Stopping rules for a parallel genetic algorithm

Ioannis G. Tsoulos*, Alexandros Tzallas and Markos Tsipouras

Department of Informatics and Telecommunications,
University of Ioannina, Greece
Email: itsoulos@teiep.gr
Email: tzallas@teiep.gr
Email: markos.tsipouras@gmail.com
*Corresponding author

Vasileios Christou

School of Computer Science,
The University of Manchester,
Oxford Rd, M13 9PL,
Manchester, UK
Email: bchristou1@gmail.com

Dimitrios Tsalikakis

Department of Engineering Informatics and Telecommunications,
University of Western Macedonia, Greece
Email: dtsalikakis@uowm.gr

Abstract: A novel method for the implementation of parallel genetic algorithms is introduced to locate the global minimum of a multidimensional function inside a rectangular hyperbox. The algorithm relies on a client – server model and incorporates an enhanced stopping rule. A number of experiments were conducted in order to measure the effects in termination by using the termination rule either on server machine or on clients. The method is tested on a series of well – known test functions as well as neural network training and the results was compared against another parallel genetic algorithm method. The results from the experiments are reported in terms of test error and amount of generations.

Keywords: genetic algorithms; parallel algorithms; stopping rules; optimisation.

Reference to this paper should be made as follows: Tsoulos, I.G., Tzallas, A., Tsipouras, M., Christou, V. and Tsalikakis, D. (2020) 'Stopping rules for a parallel genetic algorithm', *Int. J. Computational Intelligence Studies*, Vol. 9, Nos. 1/2, pp.146–160.

Biographical notes: Ioannis G. Tsoulos holds the position of Assistant Professor of Computational Intelligence at the Department of Informatics and Telecommunications, University of Ioannina, Greece. He received his BS in Computer Science from the University of Ioannina, Greece in 1998, MSc and PhD degree from the same university in 2001 and 2006, respectively. He has

published more than 60 peer-reviewed manuscripts. He has received more than 800 cross-references. He serves as reviewer for several scientific journals and conferences. His research interests include global optimisation, neural networks, genetic algorithms, genetic programming, biomedical signal and image processing.

Alexandros Tzallas holds the position of Lecturer of Computer Programming at the Department of Informatics and Telecommunications, University of Ioannina, Greece. He received his BSc in Physics from the University of Ioannina, Greece in 2001 and PhD in Medical Physics from the University of Ioannina, Greece in 2009. He has published more than 100 peer-reviewed manuscripts, five book chapters and he is the editor of two books. He has received more than 1950 citations. His research interests include: EEG, wearable devices, biomedical signal and image processing, biomedical engineering, decision support and medical expert systems and biomedical applications.

Markos Tsipouras received his Diploma in Computer Science and MSc and PhD in Biomedical Informatics in 1999, 2002 and 2008, respectively, from the Department of Computer Science, University of Ioannina, Greece. Also, he received a Natural Sciences Diploma from the Hellenic Open University in 2013. He has been employed as a Visiting Lecturer/Assistant Professor at various universities and technological institutes and he has participated in more than 20 European and national R&D projects. His research interests include medical informatics, digital processing of biomedical signals and images, fuzzy logic, data mining and machine learning.

Vasileios Christou received his BSc at Telematics and Administration Department in 2008 from Technological Education Institution of Epirus, Greece and MSc in Technology of Education and Digital Systems in 2012 from University of Piraeus, Greece. He is doing his PhD at the Computer Science Department at University of Manchester, UK. His current research interests include artificial intelligence, neural networks and evolutionary computing.

Dimitrios Tsalikakis holds the position of Lecturer of Modelling of Electrophysiological Data at the Department of Informatics and Telecommunications Engineering, University of Western Macedonia, Greece. He received his Diploma in Mathematics from the University of Ioannina and the PhD from the Department of Cardiology, Medical School from the University of Ioannina, Greece in 2006. His research interests include cardiac modelling, monophasic action potential analysis, automated systems analysis, virtual instrumentation, and scientific computing.

1 Introduction

There is a need in many scientific and real world applications to locate the global minimum of a function $f(x)$: $\Omega \subset R\ n \rightarrow R$ with $x_i \in [l_i, r_i]$, $I = 1, ..., n$. These applications include problems from many different areas such as physics (Duan et al., 1992; Wales and Scheraga, 1999), astronomy (Charbonneau, 1995; Metcalfe and Charbonneau, 2003), economics (Gaing, 2003; Maranas et al., 1997), medicine (Lee, 2007; Cherruault, 1994), etc. The methods developed to handle optimisation problems are divided into two main categories: deterministic and stochastic. The main representative of the first category is

the so-called interval method (Wolfe, 1996; Csendes and Ratz, 1997). Nevertheless, the majority of global optimisation methods fall in the second category where there are controlled random search methods (Price, 1983; Křivý and Tvrdík, 1995), simulated annealing methods (Ingber, 1989; Eglese, 1990), differential evolution methods (Storn and Price, 1997), particle swarm optimisation methods (Poli et al., 2007; Trelea, 2003), genetic algorithms (Goldberg, 1989; Michaelewicz, 1996), etc.

