REALIZÁCIA PARALELNÝCH GENETICKÝCH ALGORITMOV V MATLABE POMOCOU PARALLEL COMPUTING TOOLBOXU

S.Kajan, I.Sekaj, M.Oravec

Ústav riadenia a priemyselnej informatiky, Fakulta elektrotechniky a informatiky,
Slovenská technická univerzita v Bratislave, Slovenská republika

Abstrakt

Článok opisuje využitie Parallel Computing toolboxu v prostredí Matlab pri realizácii výpočtu paralelných genetických algoritmov (PGA), ktoré sú optimalizačným prístupom. Tieto algoritmy sú počítané vo viacerých izolovaných uzloch, ktoré bežia na viacerých počítačoch. Komunikácia medzi týmito uzlami môže byť definovaná pomocou rôznych architektúr. V článku je demonštrovaná aplikácia PGA pri návrhu parametrov rozvetveného regulačnej obvodu. Regulačný priebeh, ktorý je súčasťou optimalizovanej účelovej funkcie je simulovaný v prostredí Simulink.

1 Úvod

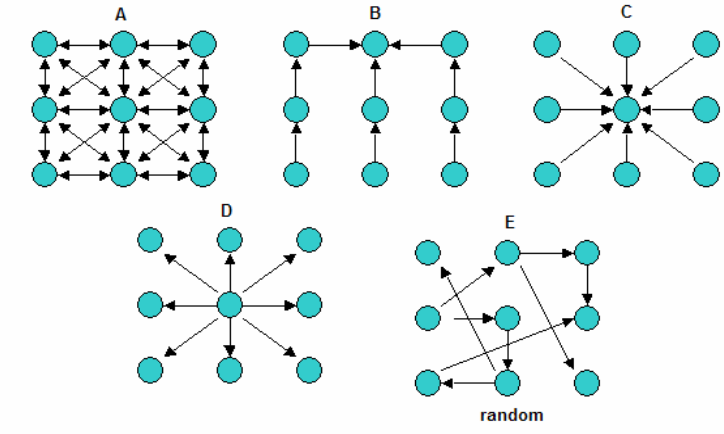
Genetické algoritmy (GA) sú výkonný stochastický optimalizačný resp. prehľadávací prístup, ktorý napodobňuje evolúciu v prírode za účelom riešenia praktických problémov. Sú od svojej podstaty paralelným algoritmom, nakoľko riešenie hľadajú súčasne na celej skupine – populácii jedincov. Napriek tomu sa pri výpočte veľmi zložitých úloh stáva, že riešenie uviazne v lokálnom extréme (v suboptimálnom riešení). Okrem toho, doba výpočtu môže byť príliš dlhá, čo je charakteristickou vlastnosťou GA. Uvedené nedostatky je možné odstrániť ďalšou úrovňou ich paralelizácie – paralelnými GA (PGA). Takáto paralelizácia zvyšuje diverzitu génov (hľadaných prvkov, parametrov) v populácii potenciálnych riešení v rámci celého PGA. Zvyšovaním diverzity sa znižuje pravdepodobnosť uviaznutia riešenia v lokálnom extréme, teda naopak zvyšuje sa pravdepodobnosť nájdenia globálneho extrému. Nevýhodou je zvýšená výpočtová náročnosť a teda aj narastajúci výpočtový čas. Toto je možné kompenzovať alebo úplne eliminovať rozložením výpočtového výkonu na viac počítačov. Možným riešením PGA na viacerých počítačoch v prostredí Matlab je využitie Parallel Computing toolboxu. V našom prípade bola optimalizácia parametrov rozvetveného regulačného obvodu pomocou PGA realizovaná v prostredí Matlab ver. 7.3, kde boli k dispozícii knižnice Distributed Computing a Matlab Distributed Computing Engine ver. 3.0. [6, 7].

2 Princíp PGA

Genetický algoritmus (algoritmus v jednom uzle PGA) pozostáva z týchto krokov: 1. inicializácie počiatočnej populácie, 2. vyčíslenie účelovej funkcie pre všetkých jedincov populácie, 3. výber jedincov na modifikáciu, 4. mutácia a kríženie vybraných jedincov, 5. tvorba novej populácie, skok na bod 2. alebo koniec.

