Computing and Informatics, Vol. 28, 2009, 1001–1013, V 2009-Mar-2

GLOBAL IMPACT BALANCING IN THE HIERARCHIC GENETIC SEARCH

Paweł Jojczyk, Robert Schaefer

AGH University of Science and Technology
Department of Computer Science
Mickiewicz Avn. 30
30-059 Kraków, Poland
e-mail: pawel.jojczyk@gmail.com,schaefer@agh.edu.pl

Revised manuscript received 9 December 2009

Abstract. The new *Globally Balanced Hierarchic Genetic Strategy* (GB-HGS) was introduced as a tool for solving difficult global optimization problems. This strategy provides a multi-deme economic stochastic search with an adaptive accuracy that allows many local extremes of the objective to be found. The strategy was designed according to the Multi Agent System (MAS) paradigm. The novelty of GB-HGS derives from its control of the search impact performed by various demes on the basis of the global information gathered and exchanged among the computing agents. This mechanism is applied together with the local profiling of the computational process already used in the previous versions of hierarchic genetic computations. The new strategy exhibits better efficiency, especially in the second phase of computations, when the promising regions containing the global extremes are encountered (see Figures 3, 4).

1 INTRODUCTION

The *Hierarchic Genetic Strategy* (HGS) invented by Kołodziej and Schaefer [11] tried to comprehend the advantages of both multi-deme (see e.g. Skolicki and De Jong [13]) and adaptive accuracy genetic searches (see e.g. Schraudolph and Belew [12], Whitley, Mathias and Fitzhorn [14]). Due to the synergy of both approaches HGS has become the accurate, stochastic global optimization algorithm with moderate computational cost. It has been intensively tested and exhibits exceptional accuracy especially in the case of difficult multimodal benchmarks (see e.g. [8, 15]).

The paper is focused on the new agent-oriented version of HGS, called *Globally Balanced Hierarchic Genetic Strategy* (GB-HGS), in which the global information collected during computation and circulating among agents is utilized for the profiling search process.

The HGS maintains a tree-structured, dynamically changing set of dependent demes. The depth of this tree is restricted by the constant m. The demes close to the root (called low-order ones) perform a chaotic search with low accuracy while the high-level ones, including leafs, represent more local and more accurate searches. The role of all low-order processes is to find the promising regions in the optimization landscape and then to activate a new child-deme of the higher order there. The strategy starts with the root of the structure (the unique 1^{st} order deme). After the period of K-epochs, called the *metaepoch*, the best fitted individual is selected. This individual becomes the seed of the new population of the 2^{nd} order which is obtained by the procedure called *sprouting* operation. Sprouting can be generalized in some way to all branches of a population's tree (demes of a higher order up to $m-1$). Sprouting is performed conditionally, if there is room for the new deme among the existing child-demes. The redundant demes that search in the same region are reduced by the *branch reduction* operation.

We will discuss the real-number encoding implementation called HGS-RN [15, 10]. Both genotypes and phenotypes appearing in each branch are the vectors of real entries belonging to the N dimensional hypercube $[a,b]^N \subset \mathbb{R}^N$. The genotype universa U_j for branches of various orders j are obtained by the proper scaling that uses the scaling coefficients $+\infty > \xi_1 > \cdots > \xi_m = 1$, so

$$U_j = \left[0, \frac{b-a}{\xi_j}\right]^N, \ j = 1, \ldots, m. \tag{1}$$

Next, we may define the set of encoding and re-scaling functions

$$\text{code}_j : U_j \ni \left\{x^i\right\} \to \left\{\xi_j x^i + a\right\} \in \mathcal{D}, \ j = 1, \ldots, m \tag{2}$$

$$\text{scale}_{i,j} : U_i \ni \left\{x^k\right\} \to \left\{\frac{\xi_i}{\xi_j} x^k\right\} \in U_j, \ i, j = 1, \ldots, m. \tag{3}$$

