GENETICKÉ ALGORITMY S GENOMEM TVOŘENÝM REÁLNÝMI ČÍSLY (REAL-CODED GENETIC ALGORITHMS)

Vladimír Hlaváč[1]

[1] *FS ČVUT v Praze, hlavac@fs.cvut.cz*

Abstrakt: Řada problémů vede na hledání kombinace reálných čísel, například parametrů modelu nebo hodnot popisujících vhodné nastavení. Tyto jednotlivé proměnné tvoří n-rozměrný prostor, ve kterém hledáme řešení. Schopnost hledat řešení se obvykle demonstruje na nejvýše třech hledaných hodnotách a na dobře podmíněném problému, který neobsahuje příliš mnoho lokálních extrémů. Právě v případech, kdy tyto dvě vlastnosti nejsou splněny, mohou dobře posloužit evoluční nebo hejnové algoritmy, které jsou v zásadě výpočetně náročnější, ale neselhávají ani za velmi složitých podmínek. Článek popisuje především nejzákladnější z nich, genetické algoritmy s reálným genomem a metodou křížení BLX-alfa a jeho modifikací, inteligentním křížením, a jako příklad hejnového algoritmu pak SOMA.
Klíčová slova: Genetické algoritmy, Křížení, BLX-α, Diverzita, .

Abstract: There are many problems defined as a searching for a combination of real numbers, for example model parameters or searching for setting of parameters. Those real numbers create n dimensional search space. The ability for finding a solution is typically demonstrated on at most three searched values and well defined problem, without too many local optima. Just in the other cases, evolutionary algorithms or the swarm intelligence algorithms can be successfully used. Those methods are more time demanding, but can work in very complicated conditions. The article describes the most fundamental of them, real valued genetic algorithms, BLX-α, and its modification, known as the intelligent crossover, and the SOMA algorithms as a swarm particle algorithm example.
Keywords: Genetic algorithms, Crossover, BLX-α, Diversity.

1 Úvod

Genetické algoritmy, nebo obecně evoluční výpočetní techniky [1], používají pro hledání řešení udržování populace tzv. kandidátních řešení. Již známá řešení se vhodným způsobem kombinují (vytvářejí se jejich potomci) a naopak, nejhorší řešení jsou z populace vyřazována. Terminologický rozdíl oproti hejnovým algoritmům je v tom, že u hejnových algoritmů se jednotlivá řešení stále upravují podle úspěšnoti „okolních“ řešení (v obou případech si řešení lze představit jako bod v mnoharozměrném prostoru). V obou případech se kvalita řešení posuzuje podle vhodně definované účelové funkce.

Velikost populace je tedy u hejnových algoritmů zpravidla stálá, zatímco u genetických se může měnit. U genetických algoritmů lze ovšem realizovat evoluci použitím tzv. turnajové selekce, která také trvale zachovává velikost populace. Typické řešení turnajové selekce (podle [2], str. 137) je výběr dvou dvojic jedinců z populace. Z každé dvojice se vybere lepší jedinec, horší se z populace vyřadí. Místo vyřazených jedinců se umístí dva potomci, vytvoření křížením nevyřazených jedinců. Ti v populaci také zůstávají (celkový počet jedinců je tedy zachován). Oproti běžnější ruletové selekci není třeba populaci řadit a ušetří se tedy často velmi náročné porovnání. Turnajová selekce ale neumožňuje řídit evoluční tlak a tím genetické algoritmy ztrácí hlavní výhodu oproti hejnovým algoritmům.

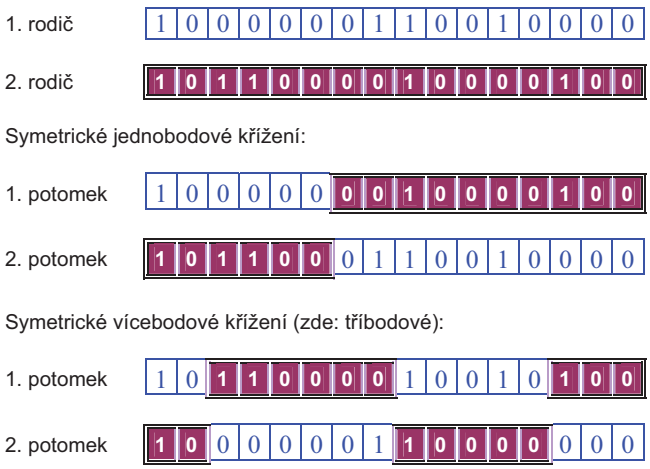
Rozdílný přístup vychází z použití ruletové selekce. Zde je v paměti pro hledané jedince vyhrazen podstatně větší počet míst, než kolik je jedinců v základní populaci. Pro vytváření nového jedince se používá křížení nebo některý typ tzv. mutace. Po vytvoření nových jedinců do volných míst je populace opět seřazena podle hodnoty

účelové funkce a v populaci je ponecháno opět jen tolik jedinců, kolik tvoří základní populaci. U ruletové selekce jsou navíc pro křížení vybíráni s vyšší pravděpodobností jedinci s lepší hodnotou účelové funkce. Pravděpodobnost výběru může obsahovat i samotnou hodnotu účelové funkce. Použití pouze pořadí v populaci ovšem dává lepší výsledky. Pokud je použito exponenciální rozložení pravděpodobnosti výběru jedince pro křížení, lze evoluční tlak definovat jako poměr pravděpodobnosti výběru dvou po sobě jdoucích jedinců v populaci. Při praktických testech vychází jako vhodný poměr čísla v rozsahu 1,01 až 1,001, v závislosti na velikosti základní populace. Podrobněji viz [3], kapitola 4.

Hejnové algoritmy svým názvem vedou k představě algoritmu jako udržování množiny řešení, pro které se v každém kroku vždy vyčíslí účelová funkce a pak se jednotlivé body posunou ve směru k jedincům, kteří aktuálně dosahují lepších hodnot účelové funkce. Stejnou představu je třeba uplatnit i na genetické algoritmy. V obou případech se vytvoří jakýsi mrak řešení, který prohledává blízký prostor a posouvá se ve směru, kterým se řešení zlepšuje. Z této představy vyplývá také to nejdůležitější, co musíme při udržování populace hlídat. Pokud se hejno bezhlavě slétne k nejlepšímu řešení, máme algoritmus, který rychle a spolehlivě uvízne v prvním lokálním minimu účelové funkce. Pokud máme problém, kde nejsou lokální optima, pak tato situace nenastává, ale z druhé strany máme lepší (a řádově rychlejší) metody, jak najít jediné řešení, například gradientní a iterační metody. Proto je třeba udržovat velikost prohledávaného prostoru. Ostatně český pojem „hejno“ evokuje hejno ptáků, kde také jedinci nemohou být příliš blízko sebe. Anglický pojem „swarm optimization“ (swarm je „částice“) ovšem nic podobného nezahrnuje. U genetických algoritmů tomuto problému odpovídá pojem diverzita. Pokud si všechna řešení začnou být podobná, veškerý vývoj se zastaví. Tato situace se označuje jako zamrznutí populace. Algoritmus to zpravidla řeší tak, že se při poklesu diverzity pokusí použít větší množství mutací, aby obnovil vývoj genomu ([4], str.149). Většina mutací (náhodných změn v genomu) ovšem vede na tvorbu defektních jedinců. Pokud mají nějak ovlivnit populaci, musí se i tito jedinci použít pro křížení (to vede ke změně pořadí operací, kdy se mutace provádí před křížením a výslední mutanti jsou náhodně zařazováni do populace, zatímco ostatní jsou posouváni na její konec). Další ochranou proti zamrznutí populace je vyřazování jedinců, kteří jsou starší než stanovený počet cyklů (např. 100) [5]. Proti vyřazením a mutacím bývá chráněno dosud nejlepší nalezené řešení. Tento přístup se označuje jako elitismus.