Genetic algorithms are naturally inspired algorithms for global optimisation and they use procedures such as natural selection, reproduction and mutation. They create a population of candidate solutions, called chromosomes. Chromosomes are iteratively evolved through selection, crossover and mutation until some stopping criteria are met. Genetic algorithms have been applied with success to many optimisation problems such as placement of wind turbines (Grady et al., 2005), water distribution (Prasad and Park, 2004), economics (Min et al., 2006), neural network training (Whitley et al., 1990), etc. The most significant advantages of genetic algorithms are:

1 They can be used easily in problems with multiple local minima.

2 They can be used in cases where the objective function is not smooth and the calculation of the derivative is difficult if not impossible.

3 They can be applied in cases where the dimension of the objective function is too large or when there is noise in the calculation.

Although, genetic algorithms have many practical applications they are tend to be relative slow requiring a great amount of CPU cycles to finish. Also, genetic algorithms are usually parallel in their design nature, because the evaluation of the objective function or the application of genetic operators can be implemented in parallel. During the past years, many researches have proposed applications of parallel genetic algorithms such as in heat exchanging problem (Hilbert et al., 2006), RNA folding (Shapiro et al., 2001), the container load problem (Gehring and Bortfeldt, 2002), combination optimisation problems (Han et al., 2001), vehicle routing problem (Berger and Barkaoui, 2004), etc. In most cases of parallel genetic algorithms, a server-client model is utilised, such as the island model (Kurdi, 2015; Tsoulos, 2008), where each machine denoted as client, runs a local genetic algorithm with separate sub-population. The machine denoted as server gathers from clients periodically the best discovered minimum. This article focuses on termination rules for the parallel genetic algorithms and more specifically we utilise a termination rule introduced in Calégari et al. (1997). The main goal of the study was to measure the effects in termination by using the termination rule either on server machine or on clients.

The rest of this article is organised as follows: in Section 2 the proposed method is given in detail, in Section 3 the results from different applications of the proposed termination rule are presented and in Section 4 some conclusions and thoughts for future work are outlined.

2 Method description

In this section, the proposed termination rule as well as the used algorithms in server and client side are outlined in detail.

2.1 The termination rule

The termination rule proposed in Tsoulos (2008) was used as an observation of the variance of the best discovered value f_l. The algorithm computes the variance $\sigma^{(iter)}$ of f_l at every *iter* generation. If no new minimum found for a number of generations, then it is an evidence that the global minimum is already found or the algorithm cannot advance further. At this point, the algorithm should terminate. Hence, the algorithm terminates when:

$$\left|f_h - f_l\right| \le e \; OR \; \sigma^{(iter)} \le \sigma^{(last)}/2 \; OR \; iter > ITERMAX \tag{1}$$

where last stands for the generation where the current best value f_l was discovered for the first time. The value e is a small positive number (usually $e \le 10^{-7}$) and *ITERMAX* is the predefined maximum number of allowed iterations (such as *ITERMAX* = 2000). The above termination rule is general enough to be applied not only in genetic algorithms but in any global optimisation procedure. Thus it has been applied with success in PSO algorithms (Tsoulos and Stavrakoudis, 2010). In the proposed study, the termination rule is applied either on the server-side algorithm or on client-side algorithm.

2.2 Server side algorithm

The machine denoted as server is responsible to gather from client machines the corresponding best discovered values. The algorithm executed on server is outlined below.

Initialisation step

1. **Set** $g_m = \infty$ as the global minimum value.
2. **Set** N the number of clients.
3. **Set** iter = 0 the number of iterations.

Check step

1. **Check** the current termination rule.
2. **If** the proposed termination rule has not be used in server side, then simply check if all clients have finished.
3. **If** the termination rule has satisfied, **then**
    1. **Report** g_m as the global minimum.
    2. **Terminate**
4. **EndIf**

Loop step

1. **For** i = 1 … N **Do**
    1. Obtain the minimum g_i from the client i
    2. **If** $g_i < g_m$ **Then** $g_m = g_i$
2. **EndFor**
3. **Goto** Check Step

2.3 Client side algorithm

On each client a genetic algorithm as described in Tsoulos (2008) is applied with the steps listed below.

