozdíly mezi jedinci jsou větší, naopak podle pořadí je zastoupení jedinců mnohem rovnoměrnější. Například rozdíl pravděpodobnosti prvního a druhého jedince podle hodnocení je přes 20 %. Pokud však je použit přístup s pořadím, rozdíl mezi jedinci je pouze 10 %. Graficky je toto vidět na dalším obrázku (viz Obrázek 3.8).



Obrázek 3.8 Srovnání pravděpodobnosti podle hodnocení a podle pořadí [2]

Nevýhoda použití srovnání podle pořadí jedinců nastává, když se v populaci nachází jedinci bez výrazných odlišností. Aplikováním srovnání podle pořadí se naopak rozdíly mezi jednotlivci výrazně zvětšují a tento způsob přestává být vhodným.

3.3.3 Turnajová selekce

Poslední často používanou metodou pro vybírání rodičovských párů je turnajová selekce [2]. Turnajová selekce se nijak neliší například od sportovních turnajů a jejich závěrečné vyřazovací části [19], při které proti sobě stojí vždy dva soupeři, a lepší z těch dvou postupuje v turnaji dále. V rámci turnaje jsou jednotlivci porovnáváni podle hodnoty fitness. Vítězem turnaje,
je ten jedinec, který došel v turnaji nejdále (má nejvyšší ohodnocení) a je vybrán k další reprodukci [20]. Čím větší je počet jedinců účastnících se turnaje, tím větší je kladen důraz na silnější jedince a slabší jedinci jsou potlačováni, jelikož nemají takovou šanci se probojovat finále a celkově turnaj vyhrát.

3.4 Křížení

Následujícím krokem je samotná reprodukce, která je prováděna dvěma základními operátory, a to mutací a křížením [16]. Pro určité problémy, jako je např. problém obchodního cestujícího, byly navrženy speciální operátory křížení. Tyto operátory budou popsány u řešení tohoto problému.

3.4.1 Bodové křížení

Prvním typem křížení je křížení bodové. Z populace jsou vybráni vždy dva rodiče, kteří utvoří pár [21]. V chromozomech je náhodně vybrán gen, od kterého se vzájemně vymění postupně všechny geny až do konce chromozomů [22]. Nemělo by smysl vybírat poslední gen, jelikož by noví jedinci byli pouhými kopiemi svých rodičů. Takto tedy vzniknou dva nový jedinci přenášející informace zároveň od obou rodičů. Noví jedinci mohou například vzniknout takto:

	chromozom	nový chromozom
1. pár	(0, 1, 1, 0 | 0, 1, 0, 0) (1, 1, 1, 0 | 0, 1, 0, 1)	(0, 1, 1, 0 | 0, 1, 0, 1) (1, 1, 1, 0 | 0, 1, 0, 0)
2. pár	(1, 1 | 1, 0, 0, 1, 0, 1) (0, 0 | 1, 1, 1, 0, 0, 1)	(1, 1 | 1, 1, 1, 0, 0, 1) (0, 0 | 1, 0, 0, 1, 0, 1)

Tabulka 3.7 Vznik nových jedinců křížením

Ne vždy je tento operátor použit, ale pravděpodobnost použití tohoto operátoru se pohybuje v intervalu $\langle 0{,}6; 1\rangle$ (viz lit. [15]). Pokud operátor křížení není použit, novými jedinci jsou prohlášeny přímé kopie rodičů.

3.4.2 K-bodové křížení

Zobecněným případem bodového křížení je případ, kdy místo jen jednoho bodu, od kterého se vymění jednotlivé geny v nových jedincích, se zvolí více míst. Toto se označuje jako k-bodové křížení, kde *k* je počet zvolených bodů [21].

jedinci před křížením	jedinci po křížení
(**0** | **1, 0** | **1, 0, 1** | **0, 1**)	(**0** | 1, 1 | **1, 0, 1** | 0, 0)
(1 | 1, 1 | 0, 0, 1 | 0, 0)	(1 | **1, 0** | 0, 0, 1 | **0, 1**)

Tabulka 3.8 k-bodové křížení

V tabulce - Tabulka 3.8 k-bodové křížení - je vidět příklad tříbodového křížení. Chromozomy byly rozděleny na 4 řetězce a následně došlo k výměně jednotlivých úseků.

3.4.3 Uniformní křížení

U *k*-bodového křížení lze ještě identifikovat jeden typ křížení. Pokud číslo k bude rovno $N - 1$ (N je celková délka chromozomu), jedná se o tzv. uniformní křížení [2]. Chromozomu se přiřadí s pravděpodobností $P = 0{,}5$ hodnota genu od prvního nebo druhého rodiče. A druhý jedince získá vždy hodnoty genu od druhého rodiče. Uniformní křížení je vidět v Tabulka 3.9 Uniformní křížení pod tímto odstavcem.

jedinci před křížením	jedinci po křížení
(**0, 1, 0, 1, 0, 1, 0, 1**)	(**0**, 1, 1, 0, **0**, **1**, 0, 0)
(1, 1, 1, 0, 0, 1, 0, 0)	(1, **1**, **0**, **1**, 0, 1, **0**, **1**)

Tabulka 3.9 Uniformní křížení

3.4.4 Operátor křížení pro permutační kódování

Všechna křížení lze uplatnit i při jiném kódování než binárním. Vše funguje úplně stejně pro celá čísla i reálná čísla s desetinou čárkou. Hodnoty genů se v jednotlivých chromozomech vymění bez sebemenších problémů.

Do kolize však přichází operátor křížení u permutačního kódování. Jak bylo vysvětleno výše, permutační kódování se používá pro plánování tras (např. se tímto problémem zabývá [23]), kdy uživatel musí projet všemi místy na mapě, ale žádné nesmí projet dvakrát. Geny představují jednotlivá místa tak, jak je uživatel projede za sebou. V tomto okamžiku by mohlo dojít k tomu, že se v jednom chromozomu ocitnou geny se stejnou hodnotou a znamenalo by to, že jedno místo bylo navštíveno dvakrát. Pro příklad je tento efekt zobrazen v následující tabulce:

jedinci před křížením	jedinci po křížení
(5, 1 | 4, 2 | 3, 6)	(5, 1 | 5, 3 | 3, 6)
(4, 2 | 5, 3 | 6, 1)	(4, 2 | 4, 2 | 6, 1)

Tabulka 3.10 Permutační kódování a křížení

Křížením došlo k tomu, že v prvním nově vzniklém jedinci by uživatel projel místa 5 a 3 dvakrát a vůbec by neprojel místy 4 a 2. U druhého jednice je tomu přesně naopak. Tímto by nebyla splněna základní podmínka, proto je třeba po křížení dále nově vzniklé jedince upravovat.

