Dual Population Genetic Algorithm (GA) versus OpenMP GA for Multimodal Function Optimization

A. J. Umbarkar
Department of Information Technology,
Walchand College of Engineering,
Sangli, MS, India.

M. S. Joshi
Department of Electronics Engineering,
Government College of Engineering,
Aurangabad, MS, India

ABSTRACT
Genetic algorithms (GAs) are useful for solving multimodal problems. It is quite difficult to search the search space of the multimodal problem with large dimensions. There is a challenge to use all the core of the system. The Dual Population GA (DPGA) attempts to explore and exploit search space on the multimodal problems. Parallel GAs (PGAs) are better option to optimize multimodal problems. OpenMP GA is parallel version of GA. The Dual Population GA (DPGA) uses an extra population called reserve population to provide additional diversity to the main population through crossbreeding. DPGA and PGA, both provide niching technique to find optimal solution. Paper presents the experimentation of DPGA, OpenMP GA and SGA. The experimentation results show that the performance of the OpenMP GA is remarkably superior to that of the SGA in terms of execution time and speed up. OpenMP GA gives optimum solution in comparison with OpenMP GA and SGA for same parameter settings.

Keywords
Genetic Algorithm (GA), Dual Population GA (DPGA), Serial DPGA, Open Multi Processing (OpenMP), Multimodal Function, Non-linear optimization problems.

1. INTRODUCTION
Genetic algorithms are based on theories of natural genetic and natural selection. GAs are useful to solve optimization problems. GA works on set of solutions and applies selection, crossover, mutation and survival selection iteratively to get the optimal solution. Many Parallel GAs are proposed by various researchers to find optimal solution, speedup the searching time and utilize the resources like CPU (loosely or tightly coupled systems), memory etc. efficiently. PGAs are developed by suggesting niching technique to solve problem effectively.

The Moore's law, Amdahl's law and Gustafson's law say that the algorithm and functionality of CPU will improve in years of time [1].

With the multicore processors getting cheaper and common, one cannot ignore their importance anymore. Multicore systems have multiple processing cores on same chip while multiprocessor systems have multiple chips inside a system. GAs need to be explored on multicore and HPC (High Performance Computing) computing paradigm. Many GA works uses multicore system for GAs but does not care about their CPU/core utilization. The algorithmically parallel DPGA but implemented sequential in C language, need to compare with programming parallel version of GA.

Paper experimented serial DPGA algorithm, which is by algorithm design, parallel, but implemented serially in C language. OpenMP GA is simple serial GA but implemented parallely in C language using OpenMP pragmas.

Paper focuses on issue of superiority of parallel programming implementation of serial algorithm versus design of parallel algorithm and serial implementation with respect to quality solution, processing speed and CPU utilization.

The paper is organized as follow. Section 2 explains the related works. Section 3 describes the algorithm implemented and information about programming options. Section 4 reports the test data and experimental results for multimodal optimization problems and compares serial DPGA, OpenMP GA and SGA. Finally, section 5 provides conclusions.

2. RELATED WORKS
Literature survey given in [2], [3] are on advances, computing trends, application and perspective of Parallel Genetic Algorithm (PGA) [4]. In computing trends the important issues are architecture, OS, topologies and programming (Libraries). The programming languages are facilitated with set of special system calls or libraries like -Linda, OpenMP[1], HPF, Parallel C and OCCAM (both for transputer networks), Java using communication libraries MPI (Message-Passing Interface)[2], Express MPI, PVM (Parallel Virtual Machine), POSIX threads and Java threads on SMP machines. There are many parallel programming options available in footnote [3, 4].

Function optimization is carried out using many parallelizable metaheurestics such as Artificial Immune Algorithms (AIA) [5], Ant Colony Optimization (ACO) [6], Particle Swarm Optimization (PSO) [7], Artificial Neural Network (ANN) [8], Differential Evolution (DE) [9], Harmony Search (HS) [10], Bacteria Foraging Optimization (BFO) [11], Shuffled Frog Leaping (SFL) [12], Artificial Bee Colony (ABC) [13], Biogeography-Based Optimization (BBO) [14], Gravitational Search Algorithm (GSA) [15], Grenade Explosion Method (GEM) [16], Teaching–learning-based optimization (TLBO) [17], etc. Performance of parallel metaheurestics for function optimization need to be checked on computing paradigms such as Clusters computing, MPPs, Grids computing, GPGPU, cloud computing and Multicore/ HPC. Many metaheurestic's performances are improved and checked on latest computing paradigms but as per the NFL (No Free Lunch) theorem [18], there is always chance that the particular metaheurestic performance might improve in terms of function evaluation, speedup [19] etc.

