Evolučné algoritmy

Úloha č.4 (2000/2001):

Porovnajte účinnosť zakódovania riešenia genetického algoritmu pre TSP (úlohu obchodného cestujúceho), kde mestá sú na pravouhlej mriežke a kódovanie je:

a) klasická permutácia
b) pri stálom štartovacom bode pri strede mriežky budú jednotlivé vstupy chromozómu čísla od 1 do 3, ktoré značia poradie ďalšej cesty z daného bodu pri počítaní zľava od vstupnej cesty (teda 1=vľavo, 2=rovno,3=vpravo). Pokiaľ sa algoritmus dostane „nožičkou" Téčka na hranu, 2 znamená napravo a 3 návrat späť. Chromozóm bude mať dĺžku odpovedajúcu ideálnemu riešeniu, funkčná hodnota riešenia bude zostavéná z členu určujúceho počet navštívených miest spolu s inverziou vzdialenosti posledného mesta od východzieho.

Vypracoval:
Martin Kanich
MFF UK, Bratislava, 5.ročník
kanci@pobox.sk
http://www.kanci.miesto.sk/ea/

Riešenie:

Táto úloha patrí medzi kombinatorické optimalizačné úlohy [1,str.92]. Z toho vychádzajúc na začiatku simulácie vytvoríme požadovanú populáciu chromozómov, ktoré reprezentujú jedincov v populácii, podľa funkcií zo zadania. Z [1] vyplýva, že klasická permutácia bude reprezentovať postupnosť, v ktorej mestá navštívime. Graf si zadefinujeme ako mriežku, kde mestá sú spojené len vodorovne a zvysle a ich hodnota(cena) je jedna. Takto sa dá úloha a) porovnať s b). Avšak teraz už vidíme, že a) je menej konkrétna na graf, avšak b) je špecializovaná na mriežku. Predpokladám teda z toho, že ak obmedzený pohyb nezhorší možnosť dosiahnutia optimálneho výsledku, rýchlosť najdenia výsledku by mala byť vyššia, vzhľadom na to, že nebudeme uvažovať „šikmé" cestičky (cesty, keď musíme prejsť cez iné mestá/mesto, aby sme sa dostali do cieľa). Na druhej strane však z b) vyplýva, že sa nám môže stať, že nenaštívime všetky mestá, čo na začiatku samozrejme očakávame, rovnako ako „vyhynutie" „ignorantov" (cesty, čo nerátajú s veľa mestami spĺňajú najmenej zadanú úlohu a teda zrejme vyhynú a v ďalšej generácii sa už nebudú vyskytovať).

Popíšme teda situáciu pre a) konkrétnejšie. Majme populáciu chromozómov, kde každý reprezentuje cestu. Na to, aby išlo o genetický algoritmus musíme definovať operáciu mutácie a operáciu kríženia medzi jednotlivcami. Presne ako v [1] platí, že najprv zvolíme miesto *c*, kde zkrížime chromozómy (permutácie) *P*,*Q*:

$$P' = (P_1,\ldots,P_{c-1},P_c,\ldots,P_{size})$$
$$Q' = (Q_1,\ldots,Q_{c-1},Q_c,\ldots,Q_{size})$$

Samozrejme, že po skrížení dvoch permutácii nastáva vo väčšine prípadov rozpor z definíciou permutácie a teda P' a Q' viac nereprezentujú chromozóm. Nasleduje teda korigujúca časť algoritmu, ktorá hľadá výskyt P_i i<c v Q_j j>=c a naopak. Každú takú dvojicu(P_a,Q_b), čo nájde, vymení. Je zrejmé, že po tejto oprave už ide o permutáciu. Operáciu mutácie zadefinujeme tiež klasicky, pre každé miesto v chromosome hodená kocka rozhodne, či zmutuje, alebo nie a ak áno, náhodne sa vyberie ďalšie miesto, s ktorým sa vymení. Môže síce nastať situácia, že pôjde o zmutovaný chromozóm, ale bude rovnaký ako predtým, to však môžme považovať za nenastanie mutácie.

```
for i:=0 to MAX_CHROM-1 do
  if Random <al_pmut then begin
    rnd := random(MAX_CHROM);
    tmp := chrom[i];chrom[i] := chrom[rnd];
    chrom[rnd] := tmp;
end;
```

