International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 17 (2015) pp 37707-37716
© Research India Publications. http://www.ripublication.com

Parallel Computing in Genetic Algorithm (GA) with The Parallel Solution of n Queen's Problem Based on GA in Multicore Architecture

Sanjay Saxena
PhD Scholar, School of Biomedical Engineering Indian Institute of Technology (BHU) Varanasi
ssaxena.rs.bme11@itbhu.ac.in

Dr. Neeraj Sharma
Associate Professor, School of Biomedical Engineering Indian Institute of Technology (BHU) Varanasi
neeraj.bme@itbhu.ac.in

Dr. Shiru Sharma
Assistant Professor, School of Biomedical Engineering Indian Institute of Technology (BHU) Varanasi
shiru.bme@itbhu.ac.in

Abstract
Genetic algorithm (GA) is a commanding stochastic optimization technique, which mimics the progression in the living nature with the aim to solve practical problems. It has been successfully applied to solve the several techniques like problem of scheduling, optimization, feature extraction etc. Effective execution of high performance computing or parallel processing makes this technique more valuable. Parallel implementation is very advantageous for the different time consuming process steps of GA. Till now, there have been proposed numerous approach of parallel implementation of genetic algorithm and plenty of research is going on. Main objective of this paper is to summarize existing techniques of parallel accomplishment of GA with the brief introduction of parallel computing and its tools and techniques and to demonstrate significant contribution of parallel computing in GA. This paper also proposed a parallel solution of n queen's problem with significant speedup using GA on easily available multi core architecture. Another main purpose of this paper is to describe the application of parallel GA in medical imaging.

Keywords: Parallel Computing, Genetic Algorithm (GA), n Queen's Problem, Image Processing, Medical Imaging, High Performance Computing.

I. Introduction

High speed computing plays a very essential role in today's era. Parallel computing is a gifted approach to meet the increased requirement for high speed computing [1-6] as it provides concurrency and efficient exploitation of non local resources. GA is very computationally expensive [7] and very time consuming [1] so it is very advantageous to implement parallel computing in this for fast and prominent results [1-5]. Parallel Implementation of GA is very valuable as GA is used in several algorithms like pattern classification, feature extraction [72] and optimization techniques etc. There have been various algorithms developed of parallel implementation of this technique using CPU and GPUs. Main motive of this paper is to provide the efficient description of existing techniques of GA & providing the brief concepts of parallel computing and its applications in medical imaging. In this paper we also tried to show the parallel implantation of n queen's problem using GA on easily available multicore architecture or environment with significant speed up. Structure of the paper is as follows:

Next section II gives the brief introduction of genetic algorithm, parallel computing and its concepts. Section III consist broad classification of parallel GA and important contribution given by different researcher in parallel GA. Section IV describes n queen's problem based on GA and its parallel implementation. Experimental set up and results are given in section V. Relevance of parallel GA in the field of medical imaging is described in section VI. Discussion and Conclusion are discussed in section VII.

II. Brief Concepts

A. *Genetic Algorithm(GA)*

A genetic algorithm is a search technique used in computing to find true or estimated solutions for mostly optimization and search problems. It was introduced by John H. Holland in 1975. It is a stochastic optimization algorithm that is basically builds on principles of natural selection and recombination. GA initialize the initial population randomly and calculate the fitness for each individual then select parent, do crossover mutation operation, calculate fitness for each new individual, finally select the next generation and repeat this process until a termination criteria or up to predefined number of generations [9]. It maximizes the fitness of individuals by employing operations inspired by the theory of evolution.

In common terms, a GA consists of four parts [10].

1. Generate an initial population.
2. Select pair of individuals based on the fitness function.
3. Produce next generation from the selected pairs by applying pre-selected genetic operators.
4. If the termination condition is satisfied stop, else go to step

The termination condition can be either:

1. No improvement in the solution after a certain number of generations.

2. The solution converges to a pre-determined threshold.

Basic flow chart is given below [11 – 21][28].

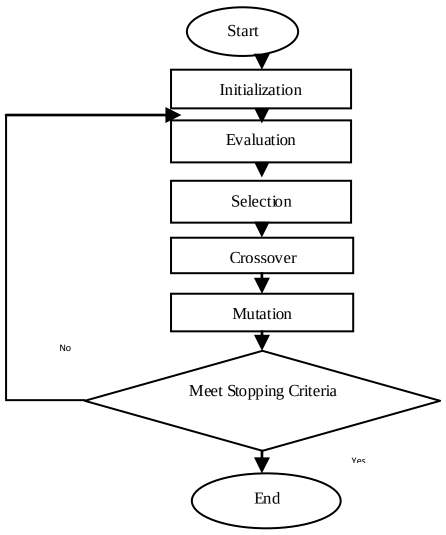

Figure 1: Basic Structure of GA

A practical disadvantage of the genetic algorithm involves longer running time on the computer to execute [28,29,74,75,76]. So parallel computing plays a very vital role in this. It can provide the efficient and fast possible solution of this short come of GA. Now we are going to discuss about parallel computing, key features, tools, techniques and its important concepts.

B. Parallel Computing

It is a type of computing in which several calculations are carried out concomitantly, it is having the principle that a large problem or a composite problem can be separated into smaller one [23]. It can be implemented in many ways of parallel computing environment as a cluster of computers that contains multiple PCs combined together with a elevated speed network, a shared memory multiprocessor by connecting multiple processors to a single memory system, a Chip Multi-Processor (CMP) contains multiple processors (called cores) on a single chip [23].
Very firstly Amdahl framed the concept of parallel computing According to Amdahl's law "if r be the fraction of work done sequentially, so (1-r) is fraction parallelizable" [21].
If number of processors are P

$$\text{Speedup} = \text{Time}(1)/\text{Time}(P) = 1/(r+ (1-r)/P) = 1/r \quad (1)$$

Speed up is the important factor or measuring parameter of parallel computing which measure how the fast is our parallel implementation.