[1] https://computing.llnl.gov/tutorials/openMP/
[2] http:// www.lam-mpi.org/
[3] http://wotug.ukc.ac.uk/parallel/
[4] http:// www.openmp.org

GAs are implemented over Clusters, MPPs [20], [21], Grids [22], [23], [24], [25], [26], [27], [28], [29], GPGPU [30], [31], [32], [33], cloud computing and Multicore/ HPC [34]. The GAs work on cluster, MPPs (Massively Parallel Processors), grid and cloud computing are in distributed environment can be called PGAs because of parallel hardware. The design of algorithm needs to be tuned with the hardware selected for implementation. The parameters to be considered are network delay, system configuration of each node in distributed system. Network delay plays important role on the performance of algorithm in case of distributed system. The programming on such parallel hardware is called loosely coupled programming. In case of Multicore/ HPC and GPGPU, also called PGAs, the design of algorithm needs to be tuned with the hardware selected for implementation. No such parameters need to be considered in tightly coupled system. The programming on such parallel hardware is called as tightly coupled programming. PGAs have capacity to run in parallel on many computing paradigms but finding suitable hardware and software will give optimal solution with optimal resource utilization.

3. ALGORITHMS

The three algorithms experimented on multimodal test functions are SGA, OpenMP GA and DPGA. Section 3.1 explains the OpenMP GA and section 3.2 explain the DPGA.

3.1 OpenMP GA

OpenMP (Open Multiprocessing) is an API that supports shared memory multiprocessing (tightly coupled) programming in C, C++, and Fortran, on most processor architectures and operating systems, including Solaris, AIX, HP-UX, GNU/Linux, Mac OS X, and Windows platforms.

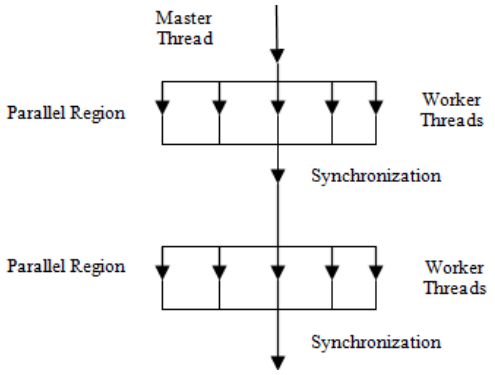

Figure 1 :Illustration of multithreading with the master thread forks off a number of threads for execute code in parallel.

Open MP library works on the concept of thread. Library has many pragmas that parallelize the loops of the algorithm along with knowing the number of cores available on multicore processor. The implementation of SGA with OpenMP multithreading will parallelize the algorithm. The master thread forks a specified number of slave threads and tasks are divided among them. The threads then run concurrently, with the runtime environment allocating threads to different processors or cores. The section of code which is meant to be run in parallel is marked accordingly, with a preprocessor directive that will cause the threads to form before the section is executed. After the execution of the parallelized code, the threads join back into the master thread, which continues onward to the end of the program. Figure 1 illustrates the multithreading with the master thread forks off a number of threads to execute code in parallel.

The flowchart of GA is given in figure 2. In OpenMP GA the various operators of SGA such as random number generator, initialization, fitness calculation are parallelized using pragmas, which reduce the amount of time to get the optimal solution and increases the CPU utilization on multicore system.