Základní představa zápisu genomu byla prostá kombinace nul a jedniček (binární genom), kdy křížení reprezentovalo střídavé kopírování genomu z jednoho nebo druhého rodiče [6]. Typické je symetrické třídění, kdy nevyužité části genomu vytvoří druhého jedince. Binární genom nebere ohled na zakódované informace ani na jejich hranice. Například rodiče mají hodnotu nějakého parametru jeden 192 a druhý 189, tedy binárně 11000000b a 10111101b. Kombinací například po změně zdroje na 3. bitu vznikne 11111101b a 10000000b, tj. 253 a 128. Chybí zde tedy jasný vztah mezi rodiči a potomky.

Výhodou binárního genomu je, že jej mohou tvořit slepená binární čísla různé délky. Zápis celých, např. 32bitových čísel, by zde zbytečně zpomaloval evoluci. Pokud například hledáme den a měsíc, což je v rozsahu do 31 a do 12, vyhradíme v genomu jen 5, resp. 4 bity. V základní verzi genetických algoritmů křížení hranice jednotlivých znaků ignoruje. To z druhé strany umožňuje změnu významu částí genomu.



Obr. 1.: Ukázka symetrického křížení s binárním genomem [3]. Význam jednotlivých bitů často závisí na kontextu.

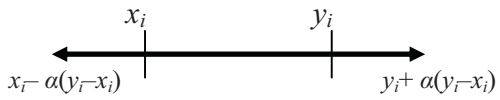
2 Hledání kombinace reálných čísel

2.1 BLX-α

Genetické algoritmy s genomem tvořeným reálnými čísly jsou definovány tak, že jednotlivé geny tvoří čísla (často v nějakém rozsahu). Křížení a mutace se od binárního genomu liší; použijí se číselné hodnoty a ty se kombinují. Nemůžeme vzít prostě průměrnou hodnotu obou čísel, to bychom ztratili diverzitu. O něco lepší přístup je vzít číslo z prvního rodiče s pravděpodobností p_1 a z druhého rodiče s pravděpodobností $p_2 = 1 - p_1$:

$$c_i = p_1 a_i + (1 - p_1) b_i \tag{1}$$

Index i rozlišuje jednotlivé složky genomu. Popsáno již v [7]. I v tomto případě dochází k jistému omezení diverzity. Protože by po chvíli došlo ke zprůměrování všech hodnot a k zastavení vývoje, bylo navrženo následující opatření: Představme si, že jednotlivá čísla z genomů obou rodičů představují dva protilehlé vrcholy, definující v prostoru n-rozměrný kvádr. Za novou kombinaci čísel bude vzata taková n-rozměrná souřadnice, která leží kdekoli nejen v takto vymezeném kvádru, ale v kvádru, který má stejný střed a polohu, ale je (1+α)-krát ve všech dimenzích zvětšen. Protože bod je vybírán kdekoli v tomto prostoru, je metoda označována jako slepé křížení, blind crossover, s parametrem α, ve zkratce BLX-α [8]. Obr. 2 znázorňuje hodnotu souřadnice pro výchozí hodnoty x_i a y_i.



Obr. 2.: Ukázka křížení jedinců X a Y pro jednotlivou souřadnici, kdy výsledná hodnota je kdekoli na dané úsečce.

Pro výpočet jednotlivých souřadnic v zásadě platí (1), ale parametr je třeba náhodně volit v rozsahu rozšířeném o hodnotu α:

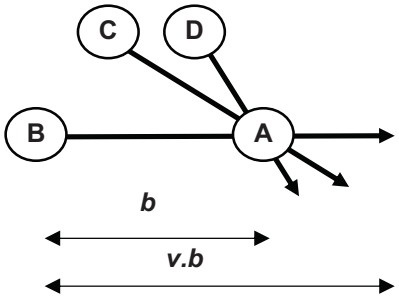
$$z_i = k_i x_i + (1 - k_i) y_i, k_i \in \langle -\alpha; 1 + \alpha \rangle. \tag{2}$$

2.2 Inteligentní křížení

U BLX-α se koeficient výběru pro každou souřadnici generuje náhodně znovu. Pokud použijeme pro všechny souřadnice stejnou hodnotu náhodného čísla, získáme tzv. inteligentní křížení (popsáno v [9]). Zde řešení leží na přímce, dané dvěma rodičovskými řešeními, ale jeho poloha může být i mimo jimi danou úsečku. Pro snadné srovnání můžeme míru o jakou může být řešení generováno mimo tuto úsečku označit opět α. Časté je ovšem nesymetrické rozšíření, například více do kladných hodnot než do záporných.

2.3 SOMA

Self-Organizing Migrating Algorithm (SOMA) [1] pracuje v podstatě podobně. SOMA je ovšem hejnový algoritmus, takže nemůže generovat nové členy populace. Místo toho mění jednotlivá řešení. Nejprve určí nejlepší řešení (vedoucí, leader) a pak všechna ostatní posune ve směru k tomuto řešení. Pokud postupujeme podle obdobného vzorce, jako je (2), můžeme řešení o jistý úsek naopak posunout ve směru od nejlepšího řešení. Tím je u této metody zajištěna diverzita. Lépe by ovšem bylo diverzitu zajišťovat v opačném směru, a interval by tedy měl být rozšířen nesymetricky (rozšíření nejprve zlepšuje úspěšnost metody při hledání řešení, vyšší hodnoty snižují stabilitu).



Obr. 3.: Schematické znázornění algoritmu SOMA [10]. Nejlepší řešení A se neposouvá, B, C a D se mění.

Existuje řada modifikací algoritmu SOMA. Program představený v [11] používá variantu, kdy jsou vybrány náhodně jen čtyři prvky populace, z nich se vybere nejlepší a ostatní tři jsou posunuty – náhodná vzdálenost je generována v zadaných mezích, osvědčily se nesymetrické až dvojnásobné ve směru nejlepšího řešení, s možností až 50% kroku směrem od nejlepšího řešení.

