Case study z predmetu Evolučné algoritmy

Problém obchodného cestujúceho na ortogonálnej štvorcovej mriežke

Peter Sýkora <psykora@gmail.com>

Vyučujúci: Doc. RNDr. Jiří Pospíchal, DrSc.
Slovenská technická univerzita
Fakuta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava, Slovakia
21. júna 2006

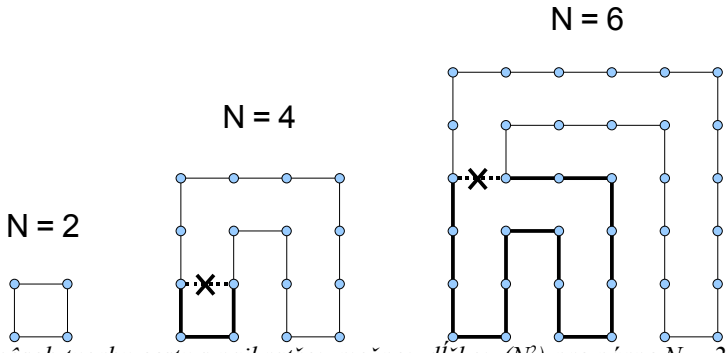
Abstrakt. Tento dokument analyzuje možné riešenie problému obchodného cestujúceho pomocou genetického algoritmu a zaoberá sa pororovnaním dvoch spôsobov kódovania cesty a to kódovanie pomocou celočíselnej permutácie a pomocou binárneho vektora. Pre jednoduchšiu verifikovateľnosť optimálnosti nájdenej cesty je použité rozmiestnenie bodov na ortogonálnej štvorcovej mriežke.

Úvod

Problém obchodného cestujúceho (TSP) je typickým kombinarickým problémom. Možno ho sformulovať nasledovne: Daný je počet miest, ktoré ma obchodný cestujúci navštíviť. Ďalej sú zadané ceny spojenia resp. presunu z jedného mesta do druhého. Úlohou obchodného cestujúceho je navštíviť každé z daných miest práve raz, vrátiť sa do mesta, z ktorého vyštartoval, a to tak, že výsledná cena cesty (permutácie postupne navštívených miest) bude minimálna možná. Pre účely tejto práce budeme používať grafy, v ktorých existuje priama cesta z každého uzla do každého a zároveň presun z mesta A do B je ohodnotený rovnakou cenou ako presun z mesta B do mesta A pre každú dvojicu uzlov (A,B) . Takýto graf sa nazýva symetrický úplný.

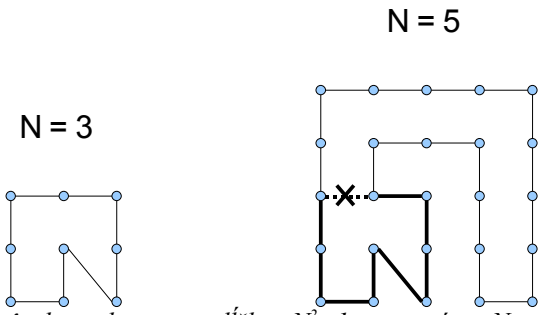
TSP nie je riešiteľný v polynomiálnom čase (až na niektoré špeciálne prípady akým je napríklad aj ortogonálna štvorcová mriežka) a preto sa pre veľký počet uzlov grafu používajú rôzne rýchle heuristické metódy, ktoré však nezaručujú nájdenie optimálneho riešenia. Jednou zo stochastických heuristických metód je použitie genetického algoritmu.

V prípade, že sú uzly rozmiestnené na ortogonálnej mriežke rozmeru NxN, pričom vzdialenosť medzi uzlami je definovaná ako súčet absolútnych hodnôt rozdielov jednotlivých súradníc(metrika typu Manhattan), je verifikácia nájdenia najkratšej cesty jednoduchou záležistosťou. Pre párne N menšie rovné ako 2 je najkratšia cesta rovná N^2. Obr. 1 ukazuje spôsob, ktorým sa dá takáto cesta vždy zostrojiť. To, že neexistuje kratšia cesta je dokázané tým, že cesta musí prechádzať cez každé mesto a najkratšia vzdialenosť medzi uzlami je 1.



Obr. 1: Spôsob tvorby cesty s najkratšou možnou dĺžkou (N^2) pre párne N = 2, 4, 6, ...

Pre N nepárne je potrebné okrem ukázania ako zostrojiť cestu z dĺžkou N^2+1 (Obr. 2) aj dokázať, že vytvorená cesta je optimálna a že neexistuje cesta s dĺžkou N^2. Keďže v prípade nepárneho N je aj N^2 nepárne a zároveň lema 1 tvrdí že cyklus skladajúci sa z hrán dĺžky 1 má vždy párnu dĺžku (čo sa nedá v prípade nepárneho počtu uzlov zostrojiť), z toho vyplýva že aspoň jedna hrana v cykle musí mať dĺžku väčšiu ako 1.

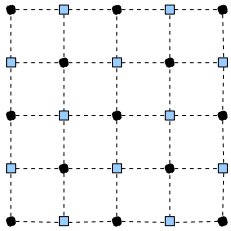


Obr. 2: Spôsob tvorby cesty s dĺžkou N^2+1 pre nepárne N = 3, 5, 7, ...

Lema 1: Na ortogonálnej štvorcovej mriežke, kde sú uzly mriežky spojené len so susednými (hrany dĺžky 1), sa každý cyklus skladá z párneho počtu uzlov (a takisto hrán).

Dôkaz: Vyplýva to z bipartitnosti grafu (uzly sú rozdelené na dve skupiny po uhlopriečkach, ako vidno na Obr. 3). medzi dvoma uzlami z jednej skupiny musí byť nepárny počet uzlov po oboch cestách v cykle (aj smere aj proti smeru hodinových ručičiek). Súčet nepárneho a nepárneho čísla dáva číslo párne a preto je v každom cykle párny počet uzlov.

