I.J. Information Engineering and Electronic Business, **2015, 1, 59-65**
Published Online January 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2015.01.08

OpenMP Dual Population Genetic Algorithm for Solving Constrained Optimization Problems

A. J. Umbarkar
Department of Information Technology, Walchand College of Engineering, Sangli, 416-416, India
Email: anantumbarkar@rediffmail.com

M. S. Joshi
Department of Computer Engineering, Jawaharlal Nehru Engineering College, Aurangabad, MS, India
madhuris.joshi@gmail.com

P. D. Sheth
Department of Information Technology, Walchand College of Engineering, Sangli, 416-416, India
Email: pranalisheth@gmail.com

Abstract—Dual Population Genetic Algorithm is an effective optimization algorithm that provides additional diversity to the main population. It deals with the premature convergence problem as well as the diversity problem associated with Genetic Algorithm. But dual population introduces additional search space that increases time required to find an optimal solution. This large scale search space problem can be easily solved using all available cores of current age multi-core processors. Experiments are conducted on the problem set of CEC 2006 constrained optimization problems. Results of Sequential DPGA and OpenMP DPGA are compared on the basis of accuracy and run time. OpenMP DPGA gives speed up in execution.

Index Terms—Dual Population Genetic Algorithm (DPGA), Open Multiprocessing (OpenMP), Constrained Optimization Problems (COPs), High Performance Computing (HPC), Meta-heuristic Algorithms, Function Optimization.

I. INTRODUCTION

Dual Population Genetic algorithms (DPGA) is population based search algorithm that obtains solution to optimization problems. It is a diversity based algorithm that addresses diversity as well as premature convergence problems of Genetic Algorithm (GA). DPGA uses a reserve population along with the main population. This provides additional diversity to the main population. With the existence of two populations, execution time required for evolution is further increased. Recent advancement in High Performance Computing (HPC) includes programming for multi-core processors. These processors can be employed in parallel for faster evolution of the reserve as well as the main population. Information exchange between populations takes place through inter-population crossbreeding [1].

In this paper, Maximum Constraints Satisfaction Method (MCSM) along with DPGA to solve Constrained Optimization Problems (COPs) are used. MCSM is a novel technique which tries to satisfy maximum constrains first and then it tries to optimize an objective function.

OpenMP is an API for writing shared-memory parallel applications in C, C++, and FORTRAN. With the release of API specifications in 1997 uses of multiple cores of multi-core CPU for parallel computing has become easy [2]. The basic execution unit of OpenMP is a thread. OpenMP is an implementation of multithreading. A master thread forks a specified number of slave threads and a task is distributed among them. The designers of OpenMP developed a set of compiler pragmas, directives, and environment variables, function calls which are platform-independent [3]. In application exactly where and how to insert threads are explicitly instructed by these constructs.

Section II gives a brief literature review of DPGA and COPs. Section III describes the proposed algorithm DPGA for solving COPs using OpenMP. Section IV presents experimental results and discussion. Section V gives some conclusions and exhibits future scope.

II. LITERATURE REVIEW

Dual Population Genetic Algorithm for solving COPs is an open research problem. Therefore we studied literature about evolution of DPGA. We have also surveyed how other Evolutionary Algorithms applied to solve COPs.

Park and Ruy (2006) [4] introduced DPGA. It has two distinct populations with different evolutionary objectives: The Prospect (main) population works like population of the regular genetic algorithm which aims to optimize the objective function. The Preserver (reserve) population serves to maintain the diversity. Park and Ruy (2007) [5] proposed DPGA-ED that is an improved design-DPGA. Unlike DPGA, the reserve population of DPGA-ED evolves by itself. Park and Ruy (2007) [6] proposed a method to dynamically set the distance between the populations using the distance between good parents. Park and Ruy (2007) [7] exhibited DPGA2 that utilizes two reserve populations. Park and Ruy (2010) [1] experimented DPGA on various classes of problems using binary, real-valued, and order-based representations. Umbarkar and Joshi (2013) [8] compared DPGA with OpenMP GA for Multimodal Function Optimization. The results show that the performance of OpenMP GA better than SGA on the basis of execution time and speed up.