Mechanizmus úprav probíhá zhruba takto:

- provede se křížení od zvoleného bodu, ostatní pole zůstávají volná
- vyplní se hodnoty genů od původních rodičů, které nejsou v kolizi s geny, které už nový chromozom obsahuje
- doplní se zbylé hodnoty podle toho, jaké číslo odpovídá jakému

Pro chromozomy uvedené v tabulce by postup vypadal následovně:

- vytvořily by se chromozomy (*, *, 5, 3, *, *) a (*, *, 4, 2, *, *)
- doplnění hodnot původních rodičů: (*, 1, 5, 3, *, 6), (*, *, 4, 2, 6, 1)
- zbylé hodnoty podle stanovení výměn $(5 \leftrightarrow 4, 3 \leftrightarrow 2, 1 \leftrightarrow 6)$

Na výměny se přišlo následovně. Z křížení víme, že č. 5 odpovídá č. 4 a č. 3 odpovídá č. 2, a poslední dvě čísla si samozřejmě musí odpovídat. To si lze ověřit v dalším kroku, kde byly chromozomům doplněny hodnoty od rodičů. Je vidět, že v posledním genu si odpovídá č. 6 a č. 1. Složitější řešení výměn by nastalo, pokud by uživatel musel projet více míst. A bez tohoto odvození by nebylo možné zbylé položky doplnit.

jedinci před křížením	jedinci po křížení
(5, 1 | 4, 2 | 3, 6)	(4, 2 | 5, 3 | 6, 1)
(4, 2 | 5, 3 | 6, 1)	(5, 3 | 4, 2 | 1, 6)

Tabulka 3.11 Výsledné chromozomy při použití permutačního kódování

3.5 Mutace

Další kombinačním operátorem, kterým může a nemusí být vystaven nově vzniklý chromozom, je mutace. Rozdílem oproti operátoru křížení je to, že mutace probíhá v rámci jednoho chromozomu a chromozomy nemění geny mezi sebou [21]. Obecně platí, že mutaci podléhá podstatně menší počet jedinců, než je tomu u křížení, jelikož mutace probíhá s velmi malou pravděpodobností [15].

3.5.1 Základní mutace

Jak bylo řečeno, mutace probíhají s velmi malou pravděpodobností, např. podle lit. [15] mutace probíhá s pravděpodobností v intervalu $\langle 0{,}05; 0{,}15 \rangle$. Při mutaci dochází náhodně ke změně hodnoty určitého genu chromozomu. Při binární kódování se z 1 stane 0 a naopak.

jedinec před mutací	jedinec po mutaci
(0, 1, 1, 0, 0, 1, 0, 1)	(0, 1, 1, 0, 0, 1, 0, 1)
(1, 1, 1, 0, **0**, 1, 0, 0)	(1, 1, 1, 0, **1**, 1, 0, 0)
(1, 1, 1, 1, 1, 0, 0, 1)	(1, 1, 1, 1, 1, 0, 0, 1)
(**0**, 0, 1, 0, 0, 1, 0, 1)	(**0**, 0, 1, 0, 0, 1, 0, 1)

Tabulka 3.12 Přehled jedinců před a po mutaci

3.5.2 Mutace pro permutační kódování

U permutačního kódování nastává problém také s mutací. U ní dochází k tomu, že náhodně je vybrán jeden gen chromozomu a je u něj změněna hodnota. Nastává ten samý problém, jako u křížení, kde se může stát, že chromozom po mutaci bude obsahovat 2 stejné hodnoty. Při mutaci lze tento problém relativně jednoduše vyřešit. Místo změny hodnoty pouze jednoho genu, se provede vzájemná výměna hodnotu genů dvou.

jedinec před mutací	jedinec po mutaci
(5, **1**, 4, **2**, 3, 6)	(5, **2**, 4, **1**, 3, 6)

Tabulka 3.13 Mutace u permutačního kódování

3.5.3 Inverze

Posledním často využívaným genetickým operátorem u permutačního kódování je inverze [2]. Při inverzi dochází k otočení části řetězce mezi dvěma zvolenými body.

jedinec před inverzí	jedinec po inverzi
(5, **1, 4, 2, 3,** 6)	(5, **3, 2, 4, 1**, 6)

Tabulka 3.14 Inverze u permutačního kódování

V tabulce - Tabulka 3.14 Inverze u permutačního kódování - je vidět mechanismus inverze. Druhý gen se přesunul na pozici čtvrtého a ten se přesunul naopak na pozici druhého. A také si vyměnil pozici třetí gen se čtvrtým.

Použití inverze pro jiné kódování není vyloučeno, ale je třeba si uvědomit fakt, že inverze zaměňuje pořadí jednotlivých genů a ne jejich hodnotu. Proto použití inverze u jiného typu kódování tedy není vůbec nutné, jelikož je to operace navíc, která stojí drahocenný výpočetní čas a nijak zvlášť účinnost genetického algoritmu nezvyšuje [2].

3.6 Nová populace

Genetický algoritmus pracuje s celou populací řešení, ne pouze s jedním. V předchozích kapitolách byla vysvětlena celá cesta od začátku algoritmu, přes přidělení hodnocení fitness až k selekci jedinců, kteří byli použiti k následné reprodukci. Pomocí genetických operátorů je reprodukce provedena a vzniká nová populace. Nicméně dochází ke střetu právě nově vzniklých jedinců s generačně staršími rodiči.

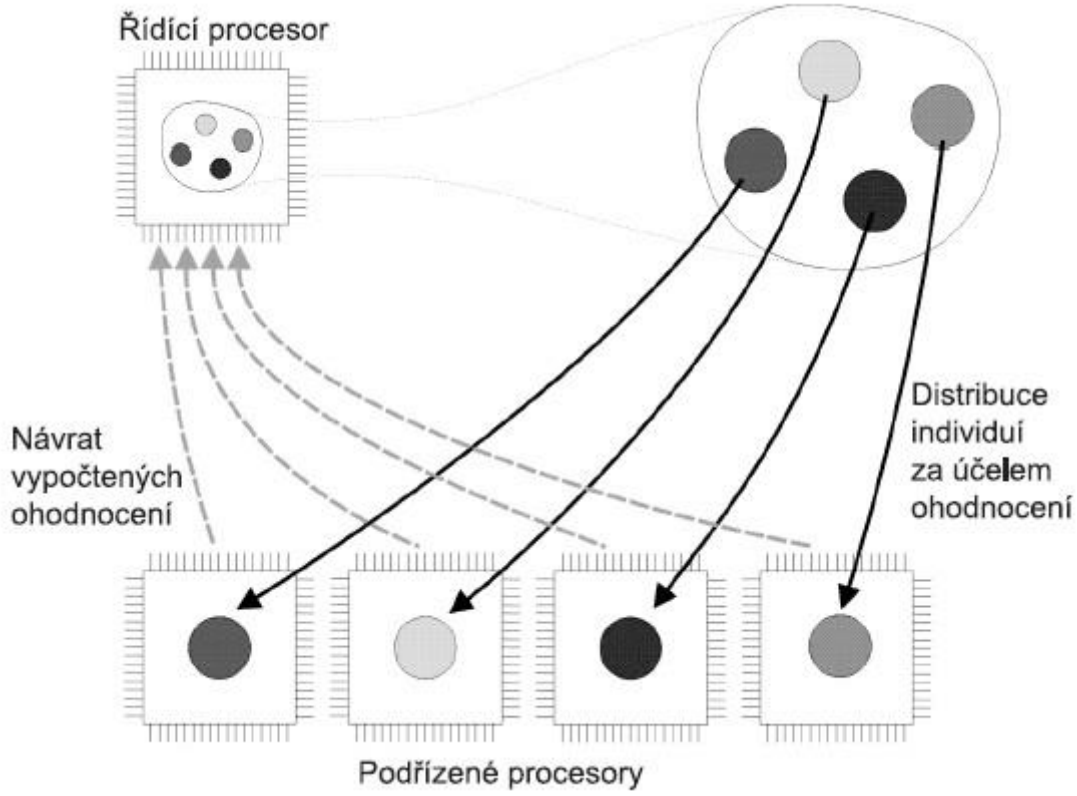
U GA je přijat efekt přímé evoluce. Znamená to to, že jedinec ovlivňuje celou populaci pouze v období jedné generace [2]. Tedy od svého vzniku k přidělení hodnocení, následné selekci a reprodukci, až po samotný zánik. Tímto způsobem však může docházet k tomu, že pokud jedinec s obstojným hodnocením není selekčním mechanismem vybrán k reprodukci, je pro GA tento jedinec zcela ztracen a jeho možní potomci také. Není vyloučeno, že může vzniknout nějakými mutacemi a křížením z jiných jedinců.