Základným princípom paralelných GA je súčasné zbiehanie viacerých subpopulácií (uzlov) – relatívne izolovaných GA, ktoré si navzájom vo vhodne zvolených okamihoch, zvoleným spôsobom vymieňajú informácie [1, 3]. Každá subpopulácia je počítaná svojím vlastným GA. Po istom počte generácií si jednotlivé subpopulácie na základe zvolených väzieb vymenia svoje priebežne najlepšie riešenia, potom opäť pokračujú izolovane ďalej. Táto výmena genetickej informácie sa nazýva „*migrácia*“. Migrácia môže byť jednosmerná alebo obojsmerná a môže nastať medzi dvojicami alebo skupinami subpopulácií. Okamihy migrácie nemusia byť periodické, ale môžu byť vyvolané splnením vhodných podmienok, alebo dokonca môžu byť náhodné. Štruktúra migračných väzieb medzi subpopuláciami môže byť rôzna, pričom nemusia byť prevázbené všetky subpopulácie navzájom. Ich

štruktúra môže byť statická ale aj dynamická. [2, 5]. Rôzne architektúry väzieb medzi subpopuláciami sú zobrazené na obr.1. Každý krúžok predstavuje jednu subpopuláciu s vlastným GA.



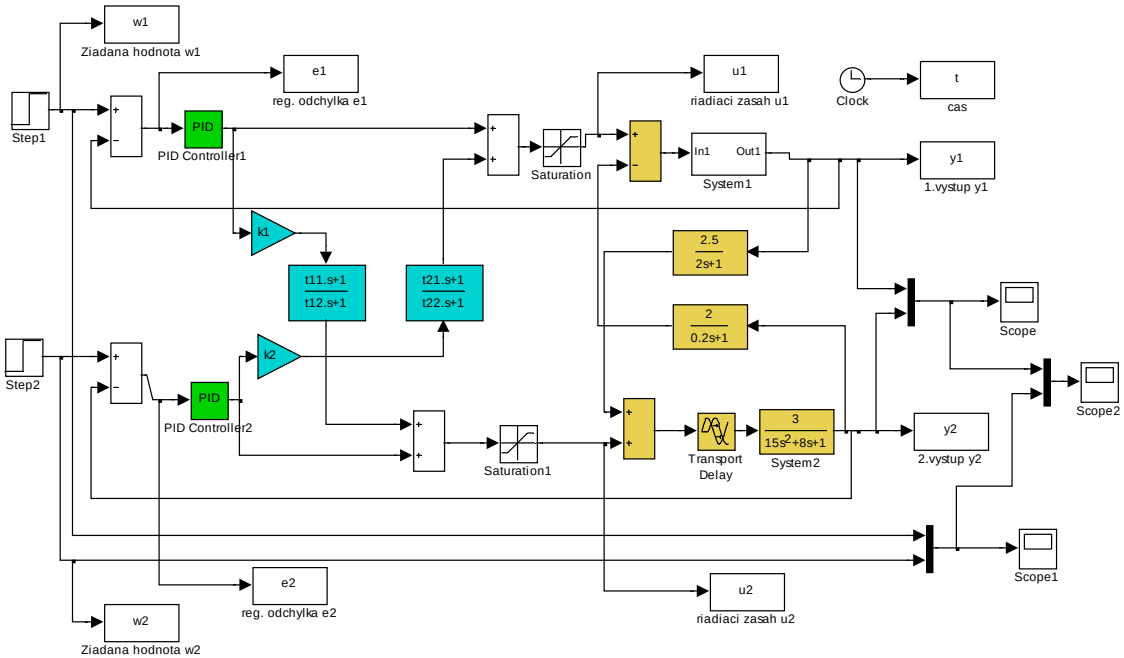
Obr.1: Rôzne architektúry PGA

3 Návrh rozvetveného regulačného obvodu pomocou PGA

Cieľom aplikácie PGA bol návrh parametrov rozvetveného regulačného obvodu. Jednalo sa o optimalizáciu parametrov dvoch PID regulátorov systému s 2 vstupmi a 2 výstupmi a 2 korekčných členov, ktorých úlohou je eliminácia vplyvu vzájomných interakcií oboch výstupov. K účelu optimalizácie bola navrhnutá kriteriálna funkcia, pozostávajúca zo sumy absolútnych hodnôt regulačných odchýliek e_1, e_2 a váhovaných derivácií výstupov systému dy_1, dy_2

$$J = \sum_{k=1}^{Nsim} \left(|e1_k| + |e2_k| + \alpha |dy1_k| + \alpha |dy2_k| \right) \tag{1}$$