The basic and classical constrained optimization methods include penalty function method, Lagrangian method [9] and Sequential Quadratic Programming (SQP) [10]. These are local search methods which can find a local optimal solution.

Recent trend is to ensure that the use of evolutionary algorithms to solve constrained optimization problems [11, 12]. Comparing with the traditional nonlinear programming approach, evolutionary algorithms need less information such as gradient (derivatives), as well as it is a global searching approach. According to Koziel and Michalewicz [13], these algorithms can be grouped into four categories:

i. Methods based on preserving feasibility of solutions by transforming infeasible solutions to feasible solutions with some operators [14]
ii. Methods based on penalty functions which introduce a penalty term into the original objective function to penalize constraint violations in order to solve a constrained problem as an unconstrained problem [15–18]
iii. Methods which make a clear distinction between feasible and infeasible solutions [19–23]
iv. Methods based on evolutionary algorithms [24-36]
v. Other hybrid methods combining evolutionary computation techniques with deterministic procedures for numerical optimization [37–40]

III. Proposed Algorithm

Sequential programming mostly can make use of only single core of multi-processor, which leaves other cores idle. To achieve complete resource utilization, algorithm can be built in parallel either by design or by multithreaded programming. This paper discusses the parallel implementation of DPGA using OpenMP. OpenMP maps threads onto physical cores of CPU; hence all the OpenMP threads run parallel and provide an optimal solution in less amount of time. This technique reduces execution time and provides speed up compared to sequential DPGA. The Sequential DPGA algorithm is indicated in Fig 2.

DPGA starts with two randomly generated populations viz., main population and reserve population. The individuals of each population are evaluated by their own fitness functions. The evolution of each population is obtained by inbreeding between parents from the same population, crossbreeding between parents from different populations, and survival selection among the obtained offspring [1].

The methodology for applying DPGA for COPs is described in this section. A detailed pseudo code is explained in Fig.2 entitled DPGA_MCSM. MCSM is a novel technique based on Deb's rule that in the category of methods searching for feasible solutions. It states any feasible solution is better than any infeasible one [23]. According to the first phase of MCSM, Pseudo Random Number Generator (PRNG) selects variables to prepare individuals which satisfy all constraints. According to the second phase of MCSM, meta-heuristic DPGA is applied for evolution of both populations via inbreeding. After crossbreeding of individuals from both populations, DPGA evolves till stopping criteria are met.

The objective function is used as fitness function for evolution of the main population. Fitness function for the reserve population is defined in another way. An individual in the reserve population is given a high fitness value if its average distance from each of the individuals of the main population is large. Therefore the reserve population can provide the main population with additional diversity.

Equation (1) describes fitness function for reserve population. Each individual of the reserve population can maintain a given distance δ from the individuals of the main population [1, 4, 5].

$$fr_{\delta}(x) = 1 - |\,\delta - d(M, x)\,| \qquad (1)$$

Where,
d(M, x): average distance $(0 \le d(M, x) \le 1)$ between the main population M and individual *x* of the reserve population
δ: desired distance $(0 \le \delta \le 1)$ between two population

The OpenMP DPGA algorithm exactly emulates the sequential algorithm stated in Fig.2 except for fitness calculation function, wherein small changes are introduced to get better performance.

This method evaluates fitness of each individual using a defined objective function. As fitness evaluation of each individual is an independent step it can be executed in parallel. OpenMP parallel directives are used to create multiple threads, ranging from 2 to 16. These threads to evaluate each individual's fitness separately. In this way, this step reduces the overall time required to evaluate the fitness of all individuals in the population. Thus it helps to reduce the amount of time required for total execution significantly.

IV. Experimental Results and discussion

The results are taken on AMD FX(tm)-8320 Eight-Core processor which is of 3.51 GHz clock speed. The machine is equipped with 16 GB RAM and hard disk of capacity 500 GB. Operating system used is CentOS Release 6.5 with Kernel Linux 2.6.32-431.6l6.x86_64 and GNOME version 2.28.2. The IDE used is Eclipse-CDT with gcc compiler.