Proto je třeba nějakým způsobem tyto jedince uchovávat pro další generace. Používá se převážně dvou možností, které se liší takřka jen počtem uchovávaných jedinců. První možností je uchovávání pouze několika nejlepších jedinců z generace. Jedná se o opravdu nejlepší jedince, dalo by se říci vlajkové lodě neboli elita populace [2].

Druhou možností je zachování naopak velkého množství jedinců populace, pouze ta nejhorší řešení jsou vyřazena a v dalších generacích se s nimi nepracuje. Tato řešení jsou nahrazena novými jedinci vzniklými reprodukcí. V GA dochází k zachování toho nejlepšího, co zatím bylo vytvořeno, bohužel si s sebou GA nese i zpomalení celkového vývoje, dochází k nárůstu výpočetního času i k zvětšení počtu generací [2].

3.7 Paralelní genetické algoritmy

Genetické algoritmy pracují, jak bylo zmíněno už několikrát, s celou populací řešení. Aby se zkrátil výpočetní čas celé operace, proces se rozloží a místo provádění výpočtu pouze na jednom procesoru, probíhá výpočet na několika procesorech zároveň.



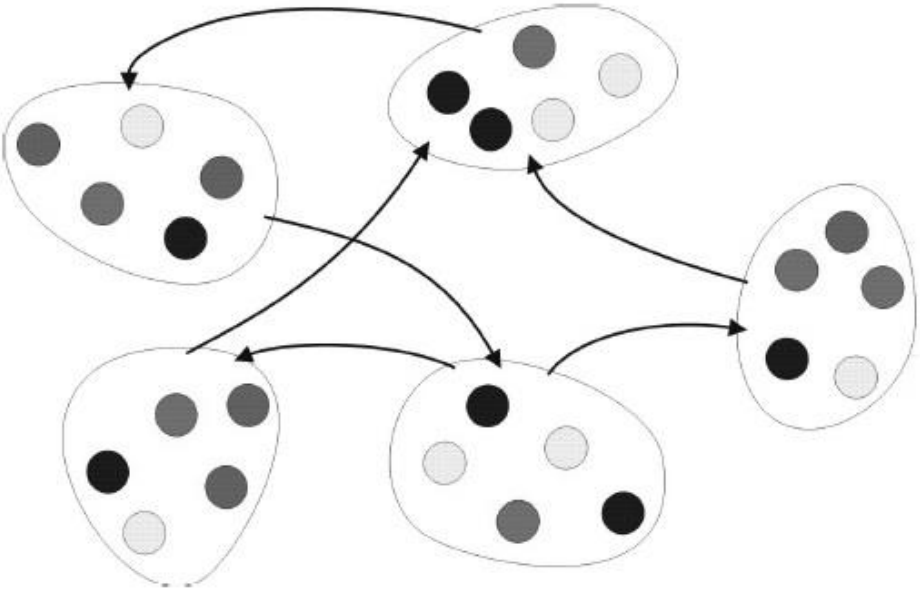
Obrázek 3.9 Rozložení populace k vypočtení hodnocení (fitness) [2]

Na obrázku (Obrázek 3.9) je vidět rozložení populace za účelem vypočtení hodnotící funkce fitness. Celkový průběh paralelního GA se od jednoduché výpočtu moc neliší. Jediným rozdílem je pouze výpočet hodnocení na podřízených procesorech. Po vypočtení hodnocení jsou jedinci posíláni na řídící procesor, kde dochází k vlastnímu mechanizmu GA, jako je selekce apod.

Úspora času nastává při stanovování hodnocení jedinců a jedná se o jednodušší metodu paralelizace algoritmu.

Druhou možností je ostrovní paralelizace. Pří této metodě je rozložena samotná populace do několika menších populací, které tvoří jaké si ostrovy (proto ostrovní populace). Tyto jednotlivé části spolu mohou komunikovat, to znamená, že občas dochází k výměně jedinců, nebo mezi ostrovy nemusí být žádná vazba a každá skupina se vyvíjí samostatně. Lze například na jednotlivé skupiny aplikovat jiné genetické operátory, jiné metody selekce a uživatel získává, rozdílné řešení daného problému.

Tento způsob decentralizace je oproti normálnímu postupu daleko zajímavější, už jen z důvodu vzniku několika možných koncových řešení daného problému při jednom použití GA. Pokud uživatel chce srovnání (více možných řešení), jednoduchý postup by bylo třeba opakovat vícekrát a bohužel by tím pádem opět došlo k nárůstu výpočetního času, což samozřejmě není chtěné. Schéma ostrovní paralelizace je vidět na následujícím obrázku.