Key Concepts or Fundamentals of Parallel Computing:
Important fundamentals of parallel computing includes

Granularity:
We can define it as the numeral of basic units. It can be coarse grained (Few Tasks of more powerful computing) and Fine Grain (Large Number of Small parts and less powerful computing). Apart from this we can defined parallel processing as two type First is Explicit in which algorithms includes instructions to specify which processes are built and executed in parallel way and Second is Implicit in which compiler has the task of inserting the necessary instructions to run the program on a parallel computer. Synchronization is also necessary for making an algorithm from sequential to parallel as it prevents from the overlapping of two or more processors. One more thing is Latency it is defined as the time conversion of information from request to accept. If we talk about scalability than It is defined as the capability of an algorithm to preserve its effectiveness by escalating the number of processors and the size of the problem in the same percentage. Now a day, GPUs and CUDA are having significant roles in high performance computing [21-27].

C. GPU (Graphical Processing Unit) [29-37][62][63][64]:

A GPU is an assorted chip multi-processor which is specially tuned for graphics. It contains a large amount of dedicated ALUs then CPUs. GPUs also contain extensive support of Stream Processing paradigm. It is related to SIMD (Single Instruction Multiple Data) processing. Each processing unit on GPU contains local memory that improves data manipulation and reduces fetch time.

D. CUDA (Computed Unified Device Architecture) [37 - 44]

It is a set of developing tools to create applications that will perform execution on GPU (Graphics Processing Unit). It is very necessary as it provides ability to use high-level languages such as C with future support of C++ to develop application that can take advantage of high level of performance and scalability that GPUs architecture offer. Compiled code will run directly on GPU. CUDA was developed by NVidia and as such can only run on NVidia GPUs of G8x series and up. Firstly it was released on February 15, 2007 for PC and Beta version for MacOS X on August 19, 2008.
It is having following tools

1. The nvcc C compiler.
2. CUDA FFT (Fast Fourier Transform) and BLAS (Basic Linear Algebra Subprograms for linear algebra) libraries for the GPU.
3. Profiler.
4. An alpha version of the gdb debugger for the GPU.
5. CUDA runtime driver.
6. CUDA programming manual.

It is having several applications in high performance computing like Seismic Database - 66x to 100x speedup given in [47], Molecular Dynamics - 21x to 100x speedup defined in [48]__MRI processing - 245x to 415x speedup in [49] Atmospheric Cloud Simulation - 50x speedup given in [50].Apart from this there are some more tools and techniques are available for implementing parallel processing such as Open CV, Open CL , Parallel Computing Toolbox of

MATLAB, Multithreading of Java and many more with their benefits and limitations discussed in [77].

III. Broad Classification of Parallel GA and Existing Literatures based on Parallel GA

There are different ways of exploiting parallelism in GAs given by different researchers: master- slave models, fine-grained models, island models, and hybrid models, Coarse-grained Model, Dual Species Model. Brief description of this implementation is given below.

A. *Master Slave Models[1][51-53][59]*

Basic model of the master slave modal is given in the following figure. In this approach like simple GA, there is a panmictic population, but the evaluation of fitness is distributed among several worker processors. It is also known as global parallel GA as in this type of parallel GA, selection and crossover considers the entire population.

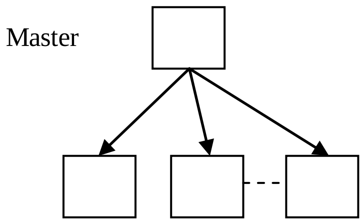

Figure 2: Master Slave Modal

B. *Multiple-Deme Parallel Gas [54 - 56]*

It is also known as multiple populations Genetic Algorithm. It consists on several subpopulations (demes) which exchange individuals occasionally. It is controlled by numerous parameters. It is very famous but difficult to understand because the effects of migration are not fully understood. It introduces the basic changes in the operation of the GA and it is having different behavior than simple GAs. Since the computation to communication ratio is usually high, they are occasionally called coarse-grained GAs [59].Basic architecture is shown in the following figure.

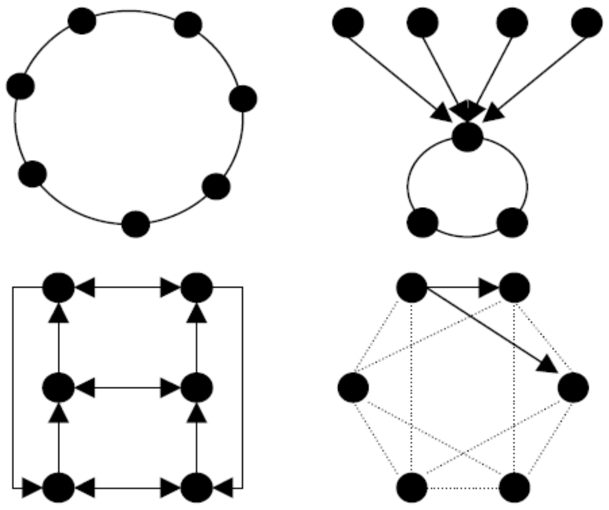

Figure 3: Multiple-deme parallel GA [6]

C. *Fine-Grained Models[57][59]*

This type of approach is suited for massively parallel computers and consists of one spatially-structured population. In this model Selection and mating are restricted to a little neighborhood, but neighborhoods overlap permitting some interaction among all the individuals. The ideal case is to have only one individual for every processing element available.

D. *Island Models[58]*

It is having several subpopulations that exchange individuals occasionally this exchange is called migration and it is controlled by numerous parameters like frequency of migration, number and destination of migrants, and the selection method for choosing the migrants.

E. *Hybrid Model[51 - 60]*

It consists of the combination of Parallel Genetic Algorithm with other optimization technique.

F. *Coarse-grained Model[1][59]*

It divides the population into subpopulations which can be computed on separate processors. The subpopulations then have an amount of migration between them.