Indukciou sa dá dokázať, že na ceste medzi uzlami z jednej skupiny je nepárny počet uzlov. Princíp začína s cestou medzi dvoma uzlami z jednej skupiny, kde medzi koncovými bodmi je uzol z druhej skupiny. Pre predĺženie cesty je potrebné pridať medzi koncové body do cesty vždy aspoň 2 uzly a to konkrétne jeden z jednej skupiny a druhý z druhej skupiny.



Obr. 3: Ortogonálny mriežkový graf so zvýraznenými skupinami nesusedných bodov

Metódy

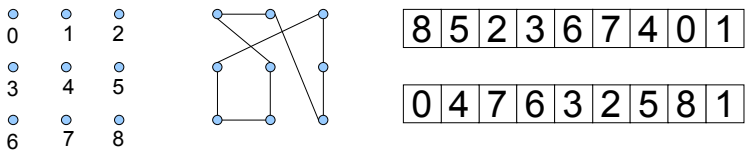
V tejto časti sú uvedené použité metódy pre riešenie problému obchodného cestujúce na ortogonálnej štvorcovej mriežke. Použité sú 2 možné reprezentácie cesty.

Kódovanie cesty

Cesta obchodného cestujúceho je ako už bolo spomenuté permutácia uzlov v poradí v akom budú navštívené Permutácia, ktorá vznikla cyklickým posuvom inej permutácie kóduje tú istú cestu, pretože nie je dôležité v ktorom meste obchodný cestujúci začne. Takisto permutácia, ktorá vznikla obrátením poradia prvkov kóduje tú istú cestu, pretože nie je dôležité či prejde obchodný cestujúci mestá v opačnom poradí. V prípade, že graf obsahuje *n* miest existuje pre každú cestu 2*n* rôznych permutácií.

Permutácia

Kódovanie cesty pomocou celočíselnej permutácie prvkov 0 , 1, ..., *n*-1 ($n=N^2$, teda ak *N* je počet uzlov na hrane mriežky, porom *n* je počet uzlov celého grafu).



Obr. 4: Ukážka cesty a jej možné zakódovania pomocou permutácie celých čísel pre N=3

Binárny vektor

Pri tejto metóde kódovania sa všetky operácie okrem vyhodnocovania fitness dejú nad binárnym vektorom. Cesta (dĺžky *n*) je zakódovaná pomocou binárneho vektora dĺžky *kn*, kde *k* je také celé číslo, ktoré zabezpečuje, aby binárny vektor dĺžky *k* bol schopný reprezentovať všetky celé čísla menšie ako *n*. Chromozóm sa teda skladá z *n* vektorov dĺžky

k. Binárny vektor $a=(a_1, a_2, \ldots, a_k) \in \{0,1\}^k$ dĺžky *k* sa transformuje na na celé čislo z intervalu <0, 2^k-1> nasledovnou funkciou:

$$integer(a)=\sum_{i=1}^{k} a_i 2^{k-1} = a_1 2^{k-1} + a_2 2^{k-2} + \ldots + a_{k-1} 2 + a_k \quad .$$

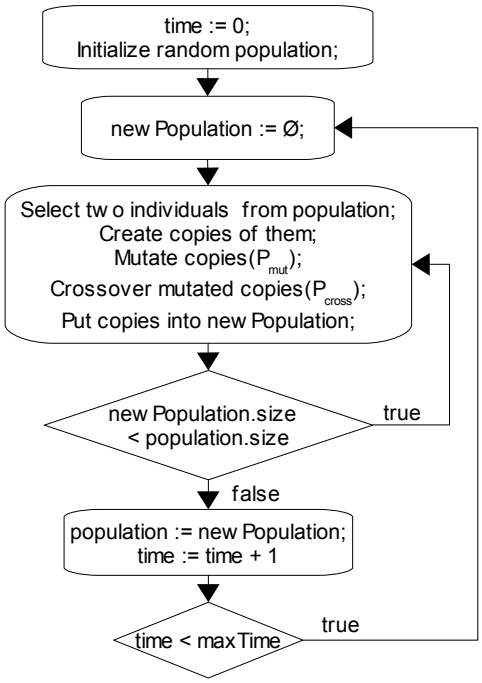
Chromozóm sa postupne pretransformuje na vektor celých čisel dĺžky *n* (môžu byť aj duplicity). Vektor je potom usporiadaný tak, že sa ku každému číslu poznačí pozícia v pôvodnom neusporiadanom vektore. Permutácia je potom vytvorená tak, že sa každé číslo v usporiadanom vektore nahradí číslom jeho pôvodnej pozície. Ohodnotenie fitness jedinca sa potom vykonáva ako pri celočíselnej permutácii.

Genetický algoritmus

Na začiatku sa inicializuje populácia vygenerovaním náhodných chromozómov. Vytvorenie celočíselnej permutácie funguje nasledovným spôsobom. Vygeneruje sa vektor náhodných reálnych čísel požadovanej dĺžky permutácie. Vektor následne zotriedime, pričom ku každému číslu si poznačíme jeho originálnu pozíciu v pôvodnom vektore. V takto zotriedenom vektore nahradíme potom tieto čísla ich pôvodnými pozíciami.

Vytvorenie binárneho vektora je jednoduchšie a v podstate ide len o naplnenie vektora požadovanej dĺžky náhodnými bitmi. Tento vektor sa transformuje na celočíselnú permutáciu len pri vyhodnocovaní fitness.

Po inicializácii populácie začína evolučná slučka, ktorá končí prejdením určeného počtu generácií daného parametrom *maxTime*. V každej generácii sa najprv vytvorí nová prázdna populácia. Z predchádzajúcej populácie sa vyberajú postupne dvaja jedinci (inicializovaná populácia musí mať párny počet jedincov, aby sa zabezpečila konštatná veľkosť populácie po celý čas trvania algoritmu) zvolenou výberovou metódou, ktorí sú následne skopírovaní do novej populácie a definovanými pravdepodobnosťami zmenení operáciami kríženia a mutácie. Tento proces sa opakuje dovtedy pokiaľ je nová populácia menšia ako stará. Potom sa stará populácia zahodí a evolučná slučka pokračuje dalšou generáciou.