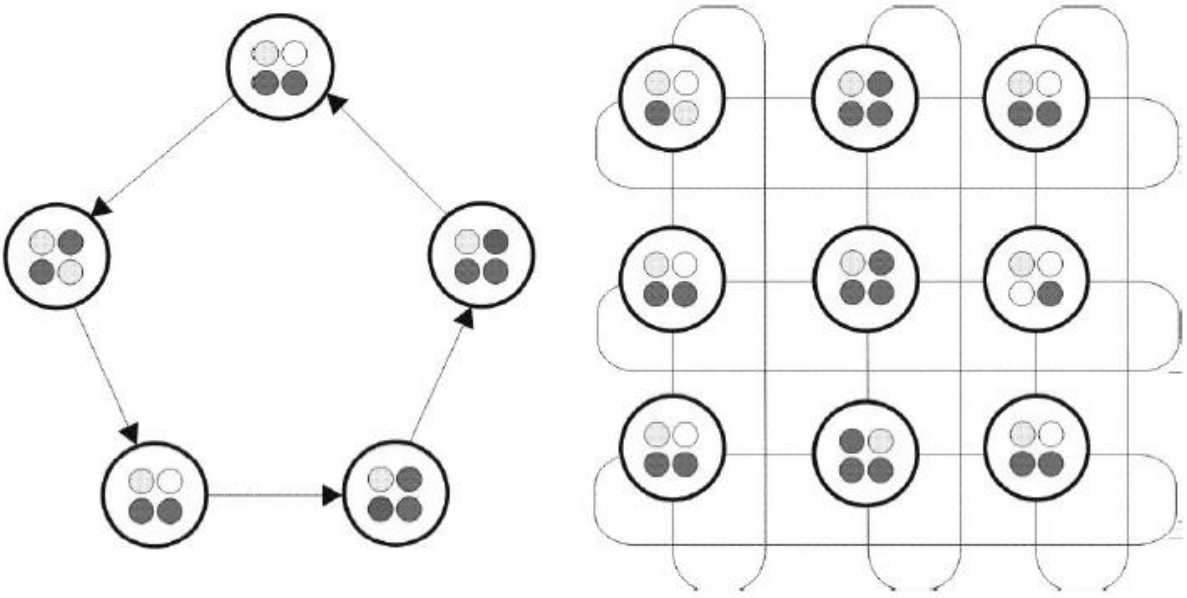


Obrázek 3.10 Ostrovní paralelizace [2]

V tomto případě jsou mezi jednotlivými ostrovy znázorněny vazby, jako komunikace mezi nimi. Pokud by vazby mezi jednotlivými ostrovy zmizely, lze pohlížet na každý ostrov jako na samostatný genetický algoritmus a lze ovlivňovat jeho vývoj samostatně.

Co se musí při ostrovní paralelizaci stanovit, je určitě počet jedinců jednoho ostrova. Toto se stanovuje podle výpočetních možností, které jsou uživateli k dispozici. Dalším faktorem je vymezení komunikace mezi jednotlivými ostrovy. Aby bylo stanoveno, na jaký ostrov se vybraný jedinec přesune. Na obrázku - Obrázek 3.10 - je komunikace zvolená náhodně a není v ní žádné velké uspořádání. Často však bývá komunikace uspořádaná např. do kruhu, do hvězdy nebo se jedná o složitější propojení, jako je maticové uspořádání (viz. Obrázek 3.11).

U komunikace je také důležité stanovit interval, v jakém bude docházet přesunu jedinců mezi ostrovy, kolik bude celkově probíhat migrací, zda jedinci z původních ostrovů budou vymazáni nebo jak budou jedinci k migraci vybíráni.



Obrázek 3.11 Kruhové a maticové uspořádání komunikace [2]

3.8 Hybridní genetické algoritmy

V kapitolách předcházejících byl vysvětlován princip fungování genetického algoritmu. Několikrát byly zmíněné jeho silné a slabé stránky, kdy tento typ optimalizační metody použít, kdy naopak ne.

Už z názvu hybridní GA vyplývá, že se nebude jednat čistě o genetické algoritmy, ale o spojení např. s jinou optimalizační metodou, za účelem zvýšení efektivnosti výpočtu řešení daného problému. Jde tedy o to využít silných stránek GA (velký prohledávaný prostor, univerzálnost, přijatelný čas výpočtu, atd.) a obohatit algoritmus o věci, jako jsou speciální genetické operátory, speciální ohodnocující funkce, které byly přejaté z metod aplikovaných na daný problém dříve. Hybridní GA jsou šité na míru určitému problému, proto je ho třeba dokonale znát.

Způsob jak by měl být hybridní GA tvořen je zhruba takový:

- reprezentace jedinců by měla vycházet z konkurenčního algoritmu
- mělo by se využívat co nejvíce kladných věcí přejatých od konkurenčního algoritmu
- genetické operátory by měly být v souladu s danou reprezentací jedinců

Na začátku je vhodné hybridnímu GA dát řešení, které bylo vypočteno v konkurenčním algoritmu. Tím se zaručí, že nový algoritmus nemůže dosáhnout horšího řešení, než tomu bylo v předcházejícím výpočtu.

V dalším kroku je třeba z vytvořené populace počátečních řešení vybrat rodičovské páry vhodné k reprodukci. K tomu dochází pomocí vhodně upravených selekčních mechanizmů opět vytvořených na základně znalostí z předchozího algoritmu. V následné reprodukci jsou použity genetické operátory, které nejsou tak jednoduché. Jsou modifikované díky znalosti daného problému, čím se stávají samozřejmě efektivnějšími. Tímto dojde k vytvoření nové populace a celý proces se znovu opakuje, dokud výsledek není globálním optimem prohledávaného prostoru (v ideálním případě). Schematicky se tento postup nijak zvlášť neliší od běžného GA. Pouze došlo k zdokonalení díky vědomostem uživatele a díky následné aplikaci těchto znalostí.

Takto upravený genetický algoritmus může dosáhnout mnohem lepších výsledků v kratším čase, než by tomu bylo u obecného algoritmu.

3.9 Použití genetických algoritmů

U optimalizace GA za zmínku stojí také použití. Různé metody optimalizace popsány výše jsou prováděny za účelem zefektivnění celého procesu. To by ovšem nefungovalo, pokud by vše nebylo řízeno logickým úsudkem uživatele.

Pro GA je důležité vycházet z toho, k čemu jsou určené. Tedy k prohledávání širokého prostoru. Je zaručena různorodost v celém prohledávaném prostoru. A jako výsledek se považuje optimální řešení v relativně přijatelném čase.

Problémem, který tento obrovský rozměr znázorňuje, je např. umístění *n* dam na šachovnici o *n* sloupcích a *n* řádcích tak, aby ani jedna dáma nebyla ohrožena postavením jiné dámy (více lze nalézt v [14]). V tomto případě prohledávaný prostor roste exponenciálně a řešení by se jiným, než genetickým algoritmem dostávalo těžko [2].

Proto znalost konkrétního problému, na který má být použitý GA, je nezbytná. Jak bylo zmíněno, GA je velmi obecnou metodou a pokud existuje pro určitý problém speciální algoritmus, určitě je vhodné jej použít. Vše záleží na správné volbě uživatele. Jestliže zvolený GA u daného problému nevykazuje dostačujících výsledků, nemusí se to znamenat, že algoritmus pracuje nesprávně, častěji se jedná spíše o špatné nastavení ze strany uživatele. A to je třeba míti na paměti a ještě před rozhodnutím zda genetický algoritmus použít.

4 Problém obchodního cestujícího

Tento příklad je jedním z nejznámějších problémů, na kterém se dají uplatnit GA, a zabíhá do praktických odvětví, jako je logistika a plánování [2], konkrétně např. stanovení optimálního postupu osazování součástek do desky plošných spojů s minimalizací neefektivních pohybů osazovací hlavy [15]. Problém popsaný v [23] je toho zářným příkladem, jelikož z obecného problému obchodního cestujícího vychází.