G. *Dual Species Model[1][60]*

As show in the following figure this modal generates a population which is further divided in to two sections Operations on the subpopulation 1(p1) and subpopulation 2(p2) are made with different parameters for the crossover and mutation. In p1, an individual is crossed more frequently with similar individuals, than with individual with fewer similarities. p1 mutate with the general mutation operator. In p2, individuals with fewer similarities are crossed at a higher rate and mutate with a greater mutation operator, than in p1 [1].

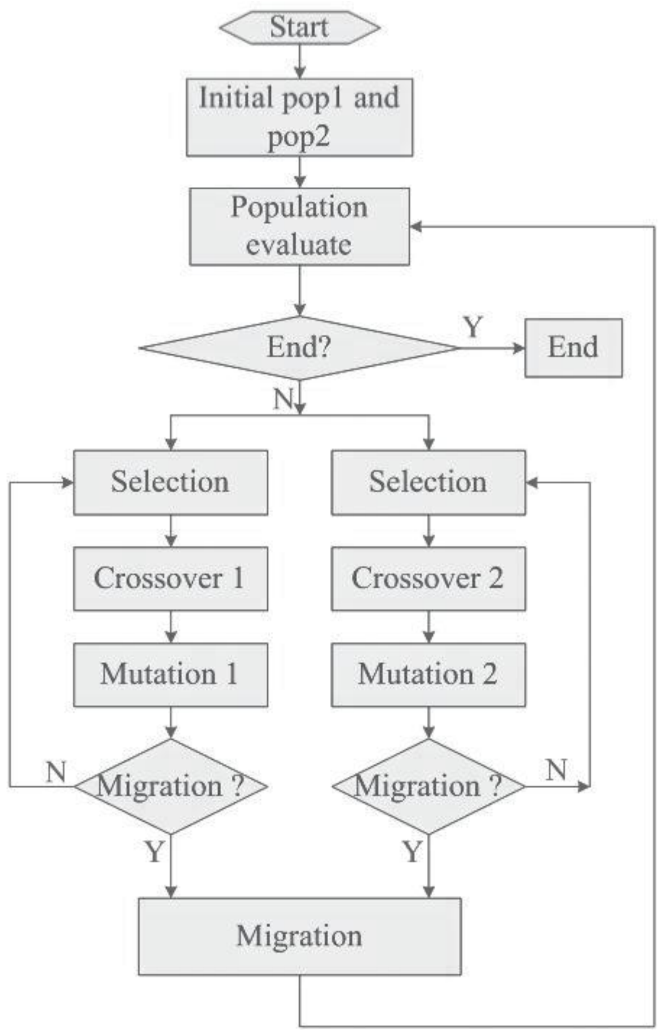

Figure 4: Flow chart of Dual species parallel GA[1][60]

H. Hierarchical Parallel GA

When two or more parallel GA is implemented concurrently than it is said to be Hierarchical Parallel GA [59].

Now we are going to give some of the important contribution by different researcher in the field of parallel genetic algorithm. Apart from these there are so many other worked has been done in this field.

1. Multiprocessor Scheduling Using Parallel Genetic Algorithm [28]

In this research authors worked on the steps of Fitness Evaluation of GA as it is the most time consuming GA operation for the CPU time, which affect the GA performance. The implemented synchronous master-slave algorithm outperforms the sequential algorithm in case of complex and high number of generations' problem.

Following is the flow chart of the proposed system [28].

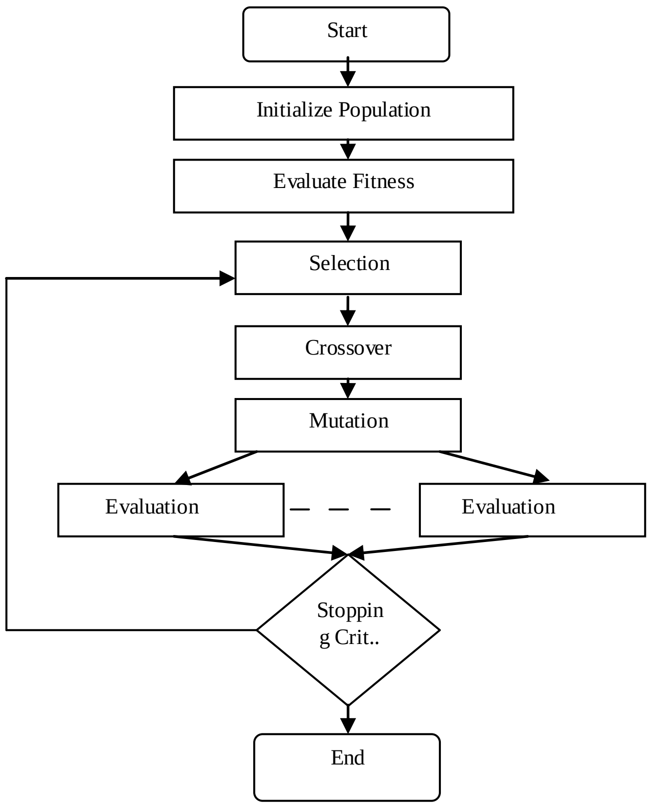

Figure 5: Proposed approach Flow Diagram [28]

The proposed parallel genetic algorithm is based on the sequential algorithm on [61] and uses synchronous master-slave GAs parallelization. The master-slave GAs parallelization never affects the behavior of the algorithm, so there is no need for huge modifications and it has the full advantage of searching the whole of search space. According to the obtained results, the proposed parallel algorithm outperforms the sequential algorithm in case of complex and high number of generation problems. In smaller problems, it is not preferred to use the parallel algorithms.

2. Parallel Genetic Algorithm on the CUDA Architecture [29]

This paper describes the mapping of the parallel island based GA with unidirectional ring migrations to NVidia CUDA software model.