The initial population for the newly sprouted branch of degree $j+1$ is randomly chosen according to the N-dimensional probability distribution

$$\mathcal{N}\left((\text{scale}_{j,j+1}(y))_1, \sigma_j^{sprout}\right), \ldots, \mathcal{N}\left((\text{scale}_{j,j+1}(y))_N, \sigma_j^{sprout}\right) \tag{4}$$

where $\mathcal{N}(\cdot,\cdot)$ represents the one-dimensional random variable with the normal probability distribution parametrized by its mean and standard deviation. Moreover, y stands for the best adapted individual (the seed individual) in the parental deme of the order $j = 1, \ldots, m-1$ while σ_j^{sprout} is the standard deviation specific for the branch of the order j.

Let P be a deme of the order $j = 1, \ldots, m-1$ and y its best fitted individual currently distinguished after a metaepoch. Sprouting is not activated if there exists

a population P' of the order $j+1$ which satisfies

$$d(\bar{y}, \text{scale}_{j,j+1}(y)) < c_{j+1} \tag{5}$$

where $\bar{y}$ is the genotypes' average in the population P', c_{j+1} is the branch comparison constant for $j+1$ order branches and d stands for the Euclidean distance in $\mathbb{R}^N$. The sprouting of the new branch may also be prohibited if the assumed maximum number of child-demes $max\_children_j$ for the particular parental deme of the j-th order may be exceeded. Similarly, based on the arithmetic averages of genotypes in two demes, we can define the *branch reduction* operation. Let $\bar{x}$ and $\bar{y}$ be the arithmetic averages of genes in two demes P, P' of the same order $j > 1$. If $d(\bar{x}, \bar{y}) < c_j$, then P, P' are reduced to the single deme of the same order j by common selection. Let us introduce the set Ch_P of all demes of the order $j+1$ being the children of the deme P of the order j. This set may also include demes that were terminated by the local stop condition. The set $\overline{Ch}_P \subset Ch_P$ will contain all child-demes of P currently processed. The conditional sprouting operation may be represented as the function of the parental deme P and its set of children Ch_P that turns back $sprout(P, Ch_P)$, being the new child-deme if the sprouting condition is satisfied and the empty deme otherwise.

The simple evolutionary techniques based on the normal phenotypic mutation and arithmetic crossover are utilized in all branches. The proper repairing operations are used to return the individuals obtained by the crossover and mutation outside the genotype universum (see e.g. [1] and [10] for details). In order to check the *local stop condition* for each branch, except the root, the progress of mean fitness is monitored. The processing of the particular deme is suspended if a lack of satisfactory mean fitness progress is observed. The applied progress conditions are rather restrictive in order to avoid the prolonged, expensive processing of the single deme. The *global stop condition* for the whole strategy depends on the evolution progress in leafs. The whole strategy is stopped if the satisfactory individuals are found in the union of leaf-demes.

The advantages of HGS-RN are caused mainly by its following features: The final solutions are found by leafs that search locally (small standard deviation of the mutation operation) with highest accuracy. Leafs are created only in the promising regions selected by the hierarchy of demes. The redundancy among demes is reduced by the conditional sprouting and the branch reduction operations. Generally, these mechanisms protect against running more than one deme in the same admissible set region.

The basic version of HGS-RN (see [15]) assumes synchronization of all branches after each metaepoch, while the branches are processed in parallel between these checkpoints. At each checkpoint, branch comparison and sprouting is performed. This version may contain the following drawbacks: Even if the metaepochs can be run for the separate demes in parallel, the synchronization after each metaepoch may cause significant time losses. The redundancy reduction is performed only among the siblings (demes being the children of the same parental deme). The total

redundancy reduction that consists of comparison of all demes of the same order may result in enormous computational costs.

The first agent-oriented system (see e.g. Ferber [2]) based on the HGS-RN strategy was made by using the OCTOPUS platform [9]. Each OCTOPUS computational agent that maintains a single deme is an independent software unit that can migrate across the computer network searching for CPU and RAM resources. Only the local profiling of the deme searching impact by using the conditional sprouting was implemented. An interesting feature of this project was the possibility of automatic scheduling of computing agents in the distributed environment by using the diffusion-like algorithm. The papers [9, 3, 4] show the speedup, scalability and other advantages of such an approach.