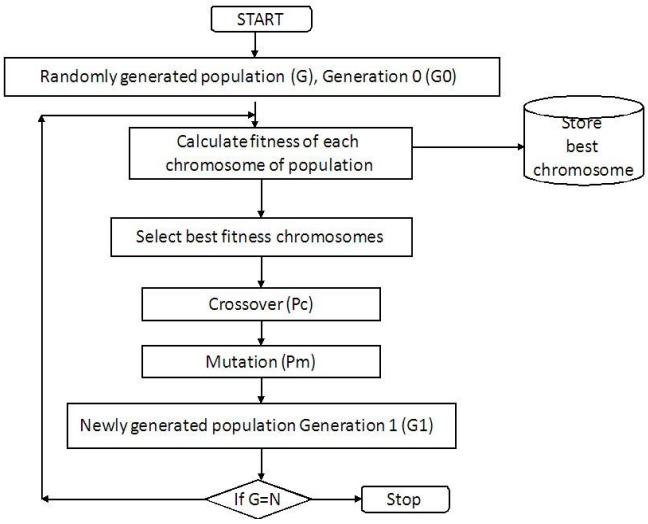

Figure 2: Schematic diagram of GA

Fitness Functions for SGA and OpenMP GA [35], [36] are the multimodal test functions. For example equation (1) is the Rosenbrock Test Function F(x) as an optimization problem. The global optimum lies inside a long, narrow, parabolic shaped flat valley.

$$f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \quad (1)$$

Test area -2.048<= x_i <= 2.048; where i=1,…, n; where n: Dimension. Its global minimum equal $f(x) = 0$ is obtainable for x_i, where $i = 1,\ldots,n$.

3.2 DPGA

DPGA is by design parallel but implemented serially [37]. DPGA has two distinct populations called main population M_p and reserve population R_p. Main population is having same role as that of the population of an ordinary GA. Main population aim to evolve to find a good solution with a high fitness value. The reserve population provides additional diversity to the main population. The reserve population is employed as a reservoir for keeping chromosomes which are different from those of the main population.