We take standard problems to conduct experiments from Problem Definitions and Evaluation Criteria for the

Congress on Evolutionary Computation 2006 Special Session on Constrained Real-Parameter Optimization problem [41]. In this report, 24 benchmark functions are described. Guidelines for conducting experiments, statistical parameters and its formulae, performance evaluation criteria are given at the end of this report.

Table 1. Function Description

Function	D	Type	LI	NI	LE	NE
g01	13	Quadratic	9	0	0	0
g04	5	Quadratic	0	6	0	0
g06	2	Cubic	0	2	0	0
g08	2	Nonlinear	0	2	0	0
g09	7	Polynomial	0	4	0	0
g10	8	Linear	3	3	0	0
g18	9	Quadratic	0	13	0	0
g24	2	Linear	0	2	0	0

Table I describes 8 functions from CEC 2006 that we have implemented using DPGA_MCSM. It describes function name, dimension (D), the type of function, no. of linear inequality constraints (LI), no. of non-linear inequality constraints (NI), no. of linear equality constraints (LE) and no. of non-linear equality constraints (NE).

Table 2. Parameter Settings

Crossover Rate	0.80	Elitism Rate	0.10
Mutation Rate	0.09	Crossbreed Rate	0.10
Main Pop. Size	1000	Reserve Pop. Size	1000

Table II describes the parameter settings used for experimentation. All 8 functions use parameter values from Table II. Consecutive 30 runs are calculated for each function keeping these parameter values constant. Size of the main population as well as the reserve population is taken as 1000. Crossover rate, elitism rate, mutation rate and crossbreed rate are kept identical for both the main population and the reserve population.

The performance of OpenMP DPGA is checked based on the quality of solution found and speed up given by DPGA as compare to the Sequential DPGA.

1) Convergence of Proposed OpenMP DPGA Algorithms

Proposed parallel algorithm gives a solution of problems close to the optimal solution obtained as a result of the sequential algorithm. Table III shows Optimal Solutions Found (OSF) in sequential as well as OpenMP DPGA algorithm. For OpenMP DPGA, the results are taken on 2, 4, 8 and 16 cores. Results show that, OpenMP DPGA algorithm converges and produces the same and sometimes better solution as that of the Sequential DPGA algorithm.

2) Increase in Number of Cores

OpenMP DPGA for COPs using MCSM experimented on different multi-core machines. Table II gives the speed-up using 2, 4, 8 and 16 cores machine for the test problems.

Speed-up measures performance gain is achieved by parallelizing a given application over sequential implementation. The speed-up is the ratio of sequential run time to parallel run time (equation 2).

Table 3. OSF in Sequential, OpenMP DPGA

Func-tion	Sequential DPGA OSF	OSF in OpenMP DPGA			
		2 Cores	4 Cores	8 Cores	16 Cores
g01	-10.87	-10.87	-11.00	-10.88	-10.70
g04	-30490.1	-30500.2	-30288.3	-30478.4	-30145.2
g06	-6691.47	-6690.47	-6790.47	-6680.47	-6614.47
g08	-0.08918	-0.09200	-0.08888	-0.08718	-0.08874
g09	663.23	662.14	663.47	661.14	660.78
g10	5876.11	5877.11	5886.22	5864.47	5861.33
g18	-0.75926	-0.74426	-0.72614	-0.72476	-0.74696
g24	-5.43	-5.44	-5.43	-5.42	-5.41

$$S = \frac{T_s}{T_p} \qquad (2)$$

The results of speed-up obtained on 2, 4, 8, 16 cores machines by OpenMP DPGA for 8 test functions of CEC 2006 are shown in Table IV.

Table 4. Speed ups obtained by OpenMP DPGA on 2-16 cores system

Function	Speed Up by OpenMP DPGA			
	2 Cores	4 Cores	8 Cores	16 Cores
g01	1.97	2.12	2.15	2.10
g04	4.40	4.40	4.40	4.40
g06	1.40	1.60	1.40	1.40
g08	0.90	1.00	1.00	1.00
g09	1.00	1.00	1.00	1.00
g10	1.00	1.00	1.00	1.00
g18	1.00	1.10	1.10	1.10
g24	1.00	1.10	1.10	1.10

It is observed that, as the number of cores increases, speed-up increases and it remains constant limited by sequential part of the code. The corresponding graph in Fig.1 shows the speedup for the average of the same 8 test functions.