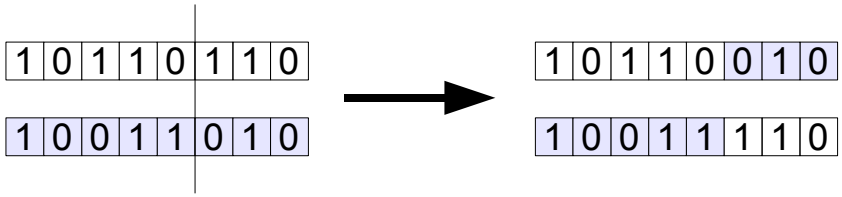
Obr. 5: Postup genetického algoritmu

Po skončení evolučnej slučky vráti algoritmus najlepšieho nájdeného jedinca. Ak je potrebné je možné počas evolúcie zbierať štatistické údaje o populácii (napr. priemerné fitness v jednotlivých iteráciách).

Operácie genetického algoritmu

Kríženie binárneho vektora

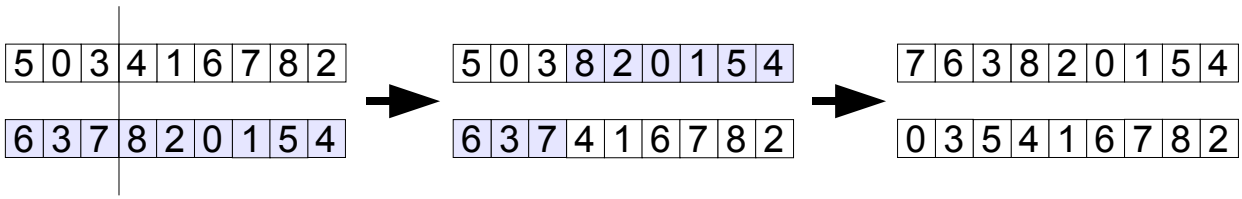
Na operáciu kríženia treba mať vybratých dvoch jedincov. Kríženie binárnych vektorov je riešené náhodne vybratým bodom prekríženia, za ktorým sú podčasti vektorov medzi sebou vymenené. Príklad takéhoto kríženia je uvedený na Obr. 6.



Obr. 6: Ukážka jednobodového kríženia binárneho vektora

Kríženie permutácie

Operácia kríženia permutácie je založená na jednobodovom krížení vektorov pričom je potrebné zabezpečiť opravu permutácií, ak sa výmenou postupností stali permutácie nepermutáciami z dôvodu duplicity niektorých prvkov.



Obr. 7: Ukážka kríženia permutácie

Oprava sa vykonáva nasledovne. Máme dva vektory P a Q dĺžky n. Nech a je bod kríženia (v príklade na obrázku je a=3). Po krížení nám z vektorov

$P = (p_0, \ldots, p_{a-1}, p_a, \ldots, p_n)$

$Q = (q_0, \ldots, q_{a-1}, q_a, \ldots, q_n)$

vznikne

$\underline{P} = (p_0, \ldots, p_{a-1}, q_a, \ldots, q_n)$

$\underline{Q} = (q_0, \ldots, q_{a-1}, p_a, \ldots, p_n)$

Môže teraz nastať situácia, že niektoré celé číslo sa vo vektore $\underline{P}$ alebo $\underline{Q}$ nachádzajú dvakrát. Definujme zobrazenie f_{PQ} tak, že zložky permutácie $\underline{P}$, ktoré sa vymenili, sú zobrazené na komponenty $\underline{Q}$, ktoré sa tiež zúčastnili výmeny

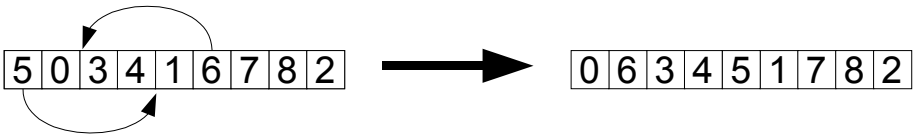
$f_{PQ} : p_i \rightarrow q_i$ (pre $i > a$)

Druhé zobrazenie f_{QP} je určené ako inverzné zobrazenie k f_{PQ}, $f_{QP} = f_{PQ}^{-1}$. Pomocou týchto dvoch zobrazení budeme opravovať „pôvodné“ časti v $\underline{P}$ a $\underline{Q}$, teda tie, ktoré ležia pred bodom kríženia. Ak zložka p_i (pre $0 \leq i < a$) nepatrí do definičného oboru zobrazenia f_{QP} (to tiež znamená, že p_i sa vyskytuje v $\underline{P}$ práve raz), potom p_i zostáva nezmenené. V opačnom prípade, ak sa p_i nachádza v definičnom obore f_{QP}, potom sa p_i zamení na prvú hodnotu iteračnej schémy $x_{k+1} = f_{QP}(x_k)$, ktorá je mimo definičnej oblasti funkcie f_{QP} (iteračné riešenie sa inicializuje $x_0 = p_i$). Podobný postup sa aplikuje aj pre opravu pôvodného segmentu v $\underline{Q}$, avšak teraz pomocou zobrazenia f_{PQ}.

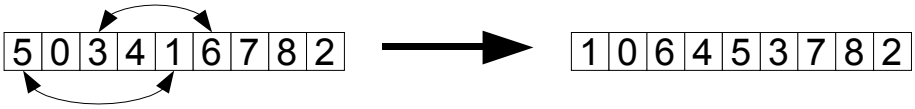
Príkladom opravy je aj Obr. 7, na ktorom boli podľa iteračnej schémy zmenené: v prvom vektore **5**→8→4→2→1→**7**, **0**→**6** a v druhom vektore **6**→**0**, **7**→1→2→4→8→**5**.

Mutácia

Pod mutáciou v genetickom algoritme sa rozumie náhodná zmena chromozómu.



Obr. 8: Ukážka mutácie permutácie presunom prvkov



Obr. 9: Ukážka mutácie permutácie výmenou prvkov

Mutácia permutácie pomocou presunu prvkov (ukážka na Obr. 8) prebieha tak, že každý prvok sa pravdepodobnosťou P_{mut} vyberie z vektora (ostatné prvky napravo od neho sa posunú o jeden prvok doľava) a umiestni sa na náhodnú pozíciu (ostatné prvky od danej pozície až po koniec vektora sa posunú o jeden prvok doprava).