Vieme vytvoriť populáciu – vygenerujeme náhodné permutácie danej dĺžky. Vieme krížiť dva chromozómy, či ich mutovať, treba ešte zadefinovať silu podľa ich významu. Fitness daného chromozómu, určujúca jeho silu/ význam, definujeme ako súčet vertikálnych a horizontálnych vzdialeností (manhattanovskú vzdialenosť) medzi dvojicami po sebe idúcich miest a tiež vzdialenost posledného mesta do prvého. Štartovacie mesto tu nie je vhodné použiť, lebo ide, na rozdiel od úlohy b), vždy o potenciálne iné mesto. Hodnoty fitness pred „prejdením“ do ďalšej generácie znormalizujem:

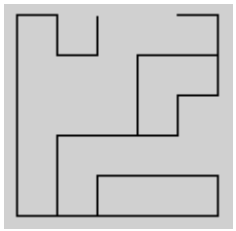
$$F'=1/(F-minF+1).$$

Nato vytvorím pole fitnesses, ktoré bude obsahovať jednotlivé Fitness chromozómov. Toto pole upravím tak, že ku každému členu pripočítam všetky predošlé, takže bude vytvárať neklesajúcu, dokonca možno povedať rastúcu postupnosť, kedže žiadna z fitness chromozómov nie je nulová po normalizácii.

V každom kroku pomocou rulety vyberiem dva chromozómy, tie zkrížim a zmutujem. Znovu vyrátam fitness, vysvietim najlepší chromozóm a doteraz zo všetkých generácii najlepšieho si označím tiež. Ako je realizovaná ruleta? Práve na to slúži úprava poľa fitnesses. Zvolím si náhodne nejakú hodnotu fitness a hľadam v tomto poli jej najbližšiu vyššiu hodnotu:

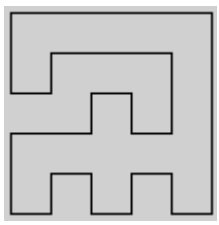
```
BinarySearch(fitnesses, Random(fitnesses[population_size-1]));
```

Ako je to v prípade b)? Chromozóm náhodne inicializujeme tak, že postupne generujeme čísla od 1 do 3. Kríženie urobíme úplne klasicky, ako keby binárne, len každý „bit“ má tri možné stavy. Mutácia tiež nie je náročná, ak sa rozhodneme „bit“ mutovať vyberiem zaň spomedzi čísel 1 až 3 náhradu, zase môže nastať situácia, že nepôjde o mutáciu, ale tiež to nespôsobuje nič viac, len nevyskytnutie sa mutácie a rovnako sa môže stať taká situácia pri krížení dvoch rovnakých chromozómov. Fitness som definoval dvojako. Raz takzvaná ClockWise fitness, vždy keď narazí Téčkom na hranu, sa zatočí doprava a potom ak nemôže ani vtedy sa pohnúť tak znova. Pri nakreslení si všetkých možností je zrejme, že to spĺňa špecifikáciu, až na symbol 1. Ten v normálnej verzii funkcii Fitness funguje korektne, akurát, že výsledky naznačujú, že „ClockWise“ je o čosi „múdrejšia“. Hodnota Fitness však nie je interpretácia pohybu, avšak až pokrytie mriežky bodmi, ktorými cesta vedia a prevrátená hodnota z dĺžky cesty z posledého miesta do prvého. Na rozdiel od a) tuto najlepšia Fitness bude najväčšia.



Takto môže vyzerať na začiatku mriežka [6x6;Fitness:32.2]

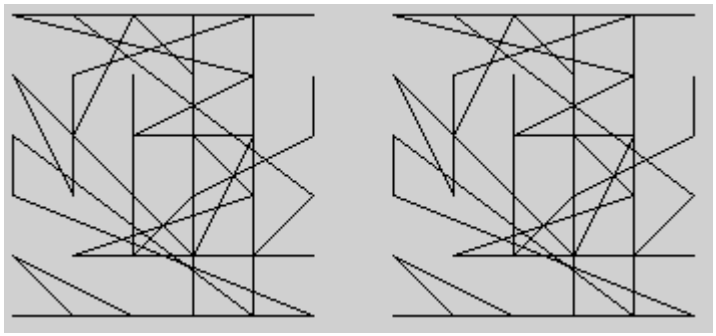
Po 349 generáciach 100 člennej komunity chromozómov sa objavy suboptimálne riešenie 36,3333. Optimálne má hodnotu 37. Samozrejme výsledok záleží od rôznych parametrov a funkcie random.Alebo 473 generácii. Nie vždy sa pri hodnotách pCross=0,7 a pMut=0,07 objaví sub-optimálne riešenie, niekedy sa objaví dokonca optimálne,



