SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Ing. Marek Linder

RIEŠENIE ZLOŽITÝCH PROBLÉMOV POMOCOU EVOLUČNÝCH VÝPOČTOVÝCH METÓD

Autoreferát dizertačnej práce

Na získanie vedecko-akademickej hodnosti
philosophiae doctor, PhD.
v odbore doktorandského štúdia
9.2.7 Kybernetika
Bratislava 2013

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia na Ústave riadenia a priemyselnej informatiky, Fakulty elektrotechniky a informatiky, Slovenskej technickej univerzity v Bratislave.

Predkladateľ: Ing. Marek Linder
Ústav riadenia a priemyselnej informatiky
Fakulta elektrotechniky a informatiky, STU v Bratislave
Ilkovičova 3, 812 19 Bratislava

Školiteľ: doc. Ing. Ivan Sekaj, PhD.
Ústav riadenia a priemyselnej informatiky
Fakulta elektrotechniky a informatiky, STU v Bratislave
Ilkovičova 3, 812 19 Bratislava

Oponenti: Prof. RNDr. Jiří Pospíchal, DrSc.
Ústav aplikovanej informatiky
Fakulta informatiky a informačných technológií
Ilkovičova
842 16 Bratislava 4

prof. Ing. Pavel Ošmera, CSc.
Ústav automatizace a informatiky
Odbor aplikované informatiky
Fakulta strojního inženýrství
Vysoké učení technické v Brně
Technická 2896/2
616 69 Brno

Autoreferát bol rozoslaný dňa .....

Obhajoba dizertačnej práce sa koná ..... pred komisiou pre obhajobu dizertačnej práce v odbore doktorandského štúdia, vymenovanou predsedom spoločnej odborovej komisie dňa ....., číslo odboru: 9.2.7, odbor doktorandského štúdia: Kybernetika, na Ústave riadenia a priemyselnej informatiky, Fakulty elektrotechniky a informatiky, Slovenskej technickej univerzity, Ilkovičova 3, 812 19 Bratislava

prof. RNDr. Gabriel Juhás, PhD
dekan Fakulty elektrotechniky a informatiky
Slovenskej technickej univerzity v Bratislave

Obsah

Anotácia

Evolučné algoritmy predstavujú univerzálny stochastický prehľadávací prístup, využívaný nielen v technických odboroch, ale aj v ekonómii, v biológii, či v dizajne. V dnešnej dobe existuje veľa prístupov k zefektívneniu EA, jedným z nich je ich paralelizácia. Nejedná sa len o efektivitu z hľadiska rýchlosti algoritmu, ale aj o efektivitu z hľadiska výkonnosti algoritmu (schopnosti dosiahnuť globálny extrém) a to najmä v prípade zložitých úloh. Cieľom tejto dizertačnej práce bolo prispieť k zefektívneniu PEA, konkrétne PGA, ako aj navrhnúť vhodnú výpočtovú platformu pre realizáciu PEA.

Abstract

Evolutionary algorithms are universal stochastic search technique used not only in technical areas, but also in economy, biology or design. Nowadays exists lot of methods contributing to effectiveness of EA, one of them is parallelization. It is not only efficiency from the view of speed, but also efficiency from the view of performance (ability to reach global optimum), especially in case of complicated problems. The aim of this work is to contribute to effectiveness of PEA (PGA) and to design suitable computing platform.

Zoznam použitých skratiek

EA – evolučné algoritmy

GA – genetické algoritmy

PEA – paralelné evolučné algoritmy

PGA – paralelné genetické algoritmy

cPGA – celulárny paralelný genetický algoritmus

1 Úvod

Pri riešení praktických úloh sa človek už oddávna stretáva s problémom optimalizácie. V priebehu času teda človek vyvinul mnoho optimalizačných metód, z ktorých mnohé sú viac či menej špecializované na konkrétny typ úlohy. Viaceré z nich majú obmedzenú schopnosť nájsť globálny extrém, či uvoľniť sa z lokálneho optima, prípadne si ich nasadenie vyžaduje príliš dlhý čas optimalizácie, ktorý nie je z reálneho hľadiska prípustný. S uvedenými problémami sa dobre vysporiadavajú evolučné algoritmy (EA). A to najmä v prípade problémov, pre ktoré neexistuje špecializovaná optimalizačná metóda. Vo viacerých prípadoch sú schopné obstáť aj v porovnaní so špecializovanými optimalizačnými metódami. Prudký rozvoj EA ide ruka v ruke s rozvojom výkonu výpočtovej techniky.

Evolučné algoritmy zaraďujeme do skupiny moderných meta-heuristických optimalizačných a prehľadávacích metód, ktoré sú založené na princípoch prežitia najprispôsobivejších jedincov v zmysle Darwinovej teórie prirodzeného výberu. Ich základom je teda biologická evolúcia, schopnosť prispôsobiť sa okolitému prostrediu, boj o vhodného partnera pre reprodukciu a dedenie vhodnej informácie. Všetko vedie k prežitiu tých najschopnejších a k zániku tých, ktorí sa v priebehu času neosvedčili. Genetické algoritmy vychádzajúc z týchto princípov predstavujú jednu z najčastejšie využívaných evolučných výpočtových techník. Prvá myšlienka pochádza zo 60-tych rokov 20.storočia, publikoval ju I. Rechenberg vo svojej práci "***Evolution strategies***" (*Evolutionsstrategie* v origináli). Myšlienka bola ďalej rozvíjaná a obhajovaná najmä J. Hollandom a jeho spolupracovníkmi ako účinný nástroj na prehľadávanie definovaného priestoru. V súčasnosti genetické algoritmy predstavujú univerzálny stochastický prehľadávací prístup, využívaný nielen v technických odboroch, ale aj v ekonómii, v biológii, či v dizajne. Genetické algoritmy sa vyznačujú schopnosťou priblížiť sa v priestore prípustných riešení daného problému ku globálnemu optimu.

Vhodným objektom pre použitie evolučných algoritmov sú okrem iného technické optimalizačné a prehľadávacie aplikácie. Ich súčasťou býva vyhodnotenie modelu alebo počítačová simulácia riešeného objektu. Tieto výpočty sú však výpočtovo/časovo náročné a použitie EA vedie k dlhým, často extrémne dlhým výpočtovým časom. Naviac problémom býva aj kvalita dosiahnutých riešení. V dnešnej dobe existuje viac prístupov k zefektívneniu EA. Jedným z najefektívnejších je ich paralelizácia. Nejedná sa len o efektivitu z hľadiska času výpočtu, ale aj o efektivitu z hľadiska výkonnosti algoritmu (schopnosti dosiahnuť globálny extrém) a to najmä v prípade zložitých úloh. Existuje viacero prístupov k paralelizácii v EA resp. genetických algoritmov (GA), ako aj viacero ich klasifikácií.

Motiváciou pri tvorbe tejto práce bola snaha urýchliť optimalizačný proces pomocou EA vďaka rozloženiu výpočtu na viacero výpočtových jednotiek, ale aj vďaka návrhu nových algoritmov a novej vnútornej architektúre paralelného EA. Tým by sa malo dospieť urýchleniu a zlepšeniu konvergencie algoritmu k riešeniu úlohy a k zlepšeniu schopnosti uniknúť z lokálneho extrému a hľadať lepšie riešenia.

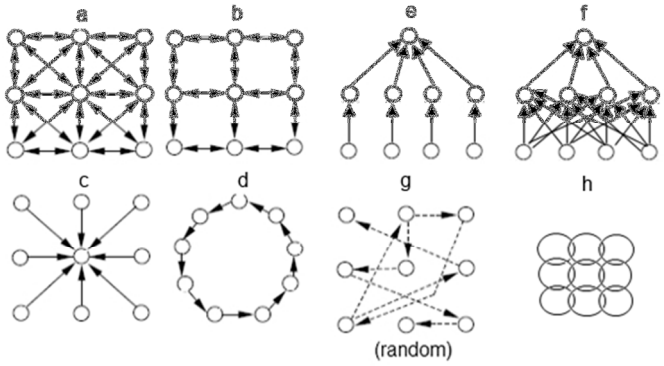
V kapitole 2 sa práca venuje prehľadu súčasného stavu PGA vo svete a tvorí teoretický podklad potrebný pre ostatné kapitoly. Kapitola 3 je venovaná použitým testovacím funkciám. V kapitole 4 je rozpracovaná analýza vlastností PGA a návrh ich nových architektúr, zároveň sa tu nachádza experimentálne porovnanie vybraných typov celulárnych PGA s reinicializáciou a bez nej. Na túto kapitolu nadväzujú kapitoly 5-6, kde sú celulárne PGA so štrukturovanou populácioua reinicializáciou použité na riešenie vybraných praktických úloh (optimalizácia pohybu robotického ramena, hľadanie optimálnej dráhy). Kapitola 7 je venovaná Analýze rozdielu

medzi jednopopulačným GA a celulárnym paralelným GA a návrhu upraveného jednopopulačného algoritmu. V kapitole 8 je navrhnutá GRID-ová výpočtová platforma pre PEA s využitím SQL databázy, experimentálne výsledky ako aj riešenie praktickej úlohy (Optimalizácia mechanicko - konštrukčnej úlohy). Vyhodnotenie výsledkov a prínosov práce sú zhrnuté v 9 kapitole.