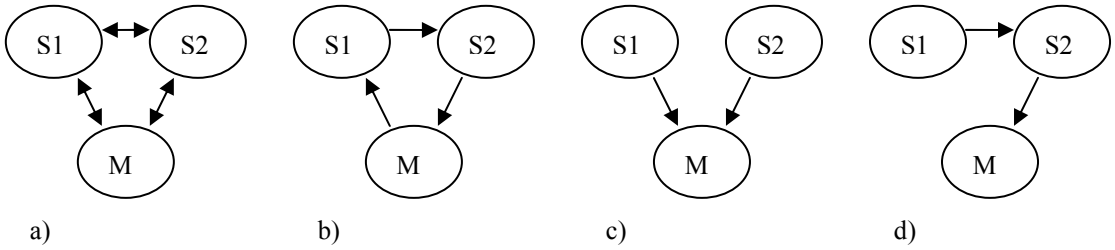
Táto bola minimalizovaná pomocou GA. Na každé vyčíslenie kriteriálnej funkcie je potrebné vykonať simuláciu regulačného obvodu znázorneného na obr.2, kde optimalizované parametre sú P1,I1,D1, P2,I2,D2,k1,t11,t12, k2,t22,t21.



Obr.2: Simulačná schéma regulačného obvodu

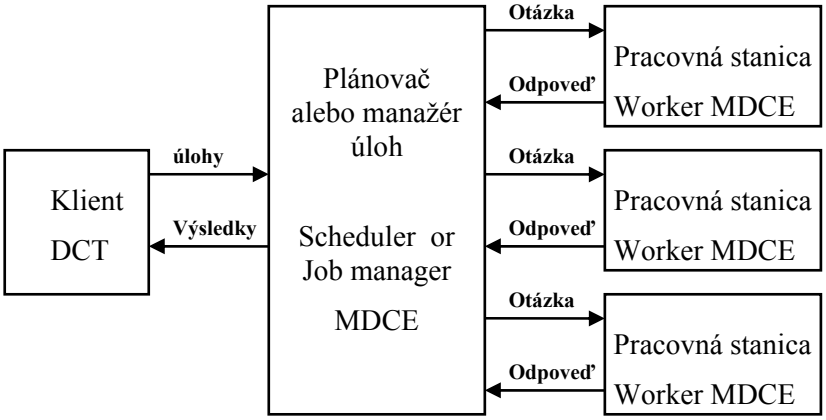
4 Realizácia PGA v Parallel Computing toolboxe

Pre účely testovania Parallel Computing toolboxu boli použité tri samostatné populácie PGA spúšťané na troch PC označované aj ako S1-Slave1, S2-Slave2, M-Master. V danom prípade jeden uzol PGA bol realizovaný v jednom PC. V uzle Master sa realizovala distribúcia výpočtov do jednotlivých počítačov a prepojenia migračných väzieb medzi jednotlivými subpoluláciami. Na obr.3 sú zobrazené použité migračné schémy.



Obr.3: Migračné prepojenia medzi subpopuláciami

Na implementáciu paralelných výpočtov v prostredí Matlab bola použitá knižnica „Distributed Computing Toolbox“ (DCT, v novšej verzii Matlabu sa nazýva už Parallel Computing Toolbox), ktorá umožňuje spúšťanie a riadenie paralelných procesov na jednom počítači s viacjadrovým procesorom. Pre vytváranie a sledovanie priebehu paralelných procesov v prostredí Matlab existuje okno „Parallel comand window“. S použitím ďalšej knižnice „MATLAB Distributed Computing Engine“ (MDCE) je možné realizovať beh paralelných procesov nielen na jednom počítači s viacerými procesormi, ale tiež na viacerých počítačoch zapojených v sieti. Na obr.4 je znázornená principiálna bloková schéma realizácie distribuovaných výpočtov v prostredí Matlab pomocou DCT a MDCE.



Obr.4 Bloková principiálna schéma distribuovaných výpočtov v prostredí Matlab pomocou DCT a MDCE

Paralelný výpočet úloh na viacerých počítačoch je možné v prostredí Matlab podľa obr.4 v princípe rozpísať do nasledovných krokov:

- Na jednom PC (klient) sa v Matlabe pomocou DCT vytvoria úlohy a spustia sa, resp. odošlú do manažéra úloh.
- Manažér úloh zabezpečí rozdelenie úloh do jednotlivých pracovných staníc, na ktorých musí byť nainštalovaný Matlab s MDCE.