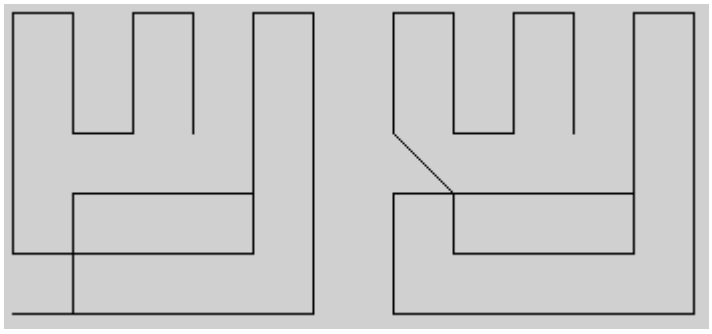
Optimálne riešenie pre 6x6

niekedy riešenie o niečo slabšie ako sub-optimálne. Moja skúsenosť s nastavovaním parametrov je taká, že čím väčšia veľkosť chromozómu, ktorá závisí priamo od veľkosti šírky

a výšky mriežky [môžte si ju počas behu programu meniť podľa potrieb], tím nižší koeficient pre pMut je vhodnejší. Ale je dosť problematické vybrať vhodný, nakoľko v neskorších generáciách bude to hlavne tento parameter, čo zabráni opusteniu doteraz najlepšieho riešenia, či naopak, populácia sa odoberie iným smerom. Ukazuje sa, že Murphyho zákon funguje a vždy sa väčší koeficient správa tak, ako by ste si nepriali, keď zvýšite, aby nezastalo riešenie priskoro: napr. pri mriežke 8x8 je koeficient 0,07 priveľký, použijeme teda 0,06. Teraz zas výpočet nedosahuje sub-optimálne riešenie v prvých generáciach, ale v neskorších zas začne oscilovať. Na výber máme navyše aj fitness funkciu korešpondujúcu so zadaním. Avšak výsledky s touto metódou sú nižšie. Je príjemné zistenie, že sa vyplatí čítať zadanie dvakrát a s odstupom času. Prináša to viaceré metódy na riešenie daného problému. Ďalej však budem uvažovať túto metódu pri porovnávaní so zadaním a). Použijeme na to mriežku 6x6 a 10x10. Najprv si však ukážeme nejaké náhodne vybraté správanie algoritmu a) na mriežke 6x6 samostatne, ešte nebudeme porovnávať.



Začiatok je „chaotický“, „nenarodí“ sa však veľa generácií a už sa to zmení k lepšiemu



Najlepšie (do generácie 4458) možnosti. Vľavo v tejto generácii, vpravo doteraz. Oba obrázky majú rovnakú fitness, avšak pravý sa objavil skôr ako ľavý. Vždy sa zobrazuje len jeden z najlepších.

Z tabuľky výsledkov, treba upozorniť, že ide len o prehľad, zo získaných údajov sa nedá urobiť záver o tejto metóde, možno vyrátať priemernú hodnotu, ktorú vie táto metóda získať z mriežky 6x6:

	Populácia	pCross	pMut	Generácia maximum	Počet iterácii	chyba
1	100	0,7	0,05	2632	38	10000	5,56%
2	100	0,7	0,05	1600	44	10000	22,22%
3	100	0,7	0,05	3834	40	10000	11,11%
4	100	0,7	0,05	9467	42	10000	16,67%
5	1000	0,7	0,05	1917	40	2000	11,11%
6	1000	0,7	0,05	830	40	2000	11,11%
7	1000	0,7	0,05	713	42	2000	16,67%
8	500	0,5	0,07	1788	38	2000	5,56%
9	500	0,5	0,09	831	40	2000	11,11%
10	500	0,5	0,075	1615	42	2000	16,67%
11	500	0,5	0,075	1079	40	2000	11,11%
							12,63%