In this authors had chosen CUDA (Compute Unified Device Architecture)framework to implement GA on GPU. Their main aim concern during designing GA accelerated by GPU is to create its efficient mapping to CUDA software model with a special focus on massive parallelism, usage of shared memory within multiprocessors and avoiding the system bus bottleneck. They assumed an island based GA with the migration along an unidirectional ring. Every individual is controlled by a single CUDA thread. The local populations are stored in shared on-chip memory on particular GPU multiprocessors (CUDA blocks). This ensures both computationally intensive execution and massive parallelism needed for GPU to reach its full potential. Their implementation also utilizes a uniform and Gaussian fast random number generators described in [2]. The proposed algorithm begins with the input population initialization on the CPU side. Then, chromosomes and GA parameters are transferred to the GPU main memory using the system bus. Next, the CUDA kernel performing genetic algorithm on GPU is launched. Depending on kernel parameters, the input population is distributed to several blocks (islands) of threads (individuals). All threads on each island read their chromosomes from the main memory to the fast shared (on-chip) memory within a multiprocessor. The process of evolution then proceeds for a certain number of generations in isolation, whereas, the islands as well as individuals are evolved on the graphics card in parallel. Each generation consists of fitness function evaluation and application of the selection, crossover and mutation. In order to mix up suitable genetic material from isolated islands, the migration is employed.

As a result Speedups up to seven thousand times higher clearly show that GPUs have proven their abilities for acceleration of genetic algorithms during optimization of simple numerical functions. The results also show that the proposed GPU implementation of GA can provide better results in the shorter time or produce better results in equal time.

3. Dynamic task scheduling using parallel genetic algorithms for heterogeneous distributed computing[75]

The proposed algorithm uses multiple processors with centralized control for scheduling. Tasks are taken as batches and are scheduled to minimize the execution time and balance the loads of the processors. According to our experimental results, the proposed parallel genetic algorithm (PPGA) considerably decreases the scheduling time without adversely affecting the maxspan of the resulting schedules.

The proposed parallel genetic algorithm involves a master scheduler, which has the processor lists and the task queue. The processors of the distributed system are heterogeneous. The available network resources between processors in the distributed system can vary over time.

A scheduling algorithm has been developed to schedule heterogeneous tasks onto heterogeneous processors on a distributed environment. It provides near-optimal schedules and adapts to varying processing resources and communication costs. The algorithm uses a dedicated master scheduler for centralized scheduling. It uses slave processors (which are not dedicated to scheduling) to parallelize the GA and thereby speed up the result. Genetic Algorithms are powerful but usually suffer from longer scheduling time which is reduced in our algorithm due to

the parallelization of the fitness evaluation, the most CPUintensive task. According to our simulation results, the proposed algorithm not only obtains similar performance as the original genetic algorithm, but also spends less time doing the scheduling. This feature also makes the proposed algorithm to be more scalable and extends its practicability. The proposed algorithm uses a straightforward encoding scheme and generates a randomized initial population. The fitness function uses the maxspan, the balance of load among the processors and the communication costs while evaluating the schedules. Stochastic sampling with partial replacement selection, an extension of the roulette wheel selection, is used to increase the possibility of survival of the fitter solutions. Cycle cross over preserves the characteristics of the parent chromosomes in the children there by leading to exploration of search space. Random swapping is done to prevent the GA to get struck in a local maximum.

4. An Asynchronous Model of Global Parallel Genetic Algorithms[76]

In this approach parallelization has been done through Multithreading. Reducing inter process communication is a key to getting high performance in parallel computing. That is the reason why the asynchronous model is used in this paper. In this there is the implementation of master-slave GA. The master creates random initial population, evaluates created individuals and starts the slaves. This is given in the following pseudo code.

```
Master thread{
initialize population;
evaluate population;
for(i=1;i<NUMBER_OF_PROCESSORS;i++){
create new Slave thread;
}
}
Slave thread{
while(termination criterion is not
reached){
select three individuals;
eliminate the worst individual of three
selected;
child=crossover(survived individuals);
replace deleted individual with child;
perform mutation with probability pm;
evaluate new individual;
}
}
```

Each slave performs whole evolution process in contrast of the traditional master-slave GA where the slaves only evaluate the fitness. This is a model of global GA, because each individual may compete and mate with any other.

The presented model of global parallel genetic algorithm in this is suitable for implementation on a shared memory computer with several processors. The difference between traditional GPGA (master-slave GA) and the described GPGA is in tasks which master and slaves perform. In the traditional GPGA slaves only evaluate individuals, while the master distributes individuals among slaves for evaluation and the master performs all genetic operators. There are so many advantages of this approach like this algorithm is simple for implementation, the exclusivity is inherently implemented, all genetic operators and evaluation is parallelized, there is no need for any

communication mechanism, it works without any synchronization, the execution time for a given number of iterations does not depend on the population size N for a constant number of iterations and speed-up factor is near the number of processors.

5. Parallel genetic algorithm optimization of die press[74]

This paper presents an optimization based GA computed in a cluster of computers with application to problems in electrical engineering. Some basic steps and technologies used for implementing this approach, as well as its advantages and disadvantages are discussed in this research. The main goal is to show the merits of distributed optimization and to determine its suitability in the area of numerical field analysis. Parallel GA optimization successfully determines the global optimum (as serial GA also does), but with at lower time cost depending on the number of workers and the complexity of the problem. An important issue is to balance the tasks between the workers in a way that the efficiency does not become undesirably low.

The dis-cussed approach for parallel GA optimization can also be implemented and will be advantageous for more time-consuming problems such as 3D models, or easily adapted to other inherently parallel optimization methods – for example, preparation of data sets for supervised neural networks training or for design of experiment method. The DCT is flexible and easy to use. Moreover, as it is directly connected to MATLAB, one can use built-in or custom functions in a distributed environment. In combi-nation with GAs it gives the electrical engineer a powerful optimization tool that implements the advantages of GAs for solving complex problems in Electromagnetics.