Initialisation step

1 **Set** iter = 0, where iter is the current number of generations
2 **Set** ITERMAX.
3 **Set** N_c as the number of chromosomes in the population.
4 **Set** pc as the selection rate and p m the mutation rate (positive values less than 1.0)
5 **Set** $g_m = \infty$, where gm is the best discovered global minimum.

Check step

1 **Check** the current termination rule.
2 **If** the proposed termination rule has not be applied, then simply check iter ≤ ITERMAX.
3 **If** the termination rule holds, then terminate and **goto Local Search step**. Otherwise continue to the following step.

Genetic operations step

1 **For** every chromosome X i in the population
  1 **Calculate** the corresponding fitness value f_i, by applying the objective function to X_i
2 **End For**
3 **Apply** the genetic operations of crossover and mutation to the population.
4 **Set** iter = iter + 1
5 **Obtain** the best value in the population, denoted as fl.
6 **If** $f_l < g_m$ **Then**
  1 **Set** $g_m = f_l$
  2 **Send** g_m to Server machine.
7 **Endif**
8 **Goto** Check Step

Local search step

1 **Apply** the local search procedure LS( ) to the best chromosome x_l. The local search procedure used in the proposed technique was a BFGS one proposed by Powell (1989).
2 **Obtain** $g_m = LS(x_l)$
3 **Send** g_m to Server machine.

3 Experiments

3.1 Benchmark functions

In order to measure the effectiveness of the proposed approach we utilise several benchmark functions from the relevant literature (Ali, 2005; Floudas et al., 1999).

3.1.1 Griewank2

The function is given by the equation $f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4,000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{(i)}}\right)$.

In our experiments we have used $n = 2$ and the global minimum is 0.0. The function has several local minima in the specified range.

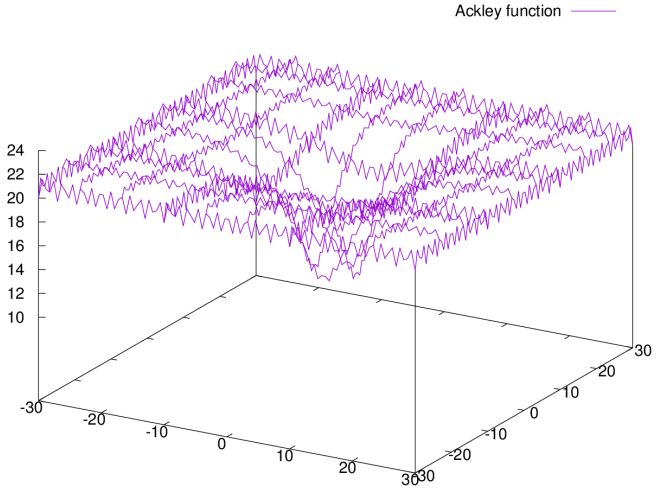
3.1.2 Ackley

The function is given by the equation

$$f(x) = -20\exp\left(-0.2\sqrt{\left(x_1^2 + x_2^2\right)} - \exp\left(0.5\left(\cos\left(2\pi x_1\right) + \cos\left(2\pi x_2\right)\right)\right)\right)$$

with $x_i \in [-30, 30]$, $i = 1, \ldots, 2$. The global minimum of the function is 0.0 *A* graphical representation of the function is shown in Figure 1.

Figure 1 The graphical representation of the function Ackley (see online version for colours)



3.1.3 CM

The function is given by the equation $f(x) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n} \cos(5\pi x_i)$ with $x \in [-1, 1]$ *n*. The value of the global minimum is –0.4 and in our experiments, we have used $n = 4$.

3.1.4 DeJoung

This function is given by the equation $f(x) = x_1^2 + x_2^2 + x_3^2$ with $x \in [-5.12, 5.12]$ 3. The value of the global minimum is 0.0.

3.1.5 Elp16

The function is given by the equation $f(x) = \sum_{i=1}^{n} i x_i^3$ with $x \in [-n, n]^n$. The value of global minimum is 0.0 and in our experiments, we have used $n = 16$.

3.1.6 Shekel7

$f(x) = -\sum_{i=1}^{7} \frac{1}{(x - a_i) * (x - a_i) + c_i}$ with $x \in [0, 10]^4$ and

$$a = \begin{matrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{matrix}$$

$$c = \begin{matrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{matrix}$$

The value of global minimum is –10.342378.

3.1.7 Test2N7

This function is given by the equation $f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i,\ x_i \in [-5, 5]$. The function has 2^n in the specified range and in our experiments we used $n = 7$. The corresponding values of global minimum is –274.163160 for $n = 7$.

3.1.8 Gkls350

The function $f(x) = Gkls\ (x, n, w)$ has w local minima as described in Gaviano et al. (2003) with $x \in [-1, 1]n$, $n \in [2, 100]$. In our experiments we use $n = 3$ and $w = 50$.