2 GLOBAL IMPACT BALANCING AMONG COMPUTING AGENTS

The new GB-HGS strategy (Globally Balanced Hierarchic Genetic Strategy) designed according to the Agent-Oriented paradigm satisfies the following assumptions:

1. The processing of each deme of the HGS-RN tree will be supervised by the *computational software agent*. It delivers and controls the CPU and RAM resources available for the deme as well as controlling the progress of its evolution. The computational agent also gathers information about the child-demes and organizes the communication service necessary for sprouting, branch reduction and other necessary duties. The computational agent is created together with the new sprouted deme. If the deme is removed or suspended, its computational agent is hibernated or still remains alive if it is necessary to control and communicate computational agents governing child-demes.
2. Each deme is processed asynchronously to each other. Each agent communicates the center of gravity of the supervised deme and the best fitness value found after every metaepoch to its parental agent.
3. Conditional sprouting is executed in the same way by using the function *sprout* and is performed locally by the computational agent. This operation is based on the set of centers of gravity of the child-demes that are received by the parental agent from its child-agents.
4. No costly comparison of all branches during branch reduction is performed. The special type agents called *Leo the Professional* asynchronously visit all populations of the same order reducing the search redundancy. If the deme P of the order less than m is reduced, then the whole branch started from P is removed.
5. Additional profiling (balancing) of the processing impact of demes is performed on the basis of global information about the evolution progress by using the mechanism of competition for resources among computing agents.

6. The whole computation is started and finished by the *environmental agent*. This agent also gathers and analyzes the computing results delivered by leafs.

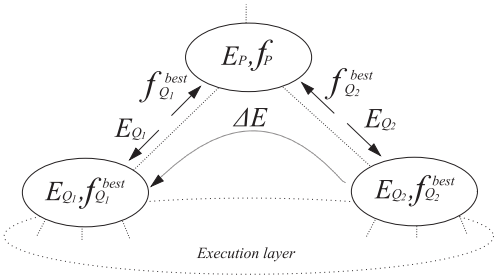

Fig. 1. Vitality transfer among the sample parental GB-HGS agent and its two child-agents

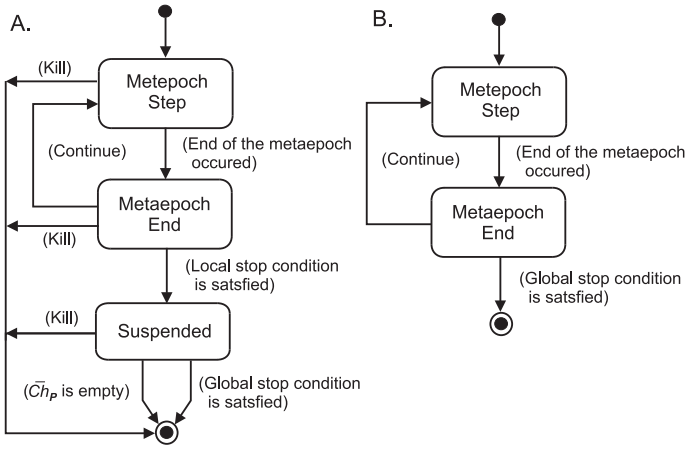

Fig. 2. State-transition diagrams of the GB-HGS computing agents: A. the agent supervising the HGS-RN deme P of the order $j > 1$, B. the agent supervising the HGS-RN deme P_{root}