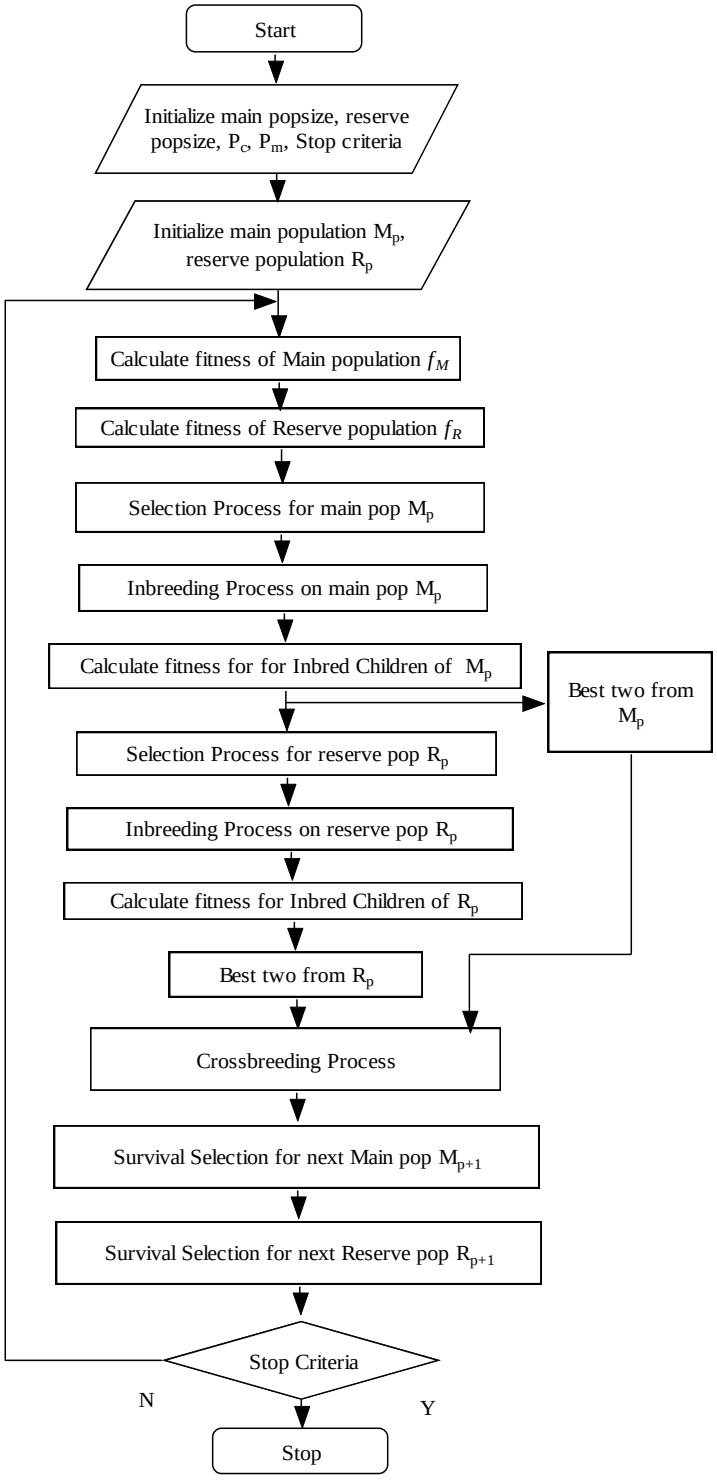

Figure 3: Schematic diagram of serial DPGA

Table 1. Parameter setting for algorithms

Initial Parameters	SGA	OpenMP GA	DPGA
Main Population size	100	100	100
Reserve Population size	--	--	100
Crossover Method	Single point	Single point	Single point
Crossover point K	1	1	1
Mutation Method	Flip Bit	Flip Bit	Flip Bit
Crossover Probability (Pc)	0.7	0.7	0.7
Mutation Probability (Pm)	0.03	0.03	0.03
Dimension (D)	5	5	5
No of Generation	2000	2000	2000

Table 2. Problem Set- continuous non linear large scale benchmark problems

Function Name	Equation	Range	Optimum value
Rosenbrock (F1)	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	[-2.048 ,2.048]	0
Ackley (F2)	$f(x) = -a.\exp(-b.\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(cx_i)) + a + \exp(1)$	[-32.768 ,32.768]	0
Griewangk (F3)	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600 ,600]	0
Schwefel (F4)	$f(x) = \sum_{i=1}^{n} [-x\sin(\sqrt{\lvert x_i \rvert}]$	[-500 ,500]	-418.9829
Rastrigin (F5)	$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$	[-5.12 ,5.12]	0

Table 3. Best Solution found and Execution Time for multimodal test functions

Test function	Best Solution Found for SGA	Best Solution Found for OpenMP GA	Best Solution Found for DPGA	Time (sec) taken by SGA	Time (sec) taken by OpenMP GA	Time (sec) taken by DPGA
F1	2.47E+00	4.79E+00	1.03E+00	11.38	4.32	3.45
F2	1.38E-02	2.55E-03	1.30E-03	11.76	10.59	10.04
F3	1.08E+00	3.25E+00	3.23E+00	10.75	4.61	3.65
F4	-1.91E+03	-1.83E+03	-1.82E+03	10.29	3.94	3.02
F5	5.18E-02	9.83E-01	4.96E-02	22.32	10.65	10

Table 4. Speed up gained for multimodal functions in comparison with SGA and OpenMP GA

Test Function	Speed up of DPGA compared with OpenMP GA (%)	Speed up of DPGA compared with SGA (%)
F1	20.13	69.58
F2	05.20	14.62
F3	20.82	66.04
F4	23.35	70.65
F5	15.12	55.20

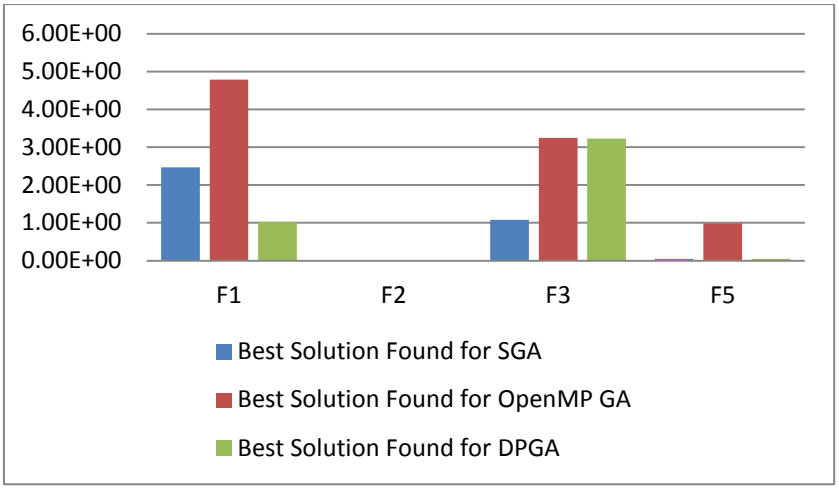

Figure 4: Best solutions found by DPGA, OpenMP GA and SGA on multimodal test functions F1, F2, F3 and F5