3 Data

Pro následující ukázky byla data generována stejnou funkcí, jako v [10]:

$$f(x, y) = k_1 \sin(k_2 x + k_3 y) \tag{3}$$

Konstanty byly opět nastaveny na hodnoty 3,71, 1,73 a poslední má hodnotu 1. Data byla vyčíslena v 555 bodech, pro hodnoty x a y náhodně generované v rozsahu $(-\pi; \pi)$. Generovaná data jsou opět zatížena malým šumem [10]. Účelová funkce byla vyčíslována jako součet čtverců odchylek v zadaných bodech.

4 Výsledky testování algoritmů

Následující grafy demonstrují základní problém, společný genetickým a hejnovým algoritmům, ztrátu diverzity. Z tohoto důvodu neznázorňují, jak se vyvíjí hledané řešení (které je reprezentováno nejlepším jedincem v populaci), ale nejmenší a největší hodnotu hledaného parametru v populaci. Úloha představuje hledání tří parametrů v datech, vygenerovaných podle funkce (3), z nichž první parametr zpravidla činí největší problémy. Proto je v následujících grafech zobrazována hodnota právě **jen** tohoto **prvního** parametru. Jak genetické, tak hejnové algoritmy představují jakýsi mrak řešení, který se pohybuje prostorem ve směru, kterým se nachází lepší hodnoty účelové funkce (zde menší součet odchylek ve všech zadávaných bodech). Pokud populace zcela ztratí diverzitu (hodnoty konstant u celé populace začnou být stejné), ztratí metoda schopnost zjistit tento směr, nebo dokonce vykonávat jakýkoli pohyb, protože míra tohoto kroku je dána jako násobek rozdílu hodnot parametrů u dvou různých řešení.

Grafy jsou s ohledem na velký počet křivek značně nepřehledné. Proto obrázek 4 ukazuje jen jeden konkrétní průběh na ukázku způsobu zobrazení. Vždy dvě křivky stejné barvy reprezentují maximální a minimální hodnotu hledaného parametru v populaci. Tenká červená čára, zvýrazněná označením bodů, pro které probíhá výpočet, znamená známé správné řešení problému. Program prezentovaný v [11] byl upraven tak, aby do log-souboru zaznamenával i nejmenší a největší prvek v populaci (jedná se o hodnotu parametru v genomu, číslo nemá nic společného s hodnotou účelové funkce).

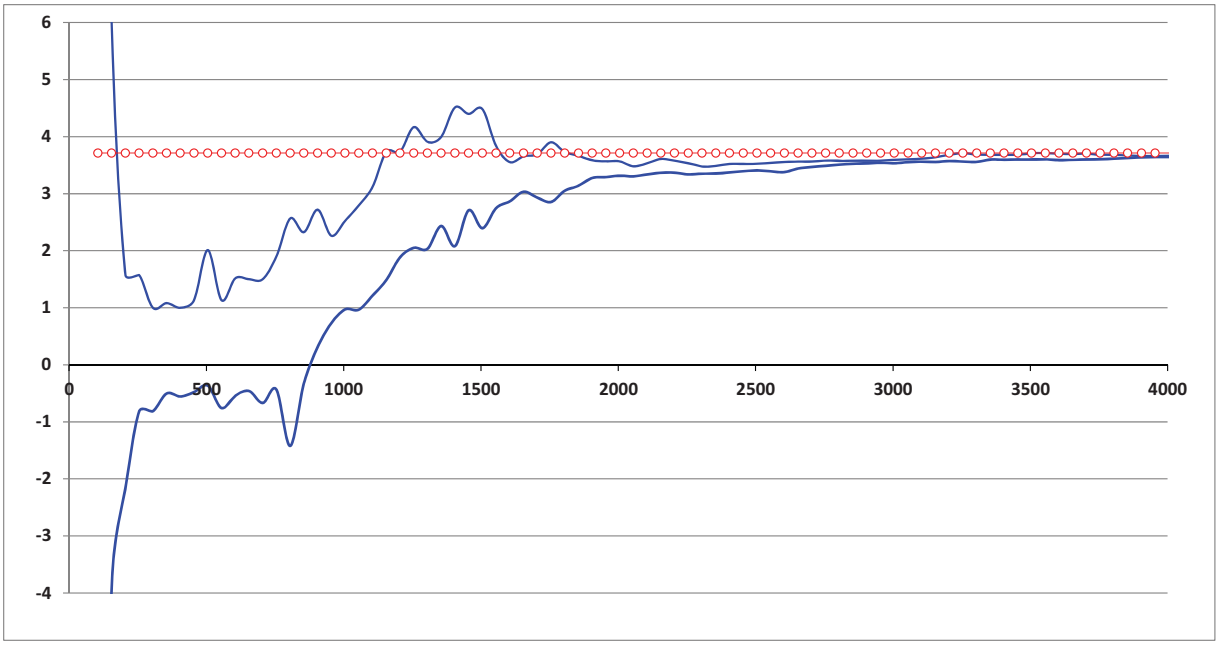
U grafů na obrázku 11 a 14 je doplněno i zobrazení nejlepšího řešení (podle hodnoty účelové funkce). Je to opět čára stejné barvy, ale na obrázku 11 je pro rozlišení tlustou čarou, na obrázku 14 bylo zvoleno různé provedení čáry (tečkovaně, čárkovaně), protože takové čáry jsou snáze viditelné, pokud se překrývají.

Právě graf na obrázku 4 nejlépe popisuje chování tohoto typu algoritmů. „Mrak řešení“ se nejprve zkoncentruje okolo zatím nejlepších řešení a pak se celý posouvá ve směru správného řešení. Když překročí nejlepší hodnotu účelové funkce, začne se okolo ní koncentrovat (rozdíly mezi jednotlivými hodnotami hledaných koeficientů se mezi různými řešeními začnou snižovat). Na konci evoluce mají všechna řešení přibližně stejnou hodnotu koeficientů a nejlepší řešení se vybere podle hodnoty účelové funkce.