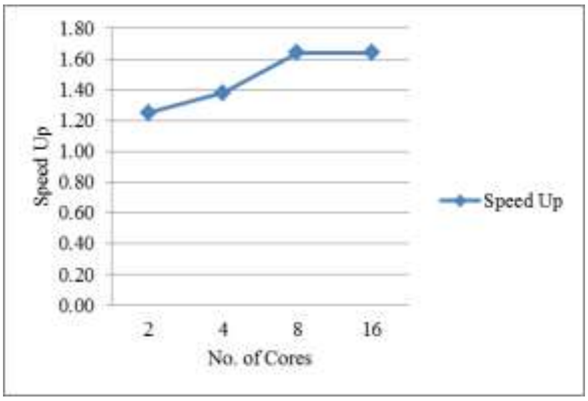

Fig.1. Speedup versus number of cores

In the above experiment, keeping problem size (population size) constant, no. of cores (processors) are increased in multiples of 2. From literature, the overhead time is time required for creating threads and some other initialization and maintenance tasks. It is formulated as

$$To = pTp - Ts \quad (3)$$

Where, To is overhead time, Tp is parallel run time required for p processors and Ts is part of code that cannot be parallelized. The overhead time is function of the increment in no. of processors. As Amdahl's Law says this increased To limits speed up and efficiency and is demonstrated in Table V.

V. Conclusion

OpenMP DPGA is a novel technique for solving COPs. Its aim is to use all available cores of current age multi-core processors. Experiments conducted using CEC 2006 problems set show increase in speed up with increase in no. of processors till certain no. of processors and then speed up remains constant. This behavior emulates Amdahl's Law.

In future, using alternative constraints handling method, parallel algorithm and high performance computing paradigm a better speed up can be achieved.

APPENDIX A.

g01: Minimize

$$f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} {x_i}^2 - \sum_{i=5}^{13} x_i$$

Subject to

$g_1(\vec{x}) = 2x_1 + 2x_2 + 2x_{10} + 2x_{11} - 10 \le 0$

$g_1(\vec{x}) = 2x_1 + 2x_2 + 2x_{10} + 2x_{11} - 10 \le 0$

$g_2(\vec{x}) = 2x_1 + 2x_3 + 2x_{10} + 2x_{12} - 10 \le 0$

$g_3(\vec{x}) = 2x_2 + 2x_3 + 2x_{11} + 2x_{12} - 10 \le 0$

$g_4(\vec{x}) = -8x_1 + x_{10} \le 0$

$g_5(\vec{x}) = -8x_2 + x_{11} \le 0$

$g_6(\vec{x}) = -8x_3 + x_{12} \le 0$

$g_7(\vec{x}) = 2x_4 - x_5 + x_{10} \le 0$

$g_8(\vec{x}) = 2x_6 - x_7 + x_{11} \le 0$

$g_9(\vec{x}) = 2x_8 - x_9 + x_{12} \le 0$

where the bounds are 0 <= xi <= 1 (i=1,……,9), 0 <= xi <= 100 (i = 10, 11, 12) and 0 <= x13<=1 .

g04: Minimize

$f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1 + x_5 + 37.293239x_1$
-40792.141

Subject to

$g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4$
$+0.0022053x_3x_5 - 92 \le 0$

$g_2(\vec{x}) = -85..334407x_1 - 0.0056858x_2x_5$
$-0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0$

$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2$
$-0.0021813x_3^2 - 110 \le 0$

$g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2$
$-0.0021813x_3^2 + 90 \le 0$

$g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3$
$+0.0019085x_3x_4 - 25 \le 0$

$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3$
$-0.0019085x_3x_4 + 20 \le 0$

where 78 <= x_1<= 102, 33 <= x_2<= 45 and 27 <= x_i<= 45 (i = 3, 4, 5).

g06: Minimize

$f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$

Subject to

$g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$

$g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$

where 13 <= x1<= 100 and 0 <= x2<= 100.

g07: Minimize

$f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2$
$+4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2$
$+7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$

Subject to

$g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0$

$g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0$

$g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0$

$g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0$

$g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0$

$g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0$

$g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0$

$g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0$

where -10 <= xi<= 10 (i = 1,…., 10).