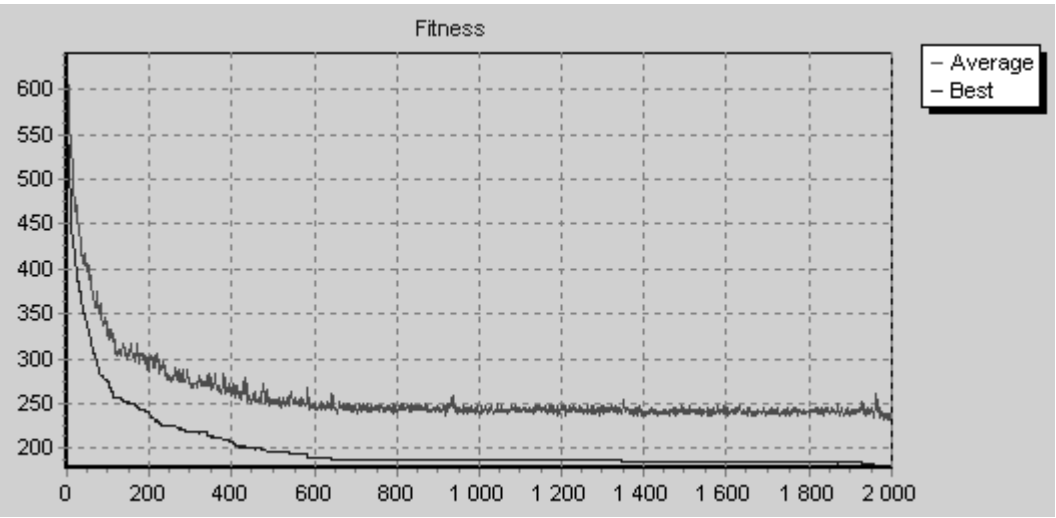
Priemerná hodnota je až 12,63%, čo je vysoké percento, avšak iterácii nebolo veľa. Je však vidno, že dokážeme dostať aj sub-optimálne riešenie a k 9. riadku treba už len dodať, že si riešenie nenašlo stale miesto v populácii. Z týchto (a ďalších) pokusov by som usúdil, že v prípade, že najdeme vhodné parameter pCross a pMut pre menšie mriežky, je skôr vhodne zväčšit o niečo populáciu ako výrazne zmenšit pMut. Takisto priveľká populácia neprinesie výrazné zlepšenia. Kým prejdeme k výsledkom zo zadania b), aby sme mohli urobiť záver, treba ešte povedať, že táto metóda dáva riešenia, ako prejsť za čo najkratší čas ***všetky*** mestá, zatiaľ čo metóda b) poskytuje čo najlepšie riešenia v danej dĺžke.

nonCW	Populácia	pCross	pMut	Generácia	maximum	Počet iterácii	chyba
1	100	0,5	0,035	88	35,2	2000	4,86%
2	100	0,5	0,035	104	35,33333	2000	4,50%
3	100	0,5	0,035	140	36,33333	2000	1,80%
4	100	0,5	0,035	132	33,33333	2000	9,91%
5	100	0,5	0,055	229	36	2000	2,70%
6	100	0,5	0,055	287	35,33333	2000	4,50%
7	100	0,5	0,055	1153	35,14286	2000	5,02%
8	100	0,5	0,055	376	35,33333	2000	4,50%
9	500	0,5	0,07	62	35,2	2000	4,86%
10	500	0,5	0,07	700	35	2000	5,41%
11	500	0,5	0,11	604	36	2000	2,70%
12	500	0,5	0,11	310	37	2000	0,00%
							4,23%