- Po ukončení úloh manažér úloh zozbiera odpovede z pracovných staníc a odošle výsledky klientovi. Na základe tohto princípu distribuovaného výpočtu z obr.4 je možné realizovať paralelný genetický algoritmus nasledovne:
- Vytvoríme si funkciu na genetický algoritmus, ktorej vstupom je matica reprezentujúca prvky populácie a výstupom je matica s prvkami novej populácie modifikovanej genetickým algoritmom po N generácií.
- Na PC klient sa vykoná inicializácia premenných a náhodne sa vygenerujú 3 nezávislé populácie pre všetky subpopulácie.
- V cykle sa paralelne pomocou DCT spúšťa funkcia s genetickým algoritmom pre 3 nezávislé populácie, pokiaľ nie je splnená ukončovacia podmienka.
- Po vrátení nových populácií funkciou je možné medzi populáciami vykonať migrácie podľa ľubovoľných migračných schém.

Pre realizáciu PGA na troch počítačoch môžeme použiť štruktúru prepojenia jedného *JobManagera* s tromi *Workermi* alebo pre každého Workera je osobitný JobManager. Na každom PC sa nachádza jeden worker. Ak sú JobManageri traja, sú umiestnení na PC Master alebo na každom PC jeden.

Z hľadiska programovania paralelného algoritmu je potrebné použiť nasledovné príkazy:

- Vytvoriť prepojenie s JobManagermi, príkazom *findResorce*
  jm= findResource('scheduler','type','jobmanager','name','menojobmanagera', 'LookURL','PC_IPadresa');
- Vytvoriť pre každého JobManaga úlohu (Job), príkazom *createJob*
  j1 = createJob(jm,'FileDependencies', {'ga_funkcia.m'});
- Vytvoriť v rámci úlohy jednu alebo viacej otázok (Task) pre každú úlohu (Job), príkazom *createTask*
  createTask(j1, @ga_funkcia,pocet_fun_vystup, {Pop,pocet_generacii});
- Spustiť riešenie úlohy, príkazom *submit(j1)*
- Testovať koniec úlohy, príkazom *waitForState(j1)*
- Prečítať výsledky úloh, príkazom *results1 = getAllOutputArguments(j1);*

5 Overenie PGA

Realizácia PGA bola testovaná na optimalizácii parametrov rozvetveného regulačného obvodu (Obr.2) s parametrami PGA ako bolo opísané v predchádzajúcich kapitolách. Riešenie optimalizačného problému je realizované v troch subpopuláciách na troch počítačoch označených ako Master, Slave1, Slave2. Bola použitá migračná schéma podľa obr.3, perióda migrácie bola 5 generácií.