Na Obr. 9 je uvedená ukážka mutácie chromozómu obsahujúceho permutáciu. Mutácia samotného génu pomocou výmeny s náhodným prvkom prebieha tak, že sa postupne každý gén s pravdepodobnosťou P_{mut} vymení s iným náhodne vybratým génom.

Obr. 10: Ukážka mutácie binárneho vektora

Pri mutácii binárneho vektora (Obr. 10) sa každý gén zmení na opačnú hodnotu (t.j. 0 na 1 resp. 1 na 0) s pravdepodobnosťou P_{mut}.

Výberové metódy

Výberové metódy zabezpečujú použiteľnosť genetického algoritmu tým, že vyberajú s vyššou pravdepodobnosťou

lepších jedincov ako rodičov pre genetické operácie, alebo pre prežitie do ďalšej generácie. Pre účely tejto práce boli použité nasledujujúce 3 metódy.

Turnajový výber

Prebieha tak, že sa z populácie náhodne vyberie určitý počet jedincov daný parametrom *tournamentSize* a porovná sa fitness jedincov. Celkovým výsledkom výberu je jedinec s najvyšším fitness. Celý proces sa opakuje dovtedy pokým nie je vybratý dostatočný počet jedincov pre genetickú operáciu.

Upravený turnajový výber

Tento výber používa turajový výber s veľkosťou turnaja 2, pričom víťaz turnaja je vybratý s pravdepodobnosťou definovanou parametrom P_{winner}, ktorý nadobúda hodnoty z intervalu <0,5;1>. Extrémy tohto parametra sú hodnota 0,5, kedy ide o náhodný výber a hodnota 1, pri ktorej ide o klasický turnajový výber s veľkosťou turnaja 2.

Použiteľnosť tohto výberu je hlavne na zníženie selekčného tlaku a teda aby sa zvýšila pravdepodobnosť výberu slabších riešení a spomalila tak konvergencia k lokálnemu optimu.

Ruletový výber

Jedinci sú mapovaní na spojité intervaly číselnej osi tak, že dĺžka intervalu každého jedinca je rovná veľkosti jeho fitness. Prvý interval začína v nule a každý ďalší interval pokračuje, tam kde predchádzajúci interval skončil. Je vygenerované náhodné číslo s rovnomerným rozdelením z intervalu (0, suma fitness všetkých jedincov) a je vybratý ten jedinec, ktorého interval obsahuje náhodné číslo, ktoré bolo vybraté. Tento proces sa opakuje až kým sa vyberie požadovaný počet jedincov. Táto technika je analógiou rulety, kde veľkosť každého výseku kruhu je ú veľkosť fitness jedinca.

Ohodnotenie fitness

Pre použitie výberových metód je potrebné ohodnotiť vhodnosť jedincov. Jedinec ktorý dosiahne cestu s dĺžkou bližšou optimálnej dosiahne vyššiu hodnotu fitness. Najkratšia vziadlenosť medzi uzlami mriežky je 1 medzi uzlami umiestnenými vedľa seba a použitá metrika je typu Manhattan a teda vzdialenosť medzi každými dvoma bodmi je celočíselná. Optimálna dĺžka je ako už bolo spomenuté N^2 pre párne N a $N^2 + 1$ pre nepárne N.

Hodnota fitness jedinca sa vypočíta pomocou nasledujúceho vzorca:

$$fitness = \frac{1}{1 + length - optimalLength},$$

kde *length* je dĺžka cesty daného jedinca a *optimalLength* je dĺžka najkratšej cesty. Najlepšie fitness je teda 1, v prípade že dĺžka cesty je rovná optimálnej.

Implementácia

Použitý program pre získanie výsledkov sme implementovali v jazyku Java. Bol riešený aj export jedincov v grafickom vektorovom formáte, pričom bola možnosť ich vykresľovať nového nájdeného najlepšieho jedinca v prípade potreby. Niektoré z týchto obrázkov boli použité aj v tomto dokumente v nasledujúcich častiach. Použili sme knižnicu na generovanie SVG (Scalable Vector Graphics) z projektu xmlgraphics.apache.org.

Čo sa týka výkonu použitého programu, v prípade použitia 32-bitového počítača typu Intel s taktom 3,0 GHz trval jeden beh genetického algoritmu pre mriežku 5x5, 2000 generácii a 1000 jedincov v populácii približne 1-2 minúty v závislosti od použitého kódovania a presných hodnôt parametrov. Triedenie pri binárnom kódovaní je implementované pomocou algoritmu so zložitosťou $O(n.\log_2(n))$.

Výsledky

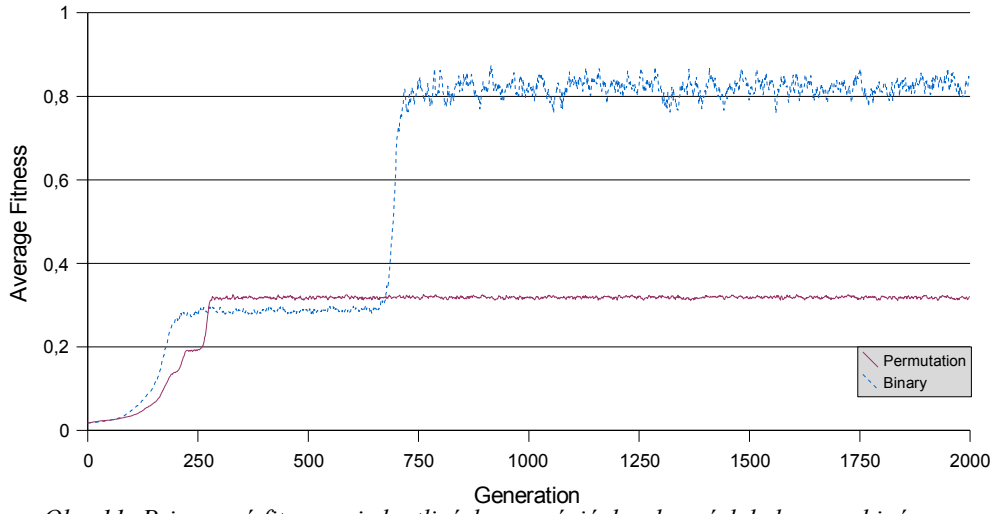
V tejto časti sú uvedené dosiahnuté výsledky. Zamerali sme sa hlavne na porovnanie dvoch spomínaných metód kódovania z hľadiska kvality dosiahnutých výsledkov a z hľadiska výpočtovej náročnosti. Pre mutáciu celočíselnej permutácie bola použitá metóda presunu prvkov, pretože dosahovala lepšie výsledky ako pri vymieňaní dvojíc prvkov.