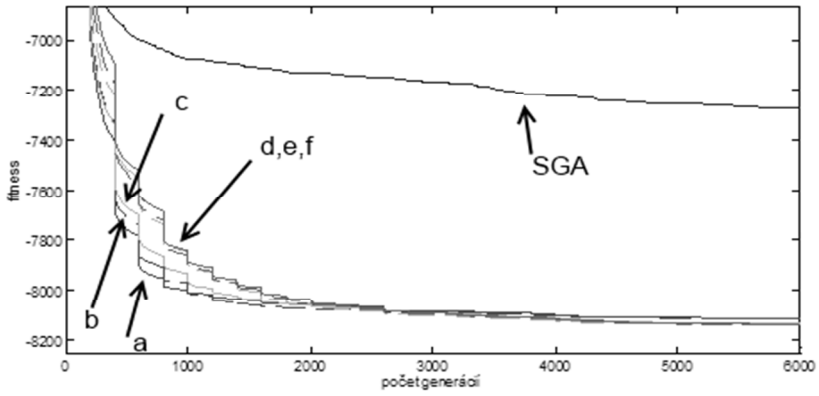
2 Analýza vlastností paralelných genetických algoritmov a návrh ich nových architektúr

2.1 Ostrovné PGA

Hrubozrnné GA nazývané aj ostrovné GA môžu priniesť zlepšenie a zrýchlenie konvergencie riešenej úlohy. Vývoj populácií prebieha v rámci ostrovov izolovane a priebeh genetického algoritmu v rámci populácie je rovnaký ako v SGA. Medzi jednotlivými populáciami, ale prebieha vo vhodných okamžikoch výmena informácií, ktorá prebieha pomocou prenosu jedinca (jedincov) medzi subpopuláciami. Možná je aj výmena informácie pomocou prekrývajúcich sa oblastí. Smer, hustota a počet migrovaných jedincov vplývajú výraznou mierou k výkonnosti algoritmu. Vzájomná komunikácia medzi ostrovmi vytvára akýsi „synergický efekt“, ktorý zvyšuje efektivitu týchto algoritmov. Príklady rôznych topológií ostrovných typov PGA sú na obrázku 2.1. Príklad priebehu fitness funkcie pri porovnaní rôznych architektúr sa nachádza na obrázku 2.2.



Obr. 2.1 Rôzne možné konfigurácie hrubozrnného/ostrovného PGA



Obr. 2.2 Priebeh fitness funkcie pri porovnaní rôznych architektúr (a-f z obrázku 2.1) hrubozrnného/ostrovného PGA a jednopopulačného GA (SGA) testovacia funkcia Eggholder pre 10 premenných (priemer z 30 opakovaní)

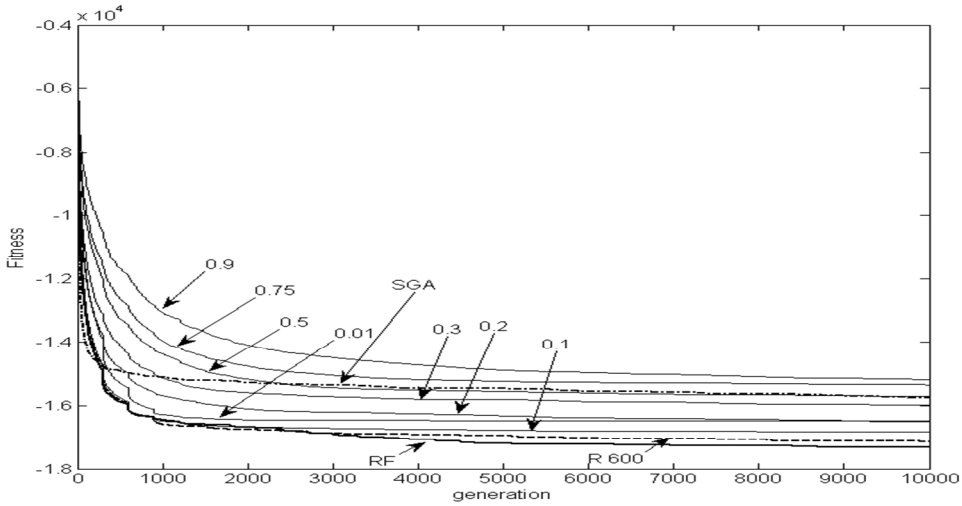
Reinicializácia v ostrovných PGA

V ostrovných PGA zreinicializovanie jedného ostrova neprináša negatívne dopady na celý GA, pretože populácia je rozdelená na ostrovy . Ako príklad môžeme uviesť porovnanie SGA a PGA napríklad so štruktúrou typu „a“ z obr. 2.1. Tento experiment porovnáva algoritmy s rôzne nastavenou mutáciou s algoritmami s pevne nastavenou mutáciou, ktoré v jednom prípade využívajú periodickú reinicializáciu po 600 generáciách výpočtu (R600), v druhom prípade reinicializáciu aplikovanú po stagnácii fitness (RF). Stagnácia fitness bola detegovaná pomocou plávajúceho priemeru diferencie fitness (2.1). Migrácia nastáva každých 200 generácií a vždy pred reinicializáciou ostrova.

$$diferencia(gen_i) = \frac{\sum_{gen_i}^{gen_{i-n}} (Fit(gen_i) - Fit(gen_{i-1}))}{n} \tag{2.1}$$

gen_i - i-ta generácia n – veľkosť okna

Reinicializácia priniesla pozitívny efekt a zlepšila priebeh konvergencie fitness, umožnila jednotlivým ostrovom dostať sa z lokálneho optima a tým sa vyhnúť predčasnej konvergencii (Obr. 2.3). Reinicializácia, ktorá bola riadená stagnáciou fitness (RF) mala ešte lepší prínos, pretože tento typ reinicializácia dokáže brať do úvahy aktuálny stav jednotlivých ostrovov.

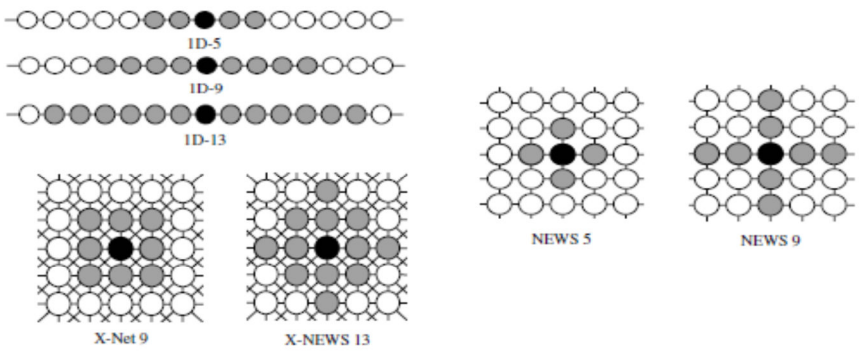


Obr. 2.3 Priebeh fitness, funkcia Eggholder, 20 premenných, priemer z 30 behov, porovnanie PGA s rôznymi mierami mutácie, RF- reinicializácia po stagnácii fitness R600 – periodická reinic.

2.2 Celulárne PGA

Ako už bolo spomenuté jemnozrnné **P**GA sa vyznačujú veľkým počtom relatívne malých subpopulácií, pričom najčastejšie jeden ostrov predstavuje jedného jedinca. Tento špeciálny prípad sa nazýva - Celulárne PGA. Informácia sa v tomto type algoritmu šíri postupne, iba pomocou kríženia s jedincami, ktorí sa nachádzajú v ich susedstve. Pohyb informácie teda pripomína difúziu. Vďaka tomu, že jedince sa môžu krížiť iba s tými, ktorí sa nachádzajú v ich okolí, nastáva v rámci populácie izolovanie jedincov vzdialenosťou. Podľa zvolenej topológie patrí každý jedinec do susedstva minimálne jedného ďalšieho jedinca, ale zvyčajne susedstvo obsahuje viacero jedincov.

Okolie bodu predstavujú jedinci, s ktorými sa môže aktuálny jedinec dostať do kontaktu. Táto oblasť môže nadobúdať rôzne tvary, napr. pre 2D prípad môžeme vyberať jedincov len v ortogonálnych smeroch, prípadne len po diagonále, kombináciou obidvoch spomenutých metód, podľa geometrickej vzdialenosti od centrálneho uzla alebo úplne inou metódou. Medzi najznámejšie tvary patria X-Net, X-News a News (Obr. 2.4) [27]. Ďalším parametrom je veľkosť týchto oblastí, môžu obsahovať len najbližších susedov, ale môže sa použiť aj väčší rozsah. Veľkosť susedstva a tým pádom počet možných partnerov je dôležitý parameter, ktorý ovplyvňuje rýchlosť šírenia informácie a tým kvalitu a rýchlosť konvergencie, ako aj možnosť uviaznutia v lokálnom extréme.