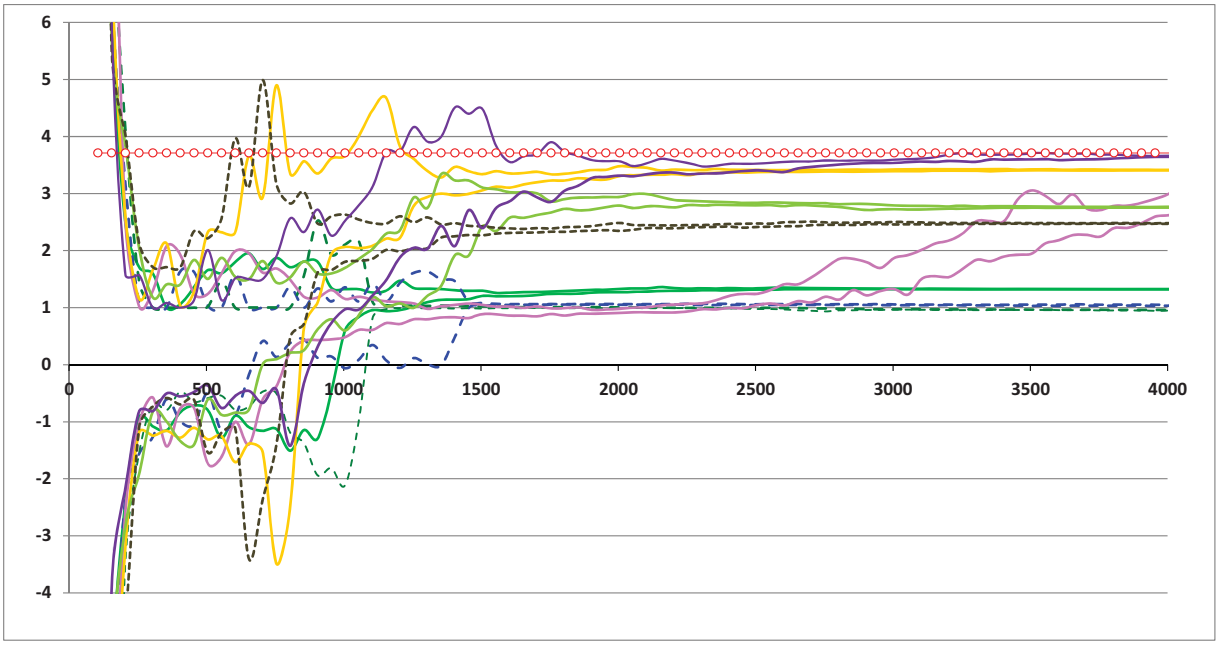
Grafy ve vodorovné ose nezačínají v nule. Je to tím, že jistý počet vyhodnocení účelové funkce proběhne hned při generování nové populace. Je to 100 vyhodnocení pro genetické algoritmy a 200 pro SOMA. Rozdíl je dán způsobem implementace těchto algoritmů a neměl by mít velký vliv na rychlost samotného výpočtu. V rámci testů bylo nastaveno 100 generací a komentáře u obrázků (nalezení správné hodnoty apod.) se vztahují k tomuto počtu generací. Zobrazený rozsah vodorovné osy (4000 vyhodnocení účelové funkce) ale odpovídá necelým 78 generacím u genetických algoritmů.

Aby byly výsledky srovnatelné, bylo vždy 50 jedinců v základní populaci. U genetických algoritmů bylo dalších 50 generováno v každé generaci (ať již křížením, nebo kombinací křížení a mutací). Pro každého nově vygenerovaného jedince je vždy ihned spočítána odpovídající hodnota účelové funkce. U genetických algoritmů jsou podle této účelové funkce seřazeni a pro další generaci je ponecháno jen 50 nejlepších. U SOMA místo toho proběhne 50 posunutí. SOMA nevyžaduje seřazení, ale pro účel zápisu do log-souboru jsou v programu nalezeni nejlepší jedinec a nejmenší a největší hodnoty prvního parametru, aby bylo možné vykreslit graf. Při praktickém použití metody lze v programu zápis do souboru a s tím spojené operace vypnout.

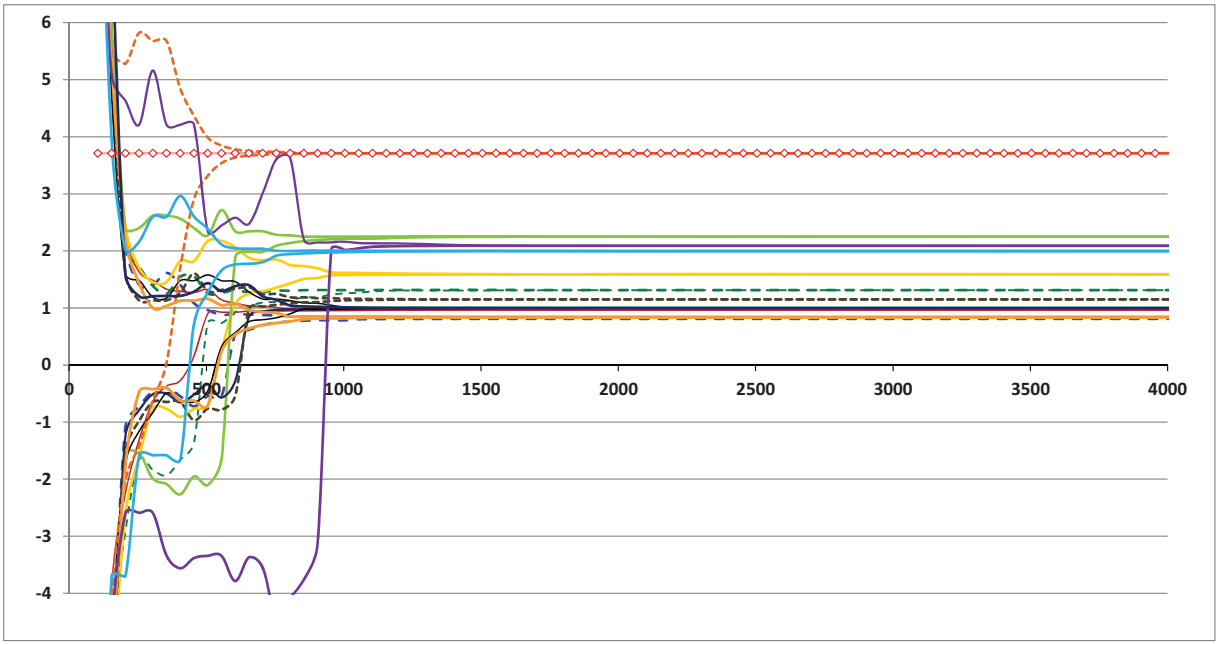
Každá metoda byla volána opakovaně a jednotlivé průběhy byly do grafů zpracovány programem MS Excel. Zde je nutné podotknout, že protože funkce sin je lichá, má úloha dvě možná řešení, kromě zadání {3,71, 1,73, 1,0} je to {–3,71, –1,73, –1,0}. V grafech by společné zobrazení vypadalo ještě více chaoticky. Proto byly průběhy, kde byly výsledné koeficienty nejlepšího jedince po sto generacích (konečné hodnoty) záporné ze zobrazení zcela vynechány. Za tento nesystematický přístup k datům se omlouvám a jsem přesvědčen, že není pro výsledné posouzení průběhů na závadu. Ve všech testovaných případech byly vždy všechny nalezené koeficienty u nejlepšího jedince stejného znaménka. Při snižování počtu grafů (z důvodu názornosti, viditelné na obr. 11, ale použité všude) byly mazány vždy poslední nahrané průběhy (a ponechané první), aby výsledky zachovaly alespoň částečnou reprezentativnost. Výjimkou je obr. 4, kde se jedná o třetí průběh.