This version is designed in terms of the AGE platform [6] which supports the design and execution of the *Evolutionary Multi-Agent Systems* (EMAS), which extend the classical model of evolutionary computation. The EMAS allow, in particular, the totally decentralized processing of individuals and populations, local performing of selection and genetic operations based on the agent appointment mechanism, etc. Each EMAS agent that is responsible for a population is equipped with the parameter $E \in \mathbb{R}_+$ called *vitality*. The AGE platform which supervises the asynchronous behavior of agents can profile the CPU utilization by the agents' actions according to the agents' vitality. A lack of vitality can result in agent hibernation or death, while a high vitality value allows the agent to compete for offspring. Vitality is

assigned by AGE to the agents according to their impact in the genetic search (e.g. according to the mean fitness of the contained individuals).

distinct f_P^{best} and compute the mean genotype $\bar{y}_P$;
send $\bar{y}_P$ to Leo the Professional;
if (local stop condition is satisfied) **then**
  $flag_P := stopped$;
else
  $flag_P := alive$;
end if
if (P is a leaf) **then**
  if ($flag_P = stopped$) **then**
    send the computation results to the environmental agent;
  end if
else
  receive messages $(flag_Q, \bar{y}_Q, f_Q^{best})$ from the child-agents $Q \in \overline{Ch}_P$;
  update $\overline{Ch}_P := \overline{Ch}_P \setminus \{Q; flag_Q = stopped\}$;
  update f_P^{best} according to the formula (6);
end if
send the message $(flag_P, \bar{y}_P, f_P^{best})$ to the parental agent;
receive the vitality E_P from the parental agent;
if (P is not a leaf) **then**
  compute the vitality for child-agents according to the formula (7);
  send E_Q for all living child-agents (e.g. $Q \in \overline{Ch}_P$);
  if $(\#\overline{Ch}_P < max\_children_j)$ **then**
    $Q := sprout(P, Ch_P)$;
    if $(\neg(Q = \emptyset))$ **then**
      $\overline{Ch}_P := \overline{Ch}_P \cup \{Q\}$; $Ch_P := Ch_P \cup \{Q\}$;
      if $(j < m)$ **then**
        $Ch_Q := \emptyset$; $\overline{Ch}_Q := \emptyset$;
      end if
    end if
  end if
end if

Algorithm 1: The GB-GS computing agent activities at the state *Metaepoch End*. The agent supervises the HGS-RN deme P of the order $j > 1$.

This idea was first applied by Jojczyk [5] for profiling the HGS-RN in the following way:

- The agent governing the HGS-RN root has the constant vitality $E_{root} > 0$.
- The new sprouted agent that is responsible for $Q \in \overline{Ch}_P$ receives the same amount of vitality $E_Q := E_P$ as is currently possessed by its parent.

- The total energy of all agents responsible for children of the deme P equals $(\#\overline{Ch}_P)E_P$.
- If the GB-HGS agent is removed or suspended, then its vitality is lost.
- Each agent supervising the leaf P distinguishes the best fitness value f_P^{best} archived by its deme at the end of the metaepoch and sends it to its parental agent.
- Each agent which is responsible for the deme P (except the agent which is responsible for the root) memorizes the best fitness value f_P^{best} archived by its deme at the end of the metaepoch. Then this value is updated by the best fitness values archived by its children and sent to its parental agent.

$$f_P^{best} := max\{f_P^{best}, f_Q^{best}; Q \in \overline{Ch}_P\} \tag{6}$$

- Each agent which is responsible for the deme P (except the agents which are responsible for the leafs) memorizes and updates the best fitness values archived by its children $\{f_Q^{best}\}, Q \in \overline{Ch}_P$. After such values are gathered (the messages containing f_Q^{best} are received from all child-agents) the new values of vitality E_Q are computed and then sent then to the child-agents.

$$E_Q := \frac{\#\overline{Ch}_P E_P}{\sum_{R \in \overline{Ch}_P} f_R^{best}} f_Q^{best},\ Q \in \overline{Ch}_P. \tag{7}$$

- The AGE platform delivers the CPU resources to the agent that supervises the deme P proportionally to the ratio $\frac{E_P}{E_{root}}$, so the higher the vitality an agent has, the faster its deme is processed.