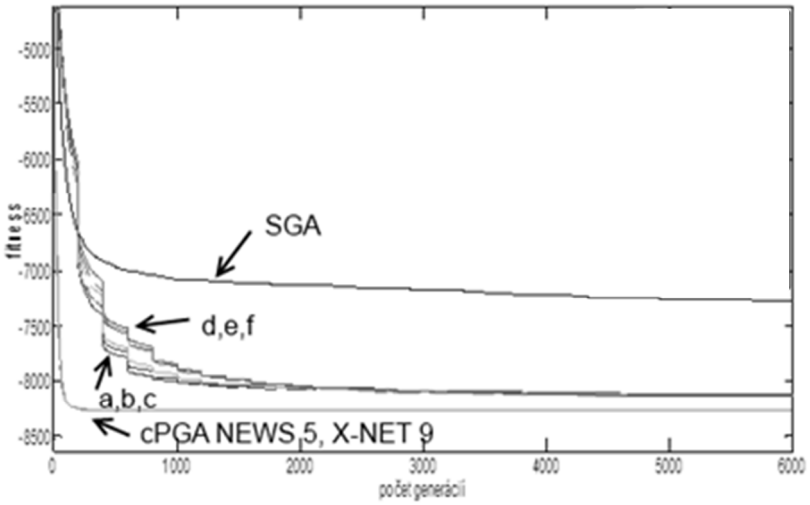
Obr. 2.4 Rôzne typy okolia, pre rôzne typy stratégií výberu suseda. Zdroj: [27]

V našich experimentoch využívame menšie typy okolia bodu a to: X-Net 9 a najmä News 5. Zároveň používame 2D architektúru s okrajmi. Ako spôsob výberu partnera do procesu kríženia využívame turnajový výber alebo náhodný výber. Ako stratégiu pre obnovu populácie sme použili synchrónne obnovenie mriežky (obnovou mriežky rozumieme zapísanie nových jedincov).

Celulárny PGA má v našom prípade nasledovný algoritmus:

- Výber suseda na kríženie
- Kríženie (jednobodové)
- Mutácia potomkov (mutácia génov v celom rozsahu, aditívna mutácia)
- Výber najlepšieho jedinca z množiny potomkov a pôvodného jedinca
- Zapísanie nového jedinca do mriežky pre nasledovnú generáciu

Pri porovnaní celulárneho PGA (cPGA) , ostrovného PGA (a-f) a jednoduchého GA (SGA) sme použili architektúry ostrovného PGA z kapitoly 2.1 (Obr. 2.1) a môžeme vidieť, že výkonnosť celulárneho PGA je pre tento typ úlohy výrazne vyššia. (Obr. 2.5).



Obr. 2.5 Priebeh fitness funkcie pri porovnaní výkonnosti celulárnych a hrubozrnných/ostrovných PGA, testovacia úloha Eggholder 10 premenných

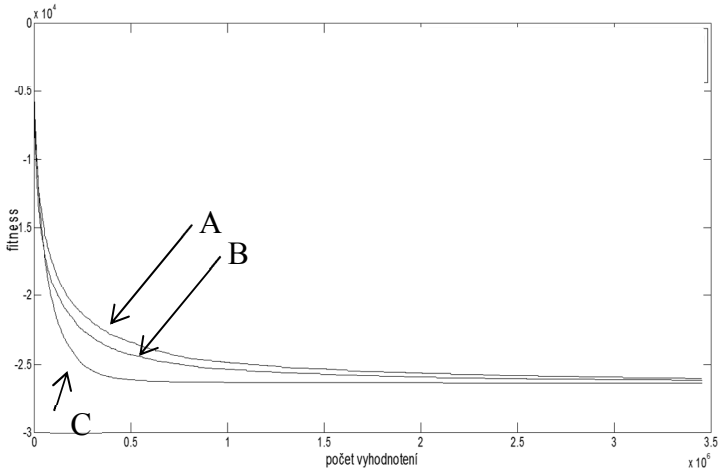
2.2.1 Celulárne PGA a počet jedincov po krížení

Na rozdiel od jednoduchého jednopopulačného GA a ostrovného PGA, kde pri procese kríženia vyberieme istý počet jedincov, skrížime ich a z tohto procesu získame rovnaký počet jedincov, v prípade celulárneho PGA, v ktorom každý ostrov predstavuje jedného jedinca, vyberieme dvoch rodičov, skrížime ich a na výstupe získame dvoch potomkov súperiacich o jedno miesto v nasledujúcej generácii. Môžeme použiť aj kríženie, ktorého výsledkom je iba jeden potomok [5] alebo môžeme podľa vopred určeného kritéria, prípadne náhodne vybrať jedného z potomkov. V prípade ak pri rozhodovaní uvažujeme hodnotu fitness nastáva otázka, ktorého potomka, vybrať, tak aby sme nestratili užitočnú informáciu. Ak sa rozhodneme vybrať obidvoch potomkov, znamená to, že počet vyhodnotení fitness narastie dvojnásobne, avšak dvojnásobný počet vyhodnotení nemusí vždy znamenať nevýhodu. V prípade, že dokážeme dosiahnuť lepší výsledok alebo ak dokážeme dosiahnuť výsledok porovnateľný, ale v polovičnom počte generácií, má táto stratégia výberu význam. S počtom jedincov, ktorých vyberieme do procesu vyhodnotenia fitness súvisí aj mutácia. V prípade, ak použijeme po krížení dvoch potomkov, môžeme vykonať mutáciu nad dvojnásobným počtom jedincov než sa nachádza v populácii. Existuje samozrejme aj možnosť vykonať mutáciu nad náhodne vybranými jedincami z populácie až po aktualizácii všetkých jedincov z populácie.

Pre porovnanie rôznych stratégií výberu potomkov po krížení sme použili tri konfigurácie:

- Po krížení sme vybrali potomka, ktorý obsahoval väčšiu časť génov z pôvodného jedinca, zmutovali sme ho a lepší z pomedzi pôvodného jedinca a potomka obsadil miesto v novej populácii – **Algoritmus A** (Obr.2.6)
- Po krížení sme vybrali potomka, ktorý obsahoval väčšiu časť génov z rodiča s lepším fitness, zmutovali sme ho a lepší z pomedzi pôvodného jedinca a potomka obsadil miesto v novej populácii [1] – **Algoritmus B** (Obr.2.6)
- Po krížení sme využili obidvoch potomkov, zmutovali sme ich a lepší z pomedzi pôvodného jedinca a potomkov obsadil miesto v novej populácii – **Algoritmus C** (Obr.2.6)

Na obrázku 2.6, je zobrazený priemer z 30 spustení troch hore uvedených stratégií výberu jedincov po krížení.



Obr. 2.6 Porovnaní rôznych stratégií výberu jedincov po krížení, testovacia úloha Eggholder 30 premenných

2.3 Reinicializácia v celulárnych PGA

Reinicializácia predstavuje perspektívny operátor v rámci paralelných genetických algoritmov. Jej použitie zvyšuje diverzitu populácie a tým umožňuje priniesť novú informáciu do GA, nové smery hľadania, prípadne umožňuje znovu objaviť informáciu už stratenú. Tieto dôsledky jej použitia dokážu pomôcť algoritmu dostať sa z lokálneho extrému. Stará populácia, ktorá stagnuje je nahradená novou náhodne vygenerovanou. Existuje viacero typov reinicializácie. Zvyčajne sa používa v ostrovných aj jednoduchých genetických algoritmoch. Jej aplikácia na celulárne algoritmy si vyžaduje niekoľko úprav. Na reinicializáciu sa môžeme pozrieť z niekoľkých pohľadov:

Prvým z nich je pohľad na podmienku uskutočnenia reinicializácie:

- Periodická reinicilizácia sa uskutočňuje po vopred stanovenom počte generácií bez ohľadu na priebeh fitness.
- Udalosťou inicializovaná reinicializácia sa uskutoční po splnení podmienky pre reinicializáciu. Môže to byť stagnácia fitness, pokles diverzity, afinity (afinita je podobnosť jedincov a je to jedna z charakteristík populácie [49]) medzi jedincami.

Ďalším pohľadom je rozsah reinicializácie:

- Celková reinicializácia – reinicializujeme celú populáciu (subpopuláciu).
- Čiastočná reinicializácia – reinicializujeme iba časť populácie (subpopulácie).
    - Jedince na reinicializáciu môžeme vybrať náhodne.
    - Jedince môžeme vybrať podľa zvoleného kritéria – podobnosť, diverzita, afinita.

Pri reinicializácii celulárnych PGA sa môžeme na reinicializáciu pozrieť ešte z pohľadu polohy reinicializovaných jedincov:

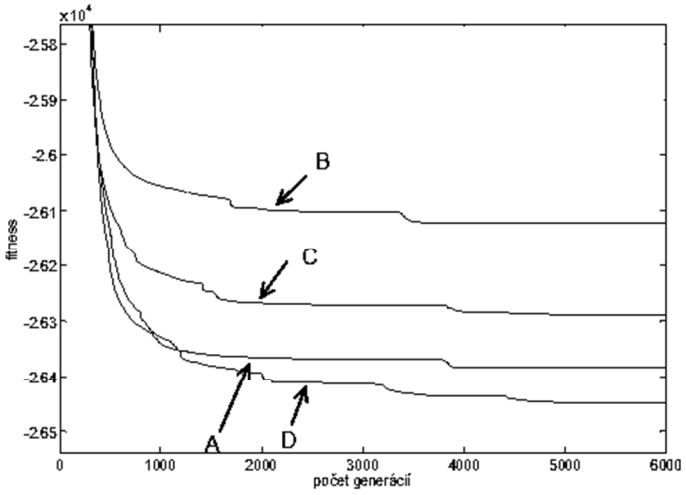
- Reinicializácia kompaktného bloku
- Reinicializácia náhodne vybraných jedincov

Vzhľadom na štruktúru celulárneho genetického algoritmu je pre nás nevýhodné reinicializovať celú populáciu. Ďalšou možnosťou je reinicializácia jednotlivých jedincov alebo istej skupiny jedincov v rámci mriežky. V prípade 2D mriežky nastáva po reinicializácii jednotlivcov ako aj oblasti jedincov rýchle ovládnutie reinicializovaných jedincov susednými, krátkodobo lepšími jedincami, ktorí neboli reinicializovaní. Jednou z možností ako zabrániť rýchlemu ovládnutiu reinicializovanej oblasti susednými jedincami a umožniť reinicializovaným jedincom vytvoriť stavebné bloky je dočasné izolovanie reinicializovanej oblasti. Táto izolácia musí byť dostatočne dlhá na to, aby sa mohli vytvoriť a presadiť stavebné bloky. V ďalšej fáze bude táto blokácia uvoľnená a vytvorené jedince si môžu vymieňať stavebné bloky so zvyškom populácie. Experimentálne porovnáme:

- celulárny PGA – Algoritmus A (Obr. 2.7),
- celulárny PGA s reinicializáciou náhodne vybraných jednotlivcov každých 300 generácií – Algoritmus B (Obr. 2.7),
- celulárny PGA s reinicializáciou bloku každých 300 generácií – Algoritmus C (Obr. 2.7),
- celulárny PGA s reinicilizáciou bloku každých 200 generácií, kde nasledovných 100 generácií je aplikovaná bariéra pre komunikáciu medzi reinicializovanými jedincami a zvyškom populácie – Algoritmus D (Obr. 2.7).