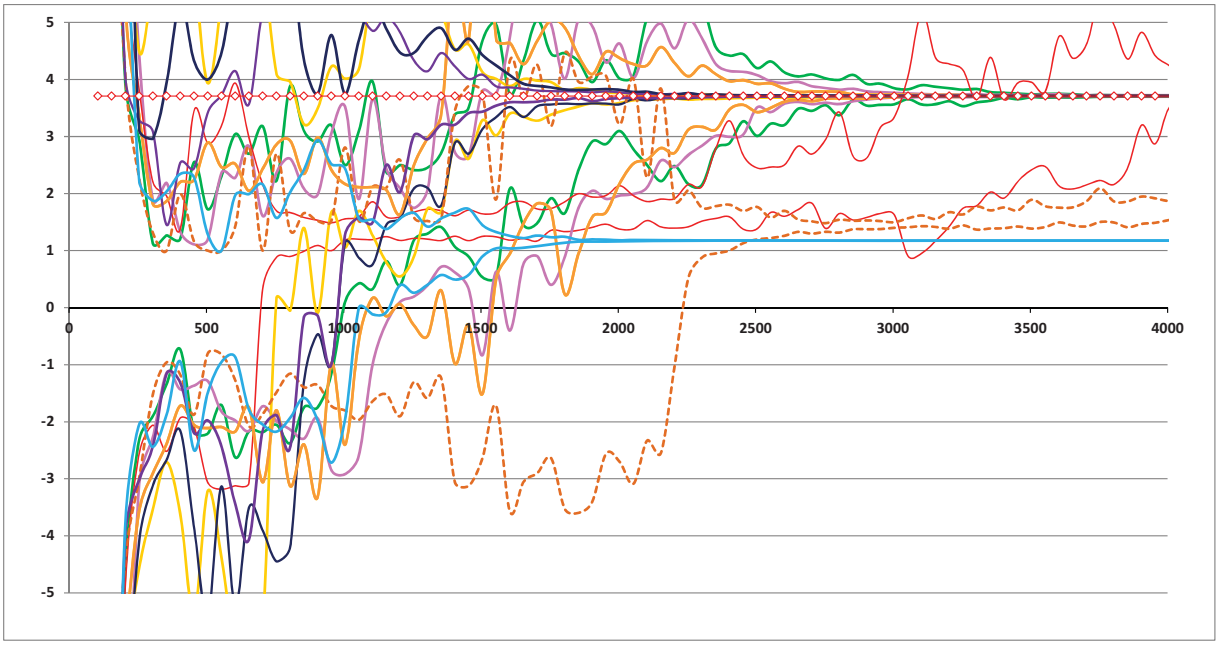
Obr. 4.: Ukázka konstrukce grafu pro jeden konkrétní průběh simulace s použitím inteligentního křížení (jedná se o jeden z grafů z obrázku 5, kde je fialovou barvou, nastavení viz obr. 5). Program hledá tři koeficienty, v grafech je zobrazena hodnota jen prvního z nich, se kterým mají popisované metody největší problémy. Vždy dvě křivky stejné barvy (resp. typu) reprezentují vždy největší a nejmenší hodnotu daného parametru v populaci v dané generaci. Červená konstanta zdůrazněná kroužky reprezentuje správnou hodnotu hledaného koeficientu (3,71), kroužky pak označují okamžiky, ke kterým jsou zobrazovány ostatní křivky. Vodorovná osa reprezentuje počet vyčíslení účelové funkce, což je nejlepší dostupné měřítko časové náročnosti metody. Pokud se ztratí diverzita, pak metoda podle obr. 2 již ztratí možnost pohybu v prostoru řešení.



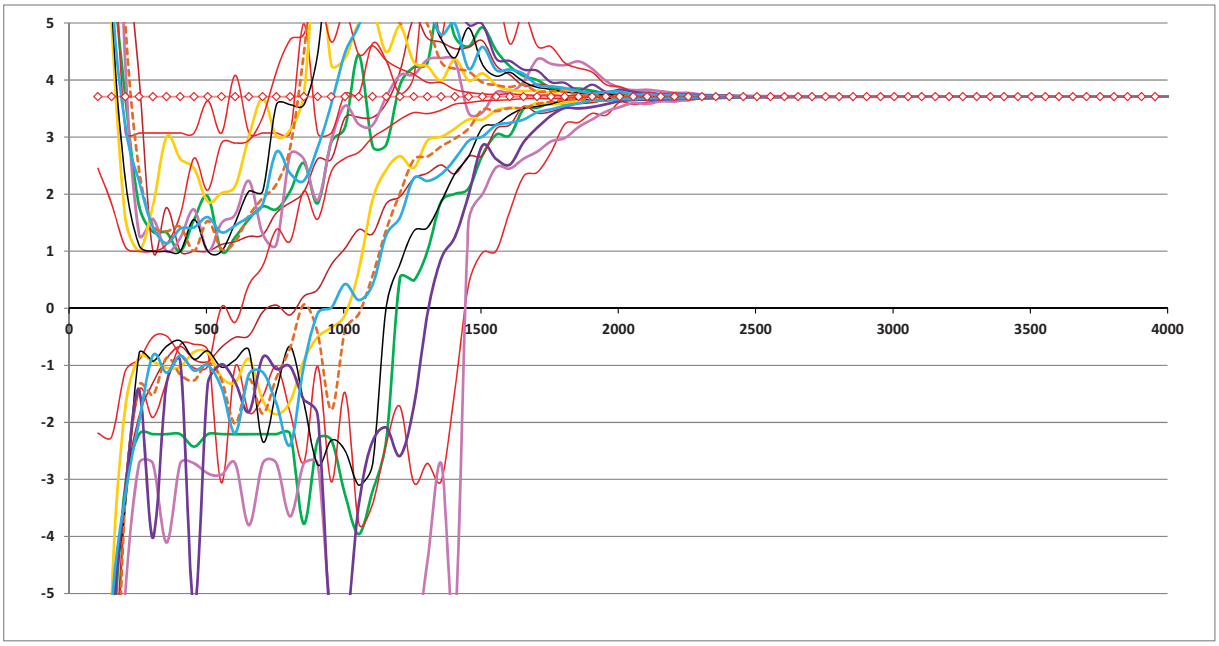
Obr. 5.: Základní nastavení pro porovnání: Velikost základní populace n=50, křížením vytvořeno x=50 potomků v generaci, evoluční tlak q=1,03, „intelligent crossover“, koeficient α=1 (velikost přesahu křížení, „jen mezi hodnotami rodičů“ = 0, může být i záporné (což by znamenalo nevybírání ani z celého rozsahu vymezeného rodiči), teoreticky se může blížit –0,5, ale pak se okamžitě ztratí diverzita a metoda zaručeně selže). Evoluční tlak vyjádřený parametrem q popisuje, kolikrát je větší pravděpodobnost výběru rodiče v populaci oproti jinému, který je v seřazené populaci o jednu pozici níže. Pravděpodobnosti výběru rodiče podle pořadí v populaci pak tvoří geometrickou řadu [3].