4.1 Představení problému

Úloha obchodního cestujícího popisuje teoretickou cestu prodejce určitého zboží jednotlivými městy, kde se snaží své zboží prodat. V každém městě se může zastavit pouze jednou a poté se vrátit do výchozího bodu odkud se na trasu vydal. Důležitým faktorem jsou vždy náklady, proto samozřejmě obchodník chce tyto náklady minimalizovat. Tyto náklady se odvíjejí obvykle od vzdálenosti mezi jednotlivými místy, ovšem náklady se mohou lišit i způsobem dopravy na trati mezi zastávkami. Posledním omezením je pravidlo, že nezáleží, jakým směrem se obchodník mezi dvěma městy pohybuje. Náklady zůstávají vždy stejné.

4.2 Řešení pomocí GA

Velkou výhodou při řešení problému obchodního cestujícího je zvolení hodnotící funkce fitness, jelikož stačí obyčejně sečíst náklady na danou cestu a tím je získáno ohodnocení příslušného jedince. Pomocí přidělené hodnoty fitness se uskutečňuje následná selekce jedinců vhodných k rekombinaci mechanizmem ruletového kola.

4.3 Kódování

Co se týče kódování, tak zde se v průběhu let při řešení objevily čtyři způsoby reprezentace individuí [2]. Jednalo se o klasické binární kódování, tzv. sousedskou reprezentaci, ordinální reprezentaci a také přirozenou reprezentaci individuí.

Při použití binárního kódování, byl algoritmus limitován omezenou možností kombinace a tím pádem i použité genetické operátory měly na celý chod značné destruktivní účinky [2]. Vše se muselo řešit pomocí opravných algoritmů a použitím speciálních genetických operátorů, kvůli kterým docházelo k nárůstu výpočetního času.

Dalším principem je sousedská reprezentace, která funguje tak, že jednotlivá města jsou očíslována přirozenými čísly a celková cesta je charakterizována vektorem, jehož délka je rovna počtu měst [2]. Číslo i se v chromozomu na j-té pozici nachází právě tehdy, když se cestující přesunuje z místa j do místa i. Vektor (4, 3, 6, 5, 2, 1) ve skutečnosti představuje cestu 1 – 4 – 5 – 2 – 3 – 6 – 1. Pro vysvětlení v prvním kole je i = 4 a nachází se na pozici j = 1. Proto první přesun probíhá z místa 1 do místa 4. V druhém kole je i = 3 a pozice j = 2 (přesun z 2 na 3). Třetí kolo: i = 6, j = 3 (přesun z 3 – 6). A tak to probíhá až do konce chromozomu, přičemž vzniká celá obchodní cesta.

Tato metoda v sobě opět skrývá několik problémů. Např. vektor (3, 5, 1, 2, 6, 4) reprezentuje cestu 1 – 3 – 1.

- První kolo: i = 3, j = 1 (přesun z 1 – 3)
- Druhé kolo: i = 5, j = 2 (přesun z 2 – 5)
- Třetí kolo: i = 1, j = 3 (přesun z 3 – 1)

Ukončením třetího kola došlo k zacyklení celé cesty, jelikož z města 3 má cesta pokračovat do města 1. Toto není přípustné řešení, jelikož nebyla splněna podmínka průchodu všemi městy
v rámci jedné obchodní cesty. Obchodník se vrátil do města 1, aniž by prošel zbylými 4 městy. Opět je tedy nutno použít speciálních genetických operátorů, které by nezpůsobovaly degeneraci cesty, jako např. bylo popsáno výše. Toto opět přináší negativní efekt na celkový výpočetní čas operace.

U ordinální reprezentace je cesta zakódována pomocí chromozomu, který můžeme označit jako seznam *N* měst, *i*-té číslo v chromozomu může nabývat hodnoty $j \in \langle 1, (N - i + 1)\rangle, j \in Z$ a tento seznam měst je vytvořen na základě tzv. referenčního seznamu *R* (viz lit. [15]). Tento referenční seznam se skládá z měst seřazených za sebou v určitém neměnném pořadí. Pro názornost je dán příklad, ve kterém je celkově 6 měst, a v referenčním seznamu jsou města seřazena vzestupně. Nyní bude popsáno dekódování např. chromozomu (1, 3, 3, 1, 2, 1).

Chromozom	Cesta	Referenční seznam R
1 3 3 1 2 1	1	1 2 3 4 5 6
1 **3** 3 1 2 1	1 – 4	2 3 4 5 6
1 3 **3** 1 2 1	1 – 4 – 5	2 3 5 6
1 3 3 **1** 2 1	1 – 4 – 5 – 2	2 3 6
1 3 3 1 **2** 1	1 – 4 – 5 – 2 – 6	3 6
1 3 3 1 2 **1**	1 – 4 – 5 – 2 – 6 – 3	3

Tabulka 4.1 Ordinální reprezentace

Obecný princip dekódování spočívá v tom, že z chromozomu se vezme postupně zleva doprava číslo k na i-té pozici, které představuje *k*-tý prvek referenčního seznamu *R*. Následně se tento prvek vyjme z referenčního seznamu a umístí se do cesty. V prvním kole je $k = 1$, proto se vezme první prvek z *R*, což je 1. Cesta začíná v městě 1 a v *R* město 1 už dále nefiguruje. V druhém kole $k = 3$, z *R* je vyjmut třetí prvek (město 4) a je přidán na cestu. Takto se postupuje až do konce chromozomu, tedy do doby, dokud není referenční seznam *R* prázdný. Tímto způsobem vznikla okružní cesta: 1 – 4 – 5 – 2 – 6 – 3 – 1.

Výhodou této metody je i fakt, že pomocí genetického operátoru jednobodového křížení vzniknou vždy chromozomy představující přípustná řešení a nenastane např. předčasné ukončení cesty, jako k tomu mohlo nastat u sousedské reprezentace [2]. Ovšem další jednoduché genetické operátory nefungují tak spolehlivě a smysluplně, proto použitím jen operátoru jednobodového křížení nebylo získáno dostačující řešení a také se od tohoto způsobu kódování upustilo [15].

Poslední a nejúspěšnější metodou proto zůstala ta nejjednodušší reprezentace a to je přirozená. U tohoto způsobu jsou jednotlivá města očíslována a v chromozomu jsou za sebe řazena v pořadí, v jakém jsou cestou projížděna. Např. chromozom (1, 2, 5, 6, 4, 3) představuje okružní cestu 1 – 2 – 5 – 6 – 4 – 3 – 1. V průběhu let se pro tuto metodu kódování vyvinuly speciální genetické operátory [15], které budou popsány v následující kapitole.

4.4 Genetické operátory

Pro řešení je zvolen princip přirozené reprezentace a města jsou do chromozomu ukládána postupně za sebou, tak jak jsou procházena na obchodní cestě.

Rekombinace probíhá použitím již známých genetických operátorů křížení a mutace, ovšem jedná se o speciální typy daných operátorů, které byly navrženy pro tento konkrétní problém.