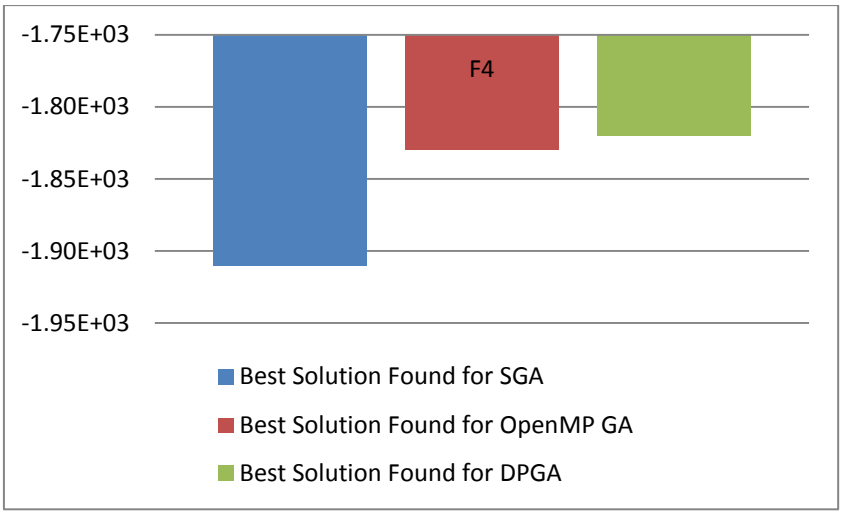

Figure 5: Best solutions found by DPGA, OpenMP GA and SGA on multimodal test functions F4

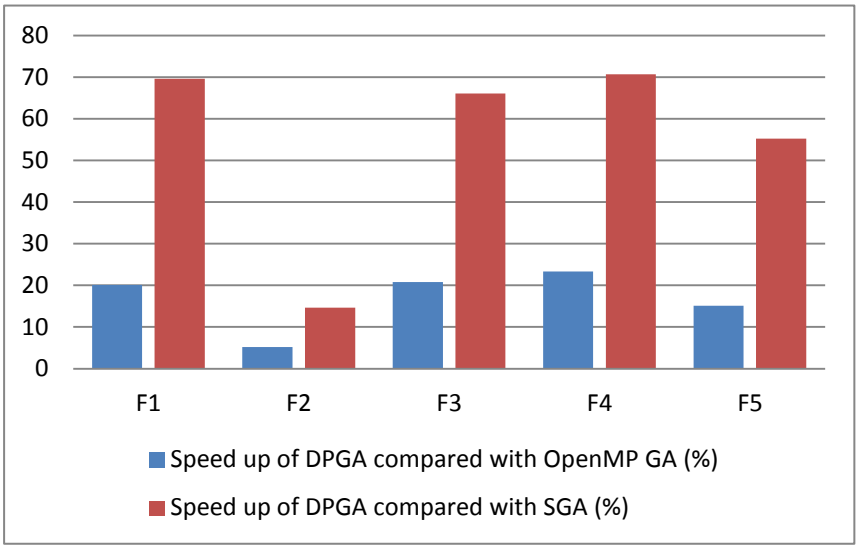

Figure 6: Speed up gained in DPGA over OpenMP GA and SGA for multimodal test functions