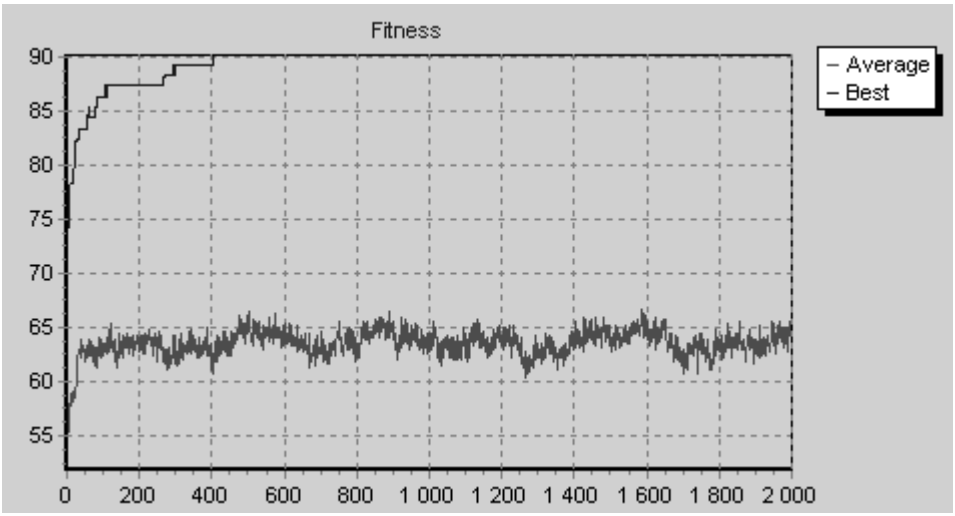
CW	Populácia	pCross	pMut	Generácia	maximum	Počet iterácii	chyba
1	100	0,5	0,035	1282	35,33333	2000	4,50%
2	100	0,5	0,035	1469	36,33333	2000	1,80%
3	100	0,5	0,035	4	34,33333	2000	7,21%
4	100	0,5	0,035	1684	35,33333	2000	4,50%
5	100	0,5	0,055	135	35,33333	2000	4,50%
6	100	0,5	0,055	594	36,33333	2000	1,80%
7	100	0,5	0,055	224	36	2000	2,70%
8	100	0,5	0,055	1977	36	2000	2,70%
9	500	0,5	0,07	327	36,33333	2000	1,80%
10	500	0,5	0,07	97	37	2000	0,00%
11	500	0,5	0,11	35	35,33333	2000	4,50%
12	500	0,5	0,11	1510	37	2000	0,00%
							3,00%

Dve tabulky výstupných hodnot pre metódu b). Prvá je pre zadanie, druhá je modifikovaná (pre okrajové body). V metóde CW sa mi podarilo dkonca pri hodnotách populácia=500,pCross=0,7 a pMut=0,13 dosiahnut pekne výsledky, kde všetky boli v 1. 2000 generáciách sub-optimálne, čiže percentuálne chyby sú priemerne nižšie ako v tabuľke. Je zrejmé, že pohraním sa s parametrami, či opakovaným spustením sa nám podarí dostať pre mriežku 6x6 sub-optimálne riešenie práve metódou b) (či jej modifikovanou verziou) ako metódou a), ktorá dáva 3-4krát horšie výsledky. Samozrejme nie je to jediné hladisko, z ktorého sa môžme na algoritmus dívať. Ďalšou zvláštnosťou algoritmu a) je jeho vytrvalosť, po veľa iteráciách dostaneme naozaj nízke hodnoty, avšak stále treba manipulovať s parametrami, čo len zvyšuje výpočtovú náročnost a na druhej strane, po určitej generácii už tažko môžme očakávať od algoritmu b) nejakú zmenu.

Pre lepšie vnímanie týchto algoritmov si ukážeme priebeh riešení pre mriežku 10x10 pre obe metódy.