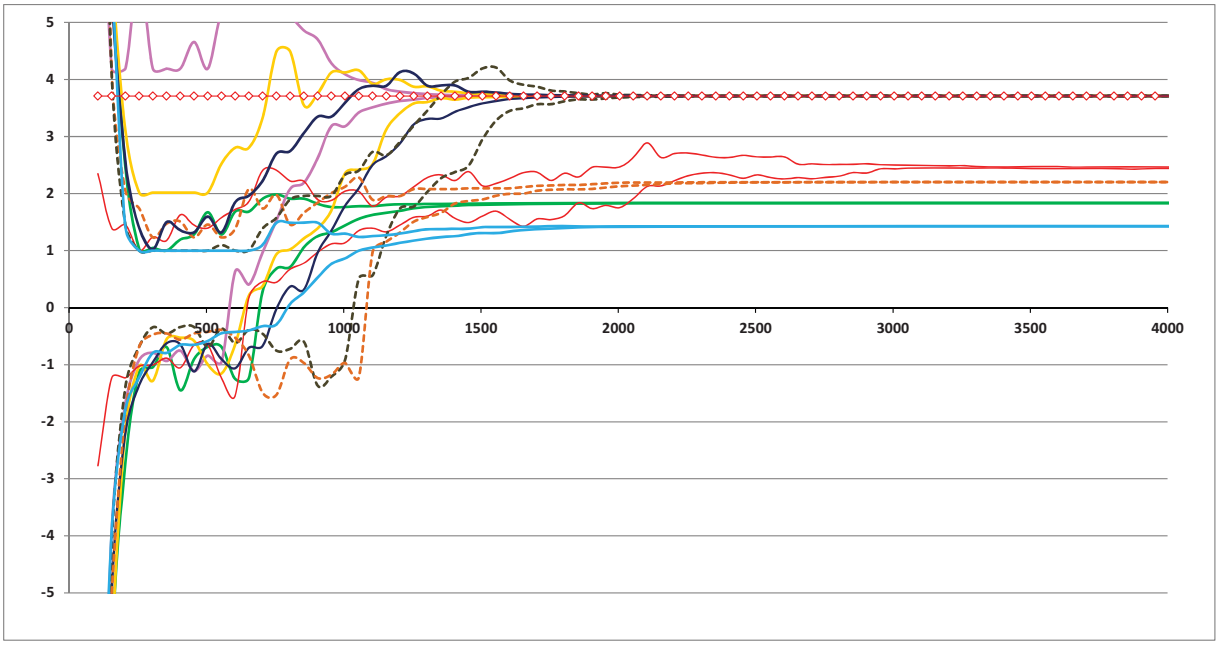
Obr. 6.: Změna hodnoty α na všeobecně doporučovanou hodnotu 0,3 [8]. Evoluce je rychlejší, ale vede na rychlé zamrznutí v bodě, který má k řešení daleko. V případě oranžové přerušované čáry bylo řešení nalezeno rychle, ale jedná se spíše o náhodu.



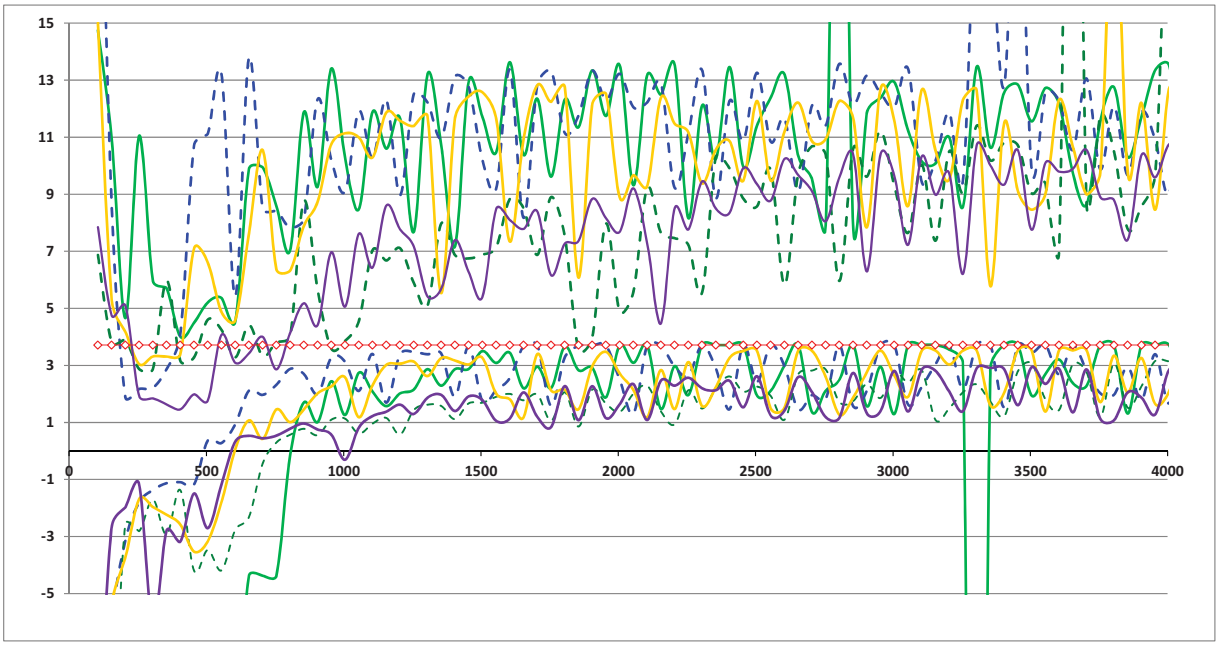
Obr. 7.: Změna hodnoty na α=2,0 . Evoluce se prodloužila, populace nemá tendenci konvergovat, ale ve většině případů nakonec nalezne správné řešení. Velká hodnota koeficientu α zde vlastně nahrazuje mutace, které jinak u genetických algoritmů zabraňují zamrznutí populace. Toto řešení není dokonalé, jak ukazuje světle modrá populace, která stačila „zamrznout“ po cca 30 generacích (graf zde zobrazuje 78 generací). Tenkou červenou čarou zobrazená populace se blíží k řešení, stejně se dá předpokládat, že je nalezne i oranžovou přerušovanou barvou vyznačená populace, která stále neztrácí diverzitu.



Obr. 8.: BLX-α. Od předchozí metody se liší tím, že pro každou ze souřadnic se náhodná hodnota v rozsahu (–α; 1+α) generuje vždy znovu. Při podrobnějším zkoumání zobrazených dat (číselně ve zdrojových datech grafu) lze konstatovat, že ve třetině případů končí 1 až 2 procenta od správné hodnoty, což na grafech není poznat. Při předchozích testech metoda občas také uvízla na chybné hodnotě (až z 10 procent).