g08: Minimize

$$f(\vec{x}) = -\frac{sin^3(2\pi x_1)sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

Subject to

$g_1(\vec{x}) = x_1^2 - x_2 + 1 \le 0$

$g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \le 0$

where 0 <= x1<=10 and 0 <= x2<= 10.

g09: Minimize

$f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2$
$+x_5^7 - 4x_6x_7 - 10x_6 - 8x_7$

Subject to

$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0$

$g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0$

$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0$

$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$
where- 10 <= xi<= 10 for (i = 1,….., 7).

g10: Minimize
$f(\vec{x}) = x_1 + x_2 + x_3$
Subject to
$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$
$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$
$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0$
$g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$
$g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$
$g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$
where 100 <= x_1<= 10000, 1000 <= x_i<= 10000 (i = 2,3) and 10 <= x_i<= 1000 (i = 4,…,8).

g18: Minimize
$$f(\vec{x}) = -0.5\begin{pmatrix} x_1x_4 - x_2x_3 + x_3x_9 - \\ x_5x_9 + x_5x_8 - x_6x_7 \end{pmatrix}$$
Subject to
$gg1(\vec{x}) = x2 + x2 - 1 \leq 0$
$gg_6(\vec{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0$
$gg_7(\vec{x}) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0$
$gg_8(\vec{x}) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0$
$gg_9(\vec{x}) = x^2 + (x_8 - x_9)^2 - 1 \leq 0$
$gg_{10}(\vec{x}) = x_2x_3 - x_1x_4 \leq 0$
$gg_{11}(\vec{x}) = -x_3x_9 \leq 0$
$gg_{12}(\vec{x}) = x_5x_9 \leq 0$
$gg_{13}(\vec{x}) = x_6x_7 - x_5x_8 \leq 0$
where the bounds are $-10 \leq xi \leq 10$ ($i = 1, \ldots, 8$) and $0 \leq x9 \leq 20$.

g24: Minimize
$f(\vec{x}) = -x_1 - x_2$
Subject to
$g_1(\vec{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$
$g_2(\vec{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$
where the bounds are 0 <= x1<= 3 and 0 <= x2<= 4.

ACKNOWLEDGMENT

We express our sincere thanks to all authors, whose papers in the area of evolutionary algorithm/DPGA and COPs is published in various conference proceedings and journals. Our special thanks to T. Park, R. Choe and K.R. Ryu for their work on Dual Population Genetic Algorithm published in various journals and conferences.