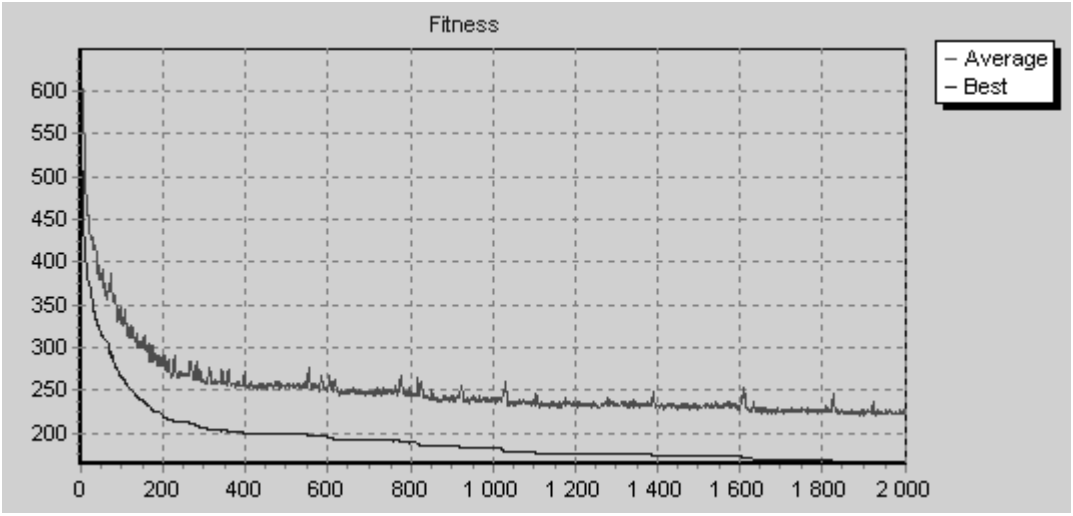


Metóda a) je pomalá, ale vytrvalá [pop=500;pCross=0,65;pMut=0,025]

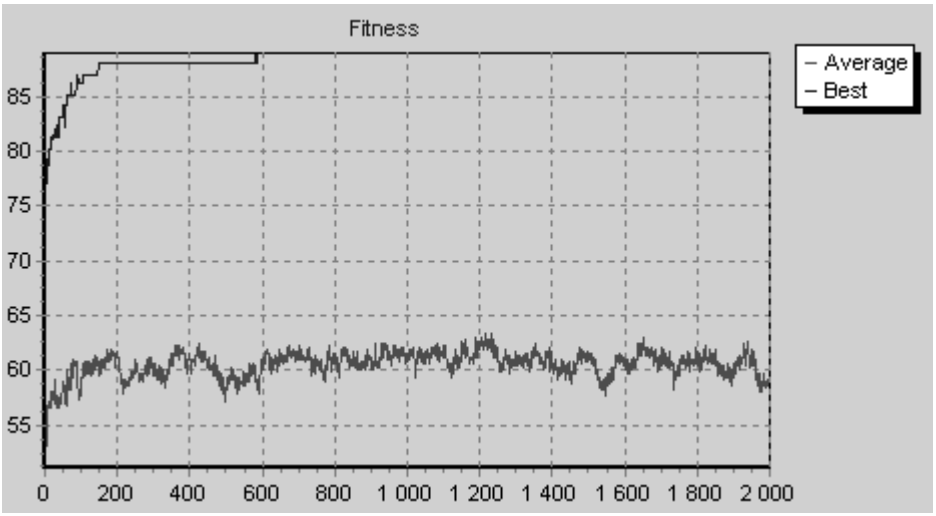


Metóda b) je rýchla...a rýchlo v koncoch [pop=500;pCross=0,7;pMut=0,04]

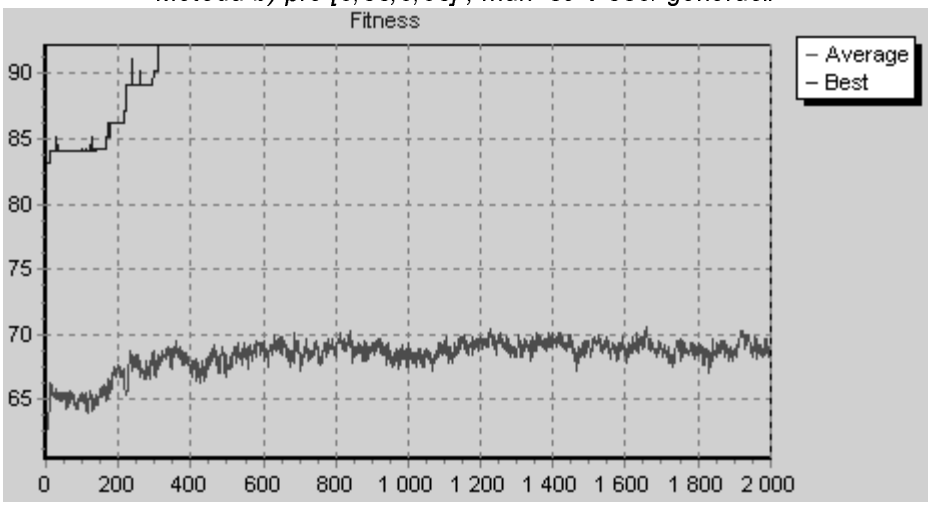
Skúsme teda ešte pár testov pre populáciu 1000 a 2000 generácii.