4.4.1 Křížení se zachováním pořadí

Prvním typem je křížení se zachováním pořadí (OX – Order Crossover) [2]. Tento způsob je typ dvoubodového křížení. OX rozdělí oba rodičovské chromozomy na 3 části. Prostřední část chromozomů se ponechá novým jedincům nezměněna a počínaje druhým řezem jsou přidělována jednotlivá města od druhého z rodičů v pořadí, v jakém se v něm nacházejí (vynechávají se města již obsažená v prostřední části nových jedinců).

Pro názornost je uveden problém obchodního cestujícího s 10 městy.

rodiče	chromozom
1.	(1, 3, 2, 5, 6, 7, 4, 9, 10, 8)
2.	(1, 4, 5, 6, 2, 3, 7, 9, 8, 10)
1. fáze	chromozom
1.	(#, #, #, 5, 6, 7, 4, 9, #, #)
2.	(#, #, #, 6, 2, 3, 7, 9, #, #)
potomci	chromozom
1.	(1, 2, 3, 5, 6, 7, 4, 9, 8, 10)
2.	(1, 5, 4, 6, 2, 3, 7, 9, 10, 8)

Tabulka 4.2 Operátor křížení se zachováním pořadí

V první fázi se zkopírují prostřední části obou rodičovských chromozomů do potomstva a ostatní místa se nechají volná. Počínaje druhým řezem se vyplní zbývající pozice městy v pořadí, ve kterém jsou u druhého rodiče. Pokud se bude tvořit první potomek, prostřední část je rovna sekvenci 5, 6, 7, 4, 9, nyní se OX ptá, co následuje po městu 9 u druhého rodiče. Je to 10 a tedy OX přiřadí na další pozici město 10. A takto projde postupně všechny zbývající pozice s vynecháním již použitých čísel měst a vytvoří nového potomka. Stejný postup opakuje i pro vznik druhého potomka.

Pro OX je nejdůležitější zachování pořadí měst, které pro řešení problému obchodního cestujícího není až tak podstatné a slibné části cesty jsou kvůli tomuto operátoru ničené, a proto tento operátor není výhodný [15].

4.4.2 Křížení s částečným zobrazením

Dalším využívaným typem křížení je tzv. křížení s částečným zobrazením (PMX – Partially Mapped Crossover) [15]. Jedná se opět o typ dvoubodového křížení, při kterém se také kopírují prostřední části řetězce. Na zbylé pozice jsou města doplňována přiřazením buď měst od druhého z rodičů, které nezpůsobují konflikt nebo pomocí částečného zobrazení, které

vzniká v prostřední části nově vznikajících řetězců. Opět je uveden příklad použití operátoru PMX pro cestu skrz 10 měst.

rodiče	chromozom
1.	(1, 4, 2, 5, 7, 3, 9, 10, 8, 6)
2.	(1, 5, 10, 8, 2, 4, 6, 3, 9, 7)
1. fáze	chromozom
1.	(1, #, 10, 5, 7, 3, 9, #, #, #)
2.	(1, #, #, 8, 2, 4, 6, 10, #, #)
částečná zobrazení	$5 \leftrightarrow 8, 2 \leftrightarrow 7, 3 \leftrightarrow 4, 6 \leftrightarrow 9$
2. fáze	chromozom
1.	(1, $5 \leftrightarrow 8$, 10, 5, 7, 3, 9, $3 \leftrightarrow 4$, $6 \leftrightarrow 9$, $2 \leftrightarrow 7$)
2.	(1, $3 \leftrightarrow 4$, $2 \leftrightarrow 7$, 8, 2, 4, 6, 10, $5 \leftrightarrow 8$, $6 \leftrightarrow 9$)
potomci	chromozom
1.	(1, 8, 10, 5, 7, 3, 9, 4, 6, 2)
2.	(1, 3, 7, 8, 2, 4, 6, 10, 5, 9)

Tabulka 4.3 Operátor křížení s částečným zobrazením

V první fázi operátor PMX zkopíruje prostřední části rodičů do nových jedinců a dále doplní na zbývající pozice ta města, která nejsou v kolizi s jinými městy v prostřední části. Pro prvního jedince je prostřední část rovna 5, 7, 3, 9 a doplněním přibyla od druhého rodiče další dvě města (město 1 a 10). Na zbývající pozice, na kterých by došlo ke konfliktu, se vypíšou částečná zobrazení (např. na druhé pozici u prvního jedince by měla být 5, ale ta se již nachází v prostřední části a proto se vypíše zobrazení $5 \leftrightarrow 8$) a následně se vybere na těchto pozicích to město, které v řetězci není ještě obsaženo. Pokud nastane takový případ, že na některých pozicích nebude známé ani částečné zobrazení, přiřadí se na tyto pozice zbývající čísla měst náhodně s podmínkou bezkonfliktnosti s jinými městy.

Výhodou operátoru PMX proti OX tkví v tom, že částečně zachovává kromě pořadí měst i jednotlivé dílčí slibné bloky cest, které jsou při řešení důležité [15].

4.4.3 Křížení s rekombinací hran

Posledním a zatím nejúspěšnějším operátorem křížení je tzv. křížení s rekombinací hran (ERX – Edge Recombination Crossover) [2]. Nejedná se o typický operátor křížení, jelikož ze dvou rodičů vzniká pouze jeden potomek. Každý úsek v tomto jedinci je přejat od jednoho rodiče. Aby bylo možno nového jedince vytvořit, je třeba pracovat s tzv. hranovou tabulkou, která složí k zobrazení sousedů každého města vyskytujících se v obou rodičích [15].

Po vytvoření hranové tabulky, se náhodně vybere město s nejnižším počtem sousedů a umístí se na první pozici potomka. Je-li těchto měst více, náhodně se vybere jedno z nich. Následně se z hranové tabulky toto město vymaže z výčtu sousedů jednotlivých měst. Volba dalšího města připadá na sousedy prvně vybraného města. Opět se vybere město s nejmenším počtem

sousedů. Takto se postupuje až do vytvoření celé obchodní cesty. Pro názornost opět uveden příklad problému s 10 městy.

rodiče	chromozom
1.	(1, 4, 2, 5, 7, 3, 9, 10, 8, 6)
2.	(1, 5, 10, 8, 2, 4, 6, 3, 9, 7)

Tabulka 4.4 Rodiče u křížení ERX

město	sousedé	město	sousedé
1	4, 5, 6, 7	6	1, 3, 4, 8
2	4, 5, 8	7	1, 3, 5, 9
3	6, 7, 9	8	2, 6, 10
4	1, 2, 6	9	3, 7, 10
5	1, 2, 7, 10	10	5, 8, 9

Tabulka 4.5 Hranová tabulka pro křížení ERX

V prvním kroku se vybere náhodně město s nejmenším počtem sousedů. Z tabulky - Tabulka 4.5 – je vidět, že může být zvoleno město 2, 3, 4, 8, 9 nebo 10. Nechť je vybráno např. město 2. Nyní je město 2 vymazáno ze seznamu sousedů ostatních měst. V dalším kole padá volba na sousedské město k městu 2. Mezi sousedy města 2 patří města: 4, 5, 8.