3.1.9 Rosenbrock32

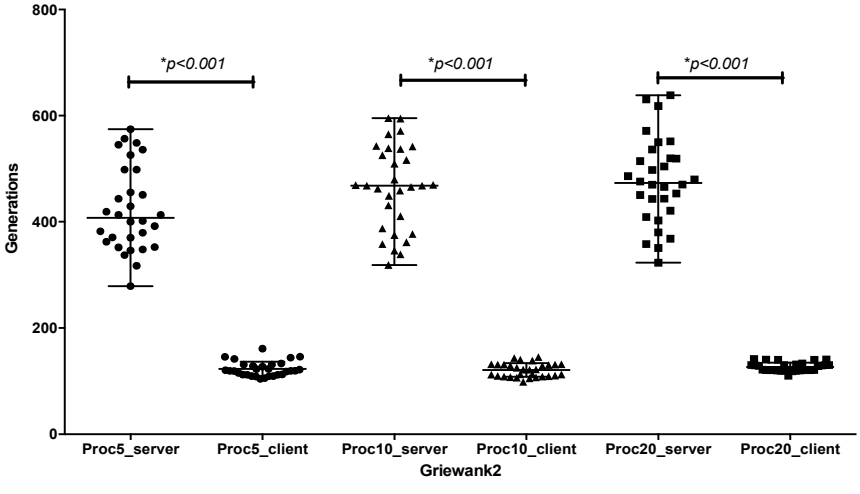
The function is given by $f(x) = \sum (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2),\ -30 \le x_i \le 30$. The global minimum is at the $x^* = (0, 0, ..., 0)$ with $f(x^*) = 0$. For our experiment we have used $n = 32$.

3.2 Experimental results

All the experiments were conducted on a series of computers with Intel I5 processors running Ubuntu Linux. The programming language used was ANSI C++ utilising Posix threads library. The parallel environment used was the Open MPI (Graham et al., 2006). The experiments were performed 30 times using different seed for the random generator each times and averages were taken. In every experiment the number of chromosomes was set to 200 and the maximum number of allowed generations was set to 2000. In all cases the average number of required generations were measured and the produced results are reported. In Table 1 the results from the usage of termination rule on server

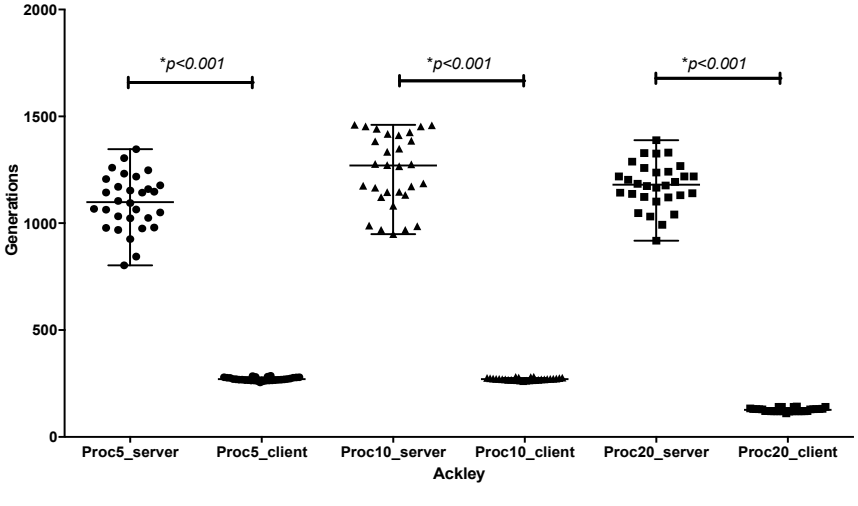
side are reported and in Table 2 the corresponding results from the usage of termination rule on client side listed. The column *function* denotes the name of the test function (as given in Subsection 3.1, the column *processors 5* denotes the results from the usage of 5 processors in the parallel setup, the column *processors 10* stands for the usage of 10 processors and the column *processors 20* denotes the results from the usage of 20 processors. Figures 2 and 3 shows the comparison of generations between server and client side for 5, 10 and 20 processors (Proc) using the Griewank2 and Ackley functions respectively. The aforementioned values of generations described as median and range and analysed using two-tailed independent samples Mann-Whitney U test.

Figure 2 Comparison of generations in server and client side for 5, 10 and 20 processors (Proc) using the Griewank2 function



Note: **p-value* from Mann-Whitney *U* test.

Figure 3 Comparison of generations in server and client side for 5, 10 and 20 processors (Proc) using the Ackley function



Note: **p-value* from Mann-Whitney *U* test.