Rozmer mriežky je 24x24 jedincov, čiže celkovo 576, kríženie poskytuje dvoch potomkov, do nasledovnej generácie postupuje najlepší z trojice potomkovia + pôvodný jedinec. Počet reinicializovaných jedincov pre mriežku je 81 jedincov resp. blok o rozmere 9x9.

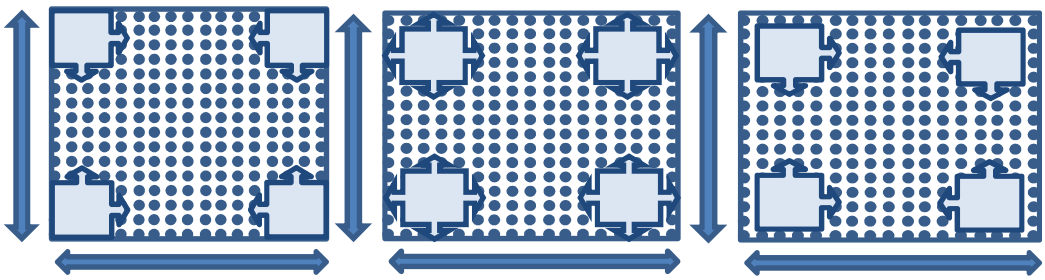
Aplikovanie reinicializácie bez aplikovania bariéry neprináša zlepšenie, dokonca môže degradovať výkon algoritmu. Aplikácia bariéry priniesla zlepšenie priebehu fitness funkcie. (Obr. 2.7). Pojem jemnozrnný v legende predstavuje celulárny PGA.



Obr. 2.7 Zobrazenie priebehu fitness pri porovnaní rôznych typov reincializácie, rozmer mriežky 24x24, testovacia úloha Eggholder 30 premenných

Evolúciou tohto riešenia je oblasť, ktorá je od zvyšku populácie oddelená polopriepustnou hranicou a umožňuje prienik informácie smerom z reinicializovanej oblasti v ktorejkoľvek generácii a pritom zamedzuje ovplyvňovaniu reinicializovaných jedincov jedincami mimo tejto oblasti. Toto riešenie umožňuje neohrozený prienik užitočnej informácie smerom k zvyšku populácie, čím

dokážeme zrýchliť a zlepšiť priebeh konvergencie. Zároveň tým zabránime vymretiu nádejných, ale krátkodobo horších jedincov, ktorý potrebujú čas a priestor, aby sa presadili. Táto oblasť nemusí byť v rámci mriežky jediná, mriežka môže obsahovať niekoľko oblastí, pričom každá z nich môže byť konfigurovaná iným spôsobom, prípadne sa môže reinicializovať v rozdielnych generáciách. Tieto oblasti sa môžu nachádzať kdekoľvek v rámci populácie, ich pozícia sa môže generovať náhodne alebo môže byť daná pevne. Jednou z možností, ako umiestniť reinicializované oblasti je ich umiestnenie v rohoch mriežky. Ďalej je možné ich umiestniť o „jedného jedinca“ od okraja, čím nám vznikne medzi reinicializovanou oblasťou a hranicou mriežky oblasť podobná 1D štruktúre. Na zvýraznenie tohto efektu 1D štruktúry je možné dokonca zakázať prienik informácie smerom z reinicializovanej oblasti do týchto miest. (Obr. 2.8) Pri takomto rozložení celulárneho PGA s reinicializovanými oblasťami a dostatočnej veľkosti mriežky a reinicializovaných oblastí nám vznikne kombinácia 1D a 2D štruktúry. Takáto forma algoritmu môže byť prínosná najmä ak nevieme s určitosťou povedať, či pri riešení problému bude výhodnejšia 1D alebo 2D štruktúra.



Obr. 2.8 Zobrazenie polopriepustných oblasti v rohoch, polopriepustných oblastí v rohoch vzdialené od okraja o „jedného jedinca“, polopriepustných oblastí v rohoch vzdialené od okraja o „jedného jedinca“

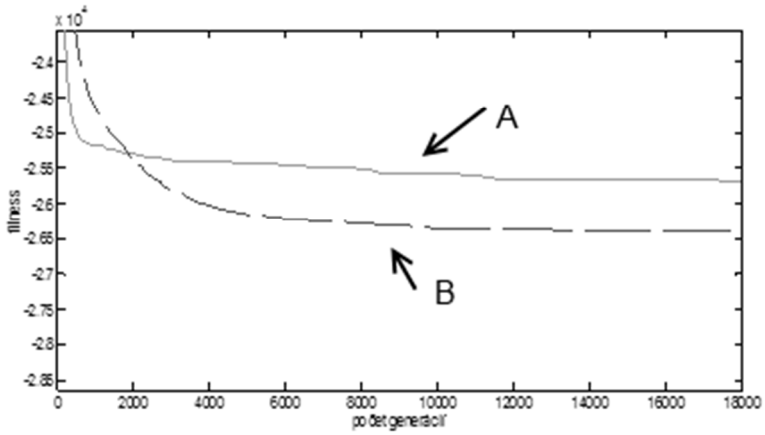
2.4 Experimentálne porovnania vybraných typov celulárneho PGA s reinicializáciou a bez reinicializácie

Pri použití reinicializácie a polopriepustných oblastí je dôležité uvažovať, v prípade že mutácia prebieha nad celou populáciou, či je potrebné udržať najlepšieho jedinca v rámci polopriepustnej oblasti alebo nie.

Teraz predpokladajme celulárny PGA nasledovným nastavením Algoritmus A:

- rozmer mriežky 16 x 16 jedincov (spolu 256 jedincov)
- kríženie poskytuje 2 potomkov
- mutácia prebehne nad oboma potomkami
- do nasledovnej generácie postupuje najlepší z trojice potomkovia + pôvodný jedinec.

Ďalej predpokladajme rovnaký algoritmus, ktorý obsahuje štyri polopriepustné oblasti o veľkosti 4x4 jedince v rohoch posunuté o jedného jedinca od kraja a reinicializáciou na základe uhla fitness. Algoritmus B. Z výsledkov pri použití testovacej úlohy Eggholder 30 je viditeľné zlepšenie konvergencie hodnoty fitness jednotlivých (Obr. 2.9).



Obr. 2.9 Zobrazenie porovnania priemerných hodnôt z 30 behov celulárneho PGA o rozmere bez reinicializovaných oblastí s celulárnym PGA s reinicializovanými oblasťami, testovacia úloha Eggholder 30 premenných

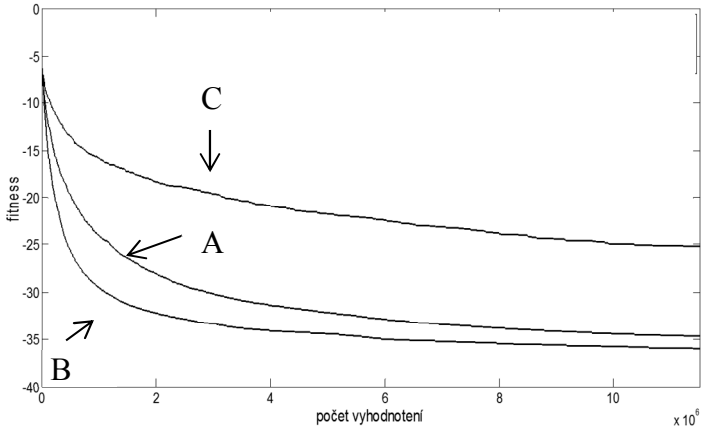
Ďalej predpokladajme celulárne PGA nasledovným nastavením:

a. celulárny, kde uvažujeme 2 potomkov – Algoritmus A (Obr. 2.10)
   - rozmer mriežky 24 x 24 jedincov (spolu 576 jedincov)
   - kríženie poskytuje 2 potomkov
   - mutácia prebehne nad oboma potomkami
   - do nasledovnej generácie postupuje najlepší z trojice potomkovia + pôvodný jedinec

b. algoritmus s rovnakou konfiguráciou, ktorý je ale vybavený štyrmi polopriepustnými oblasťami s rozmerom 4x4, ktoré sa reinicializujú na základe sklonu uhla fitness– Algoritmus B (Obr. 2.10)

c. celulárny PGA kde uvažujeme 1 potomka – Algoritmus C (Obr. 2.10):
   - rozmer mriežky 24 x 24 jedincov (spolu 576 jedincov)
   - kríženie poskytuje 1 potomka, podobnejšieho jedincovi s lepšou fitness
   - mutácia prebehne nad potomkom hneď po krížení
   - do nasledovnej generácie postupuje lepší z dvojice potomok + pôvodný jedinec

Z porovnania týchto algoritmov vychádza ako najúspešnejšia verzia s algoritmom, ktorý je ale vybavený štyrmi polopriepustnými oblasťami (Obr. 2.10).