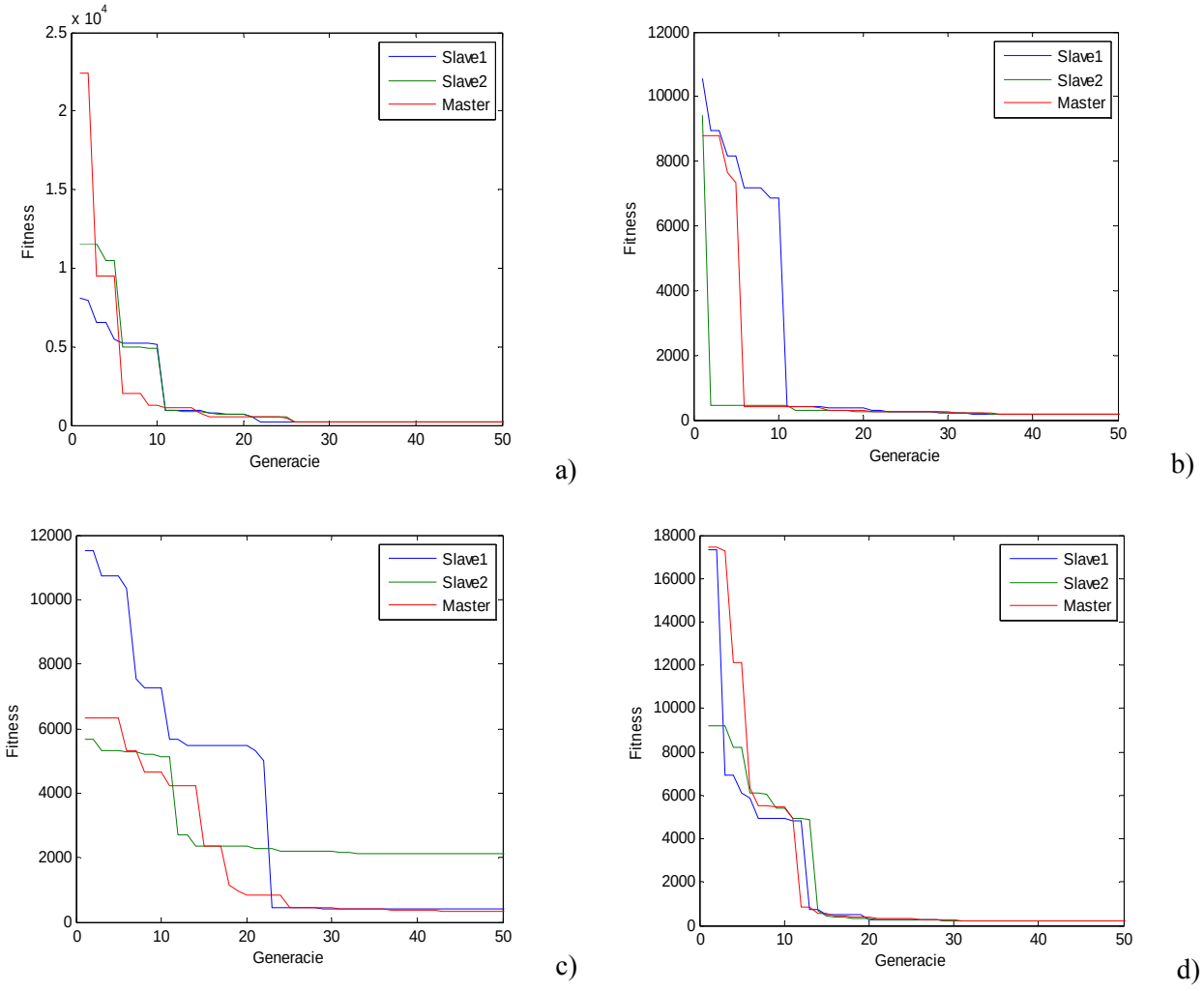
Na obrázkoch 5a) až 5d) sú zobrazené priebehy evolúcie minimalizovanej účelovej funkcie (1) pre jednotlivé GA s rôznymi typmi migrácie.

Výsledkom PGA je najlepší jedinec v uzle Master po vykonaní požadovného počtu generácií. Získané boli nasledovné parametre regulačného obvodu:

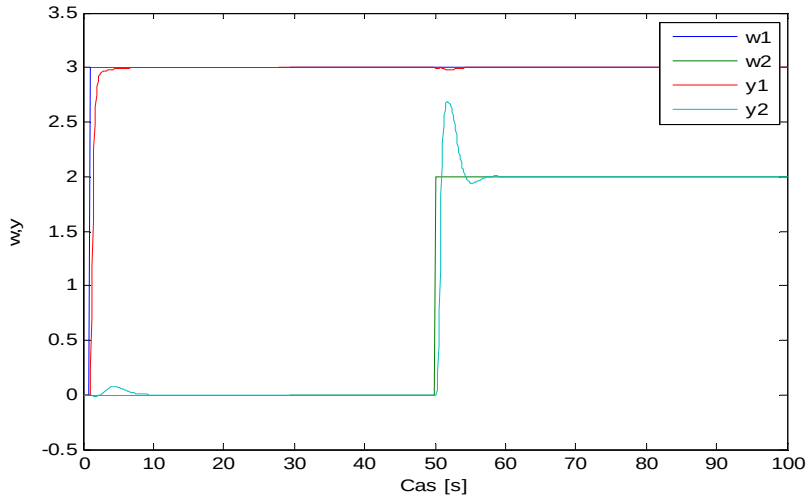
P1=53.899, I1=78.2064, D1=1.8854, P2=24.7023, I2=19.4204, D2=17.2952