Table 1 Experiments using termination rule only in the server

Function	*Processors 5*	*Processors 10*	*Processors 20*
Griewank2	357.32	353.12	310.08
Ackley	940.26	944.45	958.04
CM	115.15	122.37	99.67
DeJoung	69.96	67.05	60.83
Elp16	171.84	144.20	127.98
Test2n7	122.51	113.61	107.63
Shekel7	189.08	184.74	177.54
Gkls350	118.99	131.26	119.50
Rosenbrock32	821.19	806.59	753.41

Table 2 Experiments using termination rule in the clients

Function	*Processors 5*	*Processors 10*	*Processors 20*
Griewank2	110.4	109.82	108.9
Ackley	270.32	270.61	270.85
CM	25.59	25.63	25.98
DeJoung	19.23	19.16	19.27
Elp16	50.35	50.26	50.08
Test2n7	66.01	65.73	64.17
Shekel7	49.09	48.72	48.53
Gkls350	25.12	24.97	25.17
Rosenbrock32	351.20	353.15	346.21

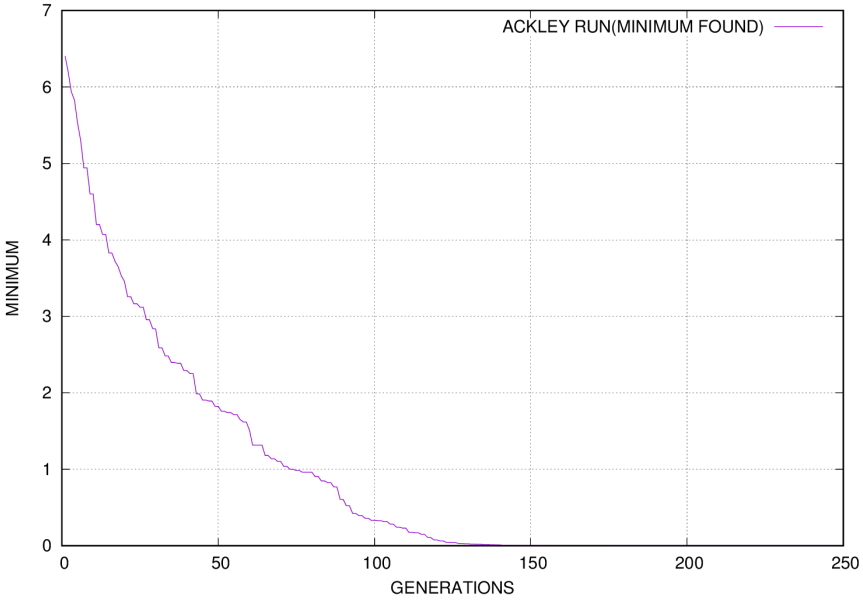
Table 3 Experiments using Galib MPI

Function	*Processors 5*	*Processors 10*	*Processors 10*
Griewank2	597.07 (0.73)	318.47 (0.77)	437.67 (0.73)
Ackley	1,650.27 (0.20)	1,289.57 (0.33)	1,369.57 (0.37)
CM	831.73 (0.90)	815.30 (0.93)	1,049.17
DeJoung	194.17	184.90	174.37
Elp16	1,709.70	1,405.83	1,138.17
Shekel7	1,801	1,734.53	1,867.43
Test2n7	901.53 (0.63)	1,015.60 (0.70)	818 (0.63)
Gkls350	291.39	287.53	280.33
Rosenbrock32	893.25	881.67	865.22

Table 4 Scalability of the method measured on Elp example

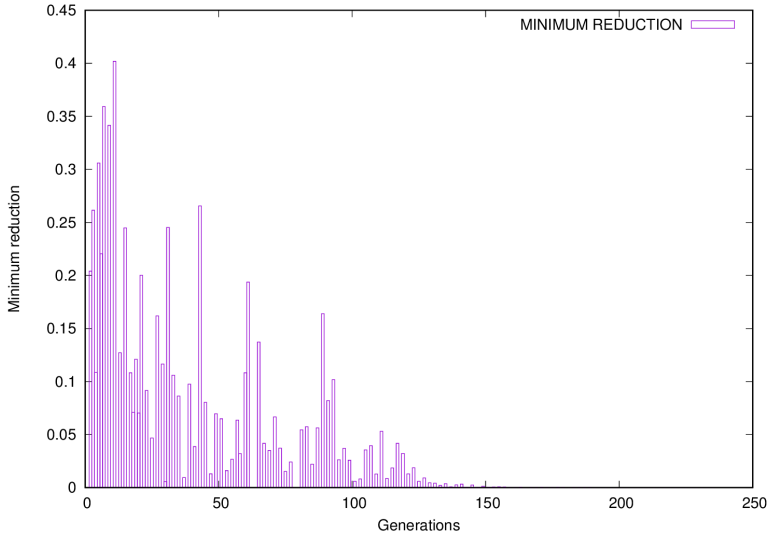
Dimension	*Termination on server*	*Termination on clients*
8	117.79	46.07
16	144.20	50.26
32	187.84	62.46
64	406.25	94.17
128	767.97	152.37

Figure 4 An example run for the Ackley function in two processors (see online version for colours)



Note: The example outlines the minimum found in every generation.