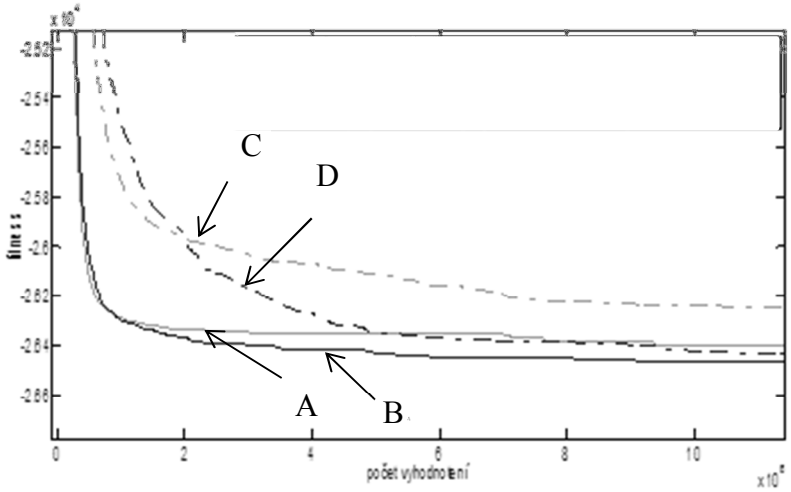


Obr. 2.10 Porovnanie priemerných hodnôt fitness z 30 behov rôznych konfigurácií celulárneho PGA. Testovacia úloha Morse 14 atómov (42 premenných)

Porovnajme štyri konfigurácie celulárneho PGA s rozmerom mriežky 24x24 jedincov kde kríženie poskytuje 2 potomkov a do nasledovnej generácie postupuje najlepší z trojice potomkovia + pôvodný jedinec:

a) mutácia nastane ihneď po krížení - Algoritmus A (Obr. 2.11)
b) mutácia nastane ihneď po krížení a mriežka obsahuje štyri reinicializované oblasti v rohoch, každá o veľkosti 6x6 jedincov. Oblasti sú posunuté od okraja o jedného jedinca s mechanizmom, ktorý zabraňuje mutácii najlepšieho jedinca v rámci reinicializovanej oblasti. - Algoritmus B (Obr. 2.11)
c) mutácia nastane až po vyhodnotení posledného jedinca v mriežke - Algoritmus C (Obr. 2.11)
d) mutácia nastane až po vyhodnotení posledného jedinca v mriežke a mriežka obsahuje štyri reinicializované oblasti v rohoch, každá o veľkosti 6x6 jedincov. Oblasti sú posunuté od okraja o jedného jedinca s mechanizmom, ktorý zabraňuje mutácii najlepšieho jedinca v rámci reinicializovanej oblasti. - Algoritmus D (Obr. 2.11)

Z porovnania týchto algoritmov vychádza, že pri funkcii Eggholder sa mutácia po vyhodnotení posledného jedinca v mriežke javí ako veľmi neperspektívna. Ale ak ju použijeme spolu s reinicializáciou, výkon algoritmu sa výrazne zlepší a v závere dosiahne podobný výsledok ako mutácia, ktorá prebieha okamžite po krížení a ktorá dosahuje od začiatku lepší výkon (Obr. 2.11).

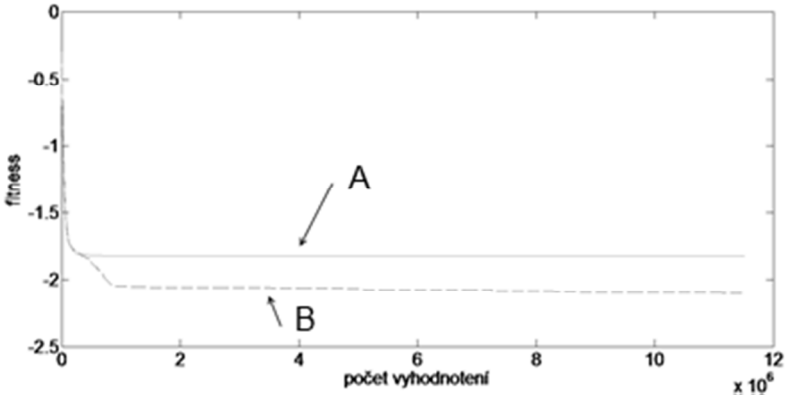


Obr. 2.11 Porovnanie hodnôt fitness pri testovacej funkcia Eggholder 30 premenných

Pri porovnaní:

a. celulárneho PGA s rozmerom 24x24, s dvoma potomkami z procesu kríženia a mutáciou hneď po krížení - Algoritmus A (Obr. 2.12)
b. celulárneho PGA s rozmerom 24x24, s dvoma potomkami z procesu kríženia a mutáciou hneď po krížení obsahujúceho štyri reinicializované oblasti o rozmere 6x6 jedincov v rohoch vzdialené od kraja mriežky o jedného jedinca. Algoritmus B (Obr. 2.12)

s využitím testovacej funkcie Fiveholes je vidieť, že reinicializované oblasti sú schopné prinášať nové riešenia a pomôcť algoritmu pri stagnácii v lokálnom extréme (Obr. 2.12).



Obr. 2.12 Zobrazenie porovnania priemerných hodnôt fitness z 30 behov celulárneho PGA bez reinicializovaných oblastí s celulárnym PGA so štyrmi reinicializovanými oblasťami o rozmere 6x6 jedincov, testovacia úloha Fiveholes 5 premenných

3 Návrh programovej GRID-ovej výpočtovej štruktúry pre PEA

Vzhľadom na to, že evolučné algoritmy patria medzi časovo/výpočtovo náročné optimalizačné metódy ako aj vzhľadom na náš zámer využívať pri optimalizácii (vyhodnotení fitness) rôzne typy simulačných softvérov (Matlab/Simulink, Adams, Ansys), v ktorých simulácie často zaberajú omnoho väčší čas, ako je beh samotného EA bolo pre nás potrebné vyvinúť vhodné riešenie, ktoré nám umožní skrátiť dobu behu algoritmu. Bežne je pomer výpočtového času medzi vyhodnotením fitness celej populácie a ostatnou réžiou PGA 10:1 a viac. Ako príklad môžeme uviesť výpočet optimalizácie parametrov PID regulátora (s pomocou jednopopulačného GA s populáciou o veľkosti 30 jedincov pri využití prostredia Matlab Simulink s periódou vzorkovania 0.001 sekundy). Vyhodnotenie fitness funkcie trvalo 99.86% času behu celého algoritmu. Našim cieľom bolo navrhnúť výpočtovú platformu pre výpočty v prostredí Matlab s prípadným využitím špecializovaných softvérov na vyhodnotenie fitness (Matlab/Simulink, Adams, Ansys), pomocou ktorej bude možné do výpočtu zapojiť rôzne počítače a pri tom využiť už existujúcu sieťovú infraštruktúru.

3.1 Grid s využitím SQL databázy

Po zvážení všetkých možností sme sa rozhodli vyvinúť vlastné riešenie na distribuované výpočty, ktoré bude zodpovedať našim potrebám. Z dôvodu čo najväčšej univerzálnosti sme zvolili Master-slave architektúru. V našom prípade komunikačnú vrstvu medzi Masterom a klientmi tvorí Microsoft SQL databáza a FTP server. Prínosom SQL databázy je jej schopnosť efektívne obslúžiť veľký počet požiadaviek, riešiť prípadné komunikačné konflikty ako aj spracovať veľké množstvo dát v relatívne krátkom čase. Okrem toho, jej využitie nám umožňuje efektívne riadenie komunikačnej záťaže. Medzi ďalšie výhody patrí jej škálovateľnosť, v prípade potreby existuje možnosť efektívne rozložiť komunikačnú záťaž na viac fyzických počítačov a neposlednom rade je možné využiť vstavané služby pre analýzu dát nad ešte bežiacim EA/PEA.

V rámci nášho gridu je možné spúšťať viacero experimentov nad jednou databázou, pričom Master procesy sa môžu nachádzať na jednom fyzickom počítači, alebo na rôznych fyzických počítačoch. Počet súčasne bežiacich experimentov je limitovaný len výkonom SQL databázy. Medzi ďalšie výhody riešenia s SQL databázou patrí to, že jediný viditeľný prvok s pevnou a zo všetkých počítačov viditeľnou IP adresou je SQL server. Medzi ostatnými počítačmi zapojenými do výpočtu nemusí existovať priama viditeľnosť a to ani zo strany SQL servera. Postačujúce je, že viditeľný je SQL server. Počítače sa môžu nachádzať za smerovačmi (routerami) s NAT, proxy serverom a môžu mať dynamicky prideľované IP adresy. Skratka NAT predstavuje spôsob úpravy sieťovej premávky cez smerovač, kde počítače nachádzajúce sa za smerovačom nie sú priamo dosiahnuteľné z IP adresy nachádzajúcej sa pred smerovačom.