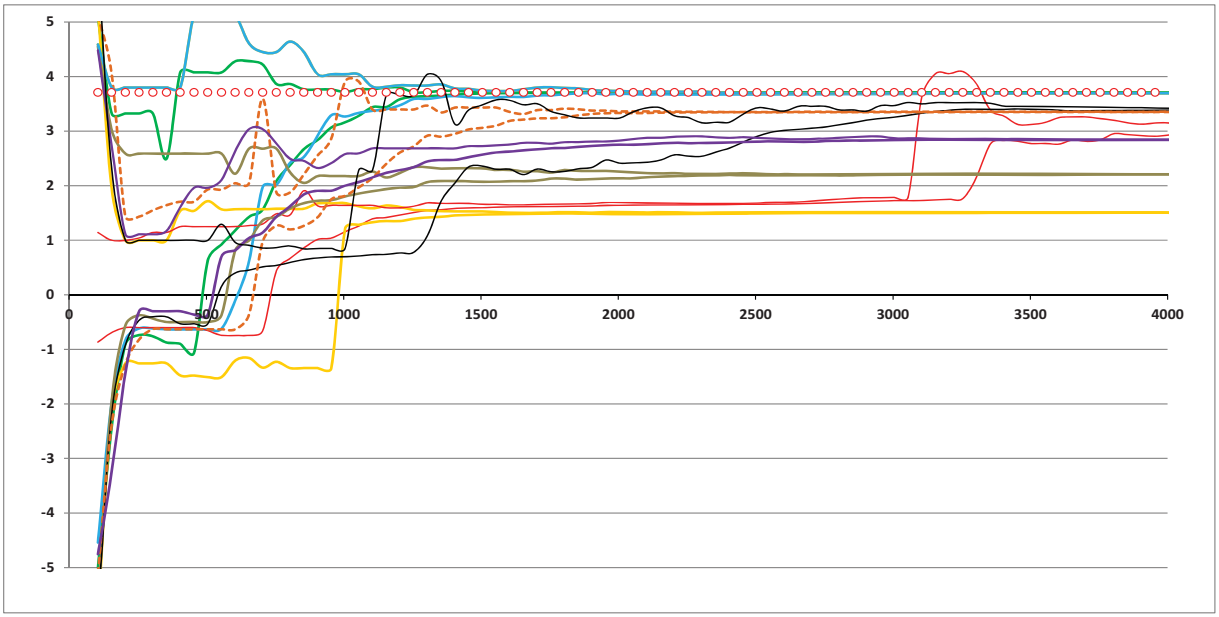


Obr. 9.: BLX-α, α=0,5. Tato hodnota je na mezi, kdy metoda ještě většinou (>50%) nalezne správné řešení.

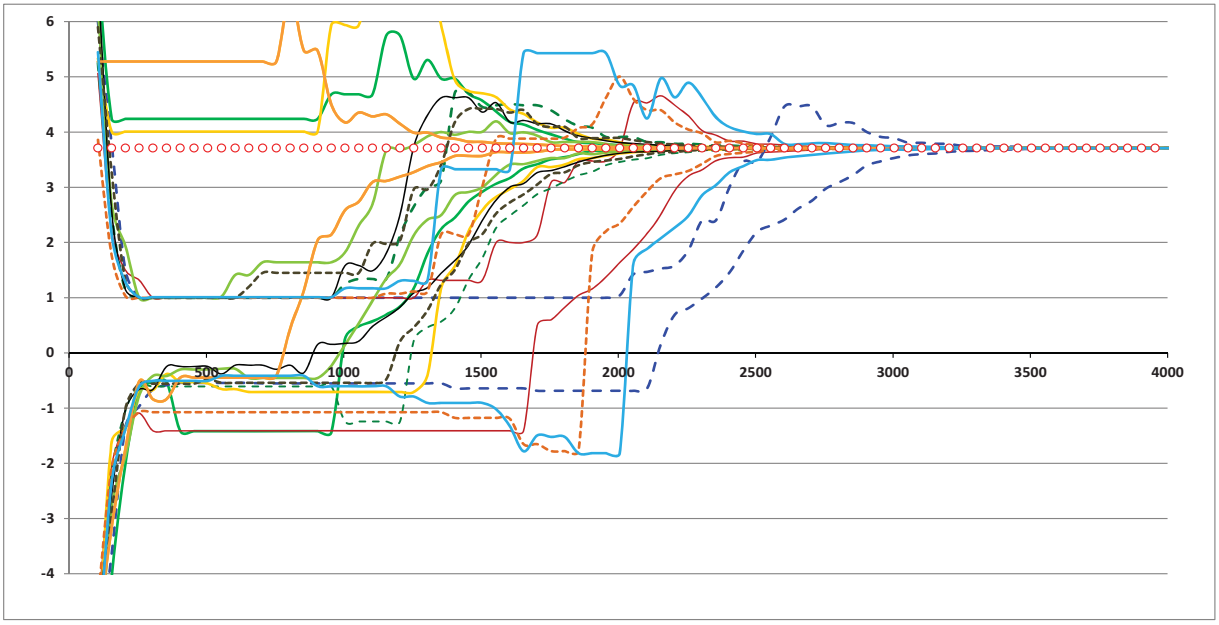


Obr. 10.: Původní nastavení (inteligentní křížení), v každé generaci 45 potomků inteligentním křížením a 5 mutací. V tomto případě není zobrazení mezí (největší a nejmenší hodnoty prvního koeficientu u jednotlivých členů populace) vhodné. Vysoká úroveň mutací (5 z 50 reprezentuje 10%) způsobuje chaotické chování a nestabilitu, nejlepší řešení ale může přesto konvergovat.