k1=-0.064, t11=29.235, t12=89.2842 k2=-0.1227, t22=54.4186, t21=19.8595

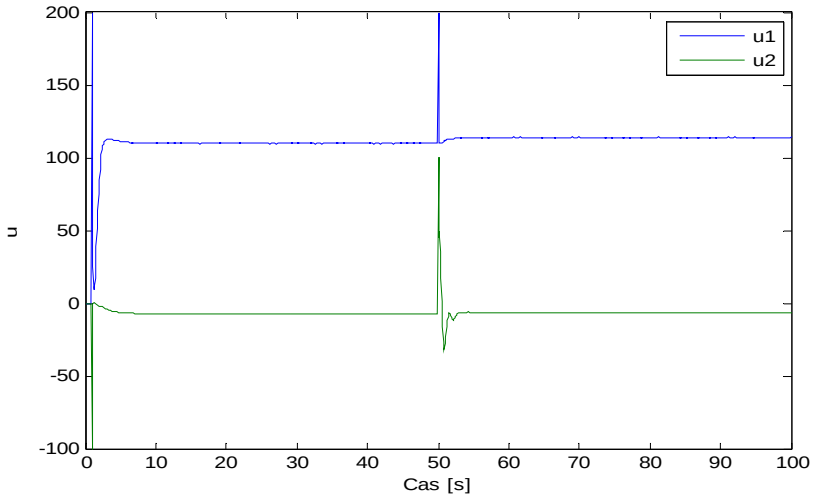
Pre uvedené parametre sú na obr.6 a obr.7 zobrazené výsledky simulácie regulačného obvodu pre výstupy systému a pre riadiace zásahy.



Obr.5 Priebehy Fitness funkcií pri optimalizácii parametrov regulačného obvodu pre typy migrácií podľa obr.3



Obr.6 Priebehy výstupov systému y1, y2 a žiadaných hodnôt w1, w2.



Obr.7 Priebehy riadiacich zásahov u1, u2

ZÁVER

Hlavným cieľom príspevku bolo ukázať technickú realizáciu paralelných genetických algoritmov na viacerých počítačoch pri riešení optimalizačných problémov použitím Parallel Computing toolboxu. Testovanie funkčnosti PGA v prostredí Matlab pre riešenie vybraných optimalizačných problémov potvrdilo, že tento prístup má veľmi dobré praktické využitie. Použitie tohto nástroja získava na význame najmä pri uskutočňovaní (veľmi) zložitých optimalizačných a prehľadávacích úloh z rôznych aplikačných domén. Riešenie rôznychch typov úloh z oblasti návrhu regulačných obvodov, robotiky, konštrukcie alebo úloh ekonomického typu (ale aj mnohých iných) nezriedka predstavuje v prípade riešenia na jednom procesore desiatky hodín až niekoľko dní. Samotná distribúcia na viacero procesorov (počítačový klaster, grid) znižuje nároky na výpočtový čas. Okrem toho vhodná konfigurácia PGA vďaka komunikácii jednotlivých uzlov pôsobí vo výpočte synergicky a dovoľuje znížiť potrebný počet vyhodnotení účelovej funkcie v rámci celého PGA v porovnaní s prípadom, kedy by bola využitá iba jedna veľká populácia aj o jeden rád. Súčasne dovoľuje nachádzať lepšie riešenia optimalizačnej úlohy a zabrániť uviaznutiu riešenia v lokálnom extréme (suboptimálnom riešení).

POĎAKOVANIE

Príspevok bol podporený grantovou agentúrou VEGA v rámci grantu č. 1/3100/06.

LITERATÚRA

[1] 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

[2] CANTÚ-PAZ, E.: *Migration polices, selection pressure, and parallel evolutionary algorithms*, in Journal of heuristics 7(4), 2001, pp. 311-334.

[3] CHIPPERFIELD, A. J., FLEMING P. J.,: *Parallel genetic algorithms*: A survey, ACSE Research Report No.518, University of Sheffield, 1994

[4] FOLTIN, M.: *Distribuované výpočty v Matlabe*, Zborník konferencie Matlab, Praha, 2004

[5] SEKAJ, I.: *Robust Parallel Genetic Algorithms with Re-Initialisation*, in *PPSN VIII*, September 18-22, Birmingham, 2004

[6] THE MATWORKS: *MATLAB Distributed Computing Engine*, System Administrator's Guide, Natick, 2006

[7] THE MATWORKS: *Distributed Computing Toolbox ver. 3.0*, User's Guide, Natick, 2006

Doc. Ing. Ivan Sekaj, PhD.
Ústav riadenia a priemyselnej informatiky, FEI STU v Bratislave, E-mail: ivan.sekaj@stuba.sk

Ing. Slavomír Kajan, PhD.
Ústav riadenia a priemyselnej informatiky, FEI STU v Bratislave, E-mail: slavomir.kajan@stuba.sk

Ing. Michal Oravec
Ústav riadenia a priemyselnej informatiky, FEI STU v Bratislave, E-mail: michal.oravec@stuba.sk