Na dosiahnutie ešte vyššieho výkonu ako dosahuje bežná MS SQL sme sa rozhodli využiť RAM disk pre fyzické uloženie tabuliek databázy. RAM disk predstavuje virtuálny disk, nachádzajúci sa v pamäti RAM, ktorý sa pre operačný systém javí ako bežný pevný disk. Výhodou takéhoto riešenia sú veľmi rýchle časy čítania a zápisu dát na ňom uložených. Toto riešenie je podobné In-memory databázam.

3.2 Vlastná realizácia gridu

Matlab umožňuje viacero spôsobov spojenia s databázou. Jeden z asi najrozšírenejších a najpoužívanejších je využitie vlastného Database toolbox-u ktorý je multiplatformný. V našom riešení ale preferujeme Microsoft OLE konektor, ktorý je rýchlejší ako Matlab Database toolbox. Na tomto mieste treba uviesť, že je možné kombinovať klientov využívajúcich Matlab Database toolbox s klientmi využívajúcimi Microsoft OLE. Klienti môžu dostávať jedincov na vyhodnotenie fitness buď po jednom alebo ich môžu dostávať po blokoch, čím dokážeme znížiť čas potrebný na komunikáciu medzi SQL databázou a klientmi. Na jednom fyzickom počítači môžeme spustiť pri dostatočne veľkej RAM rovnaký počet Matlabov, ako je počet jadier. Navrhnutý Grid predstavuje modulárne riešenie, kde sa podľa požiadaviek používateľa môžu po drobných úpravách používať rôzne typy databáz MS SQL, MySQL, Oracle a pod. a klienti môžu byť použití na rôznych operačných systémoch Windows, Linux. V prípade použitia výpočtov pod OS Linux nie je možné použiť Microsoft OLE, ale je nutné využiť Matlab Database toolbox. My preferujeme využitie MS SQL a klientov využívajúcich Microsoft OLE.

3.3 Experimentálne výsledky

Na testovanie nami navrhnutého Gridu sme využili nasledovnú konfiguráciu master a klienti boli spustení na počítači s dvoma 4-jadrovými procesormi Opteron 2354 s 16 GB RAM, SQL databáza bola spustená na počítači s CPU Athlon 6000+ procesor s 2GB RAM. Obidva počítače boli pripojené do fakultnej siete s rýchlosťou 100Mb/s a nachádzali sa na rozdielnych miestach. K tomu, že je paralelné architektúra efektívnejšia ako sekvenčné vykonávanie programu musí byť dodržaná nasledovná rovnica [18]:

$$P = \frac{T_s}{T_p} > 1 \tag{3.1}$$

Kde P je faktor výkonnosti, T_s je čas vykonania EA na jednom jadre, T_p je čas vykonania programu pomocou paralelnej infraštruktúry. Testovací EA mal nasledovnú konfiguráciu: 50 jedincov, ale len 38 bolo vyhodnocovaných, zvyšní prechádzali nezmenení z predchádzajúcej generácie a ich fitness bola známa. Počet generácií bol 1000. Počet klientov (slave počítačov) v paralelnej konfigurácii bol 6.

Pre problém kde vyhodnotenie jedného jedinca trvalo 0.2293 sekundy sme dosiahli P=5.291 a pre problém, kde vyhodnotenie jedného jedinca trvalo 2.2101 sekundy sme dosiahli P=5.432. V ideálnom prípade by sa malo P rovnať počtu procesorov (v tomto prípade 6), ale číslo je nižšie z dôvodu nákladov na komunikáciu. Efektivitu paralelnej konfigurácie môžeme definovať nasledovne:

$$Efektivita = \frac{T_s}{T_p N} \times 100[\%] \tag{3.2}$$

Kde N je počet výpočtových uzlov. Efektivita pri riešení problému kde vyhodnotenie jedinca trvalo 0.2293 sekundy bola 88.194% a pre problém, kde vyhodnotenie jedného jedinca trvalo 2.2101 sekundy bola 90.537%. Ďalší test bol vykonaný na nasledovnej hardvérovej konfigurácii: Intel Core i7 860 procesor s 4 GB RAM, využili sme 4 klientov. Nastavenie genetického algoritmu bolo nasledovné: 50 jedincov, ale len 38 bolo vyhodnocovaných, zvyšní prechádzali nezmenení z predchádzajúcej generácie a ich fitness bola známa. Počet generácií bol 100.

Tab. 3.1: Efektivita Gridu

Čas vyhodnotenia jedného jedinca [s]	T_s [s]	T_p [s]	Efektivita [%]
0,1091493	414,9744387	119,5956316	86,74
0,2028549	770,6914000	218,1047329	88,33
1,0140683	3853,2000000	1021,3769092	94,31
5,0076530	19029,0000000	5017,0713304	94,82

V prípade, že $T_n << T_{fit}$ kde T_{fit} je počítaný na N výpočtových uzloch:

$$\lim_{T_{fit}\to\infty} \frac{T_s}{T_p} = \lim_{T_{fit}\to\infty} \frac{T_{GA} + T_{fit}}{T_{GA} + T_{fit} + T_n} = N \tag{8.3}$$

kde T_{GA} je čas genetického algoritmu bez vyhodnotenia fitness, T_{fit} je čas vyhodnotenia fitness a $T_n = T_{net} + T_{SQL}$, T_{net} je oneskorenie spôsobené komunikáciou po sieti a T_{SQL} je časové oneskorenie spôsobené spracovaním na SQL serveri.

Z tabuľky 3.1 je možné vidieť, že so znižovaním času potrebného na výpočet fitness sa znižuje efektivita výpočtu. Čo je však skutočnosť, s ktorou sa stretávame pri každej paralelnej výpočtovej architektúre.

Ako výhody nami navrhnutého gridového riešenia môžeme považovať:

- efektivita
- možnosť spúšťať paralelné experimenty (viacero experimentov naraz)
- multiplatformnosť
- možnosť použiť na realizáciu voľne dostupný softvér (okrem Matlabu) a vďaka tomu znížiť náklady
- eliminovanie úzkeho hrdla pri komunikácii s Masterom
- počítače nemusia byť navzájom priamo viditeľné (okrem databázy), funkcionalita aj za počítačmi nachádzajúcimi sa za NAT a proxy serverom
- využitie existujúcej sieťovej architektúry
- možnosť použitia rôznych typov EA/PEA
- možnosť použitia rôznych typov databáz
- možnosť pripojovať a odpojovať klientov počas behu experimentu
- možnosť využitia grafických kariet pri výpočte fitness nie je limitovaná architektúrou gridu ale je závislá iba od tvorcu fitness funkcie a použitých počítačov

4 Záver

Evolučné algoritmy sa v poslednom období stále viac uplatňujú pri riešení optimalizačných problémov. Existuje veľa prístupov k zefektívneniu EA, jedným z nich paralelizácia. Napriek dlhej histórii EA je ich paralelizácia vysoko aktuálna a pritom nedostatočne preskúmaná oblasť. Stále častejšie sa objavujú snahy o zrýchlenie konvergencie a zabránenie uviaznutia v lokálnom extréme ako aj snahy o zapojenie viacerých počítačov do výpočtu EA.

Jedným zo spôsobov ako prispieť k zefektívneniu PEA je reinicializácia. Tento v literatúre veľmi zriedkavo prezentovaný operátor dokáže prispieť ku zvýšeniu diverzity v PEA. Zvyčajne sa používa v kombinácii s ostrovnými PEA. Táto dizertačná práca prezentuje aplikáciu reinicializácie na celulárne PGA. Aplikácia reinicializácie si vyžiadala špecifický prístup a nebolo možné použiť postup bežne využívaný v ostrovnom PEA, kde sú jednotlivé populácie od seba izolované. Navrhli sme štruktúrovanie populácie na oblasti v rámci celulárneho PGA, ktoré spolu s polopriepustnou bariérou samo o sebe vedie k zvýšeniu diverzity. Vďaka kombinácii štruktúrovanej populácie populácie, polopriepustnej bariéry a reinicializácie, ktorá bola aktivovaná pomocou sklonu uhla fitness funkcie sme dokázali prispieť k zlepšeniu konvergencie vybraných testovacích úloh ako aj vybraných praktických úloh (Optimalizácia pohybu robotického ramena, hľadanie optimálnej dráhy).

Na základe rozdielov v rýchlosti konvergencie a dosiahnutých riešeniach medzi jednopopulačným GA a celulárnym GA sme navrhli experiment s testovacou úlohou, kde na dosiahnutie globálneho extrému stačí kombinácia dvoch vhodných jedincov. Na základe výsledkov sme navrhli upravený jednopopulačný GA. ktorý využíva niektoré prvky z celulárneho PGA (každý jedinec skríži s náhodne vybraným partnerom, potomkovia sa zmutujú a najlepší z trojice potomkovia + pôvodný jedinec postupuje do nasledovnej generácie). Pri testovaní s využitím rôznych testovacích funkcií bol upravený GA úspešnejší ako neupravený GA. Pri porovnaní s inými GA a PGA sa potvrdil vplyv No Free Lunch teorémy. Poradie rozdielnych algoritmov bolo pre rôzne testovacie úlohy rozdielne.