Príklad 1

Prvými pokusmi sme sa snažili nájsť vhodné hodnoty parametrov genetického algoritmu. Cieľom bolo nájsť najlepšie riešenie na mriežke, pre N = 4, 5, 6, 7 a viac. Keďže sa len veľmi zriedka (rádovo raz zo 100 pokusov) podarilo nájsť optimálne riešenie pre N = 6 obmedzili sme teda hľadanie len pre N rovné 5 a menej. Pre N rovné 6 je na mriežke 6x6 = 36 bodov. To znamená, že existuje 36! permutácií čo je približne $3{,}7 \times 10^{41}$. Jedna cesta sa dá zakódovať 36x2 rôznymi

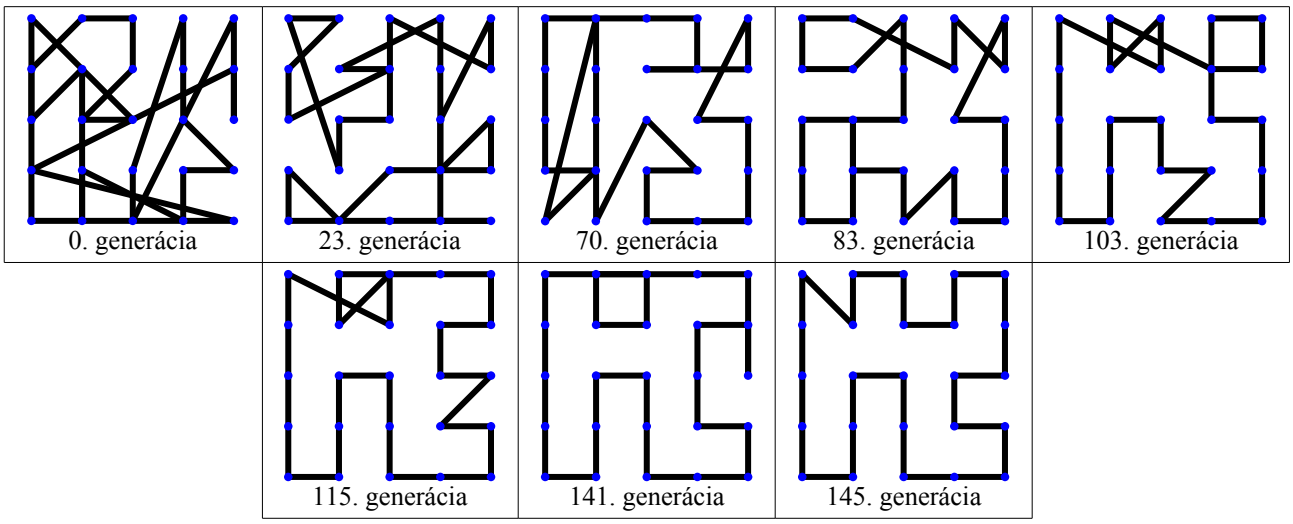
permutáciami (cyklický posuv a zároveň každú je možné kódovať v opačnom poradí).

Na nasledujúcom obrázku (Obr. 11) sú znázornené výsledky dvoch behov na mriežke 5x5, pričom pre jeden beh bolo použité binárne kódovanie (P_{cross} = 0.7, P_{mut} = 0,0008) a pre druhý kódovanie pomocou celočíselnej permutácie (P_{cross} = 0.4, P_{mut} = 0,0005). Spoločnými parametrami boli: populácia 1000 jedincov, počet generácií 2000. Pre výber vhodných jedincov bol použitý upravený turnajový výber s pravdepodobnosťou výberu víťaza turnaja s pravdepodobnosťou 0,65. Ruletový a turnajový výber konvergoval rýchlejšie no v priemere poskytoval slabšie výsledky.



Obr. 11: Priemerná fitness v jednotlivých generáciách vybraných behov pre binárnu reprezentáciu a pre reprezentáciu celočíselnou permutáciou

Z grafu je možné vidieť že binárne kódovanie permutácie dosiahlo lepšie výsledky (dokonca jediné z týchto dvoch behov našlo cestu s optimálnou dĺžkou), no porovnanie dvoch vybratých behov na porovnanie oboch metód nestačí. Na Obr. 12 je možné vidieť časť najlepších jedincov v jednotlivých generáciách pri použití binárneho kódovania. Najlepší jedinec bol nájdený už v 145. generácii.