IV. N Queen's Problem

The n-queens problem is defined as placing of n nonattacking queens on an n x n chess board. This is a simplification of the problem of putting eight nonattacking queens on a board, which was initially proposed in 1848 by M. Bezzel, who is a German chess player, in the Berliner Schachzeitung [78][79]

We can show in the following figure

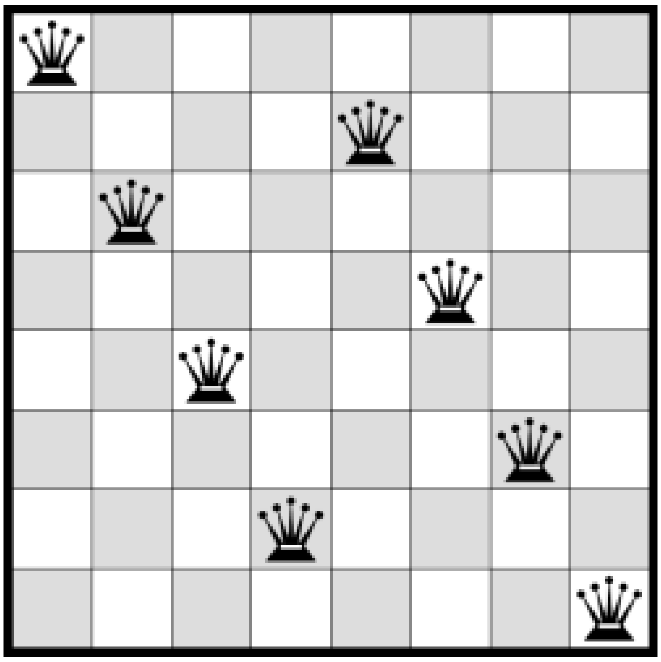

Figure 6: A solution of the 8-queens problem

Applications of N Queen's Problem

There are several applications of N queen's problem such as we can use it in memory storage scheme for conflict free access for parallel memory systems [80], Deadlock Prevention [81], parallel memory storage schemes, VLSI testing, traffic control [82], neural networks and constraint satisfaction problems [83], image processing [84] and many more discussed in [78].

Implementtion of N Queen's Problem

Till now there are several approached have been proposed for n queen's problem discussed in [78]. As we are focused on Genetic Algorithm so now we are going to describe n queen's problem using GA.

Another thing is that we considered 8 Queen's problem for practical implementation so here are we are going to describe sequential algorithm for this such as given in [85]

1. Firstly generate a population 'p' of strings with 'n' rows positions, row position generated at random for every column, representing a pattern of queens on the board.

Such as for a board of size 8 x 8, '1 6 2 5 7 4 8 3 ' is a string of size '8' belonging to population 'P'. Create 'P' such strings.

2. Now, we are having an initial population pi.
3. Pick 2 strings x and y arbitrarily from pi and apply crossover to them. While randomly picking x & y note that the likely hood of x or y getting picked is directly proportional to the value of it's fitness.
4. Now we apply crossover to x and y and produce one new string S.
5. With a little probability apply mutation to string S otherwise leave it as it is.
6. Apply steps 2 to 5 until a new population pn is generated with 'p' strings. pn acts as pi for the next iteration through steps 2 to 5.
7. Repeat steps 2 to 6 until a solution string i.e. the string with maximum fitness value representing a correct board configuration is obtained.
8. If a solution has not been found for a long time return a string with maximum fitness value amongst the generated strings.

Assessment of fitness, purpose of mutation, utilization of single or multiple subpopulations (demes), method of individuals exchange, local or global application of selection are the several parameter needs to be considered at the time of doing parallelization using GA.

In our parallel implementation for 8 queen's problem we have divided population in subpopulation into different activated cores as per choice. Genetic operations such as selection, crossover and mutation are performed by subpopulation on different cores or processors and best solutions are obtained and send to master core. And finally finest solution is obtained by master core.

Major advantage of our work is that we followed simplest procedure for solving 8 queen's problem named as Master Slave modal using most familiar tool MATLAB and found significant speed up discussed in the next section. In our approach we mainly focused on the fastest processing with efficiently using of available compute resources.

V. Experimental Setup and Results Obtained

This Section describes about the Experimental Setup developed to implement the proposed intelligent method.
We have implemented 12 Experiments with three different computing environment and obtained significant speed up.

Setup for First Experiment:
Hardware Configuration:
Processor: Intel(R) Core(TM) i7 -3770 CPU @340 GHz RAM (Random Access Memory) : 4 GB
Hard Disk Drive: 320 GB
Software Environment:
System Type: 64 Bit operating System, x – 64- based processor.
Development Tools: MATLAB, Intel Compiler 9.1, jdk – 6 – windows – i586
Parallelization Scheme in MATLAB
Multithreaded Parallelism and Explicit Parallelism
Obtained Speed up(Average of four Experiments): 201%

Setup for Second Experiment:
Hardware Configuration:
Processor: Intel Core i3 – 2350M Processor 2.30 GHz
RAM (Random Access Memory) : 3 GB
Hard Disk Drive: 320 GB
Software Environment:
System Type: 32 Bit operating System.
Development Tools: MATLAB, Intel Compiler , jdk - 6 - windows - i586
Parallelization Scheme in MATLAB [20]
Multithreaded Parallelism and Explicit Parallelism
Obtained Speed up (Average of four Experiments): 170%

Setup for Third Experiment:
Hardware Configuration:
Processor: Intel Core 2 duo T 6400 CPU @2.00 GHz
RAM (Random Access Memory) : 2 GB
Hard Disk Drive: 250 GB
Software Environment:
System Type: 32 Bit operating System, x 64 based processor.
Development Tools: MATLAB, Intel Compiler , jdk – 6 – windows – i586
Parallelization Scheme in MATLAB [20]
Multithreaded Parallelism and Explicit Parallelism
Obtained Speed up(Average of four Experiments): 145%

VI. Applications of Parallel Genetic Algorithm (PGA) in Medical Imaging [63-71]

There has been several advantage of implementing Genetic Algorithm parallelly in Medical Imaging like Medical Image Registration, Segmentation of Medical Images and etc. Registration of Medical Images is having an important role in medical image analysis. There have been numerous voxel based approaches have been proposed till now and these are mathematically almost all of them are connected with optimization problems that are extremely non-linear and non-convex. Parallel GA implementing in Mutual Information based image registration shows robust registration with high speedup. This technique is readily applicable for other voxel-based registration methods. PGA is also used for performance improvement and refining the result to segment the lateral ventricles from MR brain images. It overcome the numerical instability and is capable of generating an accurate and robust anatomic descriptor for complex objects in the human brain, such as the lateral ventricles.