cesta	2 –		
město	sousedé	město	sousedé
1	4, 5, 6, 7	6	1, 3, 4, 8
2	4, 5, 8	7	1, 3, 5, 9
3	6, 7, 9	8	6, 10
4	1, 6	9	3, 7, 10
5	1, 7, 10	10	5, 8, 9

Tabulka 4.6 Hranová tabulka po výběru města 2

Opět musí být splněna podmínka, že vybrané město musí mít nejméně sousedů. V tomto případě volba připadá buď na město 4, nebo 8, jelikož oba dva mají shodně už jen po dvou sousedech. Je tedy náhodně zvoleno město 4.

cesta	2 – 4 –		
město	sousedé	město	sousedé
1	5, 6, 7	6	1, 3, 8
2	5, 8	7	1, 3, 5, 9
3	6, 7, 9	8	6, 10
4	1, 6	9	3, 7, 10
5	1, 7, 10	10	5, 8, 9

Tabulka 4.7 Hranová tabulka po výběru města 4

Následně je město 4 vymazáno ze seznamu sousedů ostatních měst a přistupuje se k výběru dalšího města stejným způsobem. Opakování probíhá do vytvoření kompletní obchodní cesty 2 – 4 – 1 – 6 – 8 – 10 – 5 – 7 – 3 – 9 – 2.

Experimentálně bylo dokázáno, že operátor křížení ERX dosahuje nejlepších výsledků, jelikož přejímá slibné úseky cesty od svých rodičů [2]. Tento efekt se dá ještě znásobit výběrem měst, která jsou v sousedských seznamech obsažena vícekrát. Např. město 8 je sousedem pro město 10 u obou rodičů, proto kdyby došlo k tomu, že by se další město cesty muselo vybírat z více možných variant, bylo by preferováno právě město 8 před městy bez vícenásobného opakování.

V poslední řadě mezi genetické operátory používané pro řešení problému obchodního cestujícího jsou operátory mutace. Těmto mutacím podstupují s velmi malou pravděpodobností jedinci v populaci. Nejčastěji se pracuje s několika mutačními operátory zároveň, které se mezi sebou střídají [15]. Patří mezi ně:

- Zamíchání – vytvoří kompletně novou cestu
- Vložení – přesune jedno město na jinou pozici v chromozomu
- Inverze – vezme úsek chromozomu a inverzně ho otočí
- Přesun úseku – vezme náhodný úsek a přesune ho v rámci chromozomu náhodně na jiné místo

Poslední dva typy patří mezi celkem úspěšné mutace a často dochází k vylepšení jedince [15].

Závěrem je třeba říci, že pomocí takto nastavených parametrů je problém obchodního cestujícího řešitelný v rámci malého počtu procházených měst. Rozumných výsledků např. pro počet měst přesahujících tisíce, statisíce nebo i miliony by nebylo dosaženo. V lit. [15] byl problém obchodního cestujícího řešen experimentálně pro 30 měst a velikost populace 500 jedinců a použitím operátoru PRX s pravděpodobností $P_x = 0{,}95$. Optimální řešení pro takto zadanou úlohu se 19 z 20 případů podařilo nalézt do 100 iteračních cyklů. Pokud byla velikost populace zmenšena na 300 jedinců, GA optimální cestu nalezl jen zřídka a je vidět, že velmi záleží na citlivosti nastavení parametrů algoritmu. Pro úlohu se 75 městy, s velikostí populace větší než 2000 jedinců by dosahoval GA přípustných výsledků po 500 iteracích a což je neproveditelné a je třeba hledat dalších úprav a speciálních operátorů, kterým se tento efekt podaří redukovat.

Jeden takový navrhl P. Horský z MFF UK v Praze [15]. Operátor křížení pracuje se dvěma rodiči. Množina všech měst je rozdělena tak, že v jedné podmnožině jsou převzatá města od prvního rodiče a ve druhé podmnožině jsou zbývající města od druhého rodiče. Takto dojde k rozpadu cesty na úseky, které obsahují vždy jen města od jednoho rodiče. Operátor poté náhodně vezme jeden úsek a spojí ho s geometricky nejbližším úsekem. Takto postupuje až do uzavření obchodní cesty. Nejdůležitějším faktorem pro tento operátor je vhodné stanovení podmnožin. Např. pokud se na mapě města nacházejí v určité oblasti, je vhodné zvolit jako podmnožinu danou oblast, protože operátor přetne vazby na hranicích této oblasti a dále pracuje již jen s danou oblastí jako s celkem místo práce s jednotlivci v dané oblasti obsažených. Tímto dochází k zefektivnění celého výpočtu a dosažení výsledků v únosném čase [15].

5 Závěr

Tématem této práce byly optimalizační metody využitelné v rámci diskrétní simulační optimalizace. Práce se zaměřuje na specifické odvětví těchto metod a tím jsou genetické algoritmy a měla by přinést komplexní pohled na danou problematiku. V práci je vysvětlován princip fungování obecného genetického algoritmu ve své holé podstatě s popisem jednotlivých částí celého algoritmu.

Cílem této práce bylo provedení rešerše v oblasti genetických algoritmů. Tomu je věnovaná třetí část bakalářské práce, ve které jsou rozebírány jednotlivé kroky genetického algoritmu. Jsou popsány základní nejčastěji používané přístupy v oblasti kódování, hodnotící funkce fitness, selekce, křížení a mutace a jejich možné typy. Podstata jednotlivých algoritmů byla popsána a vysvětlena přímo na jednoduchých příkladech a to zejména pro názornou ilustraci jejich principu a lepší pochopení fungování daného algoritmu.

Ve čtvrté kapitole je ukázán genetický algoritmus na řešení problému obchodního cestujícího. Jsou opět popsány jednotlivé kroky genetického algoritmu a také jsou představeny speciální způsoby reprezentace, genetických operátorů křížení a mutace s jejich výhodami či nevýhodami vůči ostatním metodám.

Problém obchodního cestujícího byl vybrán s ohledem na možné využití metody v průmyslovém inženýrství. Problém se zabývá plánováním obchodní cesty skrze několik měst. Cesta, kterou by obchodník měl urazit, musí být samozřejmě nejkratší, aby bylo dosaženo nejmenších nákladů. Zde lze nalézt přímou souvislost s průmyslovým inženýrstvím, jelikož logistika a plánování je jedním z jeho odvětví. Obchodní cestu lze zaměnit například za dopravní vozík, který rozváží materiál uvnitř výrobní haly podniku nebo lze také problém transformovat na zásobování jednotlivých výrobních hal centrálním skladem podniku.

Co se týče dalších kroků, bylo by třeba teoretické fungování genetických algoritmů ověřit a otestovat prakticky. Z tohoto důvodu byly představeny ve druhé kapitole vybrané programy, které jsou schopny s genetickými algoritmy pracovat. Byl by vybrán určitý software, ve kterém by byl problém obchodního cestujícího naprogramován spolu s jednotlivými typy operací GA (selekce, křížení, mutace). Výsledkem by pak bylo komplexní srovnání jednotlivých metod spolu s odladěným nastavením genetického algoritmu, aby byl schopen generovat optimální řešení v přípustném čase.