REFERENCES

[1] T. Park, K. Ruy, "A Dual-Population Genetic Algorithm for Adaptive Diversity Control", IEEE transactions on evolutionary computation, vol. 14, no.6, pp 865-883, Dec. 2010, pp 865-883. DOI:10.1109/TEVC.2010.2043362.
[2] http://openmp.org, June 2013.
[3] https://software.intel.com, June 2013.
[4] T. Park, K. Ruy, "A Dual-Population Genetic Algorithm for Balance Exploration and Exploitation", Acta Press Computational Intelligence, 2006.
[5] T. Park and K. Ruy, "A dual population genetic algorithm with evolving diversity", in Proc. IEEE Congress Evolutionary Computation, 2007, pp. 3516–3522. DOI: 10.1109/CEC.2007.4424928.
[6] T. Park and K. Ruy, "Adjusting population distance for dual-population genetic algorithm," in Proc. Aust. Joint Conf. Artificial Intelligence, 2007, pp. 171–180. DOI:10.1109/TEVC.2010.2043362.
[7] T. Park, R. Choe, K. Ruy, "Dual-population Genetic Algorithm for Nonstationary Optimization", in Proc. GECCO'08 ACM, 2008, pp.1025-1032. DOI: 10.1145/1389095.1389286.
[8] A. Umbarkar, M. Joshi, "Dual Population Genetic Algorithm (GA) versus OpenMP GA for Multimodal Function Optimization", International Journal of Computer Applications, vol. 19, no. 64, February 2013, pp. 29-36. DOI: 10.5120/10744-5516.
[9] D. Luenberger and Y. Ye, "Linear and nonlinear programming third edition", New York, NY: Springer, 2007. ISBN 978-0-387-74503-9.
[10] P. Boggs and J. Tolle, "Sequential quadratic programming," Acta Numerica, vol.4, no.1, Jan. 1995, pp.1-51.
[11] C. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," Comput. Methods Appl. Mech. Eng., vol.191, no.11-12, Jan. 2002, pp.1245-1287. DOI: 10.1016/j.bbr.2011.03.031.
[12] H. Lu and W. Chen, "Self-adaptive velocity particle swarm optimization for solving constrained optimization problems" J. Global Optimiz., vol.41, no.3, Jul. 2008, pp.427-445. DOI: 10.1007/s10898-007-9255-9.
[13] S. Koziel, Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization", Evolutionary Computation, vol.7, no.1, 1999, pp.19–44. DOI: 10.1162/evco.1999.7.1.19.
[14] Z. Michalewicz, C.Z. Janikow, "Handling constraints in genetic algorithms, in: R.K. Belew, L.B. Booker (Eds.)", In Proc. of the Fourth International Conference on Genetic Algorithms (ICGA-91), San Mateo, California, University of California, San Diego, Morgan Kaufmann Publishers, 1991, pp. 151–157.
[15] Z. Michalewicz, M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems", Evolutionary Computation, vol.4, no.1, 1996, pp.1–32. DOI: 10.1162/evco.1996.4.1.1.
[16] A. Homaifar, S.H.Y. Lai, X. Qi, "Constrained optimization via genetic algorithms", Simulation, vol.62, no.4, 1994, pp.242–254. DOI: 10.1177/003754979406200405.
[17] J. Joines, C. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs, in: D. Fogel (Ed.)", In Proc. of the first IEEE Conference on Evolutionary Computation, IEEE Press, Orlando, Florida, 1994, pp. 579–584.
[18] Z. Michalewicz, N. Attia, "Evolutionary optimization of constrained problems", In Proc. of the 3rd Annual Conference on Evolutionary Programming, World Scientific, 1994, pp. 98–108. DOI: 10.1.1.141.2355.
[19] J. Richardson, M. Palmer, G. Liepins, M. Hilliard, "Some guidelines for genetic algorithms with penalty functions", In J. D. Schaffer (Ed.), Proc. of the third International Conference on Genetic Algorithms (ICGA-89), George Mason University, Morgan Kaufmann Publishers, San Mateo, California, June, 1989, pp. 191–197.
[20] M. Schoenauer, S. Xanthakis, "Constrained GA optimization", In S. Forrest (Ed.), Proc. of the fifth International Conference on Genetic Algorithms (ICGA-93), University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, San Mateo, California, July, 1993, pp. 573–580.

[21] D. Powell, M. Skolnick, "Using genetic algorithms in engineering design optimization with non-linear constraints", In S. Forrest (Ed.), Proc. of the fifth International Conference on Genetic Algorithms (ICGA-93), University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers, San Mateo, California, July, 1993, pp. 424–431.

[22] Z. Michalewicz, G. Nazhiyath, "Genocop III: a co-evolutionary algorithm for numerical optimization with nonlinear constraints", In D.B. Fogel (Ed.), Proc. of the Second IEEE International Conference on Evolutionary Computation, IEEE Press, Piscataway, NJ, 1995, pp. 647–651.

[23] K. Deb, "An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering", vol.186, no.2–4, 2000, pp. 311–338. DOI: 10.1016/S0045-7825(99)00389-8.

[24] S. Elsayed, R. Sarker. D. Essam, "Improved genetic algorithm for constrained optimization," in Proc. of International Conference on Computer Engineering & Systems (ICCES), 2011, pp. 111-115. DOI: 10.1109/ICCES.2011.6141022.

[25] Z. Ziqiang, C. Zhihua, Z. Jianchao, Y. Xiaoguang, "Artificial Plant Optimization Algorithm for Constrained Optimization Problems", in Proc. of Second International Conference on Innovations in Bio-inspired Computing and Applications (IBICA), 2011, pp. 120-123. DOI: 10.1109/IBICA.2011.34.