Figure 5 The minimum reduction for the function Ackley with two processors (see online version for colours)



The method is compared against GAlib (Wall, 1996) which is one of the more known genetic algorithm software and more specific a parallel variant of the software named GAlib-mpi. GAlib is tested on the same series of benchmark functions with the same settings of the proposed method. Each experiment was conducted and averages are

reported in Table 3. Each cell stands for the average number of generations required. The figures in parentheses denote the fraction of runs that located the global minimum and were not trapped in one of the local minima. Absence of this number denotes that the global minimum has been recovered in every single run.

The results indicate that the method requires a small amount of generations when the stopping rule is applied to the clients. Also, another interesting observation that requires more study is that the number of generations does not change significantly as the number of processors increase. Finally, it is evident that the method outperforms GAlib in the majority of experiments in terms of average number of generations.

As an example, run consider Figure 4 where the minimum found for every generation of the function Ackley is shown with two processors and the stopping rule is applied on the client side. Also the minimum reduction for the same function using two processors is outlined in Figure 5.

3.3 Neural network results

In order to measure the effectiveness of the proposed method additional experiments were conducted on neural network training. Neural networks are well - known parametric functions (Bishop, 1995; Cybenko, 1989) ideal for regression and classification. A neural network usually is a parametric function $N(\vec{x}, \vec{p})$ where x is the input vector and p (called weight vector) is the vector of the parameters to be discovered by some optimisation process. The optimisation procedure adapts the weight vector by minimising the error function, given as:

The vector t stands for the real output of the dataset. Three experimental datasets were used as inputs to neural network and more specific:

1 Ionosphere dataset: The ionosphere dataset contains data from the Johns Hopkins Ionosphere database. The two-class dataset contains 351 examples of 34 features each. This datasets was found in the machine learning repository in the following URL: http://www.ics.uci.edu/~mlearn/MLRepository.html.

2 BK dataset: This dataset comes from smoothing methods in statistics (Simonoff, 1996) and it used to estimate the points scored per minute in a basketball game. The dataset has 96 patterns of 4 features each.

3 The EEG dataset described in Andrzejak et al. (2001) is utilised here. The complete dataset consists of five sets (denoted as Z, O, N, F and S) each containing 100 single-channel EEG segments each having 23.6 sec duration. Sets Z and O have been taken from surface EEG recordings of five healthy volunteers with eye open and closed, respectively. Signals in two sets have been measured in seisure-free intervals from five patients in the epileptogenic zone (F) and from the hippocampal formation of the opposite hemisphere of the brain (N). Set S contains seisure activity, selected from all recording sites exhibiting ictal activity. Sets Z and O have been recorded extracranially, whereas sets N, F and S have been recorded intracranially.

All the experiments were conducted 30 times using different random seed each time. In every experiment the number of chromosomes was set to 200 and the maximum number of allowed generations was set to 2000. The number of weights used in neural network was set to 10. In all cases the average number of required generations were measured as

well as the test error of neural network. For the two classification datasets test error stands for the classification error and for BK dataset test error stands for average mean squared error. In Table 5 the results from the usage of termination rule on server side are show and in Table 6 the corresponding results from the usage of termination rule on client side are outlined. The column *dataset* denotes the name of the dataset, the column *processors 5* denotes the results from the usage of 5 processors in the parallel setup, the column *processors 10* stands for the usage of 10 processors and the column *processors 20* denotes the results from the usage of 20 processors. Additionally, the column GENS stands for the average number of required generations and the column TEST ERROR represents the average test error for every dataset.

Table 5 Results for neural networks using the proposed method with termination rule on server

Dataset	*Processors 5*		*Processors 10*		*Processors 20*	
	GENS	*Test error*	*GENS*	*Test error*	*GENS*	*Test error*
IONOSPHERE	150.35	18.66%	174.56	20.07%	208.60	20.69%
BK	823.53	1.15	698.81	1.17	631.20	1.30
EEG	168.81	35.73%	160.42	33.21%	137.34	30.80%

Judging from the results, the method can achieve the same level of test error requiring a smaller amount of generations when the stopping rule is applied only on the client side than when we use stopping rule on the server side.