Metóda a) pre [0,65;0,025] ; max=164 v 1918. generácii



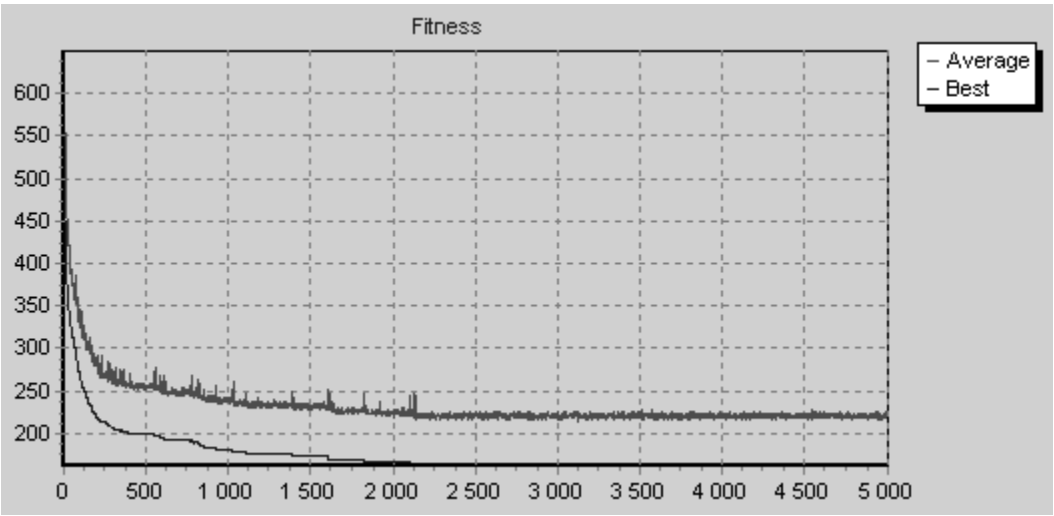
Metóda b) pre [0,55;0,05] ; max=89 v 580. generácii



Modifikovaná metóda b) pre [0,55;0,05] ; max=92,2 v 433. generácii

Vidíme, že pri malom čase CPU vieme povedať s 10% chybou riešenie, ak však máme k dispozícii viac času CPU, vieme takto ohraničiť aj shora metódou a). Priznám sa, že sa mi nepodarilo pre túto mriežku zatiaľ nájsť optimálne riešenie, avšak zas sa ako vhodnejšie rýchle riešenie ukázala modifikovaná verzia. Pomoc pre metódu b) je zjavne nízka oscilácia riešení, umožnuje súčasne držanie sub-optimálnych až nie optimálnych riešení v populácii a znich možné kombinácie môžu viesť k optimálnemu riešeniu, avšak toto optimálne riešenie sa nemusí v populácii udržať. Ďalšou možnosťou je si vytipovať isté chromozómy ako jedinečné. Máme len jednu ohodnocovaciu funkciu, avšak človeka inšpiruje častokrát viac vecí, niektoré síce len s malou pravdepodobnosťou. Nakoniec, aj "čarodejníctvo" kedysi so "slabou fitness" je dnes vhodná alternatíva k medicine, ktorá má vysokú fitness vo všeobecnosti. Pridaním niekoľko (napr. 3% z populácie) náhodne vybratých chromozómov spomedzi najmenej ohodnotených by sme mohli mať alternatívnu zásobu chromozómových reťazcov. Tieto chromozómy by boli akoby pod ochranou spoločnosti.

Zostáva však dokončit túto case study so slovami, že nechať rátať metódu a) dlho a s určitosťou čakať na dobre výsledky sa nevypláca a radšej si spustite modifikovanú metódu b). Aspoň sa vyhnete zlým výsledkom. Posledný graf z metódy a) sa vyvynul do takéhoto stavu:



Metóda a) pre [0,65;0,025] ; max=162 v 2123. generácii

Literatúra:

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

[2] **J.Procházka: *Algoritmy a štruktúry*, Vysokoškolské skriptá MFF UK, Bratislava 1998**