[26] B. Xiaojun, W. Jue, "Constrained Optimization Based on Epsilon Constrained Biogeography-Based Optimization", in Proc. of 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012, pp. 369-372. DOI: 10.1109/IHMSC.2012.184.

[27] S. Elsayed, R. Sarker, D. Essam, "On an evolutionary approach for constrained optimization problem solving", Applied Soft Computing, vol. 12, 2012, pp. 3208-3227. DOI: 10.1016/j.asoc.2012.05.013.

[28] W. Yong, C. Zixing, "Combining Multiobjective Optimization With Differential Evolution to Solve Constrained Optimization Problems", IEEE Transactions on Evolutionary Computation, vol.16, 2012, pp.117-134. DOI: 10.1109/TEVC.2010.2093582.

[29] R. Rao, V. Patel, "An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems", International Journal of Industrial Engineering Computations, vol.3, no.4, 2012, pp. 535-560. DOI: 10.1016/j.engappai.2012.06.007.

[30] M. Campos, R. Krohling, "Hierarchical bare bones particle swarm for solving constrained optimization problems" in Proc. of IEEE Congress on Evolutionary Computation (CEC-2013), 2013, pp. 805-812. DOI: 10.1109/CEC.2013.6557651.

[31] R. Datta, M. F. P. Costa, K. Deb, A. Gaspar-Cunha, "An evolutionary algorithm based pattern search approach for constrained optimization", in Proc. of IEEE Congress on Evolutionary Computation (CEC-2013), 2013, pp. 1355-1362. DOI: 10.1109/CEC.2013.6557722.

[32] L. Zhenyi and H. Qing, "A numerical gradient based technique and directed neighborhood structure for Constrained Particle Swarm Optimization", in Proc. of American Control Conference (ACC), 2013, pp. 4783-4788. ISBN: 978-1-4799-0177-7.

[33] I. Sardou, M. Ameli, M. Sepasian, M. Ahmadian, "A Novel Genetic-based Optimization for Transmission Constrained Generation Expansion Planning", IJISA, vol.6, no.1, 2014, pp.73-83. DOI: 10.5815/ijisa.2014.01.09.

[34] Y. Zhu, D. Qin, Y. Zhu, X. Cao, "Genetic Algorithm Combination of Boolean Constraint Programming for Solving Course of Action Optimization in Influence Nets", IJISA, vol.3, no.4, 2011, pp.1-7.

[35] M. Mehra, M. Jayalal, A. John Arul, S. Rajeswari, K. Kuriakose, S. Murty, "Study on Different Crossover Mechanisms of Genetic Algorithm for Test Interval Optimization for Nuclear Power Plants", IJISA, vol.6, no.1, 2014, pp.20-28. DOI: 10.5815/ijisa.2014.01.03.

[36] D. Karaboga, B. Akay, "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems", Applied Soft Computing, vol.11, no.3, April 2011, pp. 3021-3031. DOI: 10.1016/j.asoc.2010.12.001.

[37] J. Paredis, "Co-evolutionary constraint satisfaction", In Proc. of the 3rd Conference on Parallel Problem Solving from Nature, Springer-Verlag, NewYork, 1994, pp. 46–55. DOI: 10.1007/3-540-58484-6_249.

[38] I. Parmee, G. Purchase, "The development of a directed genetic search technique for heavily constrained design spaces", In I.C. Parmee (Ed.), Adaptive Computing in Engineering Design and Control-94, University of Plymouth, Plymouth, UK, 1994, pp. 97–102.

[39] H. Myung, J. Kim, D. Fogel, "Preliminary investigation into a two-stage method of evolutionary optimization on constrained problems", In J. McDonnell, R. Reynolds, D. Fogel (Eds.), Proc. of the fourth Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1995, pp. 449–463.

[40] R. Reynolds, Z. Michalewicz, M. Cavaretta, "Using cultural algorithms for constraint handling in GENOCOP, In J. McDonnell, R. Reynolds, D. Fogel (Eds.), Proc. of the fourth Annual Conference on Evolutionary Prog., MIT Press, Cambridge, MA, 1995, pp. 298– 305. ISBN: 9780262290920

[41] J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. Coello Coello, K. Deb, "Problem Definitions and Evaluation Criteria for the CEC 2006 special Session on Constrained Real-Parameter Optimization", IEEE Congress on Evolutionary Computation (CEC-2006), IEEE, Vancouver, BC, Canada, 2006.