Table 6 Results for neural networks using the proposed method with termination rule on clients

Dataset	*Processors 5*		*Processors 10*		*Processors 20*	
	GENS	*Test error*	*GENS*	*Test error*	*GENS*	*Test error*
IONOSPHERE	52.45	18.70%	53.57	20.07%	54.33	19.32%
BK	389.47	1.27	385.53	1.24	372.71	1.33
EEG	85.96	35.73%	89.71	32.29%	88.67	28.80%

3.4 Scalability of the method

In order to measure the scalability of the proposed method in terms of number of generations a series of experiments were conducted on test function Elp. The experiments were conducted using the following dimensions for the function: n = 8, 16, 32, 64, 128 The results are reported in Table 4. Again, the results indicate that the method is faster when the stopping rule is applied to the clients than when the stopping rule is applied on server. Also, the number of generations increases smoother in the first case than in the second.

4 Conclusions

We have presented a novel parallel genetic algorithm for function optimisation, based on a previous work. We have tested the efficiency of the proposed method regarding the

application of a recently introduced stopping rule. As a measure of the efficiency the average number of generations was used. Judging from the results, the method seems to be faster when the stopping rule is applied only in the client side. Future research includes incorporation of more advanced stopping rules, especially designed for a parallel environment as well as the inclusion of migration between populations in the server - client model.

References

Ali, M.M., Khompatraporn, C. and Zabinsky, Z.B. (2005) 'A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems', *Journal of Global Optimization*, Vol. 31, No. 4, pp.635–672.

Andrzejak, R.G., Lehnertz, K., Mormann, F., Rieke, C., David, P. and Elger, C.E. (2001) 'Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state', *Phys. Rev. E*, Vol. 64, No. 6, Pt. 1, pp.1–8.

Berger, J. and Barkaoui, M. (2004) 'A parallel hybrid genetic algorithm for the vehicle routing problem with time windows', *Computers & Operations Research*, Vol. 31, No. 12, pp.2037–2054.

Bishop, C. (1995) *Neural Networks for Pattern Recognition*, Oxford University Press, New York, NY, USA.

Calégari, P., Guidec, F., Kuonen, P. and Kobler, D. (1997) 'Parallel island-based genetic algorithm for radio network design', *Journal of Parallel and Distributed Computing*, Vol. 47, No. 1, pp.86–90.

Charbonneau, P. (1995) 'Genetic algorithms in astronomy and astrophysics', *Astrophysical Journal Supplement*, Vol. 101, p.309.

Cherruault, Y. (1994) 'Global optimization in biology and medicine', *Mathematical and Computer Modelling*, Vol. 20, No. 6, pp.119–132.

Csendes, T. and Ratz, D. (1997) 'Subdivision direction selection in interval methods for global optimization', *SIAM J. Numer. Anal*., Vol. 34, No. 3, pp.922–938.

Cybenko, G. (1989) 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control Signals and Systems*, Vol. 2, No. 4, pp.303–314.

Duan, Q., Sorooshian, S. and Gupta, V. (1992) 'Effective and efficient global optimization for conceptual rainfall-runoff models', *Water Resources Research*, Vol. 28, No. 4, pp.1015–1031.

Eglese, R.W. (1990) 'Simulated annealing: a tool for operational research', *Quadratic Programming Problems*, Vol. 46, pp.271–281.

Floudas, C.A., Pardalos, P.M., Adjiman, C., Esposoto, W.Z.G., Harding, S., Klepeis, J., Meyer, C. and Schweiger, C. (1999) *Handbook of Test Problems in Local and Global Optimization*, Kluwer Academic Publishers, Dordrecht.

Gaing, Z-L. (2003) 'Particle swarm optimization to solving the economic dispatch considering the generator constraints', *IEEE Transactions on 18 Power Systems*, Vol. 18, No. 3, pp.1187–1195.

Gaviano, M., Ksasov, D.E., Lera, D. and Sergeyev, Y.D. (2003) 'Software for generation of classes of test functions with known local and global minima for global optimization', *ACM Trans. Math. Softw*., Vol. 29, No. 4, pp.469–480.

Gehring, H. and Bortfeldt, A. (2002) 'A parallel genetic algorithm for solving the container loading problem', *International Transactions in Operational Research*, Vol. 9, Nos. 5–6, pp.497–511.

Goldberg, D. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Reading, Massachussets.

Grady, S.A., Hussaini, M.Y. and Abdullah, M.M. (2005) 'Placement of wind turbines using genetic algorithms', *Renewable Energy*, Vol. 30, No. 2, pp.259–270.