Pre urýchlenie výpočtov časovo náročných úloh pomocou PEA sme navrhli výpočtovú platformu pre Matlab využívajúcu SQL a FTP server. Je možné využitie Microsoft OLE alebo Matlab Database Toolbox. Táto platforma umožňuje dynamické pripájanie a odpájanie počítačov, využíva už existujúcu sieťovú infraštruktúru, dokáže eliminovať tzv. „Bottleneck“, je multiplatformná a jednotlivé počítače nemusia byť priamo viditeľné. Toto riešenie predstavuje v kombinácii s RAM diskom efektívny výpočtový Grid. Jeho činnosť a efektivitu sme overili pomocou testov ako aj pri riešení praktickej mechanicko-konštrukčnej úlohy. Využitie navrhnutého Gridu na riešenie PEA umožní efektívne riešenie časovo náročných úloh ako aj nových úloh, ktoré doteraz z časového hľadiska na našom pracovisku nemohli byť riešené.

Na záver môžeme skonštatovať, že sa nám podarilo prispieť k zefektívneniu GA resp. PGA. Aplikovali sme štruktúrovanie populácie na oblasti v rámci celulárneho PGA, reinicializáciu na celulárne PGA. Navrhli sme úpravy jednopopulačného GA, ktorý je úspešnejší ako jednopopulačný GA bez úprav a vytvorili sme efektívnu výpočtovú platformu, na ktorej sa pripravuje riešenie úloh aj z iných pracovísk nášho ústavu. Navrhnuté riešenia sme overili pri riešení praktických úloh. Táto práce predstavuje odrazový mostík pre ďalší výskum PEA/PGA na našom pracovisku.

5 Použitá literatúra

[1] Accelereyes Jacket (overené 10.7.2013) dostupný na: www.accelereyes.com

[2] Ansys - Systems & Multiphysics, 2012 ANSYS, Inc (overené 10.7.2013) dostupný na: http://www.ansys.com/

[3] Alba E. Dorronsoro, B., Cellular Genetic Algorithms, Springer-Verlag, ISBN 978-0-387-77609-5, 2008

[4] Alba E., Almeida F. , Blesa M. , Cabeza J. , Cotta C. , Diaz M. , Dorta I. , Gabarro J. , Leon C., Luna J. , Moreno L. , Pablos C. , Petit J., Rojas A. , Xhafa F., MALLBA: A library of skeletons for combinatorial optimisation, Euro-Par 2002 Parallel Processing Lecture Notes in Computer Science Volume 2400, 2002, pp 927-932

[5] Alba E., Dorronsoro B., The Exploration/Exploitation Tradeoff in Dynamic Cellular Genetic Algorithms, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 9, NO. 2, APRIL 2005

[6] Alba E., Dorronsoro B: Cellular Genetic Algorithms, Operations Research/Computer Science Interfaces Series. Vol. 42, e-ISBN: 978-0-387-77610-1, (2008)

[7] Alba E., Giacobini M., Tomassini M., Romero S.: Comparing Synchronous and Asynchronous Cellular Genetic Algorithms. In Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN VII), (2002)

[8] Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation, vol.6, no.5, pp. 443- 462, (Oct 2002)

[9] Alba E., Troya J. M.: A survey of parallel distributed genetic algorithms. Complexity, vol. 4, no. 4, pp.31 – 52, (1999)

[10] Alba, E., Luque, G. and Nesmachnow, S. (2013), Parallel metaheuristics: recent advances and new trends. International Transactions in Operational Research, 20: 1–48

[11] Arenas M. G., Collet P., Eiben A. E., Jelasity M., Merelo J. J., Paechter B., Preuß M., a Schoenauer M. 2002. A Framework for Distributed Evolutionary Algorithms. In Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN VII), Juan J. Merelo Guervós, Panagiotis Adamidis, Hans-Georg Beyer, José Luis Fernández-Villacañas Martín, and Hans-Paul Schwefel (Eds.). Springer-Verlag, London, UK, UK, 665-675

[12] Bazterra V. E., Cuma M., Ferraro M. B., and Facelli J. C. 2005. A general framework to understand parallel performance in heterogeneous clusters: analysis of a new adaptive parallel genetic algorithm. J. Parallel Distrib. Comput. 65, 1 (January 2005), 48-57.

[13] Cahon S., Melab N., Talbi E.-G. ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics, Journal of Heuristics, 10: 357–380, 2004

[14] Cantú-Paz E., Goldberg D. E.: Efficient Parallel Genetic Algorithms: Theory and Practice. Computer Methods in Applied Mechanics and Engineering, (2000)

[15] Cantú-Paz E., Goldberg D. E.: Parallel genetic algorithms with distributed panmitic population. IlliGAL Report No. 99006, (1999)

[16] Cantú-Paz E.: A summary of research on parallel genetic algorithms. IlliGAL Report No. 95007, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, (1995)

[17] Cantú-Paz E.: A Survey of Parallel Genetic Algorithms. Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, (1998)

[18] Cantú-Paz E.: Designing Efficient And Accurate Parallel Genetic Algorithms. IlliGAL Report No. 95017, (1999)

[19] Cantú-Paz E.: Implementing fast and flexible parallel genetic algorithms. in Practical handbook of genetic algorithms, (1999)

[20] CrystaDiskMark dostupný na http://crystalmark.info/?lang=en

[21] Dorronsoro B., Alba E., Giacobini M., Tomassini M.: The influence of grid shape and asynchronicity on cellular evolutionary algorithms, Proccedings on the CEC 2004, (2004)

[22] Dorronsoro, B.; Bouvry, P., "Adaptive Neighborhoods for Cellular Genetic Algorithms," Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on , vol., no., pp.388,394, 16-20 May 2011
[23] Dorronsoro B., Arias D., Luna F., Nebro A.J., a Alba E.. A grid-based hy-brid cellular genetic algorithm for very large scale instances of the CVRP. In Waleed W. Smari, editor,High Performance Computing & Simulation Confer-ence (HPCS), pages 759–765, 2007
[24] Dubreuil M., Gagne C., a Parizeau M. 2006. Analysis of a master-slave architecture for distributed evolutionary computations. Trans. Sys. Man Cyber. Part B 36, 1 (February 2006), 229-235
[25] ECJ 21, A Java-based Evolutionary Computation Research System by S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, R.l Kicinger, E. Popovici, K.Sullivan, J. Harrison, J. Bassett, R. Hubley, A. Desai, A. Chircop, J. Compton, W. Haddon, S. Donnelly, B. Jamil, J. Zelibor, E. Kangas, F. Abidi, H. Mooers, J. O'Beirne, K. A. Talukder, and J. McDermott dostupné na http://cs.gmu.edu/~eclab/projects/ecj/
[26] Eiben E. A., Schoenauer M., Laredo J. L. J., Castillo P. A., Mora A. M., a Merelo J. J.. 2007. Exploring selection mechanisms for an agent-based distributed evolutionary algorithm. In Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation (GECCO '07). ACM, New York, NY, USA, 2801-2808
[27] Eklund S. E.: A massively parallel architecture for distributed genetic algorithms. Parallel Comput. 30, 5-6 May 2004, 647-676, (2004)
[28] Giacobini M., Tomassini M., Tettamanzi A.G.B., Alba E.: *Selection intensity in cellular evolutionary algorithms for regular lattices*, IEEE Transactions on Evolutionary Computation, 2005
[29] Goga V., Linder M., Structural shape optimization using genetic algorithms – zatiaľ nepublikované
[30] Goldberg David E.. 1989. Genetic Algorithms in Search, Optimization and Machine Learning (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[31] Golub M., Jakobovic D.: A new model of global parallel genetic algorithm. Information Technology Interfaces, 2000. ITI 2000. Proceedings of the 22nd International Conference on , vol., no., pp.363-368, 16-16, (June 2000)
[32] Herrera F., Lozano M.: Gradual distributed real-coded genetic algorithms. IEEE Transactions on Evolutionary Computation, vol.4, no.1, pp.43-63, (Apr 2000)
[33] Hisao Ishibuchi, Ken Ohara, Yusuke Nojima: Implementation of Elitism in Cellular Genetic Algorithms. Department of Computer Science and Intelligent Systems Graduate School of Engineering, Osaka Prefecture University
[34] Homberger J. A Two-Level Parallel Genetic Algorithm for the Uncapacitated Warehouse Location Problem, Proceedings of the 41st Hawaii International Conference on System Sciences – 2008
[35] Chalermwat Prachya, El-Ghazawi Tarek A., LeMoigne Jacquline: GA-Based Parallel Image Registration on Parallel Clusters. In Proceedings of the 11 IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing
[36] Imade H., Morishita R., Ono I., Ono N., a Okamoto M.. 2004. A grid-oriented genetic algorithm framework for bioinformatics. New Gen. Comput. 22, 2 (January 2004), 177-186
[37] ImDisk Virtual Disk Driver (overené 10.7.2013) dostupný na: http://www.ltr-data.se/opencode.html/
[38] Janson S., Alba E., Dorronsoro B., a Middendorf M. Hierarchical cellular genetic algorithm. In J. Gottlieb and G.R. Raidl, editors,Evolutionary Com-putation in Combinatorial Optimization (EvoCOP), volume 3906 of Lecture Notes in Computer Science (LNCS), pages 111–122, Budapest, Hungary, April 2006. Springer-Verlag, Heidelberg
[39] Knysh D, Kureichik V.: Parallel genetic algorithms: a survey and problem state of the art. Journal of Computer and Systems Sciences International, (2010)