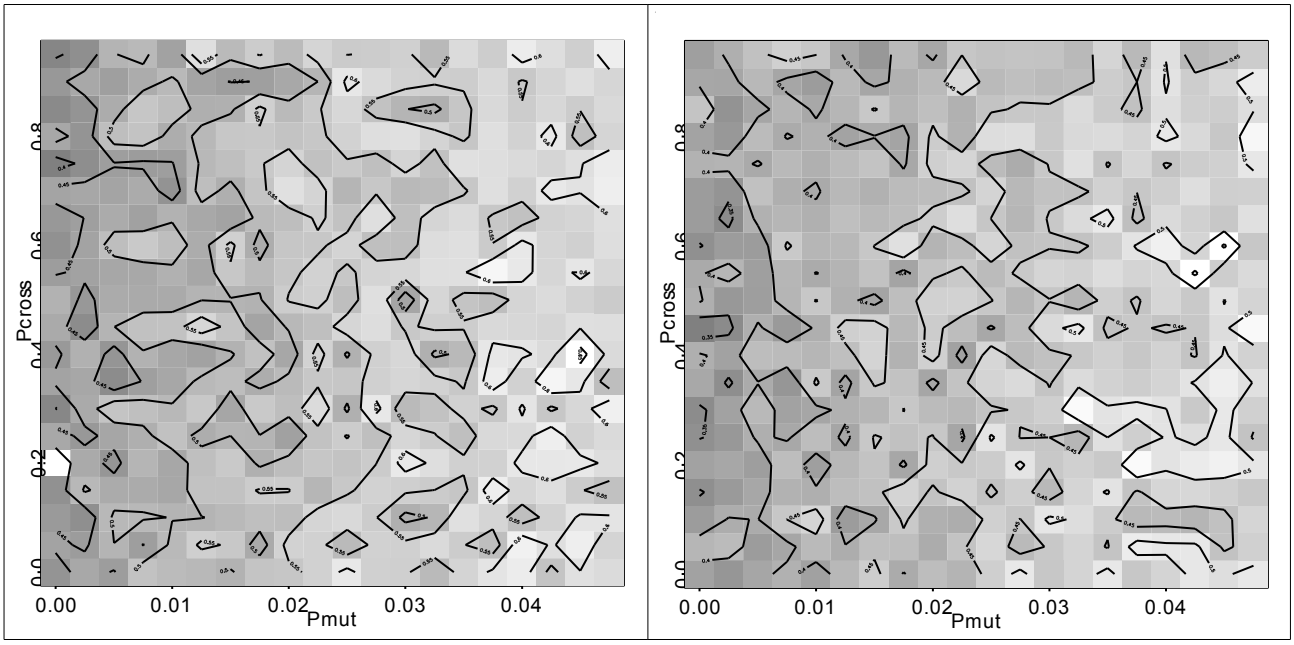
Obr. 12: Najlepší nájdený jedinec v jednotlivých generáciách (binárne kódovanie)

Hľadanie vhodných pravdepodobností kríženia a mutácie

Pokúsili sme sa zistiť vhodné hodnoty parametrov pre kríženie a mutáciu, pričom ostatné parametre boli konštanté a nastavené na nasledovné hodnoty:

- N (počet bodov na strane mriežky) = 3,
- ruletový výber,
- počet generácií = 200,
- veľkosť populácie = 20.

Testované hodnoty pravdepodobnosti mutácie boli od 0,0 do 0,0475 s krokom 0,0025 a hodnoty pravdepodobnosti kríženia od 0,0 do 0,95 s krokom 0,5. Pre každú kombináciu týchto hodnôt bolo vykonaných 100 behov pričom ako pričom ako kvalita parametrov sa hodnotil priemer maximálnej dosiahnutej fitness z týchto behoch. Pre 20 rôznych hodnôt P_{mut} a 20 rôznych hodnôt P_{cross} bolo potrebné teda vykonať 20*20*100 = 40 000 behov genetického algoritmu. Pre binárnu reprezentáciu trvala celá simulácia približne 1 hodinu a pre permutáciu približne 40 minút. Výsledky sú uvedené na Obr. 13. Svetlejšie body znamenajú vyššiu dosiahnutú maximálnu fitness (spriemerovanú zo 100 behov) a teda kvalitnejšie hodnoty parametrov.



Obr. 13: Priemerná maximálna získaná hodnota fitness zo 100 pokusov pre každú hodnoty P_{cross} a P_{mut}. V grafe naľavo je použitá binárna reprezentácia a napravo celočíselná permutácia.

Vhodné by bolo takto hľadať parametre aj pre iné veľkosti mriežky a hodnoty ostatných parametrov (poprípade použiť metaevolúciu – evolučný algoritmus pre optimalizáciu použitých parametrov). Pre N=5, veľkosť populácie 1000, počet generácií 2000, trval však jeden beh 1-2 minúty v závislosti od použitého kódovania a výberovej metódy a preto vykonanie 40 000 behov by bolo časovo veľmi náročné. Dalej v texte budú preto používané hodnoty parametrov, ktoré boli získané ručným skúšaním.

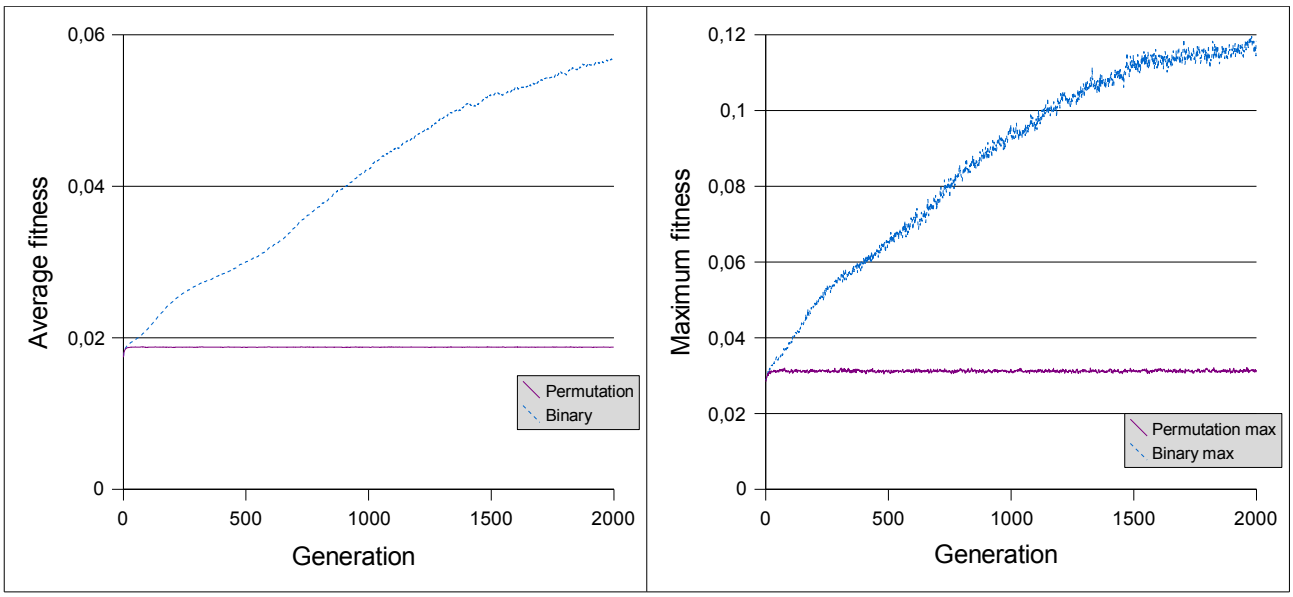
Porovnanie binárneho kódovania a kódovania celočíselnou permutáciou