Graham, R.L., Woodall, T.S. and Squyres, J.M. (2006) 'Open MPI: a flexible high performance MPI', *Parallel Processing and Applied Mathematics*, *Lecture Notes in Computer Science*, Vol. 3911, pp.228–239.

Han, K-H., Park, K-H., Lee, C-H. and Kim, J-H. (2001) 'Parallel quantum-inspired genetic algorithm for combinatorial optimization problem', *Proceedings of the 2001 Congress on Evolutionary Computation*, pp.1422–1429.

Hilbert, R., Janiga, G., Baron, R. and Thévenin, D. (2006) 'Multi-objective shape optimization of a heat exchanger using parallel genetic algorithms', *International Journal of Heat and Mass Transfer*, Vol. 49, Nos. 15–16, pp.2567–2577.

Ingber, L. (1989) 'Very fast simulated re-annealing', *Mathematical and Computer Modelling*, Vol. 12, No. 8, pp.967–973.

Křivý, I. and Tvrdík, J. (1995) 'The controlled random search algorithm in optimizing regression models', *Computational Statistics & Data Analysis*, Vol. 20, No. 2, pp.229–234.

Kurdi, M. (2015) 'A new hybrid island model genetic algorithm for job shop scheduling problem', *Computers & Industrial Engineering*, Vol. 88, pp.273–283.

Lee, E.K. (2007) 'Large-scale optimization-based classification models in medicine and biology', *Annals of Biomedical Engineering*, Vol. 35, No. 6, pp.1095–1109.

Maranas, C.D., Androulakis, I.P., Floudas, C.A., Berger, A.J. and Mulvey, J.M. (1997) 'Solving long-term financial planning problems via global optimization', *Journal of Economic Dynamics and Control*, Vol. 21, Nos. 8–9, pp.1405–1425.

Metcalfe, T.S. and Charbonneau, P. (2003) 'Stellar structure modeling using a parallel genetic algorithm for objective global optimization', *Journal of Computational Physics*, Vol. 185, No. 1, pp.176–193.

Michaelewicz, Z. (1996) *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Verlag, Berlin.

Min, S-H., Lee, J. and Han, I. (2006) 'Hybrid genetic algorithms and support vector machines for bankruptcy prediction', *Expert Systems with Applications*, Vol. 31, No. 3, pp.652–660.

Poli, R., Kennedy, J. and Blackwell, T. (2007) 'Particle swarm optimization an overview', *Swarm Intelligence*, Vol. 1, No. 1, pp.33–57.

Powell, M.J.D. (1989) 'A tolerant algorithm for linearly constrained optimization calculations', *Mathematical Programming*, Vol. 45, Nos. 1–3, pp.547–566.

Prasad, T. and Park, N. (2004) 'Multiobjective genetic algorithms for design of water distribution networks', *J. Water Resour. Plann. Manage*., Vol. 130, No. 1, pp.73–82.

Price, W.L. (1983) 'Global optimization by controlled random search', *Journal of Optimization Theory and Applications*, Vol. 40, No. 3, pp.333–348.

Shapiro, B.A., Wu, J.C., Bengali, D. and Potts, M.J. (2001) 'The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation', *Bioinformatics*, Vol. 17, No. 2, pp.137–148.

Simonoff, J.S. (1996) *Smooting Methods in Statistics*, Springer, Verlag.

Storn, R. and Price, K. (1997) 'Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces', *Journal of Global Optimization*, Vol. 11, No. 4, pp.341–359.

Trelea, I.C. (2003) 'The particle swarm optimization algorithm: convergence analysis and parameter selection', *Information Processing Letters*, Vol. 85, No. 6, pp.317–325.

Tsoulos, I.G. (2008) 'Modifications of real code genetic algorithm for global optimization', *Applied Mathematics and Computation*, Vol. 203, No. 2, pp.598–607.

Tsoulos, I.G. and Stavrakoudis, A. (2010) 'Enhancing PSO methods for global optimization', *Applied Mathematics and Computation*, Vol. 216, No. 10, pp.2988–3001.

Wales, D.J. and Scheraga, H.A. (1999) 'Global optimization of clusters, crystals, and biomolecules', *Science*, Vol. 285, No. 5432, pp.1368–1372.

Wall, M. (1996) *GAlib: A C++ Library of Genetic Algorithm Components*, Vol. 87, p.54, Mechanical Engineering Department, Massachusetts Institute of Technology.

Whitley, D., Starkweather, T. and Bogart, C. (1990) 'Genetic algorithms and neural networks: optimizing connections and connectivity', *Parallel Computing*, Vol. 14, No. 3, pp.347–361.

Wolfe, M.A. (1996) 'Interval methods for global optimization', *Applied Mathematics and Computation*, Vol. 75, Nos. 2–3, pp.179–206.