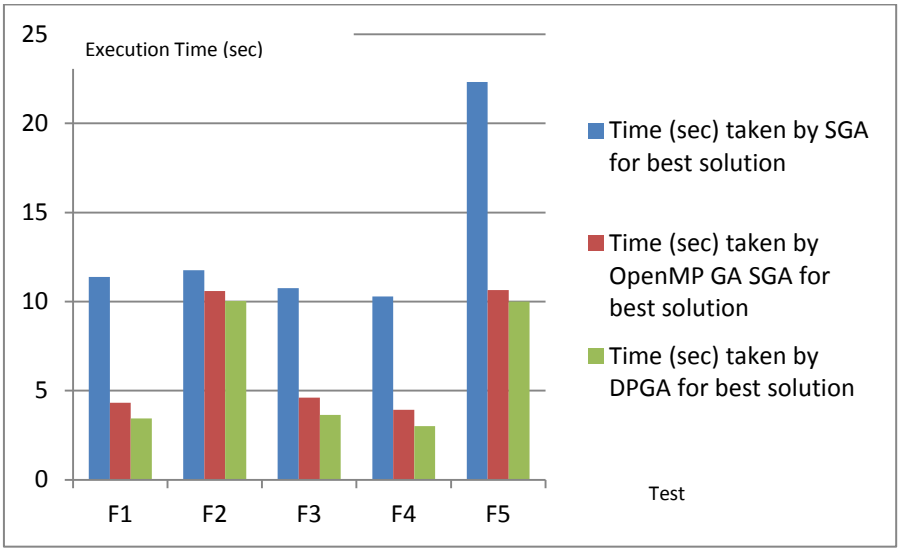

Figure 7: Execution Time (in seconds) comparison for SGA, OpenMP GA & DPGA to find BSF for multimodal test functions

Fitness Functions for Main Population $f_M(x)$ [35], [36]: The fitness function for main population is multimodal test functions. . For example equation (1) is the Rosenbrock Test Function F(x) as an optimization problem.

Fitness Function for Reserve Population $f_R(x)$: [38] [39]

$$f_R(x) = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{l}\left|m_{i,k} - x_k\right| \quad (2)$$

x_k : k^{th} gene of chromosome x from R_P.

$m_{i,k:}$ k^{th} gene of chromosome mi from M_P.

n: Population Size of M_P and R_P

l : Chromosome length.

The equation (2) calculates the hamming distance of the k^{th} gene chromosome from M_P with k^{th} gene of chromosome x from R_P and assigns the fitness for reserve populating.

The implementation of DPGA is carried out serially. Figure 3 shows the schematic diagram of the serial DPGA. SDPGA is single threaded DPGA, so the exploration, exploitation, migration etc. are sequential.

4. EXPERIMENTAL RESULTS

4.1 Test Data

The algorithm was evaluated on multimodal problems given in [35], [36]. The experimentation is carried out with five benchmark functions. The F1, F2, F3, F4 and F5 are multimodal functions. The test problem Set is shown in Table 2.

[5] http://www.cs.utah.edu/dept/old/texinfo/as/gprof.htm

4.2 Results and Discussion

In the experimentation, SGA, OpenMP GA and DPGA are implemented in C language over Linux platform on a Personal Computer (Intel Intel(R) Core (TM) i5 CPU 650 @ 3.20GHz processor with 2 cores and 4MB cache, 1 GB of memory and 300 GB hard disk). Table 1 shows the parameter setting for SGA, OpenMP GA and DPGA algorithms. Comparison between SGA and OpenMP GA is made on the basis of execution time taken and speedup obtained. Equation number (3) is the formula for speed up calculation with respect to SGA. Equation number (4) is the formula for speed up calculation with respect OpenMP GA.

$$\text{Speed Up} = \frac{(\text{Time taken by SGA} - \text{Time taken by DPGA}}{\text{Time taken by SGA}} \times 100 \quad (3)$$

$$\text{Speed Up} = \frac{(\text{Time taken by OpenMP GA} - \text{Time taken by DPGA GA})}{\text{Time taken by OpenMP GA}} \times 100 \quad (4)$$

Execution time is observed using GPprof software[5]. It is a performance analyzing and profiling tool, which collects and arranges statistics of our programs like execution time, CPU utilization, function calls, function wise memory and CPU utilization [40].

Table 3 shows the comparison based on Best Solution Found and execution time in sec by SGA, OpenMP GA and DPGA. Figure 4 is a plot of Best Solution Found verses various multimodal test functions F1, F2, F3 and F5 considered. It shows that DPGA is outperforming in comparison with SGA and OpenMP GA for multimodal test functions except F3 in terms of quality of solution. Figure 5 is a plot of best solution found verses various multimodal test function F4. It shows that DPGA is outperforming in comparison with SGA and OpenMP GA for multimodal test function F4 in terms of quality of solution.

Table 4 shows the comparison based on speed up achieved in percentage by DPGA over OpenMP GA and SGA for

multimodal test functions. Figure 6 is a plot of Speed up in percentage verses various multimodal test functions considered. It shows that DPGA outperforms in comparison with SGA and OpenMP GA for multimodal test functions in terms of speed up. It also shows that speedup achieved with DPGA against SGA is higher than OpenMP GA. Figure 7 is a plot of execution time in second verses various multimodal test functions considered. It shows that the DPGA is outperforming in comparison with SGA and OpenMP GA in terms of execution time taken to find best solutions.