The diagram that illustrates the vitality transfer among the sample parental agent and its two child-agents was presented in Figure 1. Moreover, the simplified state transition diagram of the GB-HGS computing agent that governs the HGS-RN deme $P \neq P_{root}$ is shown in Figure 2 A. In the state *Metaepoch Step* at most K evolution epochs for the deme, P is performed. The number of epochs may be less than K if the local stop condition appears. The agent can pass to this state just after it is sprouted or if the next metaepoch is run for P (Continue event). The operations performed in the *Metaepoch End* state summarize the result of the metaepoch computation of the deme P and its child-demes $\overline{Ch}_P$. The agent may pass to this state only from the state *Metaepoch Step* if the metaepoch is finished. The draft of the agent activities in the state *Metaepoch End* is reported by the pseudo-code Algorithm 1. Please note that all particular send and receive operations used here are non-blocking ones, e.g. messages are sent to the proper buffers waiting to be received by the respondent agent. Moreover, if there is no new message in the buffer, the old one is accepted while "receive" is invoked. For the sake of simplicity the details of sending and receiving signals among the GB-HGS agents are omitted. If the GB-HGS computing agent is in the state *Suspended*, then it is

still responsible for its child-agents. Its activities may be described by the similar pseudo-code as previously (see Algorithm 1) excluding statements 1–7, 18–26. The GB-HGS computing agent passes to the final state from the *Suspended* state if the global stop condition occurs and the proper signal from the environmental agent is broadcasted, or if the set of living child-demes becomes empty due to the local stop condition. The agent can also pass to the final state after the Leo the Professional agent kills it during the branch reduction process. The GB-HGS computing agent that supervises the P_{root} has only two states (see Figure 2 B). Its behavior in the state *Metaepoch Step* is the same as for higher order demes.

3 TESTING GB-HGS

The computational tests presented in this paper allow the comparison of the GB-HGS results with the results obtained by HGS-RN as well as by the single population Simple Genetic Algorithm SGA (see e.g. [10]). In order to focus on the advantages arising from the global impact balancing, both HGS-RN and GB-HGS were implemented using asynchronous agents on the AGE platform. Moreover, mechanisms of the local profiling of the genetic computations by the conditional sprouting and the branch reduction performed by Leo the Professional agents were also implemented in the same way for both strategies. The agents are scheduled by the AGE platform to the threads of the operating systems' kernel in order to execute their actions, which is similar to the mechanism of the light weight processes. The scheduling algorithm using the priorities assigned to the agents was utilized. The priorities of all agents were the same in the case of HGS-RN, while the priorities of the GB-HGS agents depend linearly on their current vitality which enables their CPU resources to be profiled. Asynchronous communication was implemented by defining the unique buffers for each agent that collects the receiving messages and the queuing mechanism for their processing.

The initial test series of the GB-GHS implementation using the well known Rastrigin, Ackley, Easom and Schwefel global optimization benchmarks were performed. The tests try to show the progress in global extreme finding as well as the ability to recognize more than one global extreme and also multiple local extremes. The results of GB-HGS were compared to the HGS-RN results as well as to the results of the Simple Evolutionary Algorithm (SEA) which operates on the single population.

The depth of the HGS-RN and GB-HGS trees was set as $m := 5$ in all experiments. The maximum number of children was established as $max\_children_j := 5, \forall j = 1, \ldots, 5$. Additionally, the population size was the same for all branches and equalled 50. The scaling coefficients ξ_j, $j = 1, \ldots, 5$ (see formula (1)) were set as follows: 5.0, 4.0, 3.0, 2.0 and 1.0, respectively. Standard genetic operations, such as normal mutation, arithmetic crossover and roulette selection, were utilized in each branch. The standard deviation σ_j^{mut} of the j^{th} order branch mutation depended on the branch order as well as on the size of the admissible domain. The standard deviation utilized by the sprouting operation (see Equation (4)) was set as