Authors' Profiles

A. J. Umbarkar is presently working as an Assistant Professor in Information Technology at Walchand College of Engineering, at Sangli, MS, India. He has completed his Bachelor of Engineering (BE) in Computer Engineering from PICT, Pune, MS, India and Master of Engineering (ME) from Computer Science and Engineering (CSE) from Walchand College of Engineering, at Sangli, MS, India.

He has 13 years of teaching experience at UG and 6 years at PG. His research interests include Parallel Genetic Algorithms, Parallel Evolutionary Algorithms and Parallel programming. He has published 18 research papers in Conferences and Journals.

Madhuri S. Joshi is presently working as a Professor, Department of Computer Engineering, Jawaharlal Nehru Engineering College, Aurangabad, M.S., India. She received her B.E. (Electronics & Telecom) from College of Engineering, Pune (1985), M. Tech. (CS) from IIT Madras (1993) and Ph.D. in 2008. Her research interests are Image Processing, Pattern Recognition, Data Mining, Operating Systems. She has published about 23 research papers in Conferences and Journals.

Pranali Dilip Sheth has completed Bachelor of Technology (B. Tech- 2007) and Master of Technology (M. Tech - 2014) in Information Technology from Walchand College of Engineering, Sangli, MS, India. Her research interests are Parallel Evolutionary Programming, High Performance Computing, Data Mining and Image Processing.

Procedure DPGA_MCSM
begin
Initialize main population M_0, reserve population R_0, crossover rate, elitism rate, mutation rate, crossbreeding rate, tour size, max generation t_{max}
Initialize M_0 of size *m*, accept individuals which satisfies all constrains
Initialize R_0 of size *n*, *n*>*m*
t: = 0
Repeat
Step I: Encoding of both population from decimal value representation to binary value representation
Step II: Fitness Calculation
a. Evaluate M_0 using objective function *fm(x)*
b. Evaluate R_0 using fitness function for reserve population (1) *fr(x)*
Step III: Inbreeding of main population and generate intermediate main population O_m via crossover
Step IV: Inbreeding of reserve population and generate intermediate reserve population O_r via crossover
Step V: Crossbreeding
a. Offspring C of size (n-m) using best individuals from O_m and O_r
b. Make $I_m = C \cup O_m$ and $I_r = C \cup O_r$
Step VI: Decoding: Decoding of both population from binary representation to decimal representation
Step VII: Evaluation
a. Evaluate I_m using *fm(x)*
b. Evaluate I_r using *fr(x)*
Step VIII: Survival selection from I_m of size m and from I_r of size n
t = t + 1
Until
fm(x) > = global optimal value or $t > t_{max}$
End
Where,
t : index of current generation — M_0, R_0: main, reserve population
fm(x): objective function — O_m, O_r: intermediate main, reserve population respectively
fr(x): fitness function for reserve population — I_m, I_r: constitute set of main, reserve population respectively
C: offspring — t_{max}: maximum generations

Fig.2. Pseudo code of DPGA_MCSM

Table 5. Speed up and efficiency achieved by OpenMP DPGA

Function	Speed Up by OpenMP DPGA				Efficiency by OpenMP DPGA			
	2	4	8	16	2	4	8	16
g01	1.97	2.12	2.15	2.10	0.99	0.53	0.27	0.13
g04	1.70	2.10	4.40	4.40	0.85	0.53	0.55	0.28
g06	1.40	1.60	1.40	1.40	0.70	0.40	0.18	0.09
g08	0.90	1.00	1.00	1.00	0.45	0.25	0.13	0.06
g09	1.00	1.00	1.00	1.00	0.50	0.25	0.13	0.06
g10	1.00	1.00	1.00	1.00	0.50	0.25	0.13	0.06
g18	1.00	1.10	1.10	1.10	0.50	0.28	0.14	0.07
g24	1.00	1.10	1.10	1.10	0.50	0.28	0.14	0.07
Average	1.25	1.38	1.64	1.64	0.62	0.34	0.21	0.10