VII. Discussion and Conclusion

This paper presented some important research work done on parallel genetic algorithm. This research shows significant contribution of parallel computing in Genetic Algorithm. Firstly this paper gave the brief introduction and fundamentals of Genetic Algorithm, Parallel computing, GPU and CUDA. After that, In this paper there is the classification of the different techniques of parallel genetic algorithm like master-slave modal, fine-grained model, multiple-deme model, island model , hybrid coarse grained model, dual species model defined by different researchers. We reviewed numerous research papers implementing parallel genetic algorithm. Some of them are analyzed in this work with those important steps. We also developed a parallel approach to solve n queen's problem based on master slave modal, which produced significant result. Then, we presented its applications in medical imaging. We found that there has been done lots of research on parallel genetic algorithm till now and many more is going on. And lots of work has to be done in the area of medical imaging. As a point of view of its future scope, it is having a great scope in the field of medical imaging to analyze medical data sets and to optimize different important techniques.

References

[1] A. Hassani and J. Treijis "Overview of standard and parallel genetic algorithms", in Proc. IDT Workshop on Interesting Results in Computer Science and Engineering (IRCSE '09) , Mälardalen University, Sweden, October 30, 2009.

[2] E. Alba, J.M. Troya, A survey of parallel distributed genetic algorithms, Complexity 4 (4) pp. 31–52, 1999.

[3] Qizhi Yu, Chongcheng Chen, and Zhigeng Pan, "Parallel Genetic Algorithms on Programmable Graphics Hardware", John Willey & Sons. vol. 4, 1999.

[4] S. Kajan, I. Sekaj and M. Oravec , "The Use of MATLAB Parallel Computing Toolbox For Genetic Algorithm-Based Mimo Controller Design" , 17th International Conference on Process Control , June 9–12, 2009.

[5] Edwin S . H . Hou, Ninvan Ansari, , and Hong Ren, "A Genetic Algorithm for Multiprocessor Scheduling" , in IEEE Transactions on Parallel and Distributed Systems. vol. 5, no. 2, Feb 1994.

[6] Nourah Al-Angari, Abdullatif Abdullatif, "Multiprocessor Scheduling Using Parallel Genetic Algorithm" , Fuzzy Sets, Information and Control, pp. 338-353, 1965.

[7] John P. Cartlidge, "An Analysis of Evolutionary Computation used in Image Processing Techniques " BSc Artificial Intelligence and Mathematics 1999-2000.

[8] Mariusz Nowostawski, and Riccardo Poli, "Parallel Genetic Algorithm Taxonomy", KES, MAY 13, 1999.

[9] C.B. Pettey, M.R. Leuze and J.J. Grefenstette; "A parallel genetic algorithm", in Proceedings of the second International Conference on Genetic Algorithms and their application (ICGA), John J. Greffenstette (Ed.), Lawrence Erlbaum Associates Publishers, 1987.

[10] Zdenˇek, Konfrˇst, "Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives", 18th International Parallel and Distributed Processing, 2004.

[11] M.S.Shyu, and J.J.Leou,"A genetic algorithm approach to color image enhancement", Volume 31, Issue 7, pp. 871–880, July 1998.

[12] Dr. R.K Bhattacharjya, "Introduction To Genetic Algorithms", IIT Guwahati, pp.12, 2012.

[13] S.K. Pal, and D. Bhandari, and M. K. Kundu , "Genetic algorithms for optimal image enhancement" , Pattern Recognition Letters , pp.261-271, March 1994.

[14] Chun-Hung, L., and Ja-Ling, W., "Automatic Facial Feature Extraction by Genetic Algorithms" in IEEE Transactions on Image Processing, Vol. 8, No. 6, June 1999, pp. 834-845, June 1999.

[15] De Jong, K.A. , "Genetic algorithms: A 10 year perspective". In Grefenstette , pp. 169-177, 1985.

[16] Filippidis, A., Jain, L.C., and Martin, N.M. , "Using Genetic Algorithms and Neural Networks for Surface Land Mine Detection" in IEEE Transactions on Signal Processing, Vol. 47, No. 1, pp. 176-186, January 1999.

[17] Fogel, D.B., and Atmar, W., "Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems". Biological Cybernetics, vol. 63, pp. 111- 114, 1990.

[18] Goldberg, D.E , "Genetic Algorithms in Search, optimization and Machine Learning" Reading, Mass. Wokingham: Addison-Wesley.

[19] Haataja, J. , "Using Genetic Algorithms for Optimization: technology transfer in action" In Miettinen et al. pp. 3-22, 1999.

[20] R.K. Mohanta, and B. Sethi, "A Review of Genetic Algorithm application for Image Segmentation", J.Computer Technology & Applications, vol 3 (2), pp.720-723.

[21] Sanjay Saxena, Shiru Sharma, Neeraj Sharma, "Parallel Computation of Mutual Information in Multicore Environment and Its Applications in Medical Imaging " IEEE International Conference on Medical Imaging, m Health & Emerging Communications System, Noida, India 2014.

[22] Qizhi Yu, Chongcheng Chen, and Zhigeng Pan, "Parallel Genetic Algorithms on Programmable Graphics Hardware" LNCS 3612, pp. 1051–1059, 2005.

[23] R. T. Rasúa, "Algoritmos paralelos para la solución deproblemas de optimización discretos aplicados a ladecodificación de señales," Ph.D. dissertation, Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, Valencia, España, 2009.