5. CONCLUSIONS

Paper is representing the fact that the parallel hardware as well as algorithm with appropriate programming language will give optimal results, optimal resource utilization with speed up. The performance of the DPGA is remarkably superior to that of the SGA and OpenMP GA. The multimodal functions take less time on DPGA than OpenMP GA and SGA to get the best solution. The quality of solution shown by DPGA is better as compare to SGA and OpenMP GA except F3 for given generation. The Speed up achieved by DPGA in comparison with SGA and OpenMP GA is remarkable. The average speed up shown by DPGA over SGA is ~55 % and over OpenMP GA is 15%

The niching technique used in DPGA gives better result than parallel GA. The algorithmic design is more important than only parallel implementation. The combination of niching technique and parallelization in GA will gives better results than the niching and parallelization alone.

Future scope is to experiment niching technique and parallelization in GA for Unimodal and multimodal function optimization.

6. ACKNOWLEDGMENTS

Authors are grateful to T. Park and K.R. Ryu, Pusan National University, Dept. of Computer Engineering, Jangjeon-Dong San 30, Gumjeong-Gu, Busan. Without their DPGA work, this work would have been impossible. Authors also express thanks to all reviewers for their valuable reviews and comments.

7. REFERENCES

[1] August A.D., Chiou K.P.D, Sendag R., Yi J.J., (2010). "Programming Multicores: Do Application Programmers Need to Write Explicitly Parallel Programs?", Computer Architecture Debates in IEEE MICRO, pp. 19-32.

[2] Konfrst Z., (2004). "Parallel Genetic Algorithm: Advances, Computing Trends, application and Perspective", In proceeding of 18th International Parallel and Distributed Processing Symposium [IPDPS'04], IEEE Computer Society.

[3] Cant´u-Paz E., (2000). "Efficient and Accurate Parallel Genetic Algorithms", Kluwer Academic Publishers.

[4] Cantú-Paz E., (2002). "A Survey of Parallel Genetic Algorithms", Department of Computer Science and Illinois Genetic Algorithms Laboratory University of Illinois at Urbana-Champaign.

[5] Farmer J.D., Packard N., Perelson A., (1986). "The immune system, adaptation and machine learning", Physica 22 pp.187–204.

[6] Dorigo M., (1992). "Optimization, learning and natural algorithms", PhD Dissertation, Politecnico di Milano, Italy.

[7] Kennedy J., Eberhart R.C., (1995). "Particle swarm optimization", Proceedings IEEE International Conference on Neural Networks, Piscataway, pp. 1942–1948.

[8] Hykin S. S., (1999). Neural Network: A comprehensive Foundation, Prentice hall, pp. 1-889.

[9] Storn R., Price K., (1997). "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces", J. Glob. Optim. 11, pp. 341–359.

[10] Geem Z.W., Kim J.H., Loganathan G.V., (2001). "A new heuristic optimization algorithm: Harmony Search Simul", the Soc. for Model and Simul. Int. 76(2), pp. 60–68.

[11] Passino K.M., (2002). "Biomimicry of bacterial foraging for distributed optimization and control", IEEE Control Syst. Mag. 22, pp.52–67.

[12] Eusuff E., Lansey E., (2003). "Optimization of water distribution network design using the shuffled frog leaping algorithm", J. Water Resour. Plan Manag. ASCE 129, pp. 210–225.

[13] Karaboga D., (2005). "An idea based on honey bee swarm for numerical optimization", Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey.

[14] Simon D., (2008). "Biogeography-based optimization", IEEE Trans Evol. Comput. 12, pp. 702–713.

[15] Rashedi E., Nezamabadi-pour H., Saryazdi S., (2009). "GSA: a gravitational search algorithm" Inf Sci 179, pp. 2232–2248.

[16] Ahrari A., Atai A., (2010). "Grenade Explosion Method-A novel tool for optimization of multimodal functions", Appl Soft Comput 10(4), pp. 1132–1140.

[17] Rao R.V., Savsani V.J., (2012). "Mechanical Design Optimization Using Advanced Optimization Techniques", Springer Series in Advanced Manufacturing, Springer-Verlag London.

[18] Yu X., Gen M., (2010) Introduction to Evolutionary Algorithms, Springer-Verlag London Limited, pp. 105-111.