Toto porovnanie bolo vykonané na mriežke 5x5 nasledujúcim postupom. Pre každé kódovanie a hodnoty parametrov sa spustilo 100 nezávislých behov pričom pre každú generáciu a každý beh sme si uchovali priemerné fitness celej populácie. Nakoniec sme pre každú generáciu vypočítali priemer zo všetkých behov a vyniesli do grafov na Obr. 14, Obr. 15 a Obr. 16. Hodnoty parametrov spoločné pre všetky grafy:

- *N* (počet uzlov na strane mriežky) = 5,
- upravený turnajový výber s pravdepodobnosťou výberu víťazného jedinca 0,65,
- počet generácií = 2000,
- veľkosť populácie = 1000.

Upravený turnajový výber bol zvolený z dôvodu dosahovania lepších výsledkov. Turnajový výber a ruletový výber konvergovali rýchlejšie k lokálnemu optimu a populácia sa veľmi rýchlo stala takmer homogénna. Bolo preto vhodné mierne znížiť selekčný tlak.

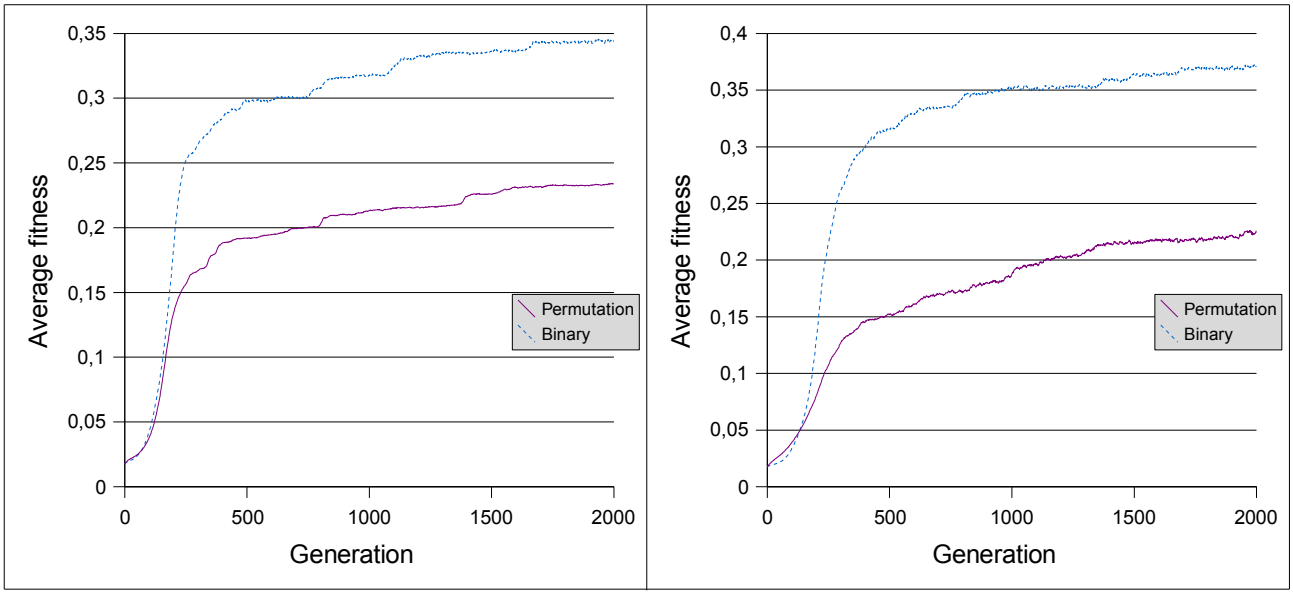
V prvých porovnaniach sme sa pokúšali spustiť simuláciu s parametrom pravdepodobnosť mutácie takým aby zmutoval v každej generácii jedinec približne raz. Pre celočíselnú permutáciu veľkosti 5x5 bola teda použitá pravdepodobnosť P_{mut}=1/25. Pre binárne kódovanie (pre kódovanie čísel 0-25 treba 5 bitov a teda celý chromozóm má 5x25 bitov) bola pravdepodobnosť P_{mut} nastavená na hodnotu 1/125. Pravedepodobnosť kríženia P_{cross} bola u oboch kódovaní rovnaká a to 0,5. Spriemerované výsledky nezávislých 100 behov sú vynesené do grafu na Obr. 14.



Obr. 14: Priebeh priemernej a maximálnej fitness (spriemerované zo 100 pokusov) pre obe použité kódovania. Pre binárne kódovanie bola hodnota pravdepodobnosti mutácie P_{mut} 0,008 a pre kódovanie celočíselnou permutáciou 0,04. Pravdepodobnosť kríženia P_{cross} bola nastavená pre obe kódovania na 0,5