[24] Sanjay Saxena, Shiru Sharma, Neeraj Sharma , "Image Registration Techniques Using Parallel Computing in Multicore Environment and Its Applications in Medical Imaging" IEEE International Conference on Computer and Communication Technology, ICCCT MNNIT (Moti Lal Nehru National Institute of Technology), Allahabad, 2014.

[25] Sanjay Saxena, Neeraj Sharma, Shiru Sharma, "Image Processing Tasks Using Parallel Computing in Multicore Architecture & its application in medical imaging" in International Journal of Advanced Research in Computer and Communication Engineering(IJARCCE), Vol. 2, Issue 4, April 2013.

[26] Sanjay Saxena, Neeraj Sharma, Shiru Sharma, "Region wise processing of an image using multithreading in multi core environment & its application in medical imaging" in International Journal of Computer Engineering and Technology (IJCET), Volume 4, Issue 4, July - August 2013.

[27] Sanjay Saxena, Neeraj Sharma, Shiru Sharma, "An Intelligent System for Segmenting an Abdominal Image in Multicore Architecture ", in IEEE, International Conference and Expo on Emerging Technologies for a smarter world at New York, USA, Held in New York, United States of America, 2013.

[28] Nourah Al-Angari, Abdullatif Abdullatif, "Multiprocessor Scheduling Using Parallel Genetic Algorithm".

[29] Petr Pospichal, Jiri Jaros, and Josef Schwarz, "Parallel Genetic Algorithm on the CUDA Architecture" Evo Applications , Part I, LNCS 6024, pp. 442–451, 2010

[30] ''NVIDIA's Next Generation CUDA Compute Architecture: Fermi,'' Nvidia, 2009.

[31] V.W. Lee et al., ''Debunking the 100x GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU '' Proc. 37th Ann. Int'l Symp. Computer Architecture , ACM, pp. 451-460, 2010.

[32] R. Kumar, D.M. Tullsen, and N.P. Jouppi, ''Core Architecture Optimization for Heterogeneous Chip Multiprocessor,'' Proc. 15th Int'l Conf. Parallel Architecture and Compilation Techniques , ACM, pp. 23-32, 2006.

[33] J.D. Owens et al., ''A Survey of General Purpose Computation on Graphics Hardware'' Computer Graphics Forum, vol. 26, no. 1, pp. 80-113, 2007.

[34] C. Kolb and M. Pharr, ''Options Pricing on the GPU '' GPU Gems 2, M. Pharr and R. Fernando, eds., Addison-Wesley, chapter 45, 2005.

[35] G. Wang et al., ‘‘Program Optimization of Array-Intensive SPEC2K Benchmarks on Multithreaded GPU Using CUDA and Brook+’’ Proc. 15th Int’l Conf. Parallel and Distributed Systems, IEEE CS, pp. 292-299, 2009.
[36] G. Shi, S. Gottlieb, and V. Kindratenko, “MILC on GPUs”, tech. report, NCSA, Univ. Illinois, Jan. 2010.
[37] J. Walters et al., ‘‘Evaluating the Useof GPUs in Liver Image Segmentation and HMMER Database Searches’’ Proc. IEEE Int’l Symp. Parallel & Distributed Processing, IEEE CS, 2009.
[38] G. Ruetsch and M. Fatica, ‘‘A CUDA Fortran Implementation of BWAVES’’ http://www. pgroup.com/lit/articles/nvidia_paper_ bwaves.pdf.
[39] W.M. Hwu et al., ‘‘Performance Insights on Executing Nongraphics Applications on CUDA on the NVIDIA GeForce 8800 GTX ’’ Hot Chips 19, 2007. http://www.hotchips. org/archives/hc19.
[40] NVIDIA: NVIDIA CUDA Programming Guide, http://developer.download.nvidia.com/ compute/cuda/2_0/docs/.
[41] “CUDA, Supercomputing for the Masses” by Rob Farber. http://www.ddj.com/architect/207200659
[42] “CUDA, Wikipedia”. http://en.wikipedia.org/wiki/CUDA
[43] “Cuda for developers, Nvidia. http://www.nvidia.com/object/cuda_home.html#.
[44] “Download CUDA manual and binaries” http://www.nvidia.com/object/cuda_get.html
[45] “NVIDIA, C.: Compute Unified Device Architecture Programming Guide. NVIDIA” Santa Clara, CA 2007.
[46] Rob Farber, "CUDA, Supercomputing for the Masses: Part 4", Dr.Dobbs, http:http://www.ddj.com/architect/208401741?pgno= 3//www.ddj.com/hpc-high-performance-computing/207402986.
[47] Article from http://www.headwave.com
[48] Article from http://www.ks.uiuc.edu/Research/vmd
[49] Article from http://bic-test.beckman.uiuc.edu
[50] Article from http://www.cs.clemson.edu/~jesteel/clouds.html
[51] ABRAMSON D., ABELA J., “ A parallel genetic algorithm for solving the school timetabling problem” , In Proceedings of the Fifteenth Australian Computer Science Conference , ACSC 15, vol. 14, p. 1–11, 1992.
[52] HAUSER R., MÄNNER R., “Implementation of standard genetic algorithm on MIMD machines” . In DAVIDOR Y., SCHWEFEL H.-P., MÄNNER R., Eds., Parallel Problem Solving fron Nature, PPSN III, p. 504–513, Springer-Verlag (Berlin), 1994.
[53] FOGARTY T. C., HUANG R., “Implementing the genetic Algorithm on Transputer Based Parallel Processing Systems” . Parallel Problem Solving from Nature, p. 145–149, 1991.
[54] GROSSO P. B., “Computer simulations of genetic adaptation : Parallel subcomponent interaction in a multilocus model”. Unpublished doctoral dissertation, The University of Michigan, 1985.
[55] PETTEY C. B., LEUZE M. R., GREFENSTETTE J. J., “A parallel genetic algorithm” . In GREFENSTETTE J. J., Ed., Proceedings of the Second International Conference on Genetic Algorithms, p. 155–161, Lawrence Erlbaum Associates (Hillsdale, NJ), 1987.
[56] TANESE R., “Parallel genetic algorithm for a hypercube”. In GREFENSTETTE J. J., Ed., Proceedings of the Second International Conference on Genetic Algorithms, p. 177–183, Lawrence Erlbaum Associates (Hillsdale, NJ), 1987.
[57] E. Cant_u-Paz. “A survey of parallel genetic algorithms”. Calculateurs Paralleles, Reseaux Et Systems Repartis, 10, 1998.
[58] Riccardo Gismondi, “Parallel Genetic Algorithm for the Calibration of Financial Modals” in HPCFal.
[59] Erick Cantú-Paz, “A Survey of Parallel Genetic Algorithms”. Om
[60] Li and M. Li. “Genetic algorithm with dual species”. In International Conference on Automation and Logistics Qingdao, China September 2008, 2008.
[61] Melanie Mitchell, “Genetic Algorithms: An Overview”, An Introduction to Genetic Algorithms, Chapter 1. MIT Press, forthcoming. 1995.
[62] Cant-Paz, E., “Efficient and Accurate Parallel Genetic Algorithms”. Kluwer Academic Publishers, Dordrecht, 2000.
[63] W.M. Wells, III, P. Viola, H. Atsumi, S. Nakajima, and R. Kikinis, “Multi-modal Volume Registration by Maximization of Mutual Information”,Medical Image Analysis, Vol. 1, pp. 35–51, 1996.
[64] F. Maes, D. Vandermeulen, P. Suetens, “Comparative evaluation of multiresolution optimization strategies for multimodality image registration by maximization of mutural information”, Medical Image Analysis, Vol. 3, No. 4, pp. 373–386, 1999.
[65] J.B.A. Maintz and M. A. Viergever, “ A Survey of Medical Image Registration”, Medical Image Analysis, Vol. 2, No. 1, pp. 1–36, 1998.
[66] Neil A. Thacker, Alan Jackson, David Moriarty, Elizabeth Vokurka, “Improved quality of re-sliced MR images using re-normalized sinc interpolation”, Journal of Magnetic Resonance Imaging, Vol. 10, No. 4, pp. 582–588, 1999.
[67] Albert Y. H. Zomaya, Parallel and Distributed Computing Handbook, McGraw-Hill, New York, 1996.
[68] A. Amini, T. Weymouth, and R. Jain, “Using Dynamic Programming for Solving Variational Problems in Vision,”IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.12, pp. 855-867, 1990.
[69] Yong Fan, Tianzi Jiang, and David J. Evans , “Volumetric segmentation of Brain Images using Parallel Genetic Algorithm”.
[70] D. L. Collins, A. P. Zijdenbos, V. Kollokian, J. G. Sled, N. J. Kabani, C. J. Holmes, and A. C. Evans, “Design and Construction of a Realistic Digital Brain