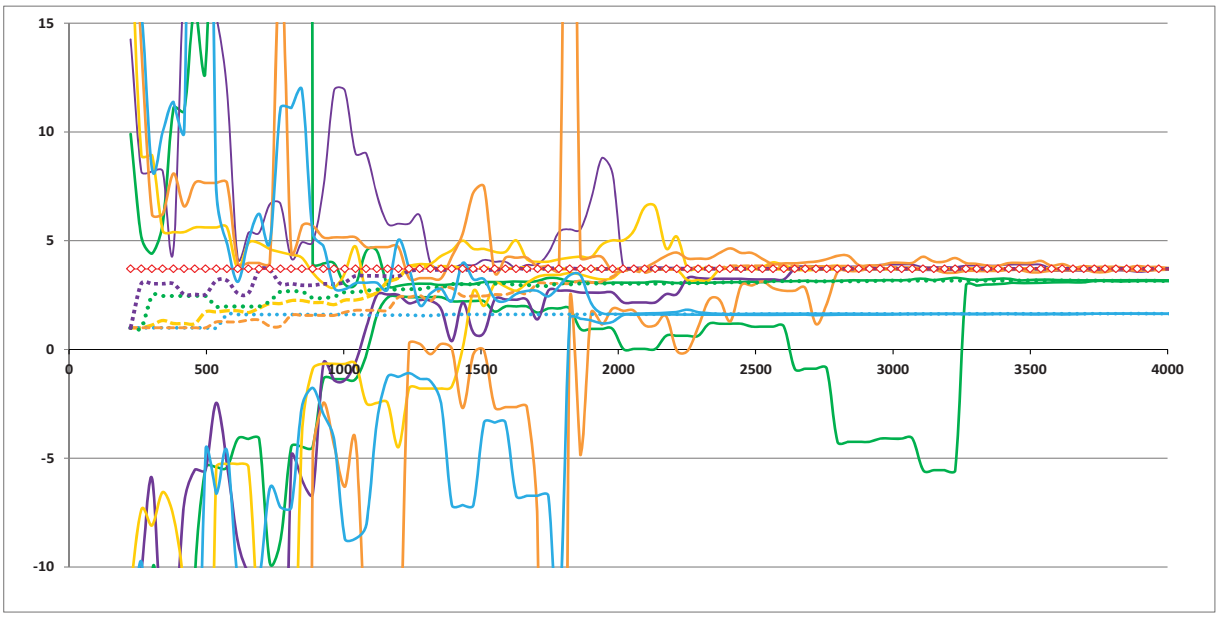
Obr. 11.: Jiné zobrazení dat z předchozího obrázku. Ponechány jen tři průběhy (oproti obr. 10 změněny barvy), ale je doplněno dosud **nejlepší řešení** (tlustá čára stejné barvy). Fialový průběh v rámci 78 generací vůbec nenalezne řešení. Je třeba zdůraznit, že zobrazena je jen první z hledaných konstant, proto se může řešení od nalezené hodnoty i vzdalovat (mohou se zlepšovat ostatní hledané konstanty). K průběhu mezí a nejlepšího řešení je třeba zdůraznit, že použití hladké čáry pro grafy je zde matoucí. Pouze v časech zobrazených červenými kroužky křivky prochází spočítanou hodnotou. Dosažené hodnoty: žlutá – nalezeno řešení (3,71), zelená – přiblížení na 3,7173, fialová – dosaženo 2,94.



Obr. 12.: 2 z 50 (4 %) potomků v generaci tvořeno mutací. V tomto případě je zobrazován stav po setřídění (mutovaní mají vliv na zobrazení, jen pokud přežijí do nové generace). Po těchto úpravách je chování algoritmu mnohem stabilnější, ale je patrné, že ve většině případů nenajde správnou kombinaci konstant. Tomu lze zabránit změnou pořadí operací, kdy mutace se použijí před křížením a výsledky se zařazují do populace místo vzorů (ty se přesouvají na konec, na své místo se mohou vrátit na konci cyklu, kdy jsou prvky znovu seřazeny). Zde nebyla tato modifikace testována.



Obr. 13.: Podobné nastavení jako v předchozím, ale metoda křížení je BLX-α. Testováno vícekrát, řešení nalezeno v tomto případě vždy.



Obr. 14.: Jen pro porovnání: jedna z možností implementace SOMA (angl. Self Organizing Migration Algorithm, česky samo-organizující se migrační algoritmus). Algoritmus udržuje vysokou diverzitu populace a nepotřebuje mutace (pozor na jiné měřítko na svislé ose). Technicky provádí turnajovou selekci. Jako hejnový algoritmus nemění počet jedinců v populaci; vylosuje čtyři jedince, z nich určí nejlepšího (toho ponechá beze změn) a ostatní posune ve směru k němu o vzájemnou vzdálenost násobenou koeficientem α, zde v rozsahu od –0,5 do 2. Protože diverzita populace je veliká, je opět doplněna hodnota nejlepšího zatím dosaženého řešení, ve stejné barvě, ale přerušovanou čarou. V případě světle modře označeného průběhu okolo 40. „generace“ algoritmus ztratí diverzitu populace (největší a nejmenší koeficient v populaci mají skoro stejnou hodnotu) a zamrzne na dosažené hodnotě. Opět je třeba připomenout, že program hledá kombinaci tří koeficientů pro nejlepší proložení bodů (regrese), ale zobrazen je jen první z nich. Řešená úloha je ale velmi nelineární. SOMA při tomto nastavení najde obvykle řešení v 80% případů.

5 Závěr

Z výsledků je patrný význam udržení diverzity populace a je názorně patrný smysl představy hejna řešení, které putuje prostorem ve směru nejlepších hodnot. Podrobněji je představena základní metoda BLX-α, která chyběla mezi ukázkami v [10]. Použitá velikost populace spadá do rozsahu, doporučovaném v literatuře (např. [4]). Při větším počtu hledaných parametrů by samozřejmě bylo nutné volit větší populace. Další, co by bylo třeba řešit, je vliv mezí, mezi kterými byla generována počáteční populace (zde v rozsahu ±10). Popisované algoritmy naleznou řešení, i když se nalézá mimo rozsah počáteční populace, ale vliv počáteční populace na rychlost konvergence by také bylo zajímavé popsat.

6 Literatura

[1] Zelinka, I. et al.: Evoluční výpočetní techniky: principy a aplikace. Praha: BEN - technická literatura, 2009.
[2] Mařík, V. et al.: Umělá inteligence (3). Vyd. 1. Praha: Academia, 2001.
[3] Hlaváč, V.: Nový algoritmus pro symbolickou regresi pomocí genetického programování s upřesňováním číselné hodnoty koeficientů. Doktorská práce. Fakulta dopravní ČVUT v Praze, 2017.
[4] Banzhaf, W. et al.: Genetic Programming, an Introduction. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998.
[5] Ghosh, A., Tsutsui, S., Tanaka, H.: Individual Aging in Genetic Algorithms. in: 1996 Australian New Zealand Conf. on Intelligent Information Systems, Adelaide, Australia.
[6] Mitchell, M.: An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press, 1996.
[7] Radcliffe, N.: Forma Analysis of Random respectful Recombination. in: Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego, 13-16 July 1991.
[8] Eshelman L.J., Schaffer J.D.: Real-Coded Genetic Algorithms and Interval-Schemata. in: Foundation of Genetic Algorithms 2, L.Darrell Whitley (Ed.). Morgan Kaufmann Publishers, San Mateo, 1993, pp. 187–202.
[9] Brandejsky T.: Genetic Programming Algorithm with constants pre-optimization of modified candidates of new population. In: Mendel 2004, Brno, 2004, pp. 34-38.
[10] Hlaváč, V.: Vyčíslování konstant progenetické programování. in: Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky 2017. ČVUT v Praze, 2017.
[11] Hlaváč, V.: A program searching for a functional dependence using genetic programming with coefficient adjustment. in: SCSP Prague, 2016.

Selected article from
Tento dokument byl publikován ve sborníku

Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky 2018
New Methods and Practices in the Instrumentation, Automatic Control and Informatics 2018
28. 5. – 30. 5. 2018, Příbram - Podlesí

ISBN 978-80-01-06477-1

Web page of the original document:
http://control.fs.cvut.cz/nmp
http://iat.fs.cvut.cz/nmp/2018.pdf

Obsah čísla/individual articles:
http://iat.fs.cvut.cz/nmp/2018/