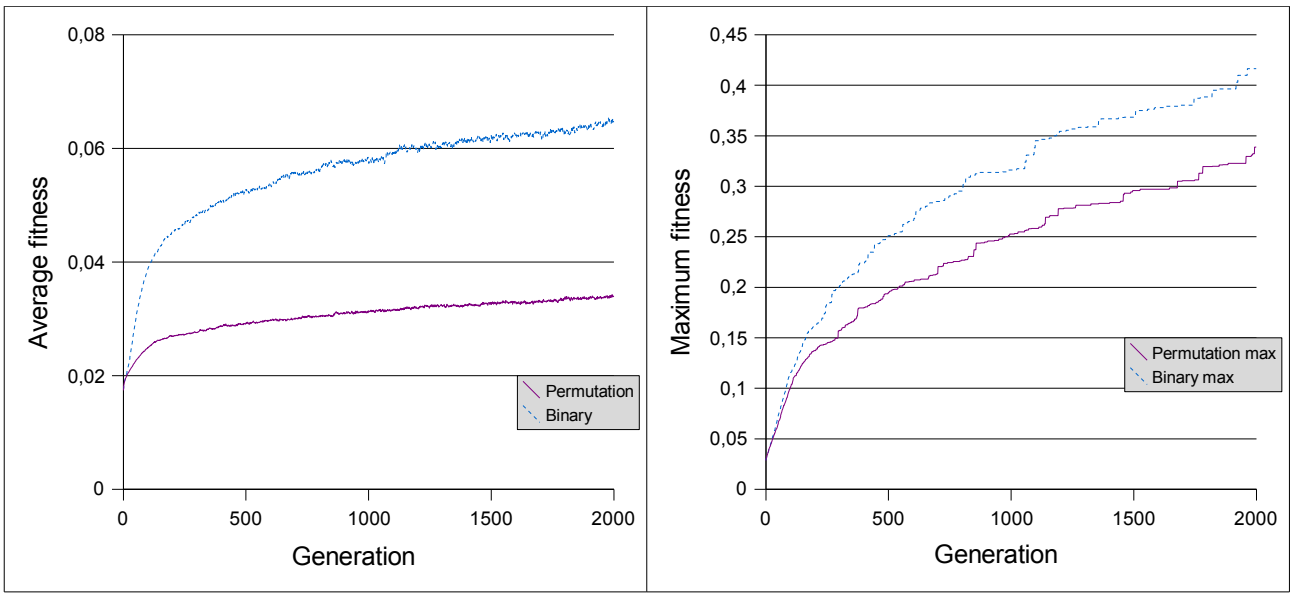
Potom sme sa snažili rovnakú simuláciu spustiť pre hodnoty parametrov, ktoré nám dávali lepšie výsledky. Priemer priemernej fitness v 100 nezávislých behoch pre každú generáciu je uvedený na Obr. 15. Pre graf naľavo boli pre obe použité kódovania použité P_{mut} = 0,0005 a P_{cross} = 0,5. Graf napravo zobrazuje zopakovaný test pre najlepšie nájdené hodnoty týchto parametrov pre obe kódovania a to P_{mut} = 0,0008 a P_{cross} = 0,7 pre binárne kódovanie resp. P_{mut} = 0,0005 a P_{cross} = 0,2 pre celočíselné kódovanie permutácie.

Z grafu na Obr. 15 možno vidieť, že binárne kódovanie dosahuje lepšie výsledky, ale treba podotknúť že jeden beh pri použití binárneho kódovania trvá takmer 2 minúty. Pri použití celočíselnej permutácie trvá jeden beh zhruba polovicu času.



Obr. 15: Priemerné fitness jedincov v jednotlivých generáciách (pre každú generáciu je uvedený priemer zo 100 pokusov). Pre graf naľavo boli použité P_{mut} = 0.0005 a P_{cross} = 0.5 pre oba spôsoby kódovania a, pre graf napravo boli použité najlepšie nájdené hodnoty týchto parametrov.

Z Obr. 14 a Obr. 15 vidno, že pre vyššie hodnoty pravdepodobností mutácie je efektívnosť (hlavne pri mutácii celočiselnou permutáciou) genetického algoritmu menšia, pretože nájdený najlepší jedinci sa rýchlo zmenia. Preto sme sa v nasledujúcom pokuse pokúsili znížiť pravdepodobnosť straty dobrých riešení tak, že časť jedincov z pôvodnej populácie sa prekopíruje do novej populácie nezmenená, a vykonať simuláciu (ktorej výsledky sú na Obr. 14) znova. Riešili sme to elitizmom a to konkrétne tak, že do novej populácie vždy automaticky prešlo 20 najlepších jedincov zo starej. Výsledky je možné vidieť na Obr. 16.



Obr. 16: Priebeh priemernej a maximálnej fitness pri testoch ako na Obr. 14, ale bol pridaný elitizmus. 20 najlepších riešení je automaticky prekopírovaných do novej populácie bezo zmeny.

Zhodnotenie

V rámci tejto práce boli analyzované metódy genetického algoritmu pre riešenie problému obchodného cestujúceho. Všetky tieto metódy boli implemetované a otestované na vhodnosť.

Genetický algoritmus bol testovaný na ortogonálnej štvorcovej mriežke s použitím dvoch spôsobov kódovania cesty a to binárne kódovanie a kódovanie pomocou permutácie (s adekvátnymi operáciami mutácie a kríženia). Testovalo sa hlavne na mriežke 5x5 a zistili sme, že binárne kódovanie dosahuje pre tento typ problému lepšie výsledky ale z dôvodu potreby implementácie triedenia (pri konverzii na permutáciu) je o niečo viac náročnejšie na výpočtový výkon. V prípade použita takých hodnôt parametrov aby v priemere každý vektor raz zmutoval v jednej generácií sa tieto ukázali byť vysoké, rýchlo sa strácali najlepšie riešenia a preto bol do programu doplnený elitizmus, pri ktorom sa časť najlepších jedincov kopírovala do populácie nezmenená.

Počas testovania sa nám nepodarilo nájsť ani raz optimálnu cestu na mriežke 7x7 a väčšej. Pre 36 bodovú mriežku (6x6) mal genetický algoritmus taktiež problémy a optimálne riešenie sa podarilo nájsť len raz zo 100 pokusov. Aby bolo možné z toho vyvodiť závery týkajúce sa použiteľnosti genetického algoritmu na riešenie problémov tohto typu by bolo potrebné vykonať viac pokusov a porovnať s inými postupmi (napr. iné evolučné algoritmy).

Použitá literatúra

1. Kvasnička, V., Pospíchal, J., Tiňo, P.: Evolučné algoritmy.Vydavateľstvo STU, Bratislava, 2000.