Phantom", IEEE Transactions on Medical Imaging, Vol.17, pp. 463-468, 1998.

[71] B. Wilkinson and M. Allen, "Parallel Programming: Techniques and Applications using networked workstations and Parallel Computers", Prentice Hall, New Jersey, 1999.

[72] Steven P. Brumby , James Theiler , Simon J. Perkins , Neal Harvey, John J. Szymanski, Jeffrey J. Bloch , and Melanie Mitchell, "Investigation of Image Feature Extraction by a Genetic Algorithm". http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.8210.

[73] Peter Zalutski, "CUDA: Supercomputing for the Masses". http://www.drdobbs.com/parallel/cuda-supercomputing-for-the-masses-part/207200659.

[74] Pavlina Mihaylova, Kostadin Brandisky, "Parallel Genetic Algorithm Optimization of Die Press". http://phd.etfbl.net/files/Works_PDF/Mihaylova%20Pavlina.pdf.

[75] R.Nedunchelian, K.Koushik, N.Meiyappan, V.Raghu, "Dynamic Task Scheduling using Parallel Genetic Algorithms for Heterogeneous Distributed Computing". http://ww1.ucmss.com/books/LFS/CSREA2006/GCA4489.pdf.

[76] Marin Golub, Leo Budin, "An Asynchronous Model of Global Parallel Genetic Algorithms". http://www.zemris.fer.hr/~golub/clanci/eis2k.pdf.

[77] Sanjay Saxena, Neeraj Sharma and Shiru Sharma, "Parallel Image Processing Techniques, Benefits and Limitations" in Press.

[78] Jordan Bell, Brett Stevens, "A survey of known results and research areas for n-queens" in Discrete Mathematics, Elsevier, vol. 309, pp. 1-31, 2009.

[79] M. Bezzel, Proposal of 8-queens problem, Berliner Schachzeitung 3 (1848) 363. Submitted under the author name "Schachfreund"

[80] C. Erbas, M.M. Tanik, V.S.S. Nair, "A circulant matrix based approach to storage schemes for parallel memory systems" in Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Processing (Dallas, Texas, Dec. 1–4, 1993), IEEE, 1993, pp. 92–99.

[81] M.M. Tanik, "A graph model for deadlock prevention", Ph.D. Thesis, Texas A & M University, 1978.

[82] C. Erbas, M.M. Tanik, Z. Aliyazicioglu, "Linear congruence equations for the solutions of the N-queens problem", Inform. Process. Lett. Vol. 41 (6) pp. 301–306, 1992.

[83] C. Erbas, S. Sarkeshik, M.M. Tanik, "Different perspectives of the n-queens problem", in Proceedings of the 1992 ACM Annual Conference on Communications, ACM Press, pp. 99–108, 1992

[84] S.-W. Yang, C.-N. Wang, C.-M. Liu, T.-H. Chiang, in: H.-Y. Shum (Ed.), "Fast motion estimation using N-queen pixel decimation", in Lecture Notes in Computer Science, vol. 2195, Spring-Verlag, Berlin, 2001.

[85] Article "How is genetic algorithm used to solve the N-Queens problem" http://www.quora.com/How-is-genetic-algorithm-used-to-solve-the-N-Queens-problem