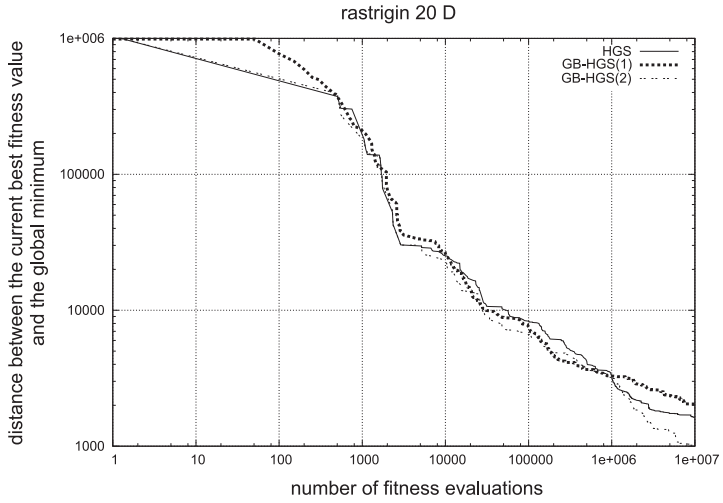

Fig. 3. The objective value regression for the best individual vs. the number of the fitness evaluation for the 20D Rastrigin benchmark

$\sigma_j^{sprout} := 2.5\sigma_j^{mut}$ for each branch order $j = 1, \ldots, 4$. The sprouting was performed if the Euclidean distance between the seed individual and the centers of gravity of all child-demes was greater than the branch comparison constant $c_j = 3\sigma_j^{mut}$, where σ_j^{mut} stands for the standard deviation of mutation for the proper degree deme. A similar condition was utilized for the branch reduction performed by the Leo the Professional agent. This operation was performed only for branches of degree greater than 2.

The first group of tests try to show how the GB-HGS mechanism can speed up the process of the best fitted individual bounding to the global extreme. The charts presented in Figures 3, 4 represent the trajectory of the best fitness with respect to the total number of fitness evaluations performed in all GB-HGS or HGS-RN demes. The ordinate of each point on these charts represents the distance between the fitness represented by the best evaluated individual and the fitness value at the global extreme of the particular problem.

The results observed for the Rastrigin and Ackley benchmarks show the similar behavior of HGS and both GB-HGS versions, GB-HGS(1) and GB-HGS(2). They are much more effective than the SEA single population one (see Figure 4). The Rastrigin function was defined in the box $[-512.0, 512.0]^{20}$ while the Ackley function in the $[-30.0, 30.0]^{10}$ domain. The standard deviations σ_j^{mut}, $j = 1, \ldots, 5$ for the Rastrigin benchmark were set as 68.27, 34.13, 22.76, 17.07 and 13.65, respectively, while for the Ackley one they were set as 4.00, 2.00, 1.33, 1.00, 0.80.

Much more interesting is the detailed comparison of results obtained by the HGS-RN and GB-HGS versions. We may observe two phases in GB-HGS trajectories for both benchmarks. In the first phase, in which the chaotic search dominates,

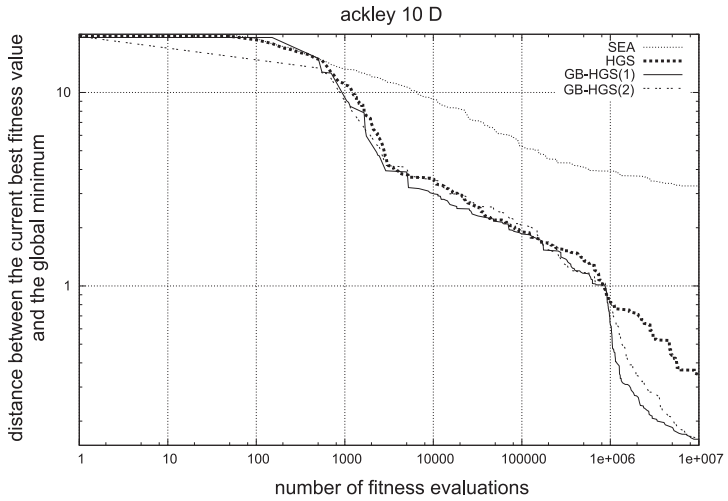

Fig. 4. The objective value regression for the best individual vs. the number of the fitness evaluation for the 10D Ackley benchmark