[19] Luque G., Alba E., (2011). Parallel Genetic Algorithms, Springer.

[20] Gagn´e C., Parizeau M., Dubreuil M., (2003). "The Master-Slave Architecture for Evolutionary Computations Revisited", Proceedings of the Genetic and Evolutionary Computation Conference, Chicago, IL, 2, pp. 1578-1579.

[21] Lin S., Punch W., Goodman E., (1994). "Coarse-grain parallel genetic algorithms: categorization and analysis", In IEEE Symposium on Parallel and Distributed Processing, pp. 27–36.

[22] Hauser R., Männer R., (1994). "Implementation of Standard Genetic Algorithms on MIMD Machines", In Parallel Problem Solving from Nature [PPSN3], pp. 504-513.

[23] Tanese R., (1987). "Parallel Genetic Algorithm for a Hypercube", In Proceedings of the Second International

Conference on Genetic Algorithms [ICGA2], pp. 177-183.

[24] Tanese R., (1989). "Distributed Genetic Algorithms", In Proceedings of the Third International Conference on Genetic Algorithms [ICGA3], pp. 434-439.

[25] Voigt H. M., Born J., Santibanez-Koref I., (1991) "Modeling and Simulation of Distributed Evolutionary Search Processes for Function Optimization", In Parallel Problem Solving from Nature [PPSN1], pp. 373-380.

[26] Voigt H. M., Born J., Santibanez-Koref I., (1992). "Hierarchically Structured Distributed Genetic Algorithm", In Parallel Problem Solving from Nature [PPSN2], pp. 145-154.

[27] Imade H., Morishita R., Ono I., Ono N., Okamoto M., (2003). "A grid-oriented genetic algorithm for estimating genetic networks by s-systems", SICE 2003 Annual Conference, 3(4-6), pp. 2750–2755.

[28] Herrera J., Huedo E., Montero R., Llorente I., (2005). "A gridoriented genetic algorithm", In Advances in Grid Computing - EGC 2005, pp. 315–322.

[29] Imade H., Morishita R., Ono I., Ono N., Okamoto M., (2004). "A grid-oriented genetic algorithm framework for bioinformatics", New Gen. Comput., 22(2), pp. 177–186.

[30] Wong M., Wong T., (2006). "Parallel hybrid genetic algorithms on consumer-level graphics hardware", in Congress on Evolutionary Computation, Canada, pp. 2972-2980.

[31] Arora R., Tulshyan R., Deb K., (2010). "Parallelization of binary and real-coded genetic algorithm on GPU using CUDA", in Congress on Evolutionary Computation, pp. 1-8.

[32] Vidal P. Alba E., (2010). "A multi-GPU implementation of a cellular genetic algorithm", in 2010 IEEE Congress on Evolutionary Computation.

[33] Oiso M., Matumura Y., (2011). "Accelerating Steady-state genetic algorithms based on CUDA architecture", , in 2011 IEEE Congress on Evolutionary Computation, pp. 687-692.

[34] Zheng L., et. al. (2011). "Architecture-based Performance Evaluation of Genetic Algorithms on Multi/Many-core Systems", In proceeding of: 14th IEEE International Conference on Computational Science and Engineering, CSE 2011, Dalian, China, pp. 321-334.

[35] Molga M., Smutnicki C., (2005) "Test functions for optimization needs- 2005, unpublished.

[36] Mohan C., Deep K., (2009). "Optimization Techniques" first edition, New Age International Publication.

[37] Umbarkar A.J. and Joshi M.S., (2012). "Serial DPGA vs Parallel Multithreaded DPGA: Threading Aspects", in Proceedings of the International Conference on Soft Computing for Problem Solving (SOCPROS 2011)" AISC 130, Springer pp.37-49.

[38] Park T., Ryu K.R., (2007). "A dual population genetic algorithm with evolving diversity", In IEEE Congress on Evolutionary Computation (CEC2007), pp. 3516–3522.

[39] Park T., Ryu K.R., (2008). "Dual population genetic algorithm for Nonstationary Optimization", Proc. Genetic Evol. Comput. Conf. (GECCO'08), pp. 1025-1032, 2008.

[40] Susan L., Graham P.B., Kessler M.K., McKusick, "gprof: a Call Graph Execution Profiler1", Electrical Engineering and Computer Science Department University of California, Berkeley, California.