[40] Kohlmorgen U., Schmeck H., Haase K.: Experiences with Fine-Grained Parallel Genetic Algorithms. Annals of Operations Research, (1996)
[41] Koumousis, V.K., Katsaras, C.P., A Saw-Tooth Genetic Algorithm Combining the Effects of Variable Population Size and Reinitialization to Enhance Performance, Evolutionary Computation, IEEE Transactions on , vol.10, no.1, pp.19,28, Feb. 2006
[42] Králik J. Modelovanie konštrukcií v metóde konečných prvkov – Systém Ansys STU Bratislava 2009
[43] Lim D., Ong Yew-Soon, Jin Yaochu, Sendhoff B., Lee Bu-Sung. Efficient Hierarchical Parallel Genetic Algorithms using Grid computing, Future Generation Computer Systems 23 (2007) 658–670
[44] Linder, Marek - Pernecký, Daniel - Sekaj, Ivan: Grid Computing in Matlab for Solving Evolutionary Algorithms. In: Technical Computing Bratislava 2012 [elektronický zdroj] : 20th Annual Conference Proceedings. Bratislava, 7.12. 2012. - Bratislava : RT Systems, 2012. - ISBN 978-80-970519-4-5. - CD-ROM, [6] s.
[45] Luque G., Alba E., Parallel Genetic Algorithms. Theory and Real World Applications, Springer-Verlag, ISBN 978-3-642-22083-8, July 2011
[46] Nowostawski M., Poli R. : Parallel Genetic Algorithm Taxonomy. In KES'99
[47] NVIDIA's Next Generation CUDA™ Compute Architecture: Fermi, Whitepaper (overené 10.7.2013) dostupný na: www.nvidia.com/
[48] Opencl-toolbox - OpenCL toolbox for Matlab (overené 10.7.2013) dostupný na: http://code.google.com/p/opencl-toolbox/
[49] Oravec M.: Návrh evolučných algoritmov, Dizertačná práca. FEI STU Bratislava, (2011)
[50] Pan, J.-S.; McINNES, F.R.; Jack, M.A., "VQ codevector index assignment, using genetic algorithms for noisy channels," Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on , vol.1, no., pp.295,298 vol.1, 3-6 Oct 1996
[51] Pernecký D. Paralelné evolučné algoritmy Bakalárska práca, 2012
[52] Pinel, F.; Dorronsoro, B.; Bouvry, P., "A New Parallel Asynchronous Cellular Genetic Algorithm for de Novo Genomic Sequencing," Soft Computing and Pattern Recognition, 2009. SOCPAR '09. International Conference of , vol., no., pp.178,183, 4-7 Dec. 2009
[53] Pospíchal P., Jaros J., and Schwarz J. EvoApplications 2010, Part I, LNCS 6024, pp. 442–451, 2010.
[54] Sarma J., De Jong K.: An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms. To appear in Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN96), Sept. 22-26, Berlin, Germany, (1996)
[55] Sarma J., De Jong K.: An Analysis of Local Selection Algorithms in a Spatially Structured. Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kaufmann, (1997)
[56] Sekaj I., Oravec M.: Paralelné evolučné algoritmy. V zborníku príspevkov V.Kvasnička a kol.: Umelá inteligencia a kognitívna veda, STU, (2011) Lin S.Ch., Punch W., Goodman E.: *Coarse-grain parallel genetic algorithms: Categorization and new approach*, IEEE Symposium on Parallel and Distributed Processing,1994
[57] Sekaj I., Perkacz J.: Some Aspects of Parallel Genetic Algorithms with Population Re-initialization. In CEC, Singapore, (2007)
[58] Sekaj I.: Robust Parallel Genetic Algorithms with Re-Initialisation. In PPSN VIII, September 18-22, Birmingham,(2004)
[59] Sekaj, Ivan - Linder, Marek - Pernecký, Daniel: Experimental Comparison of Selected Types of Parallel Evolutionary Algorithms. In: ECTA 2011 and FCTA 2011 : Proceedings of the International Conference on Evolutionary Computation Theory and Applications and International Conference on Fuzzy Computation Theory and Applications. Paris, France, 24-26 October, 2011. - : Springer, 2011. - ISBN 978-989-8425-83-6. - S. 296-302

[60] Sekaj, Ivan - Linder, Marek: Population Reinitialisation in Parallel Genetic Algorithms. In: MENDEL 2012 : 18th International Conference on Soft Computing. June 27-29, 2012, Brno, Czech Republic. - Brno : University of Technology, 2012. - ISBN 978-80-214-4540-6. - S. 13-19
[61] Schwehm M.: Parallel Population Models for Genetic Algorithms. (1996)

[62] Shyh-Chang Lin; Punch, W.F., III; Goodman, E.D., "Coarse-grain parallel genetic algorithms: categorization and new approach," Parallel and Distributed Processing, 1994. Proceedings. Sixth IEEE Symposium on , vol., no., pp.28,37, 26-29 Oct 1994
[63] Sifa Zhang, Zhenming He ,Implementation of Parallel Genetic Algorithm Based on CUDA, Advances in Computation and Intelligence Lecture Notes in Computer Science Volume 5821, 2009, pp 24-30
[64] Simoncini D., Collard P., Verel S., Clergue M.: From cells to islands: An unified model of cellular parallel Genetic Algorithms. hal-00164669, version 1, (2008)
[65] Skolicki Z, De Jong K.: The influence of migration sizes and intervals on island models. In Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO '05), Hans-Georg Beyer (Ed.). ACM, New York, NY, USA, 1295-1302., (2005)
[66] ÚVOD DO GRIDOVÉHO POČÍTANIA (cit. 10.7.2013) dostupný na: http://www.sivvp.sk/vypoctove-prostriedky/clusterove-pocitanie/
[67] Vatanutanon J., Noman N., a Iba H. 2011. Polynomial selection scheme with dynamic parameter estimation in cellular genetic algorithm. In Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO '11), Natalio Krasnogor (Ed.)
[68] Whitley D., Rana S., Heckendorn R. B.: *The island model genetic algorithm: On separability, population size and convergence*, in Journal of Computing and Information Technology, 7(1), 1999, pp.33-47.
[69] Xue Shengjun, Guo Shaoyong, Bai Dongling: The Analysis and Research of Parallel Genetic Algorithm. Wireless Communications, Networking and Mobile Computing, 2008. 4th International Conference WiCOM '08, vol., no., pp.1-4, 12-14, (Oct. 2008)
[70] Zelinka I., Oplatkova Z., Šeda M., Ošmera P., Včelař F.: Evoluční výpočetní techniky - Principy a aplikace. BEN, (2009)

Publikácie autora:

Kajan, S., Dideková, Z., Kozák, S., Linder, M.: Neural-genetic control algorithm of nonlinear systems, International Review of Automatic Control 6 (2) , pp. 206-210, ISSN: 19746059 (2013)

Linder, Marek - Sekaj, Ivan: Parallel Genetic Algorithms. In: MENDEL 2011 : 17th International Conference on Soft Computing. June 15 - 17, 2011, Brno, Czech Republic. - Brno : University of Technology, 2011. - ISBN 978-80-214-4302-0. - [7]

Linder, Marek - Kajan, Slavomír - Hypiusová, Mária: Supervisory Control system Realized Using OPC Server and Genetic Algorithm. In: Process Control 2012 : 10th International Conference. Kouty nad Desnou, Czech Republic, June 11-14, 2012. - Pardubice : University of Pardubice, 2012. - ISBN 978-80-739550-07. - Art.No. 045c

Sekaj, Ivan - Linder, Marek - Pernecký, Daniel: Experimental Comparison of Selected Types of Parallel Evolutionary Algorithms. In: ECTA 2011 and FCTA 2011 : Proceedings of the International Conference on Evolutionary Computation Theory and Applications and International Conference on Fuzzy Computation Theory and Applications. Paris, France, 24-26 October, 2011. - : Springer, 2011. - ISBN 978-989-8425-83-6. - S. 296-302

Sekaj, Ivan - Linder, Marek: Population Reinitialisation in Parallel Genetic Algorithms. In: MENDEL 2012 : 18th International Conference on Soft Computing. June 27-29, 2012, Brno, Czech Republic. - Brno : University of Technology, 2012. - ISBN 978-80-214-4540-6. - S. 13-19

Linder, Marek - Pernecký, Daniel - Sekaj, Ivan: Grid Computing in Matlab for Solving Evolutionary Algorithms. In: Technical Computing Bratislava 2012 [elektronický zdroj] : 20th Annual Conference Proceedings. Bratislava, 7.12. 2012. - Bratislava : RT Systems, 2012. - ISBN 978-80-970519-4-5. - CD-ROM, [6] s.

Linder, Marek - Sekaj, Ivan: Parallel Genetic Algorithms. In: ELITECH´11 : 13th Conference of Doctoral Students Faculty of Electrical Engineering and Information Technology. Bratislava, Slovak Republic, 17 May, 2011. - Bratislava : Nakladateľstvo STU, 2011. - ISBN 978-80-227-3500-1. - S. 1-4

Linder, Marek - Kajan, Slavomír: Remote Control of Dynamical System Using OPC Server. In: ELITECH´12 [elektronický zdroj] : 14th Conference of Doctoral Students. Bratislava, Slovak Republic, 22 May 2012. - Bratislava : Nakladateľstvo STU, 2012. - ISBN 978-80-227-3705-0. - CD-ROM, [5] s.

Sekaj, Ivan - Linder, Marek: Evolutionary Algorithm Based Power Plant Working Point Optimisation Using PLC and Simulink Model. In: Technical Computing Bratislava 2010 : 18th Annual Conference Proceedings. Bratislava, Slovak Republic, 20.10.2010. - Bratislava : RT Systems, 2010. - ISBN 978-80-970519-0-7. - CD-Rom