the behavior of GB-HGS is similar to HGS-RN. Both strategies develop the tree structure of demes obtaining similar efficiency in all branches. The mechanism of impact boosting by vitality balancing is not active. In the second phase, in which the region of the global extreme is recognized, GB-HGS begins to dominate over the HGS-RN. It may be explained by the activation of the vitality balancing that assigned more CPU resources for demes searching near the global extreme.

It is also interesting that the version of GB-HGS(2), with the relaxed local stop condition (the same condition at each level of the GB-HGS tree), is slightly better than the GB-HGS(1) version, in which the local stop condition is more restrictive for higher level branches than for lower ones.

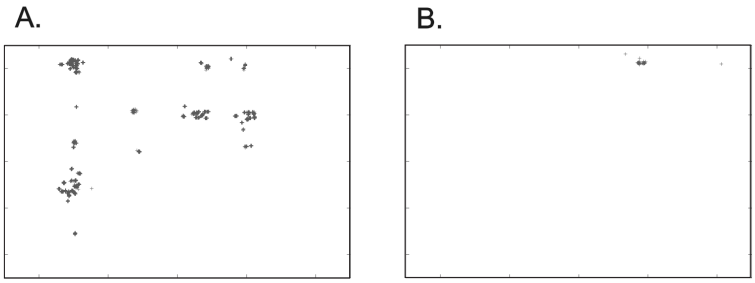

Fig. 5. Individuals distribution for the 2D Schwefel benchmark: A. the leaf populations of GB-HGS after 9 155 331 fitness evaluations, B. the SEA population after 7 703 051 fitness evaluations

In order to highlight the GB-HGS ability of concurrent finding of many local extremes, the very simple two-dimensional Schwefel benchmark was performed. The objective function was defined over the segment $[-500.0, 500.0]^2$. The standard deviations σ_j^{mut}, $j = 1, \ldots, 5$ were set now as 66.67, 33.33, 22.22, 16.67 and 13.33, respectively. The GB-HGS results were compared to the results of SEA for the same problem. Figure 5 B shows the concentration of the whole SEA population in the basin of attraction of the single, local extreme while the leaf-demes of GB-HGS occupied basins of attraction of 12 different local extremes (see Figure 5 A).

4 CONCLUSIONS

The complex multi-deme genetic strategies were created as methods to effectively solve the complicated, global optimization problems for multimodal objectives. One such strategy, called HGS, delivers the economic, global search methods with adaptive accuracy. The search process is profiled by using the local knowledge about the problem gathered by the algorithm.

The asynchronous GB-HGS strategy designed according to the multi-agent paradigm was derived from the semi-synchronous version of HGS called HGS-RN. This strategy allows additional profiling of the search process on the basis of the global knowledge gathered and exchanged among the computing agents.

The new strategy exhibits better efficiency than HGS-RN, especially in the second phase of computations, when the promising regions containing the global extremes are encountered (see Figures 3, 4). Moreover, GB-HGS possess strong global search abilities which may be seen by comparison to the results of single population searches (see e.g. Figure 5).

REFERENCES

[1] Bäck, T.—Fogel, D.B.—Michalewicz, Z. (Eds.): Evolutionary Computation. Institute of Physics Publishing, Bristol and Philadelphia, 2000.

[2] Ferber, J.: Multi-Agent Systems. Addison-Wesley Professional, 1999.

[3] Grochowski, M.—Schaefer, R.—Uhruski, P.: Diffusion Based Scheduling in the Agent-Oriented Computing Systems. LNCS, Vol. 3019, Springer 2004, pp. 97–104.