Posledním krokem by bylo přenesení takto ověřeného algoritmu do podnikového sektoru na reálnou situaci v rámci určité společnosti, popřípadě navržení možného zlepšení, s analýzou přínosu použití genetických algoritmů.

6 Citovaná literatura

[1] TVRDÍK, Josef. *Stochastické algoritmy pro globální optimalizaci a jejich aplikace ve výpočetní statistice.* Ostrava, 2003. Habilitační práce. Ostravská univerzita v Ostravě.

[2] HYNEK, Josef. *Genetické algoritmy a genetické programování*. Praha: Grada, 2008. Průvodce (Grada). ISBN 978-80-247-2695-3.

[3] RAŠKA, Pavel. *Optimalizační metody pro diskrétní simulaci výrobních systémů a výrobních procesů ve strojírenství*. Plzeň, 2012. Disertační práce. Západočeská univerzita v Plzni. Vedoucí práce Doc. Ing. Václav Votava, CSc.

[4] CHIN-CHIH HSU, J. MARTIN, S. YAMADA, H. FUJIKAWA a K. SHIDA. An effectiveness analysis of genetic operators for TSP. *Proceedings of the IEEE International Symposium on Industrial Electronics* [online]. IEEE, 1995, 766-770 [cit. 2016-04-29]. DOI: 10.1109/ISIE.1995.497282. ISBN 0-7803-2683-0. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=497282.

[5] KARASEK, Jan, Radim BURGET, Lukas POVODA, Malay Kishore DUTTA a Anushikha SINGH. Genetic programming operators for work-flow optimization in logistic distribution centers. *2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom)* [online]. IEEE, 2014, 105-109 [cit. 2016-04-29]. DOI: 10.1109/MedCom.2014.7005985. ISBN 978-1-4799-5097-3. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7005985.

[6] BEASLEY, David, David R. BULL a Ralph R. MARTIN. *An Overview of Genetic Algorithms: Part 1, Fundamentals* [online]. 1993,1-16 [cit. 2016-04-29]. Dostupné z: http://www.geocities.ws/francorbusetti/gabeasley1.pdf.

[7] K. MEFFERT a N. ROTSTAN. *JGAP* [Online]. 2002 [cit. 2016-05-05]. Dostupné z: http://jgap.sourceforge.net/.

[8] J. BROWNLEE. *Optimization Algorithm Toolkit* [online]. [cit. 2016-05-05]. Dostupné z: http://optalgtoolkit.sourceforge.net/about.php.

[9] Sourceforge.net. *GAlib Documentation* [online]. [cit. 2016-05-05]. Dostupné z: http://lancet.mit.edu/galib-2.4/.

[10] *SolveXL* [online]. 2013 [cit. 2016-05-05]. Dostupné z: http://www.solvexl.com/.

[11] A. SCHREYER. *GA Optimization for MS Excel* [online]. [cit. 2016-05-05]. Dostupné z: http://alexschreyer.net/projects/xloptim.

[12] Sourceforge.net. *List of GA Software* [online]. [cit. 2016-05-05]. Dostupné z: https://sourceforge.net/directory/development/algorithms/genetic-algorithms/.

[13] BABJAK, Jozef. *Genetické algoritmy* [online]. 2003 [cit. 2016-12-07]. Dostupné z: http://computerworld.cz/archiv/geneticke-algoritmy-19620.

[14] AGHAZADEH HERIS, Jalal Eddin a Mohammadreza Asgari OSKOEI. Modified genetic algorithm for solving n-queens problem. *2014 Iranian Conference on Intelligent Systems (ICIS)*[online]. IEEE, 2014, 1-5 [cit. 2015-12-06]. DOI: 10.1109/IranianCIS.2014.6802550. ISBN 978-1-4799-3351-8. Dostupné z:

http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6802550.

[15] MAŘÍK, Vladimír a kol. *Umělá inteligence. 3. díl*. 1. vyd. Praha: Academia, 2001. 328 s.: il. ISBN 80-200-0472-6.

[16] PETERKA, Ivo. *Genetické algoritmy*. Praha, 1999. Diplomová práce. Matematicko-fyzikální fakulta Univerzity Karlovy. Vedoucí práce RNDr. František Mráz.

[17] WEISSTEIN, Eric W. *Gray Code*. [online]. MathWorld, [cit. 2015-12-14]. Dostupné z: http://mathworld.wolfram.com/GrayCode.html.

[18] LIN, Guangming, Lishan KANG, Yongsheng LIANG a Ruhul SARKER. Analysis the Impact of Genetic Operators in Evolution Strategies. *2008 Second International Conference on Genetic and Evolutionary Computing* [online]. IEEE, 2008, 88-91 [cit. 2016-05-04]. DOI: 10.1109/WGEC.2008.107. ISBN 978-0-7695-3334-6. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4637401.

[19] RAGHAVJEE, Rushil a Nelishia PILLAY. A comparison of genetic algorithms and genetic programming in solving the school timetabling problem. *2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC)* [online]. IEEE, 2012, 98-103 [cit. 2015-12-05]. DOI: 10.1109/NaBIC.2012.6402246. ISBN 978-1-4673-4769-3. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6402246.

[20] MILLER, Brad L. a David E. GOLDBERG. *Genetic Algorithms, Tournament Selection* [online]. 1995, [cit. 2016-05-03]. Dostupné z: http://www.complex-systems.com/pdf/09-3-2.pdf.

[21] KOWALCZYK, R. Constrained genetic operators preserving feasibility of solutions in genetic algorithms. *Second International Conference on Genetic Algorithms in Engineering Systems* [online]. IEE, 1997, 191-196 [cit. 2016-05-05]. DOI: 10.1049/cp:19971179. ISBN 0852966938. Dostupné z: http://digital-library.theiet.org/content/conferences/10.1049/cp_19971179.

[22] FENGMING YE, Shingo MABU, LUTAO WANG a Kotaro HIRASAWA. Genetic Network Programming with new genetic operators. *2010 IEEE International Conference on Systems, Man and Cybernetics* [online]. IEEE, 2010, 3346-3353 [cit. 2016-05-04]. DOI: 10.1109/ICSMC.2010.5642337. ISBN 978-1-4244-6586-6. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5642337.

[23] MA, Yingjun a Xueyuan CUI. Solving the fuel transportation problem based on the improved genetic algorithm. *2014 10th International Conference on Natural Computation (ICNC)*[online]. IEEE, 2014, 584-588 [cit. 2015-12-06]. DOI: 10.1109/ICNC.2014.6975900. ISBN 978-1-4799-5151-2. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6975900.

[24] BOZIKOVIC, M., M. GOLUB a L. BUDIN. Solving n-Queen problem using global parallel genetic algorithm. *The IEEE Region 8 EUROCON 2003. Computer as a Tool* [online]. IEEE, 2003, 104-107 [cit. 2016-03-01]. DOI: 10.1109/EURCON.2003.1248159. ISBN 0-7803-7763-X. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1248159.