[4] Uhruski, P.—Grochowski, M.—Schaefer, R.: A Two-Layer Agent-Based System for Large-Scale Distributed Computation. Computational Intelligence, Vol. 24, No. 3, Blackwell Publishing (August 2008), pp. 191–212.

[5] Jojczyk, P.: Multi-Deme, Hierarchic Algorithm of Global Genetic Optimization. M. Sc. Thesis, AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Kraków 2007, Poland (in Polish).

[6] Kisiel-Dorohinicki, M.: Agent-Based Models and Platforms for Parallel Evolutionary Algorithms. In Bubak, M., van Albada, G. D., Sloot, P. M. A. and Dongarra,

J. (Eds.): Computational Science – ICCS 2004, Part III, LNAI, Vol. 3038, Springer 2004, pp. 225–236.

[7] Kołodziej, J.: Modelling Hierarchical Genetic Strategy as a Family of Markov Chains. Proc. of the Conference on Parallel Processing and Applied Mathematics, LNCS, Vol. 2328, Berlin-Heidelberg 2002, pp. 595–598.

[8] Kołodziej, J.—Jakubiec, W.—Starczak, M.—Schaefer, R.: Identification of the CMM Parametric Errors by Hierarchical Genetic Strategy Applied. In: Burczynski, T., Osyczka, A. (Eds.): Solid mechanics and its Applications, Vol. 117, Kluwer 2004, pp. 187–196.

[9] Momot, J.—Kosacki, K.—Grochowski, M.—Uhruski, P.—Schaefer, R.: Multi-Agent System for Irregular Parallel Genetic Computations. LNCS, Vol. 3038, Springer 2004, pp. 623–630.

[10] Schaefer, R.: Foundation of Genetic Global Optimization. Studies in Computational Intelligence, Series 74, Springer 2007.

[11] Schaefer, R.—Kołodziej, J.: Genetic Search Reinforced by the Population Hierarchy. In De Jong, K. A., Poli, R., Rowe, J. E. (Eds.): Foundations of Genetic Algorithms 7, Morgan Kaufman Publisher 2003, pp. 383–399.

[12] Schraudolph, N. N.—Belew, R. K.: Dynamic Parameter Encoding for Genetic Algorithms. Machine Learning Journal 1992, Vol. 9, No. 1, pp. 9–21.

[13] Skolicki, Z.—de Jong, K.: Improving Evolutionary Algorithms with Multirepresentation Island Models. Proc. of 8th Int. Conf. on Parallel Problem Solving from Nature – PPSN VIII, Birmingham, UK, LNCS, Vol. 3242, Springer 2004, pp. 420–429.

[14] Whitley, D.—Mathias, K.—Fitzhorn, P.: Delta Coding: An Iterative search Strategy for Genetic Algorithms. In: Belew, R. K., Booker, L. B. eds., Proc. of the 4th Int. Conf. on Genetic Algorithms, Morgan Kaufman 1991, San Mateo, CA, pp. 77–84.

[15] Wierzba, B.—Semczuk, A.—Kołodziej, J.—Schaefer, R.: Hierarchical Genetic Strategy with real number encoding. Proc. of the 6th Conf. on Evolutionary Algorithms and Global Optimization, Warsaw Technical University Press 2003, pp. 231–237.

Paweł Jojczyk, M. Sc. in Computer Science in 2007 at AGH University of Science and Technology; Software Engineer at IBM Polska. Research: multi-agent systems, evolutionary computation.

Robert Schaefer M. Sc. – 1979, Ph. D. – 1984 and habilitation – 1991 at the Cracow University of Technology. Professor title granted by the President of the Republic of Poland – 2003. Full Professor position in the Department of Computer Science, AGH University of Science and Technology – 2006. Author and co-author of more than 140 books, papers and conference contributions. Research: genetic algorithms in solving continuous global optimization problems, computing multiagent systems, algorithms of solving PDEs by the hp-adaptive Finite Element Method. Former research: modeling of blood flow and nonlinear flow in porous media.