Parallel and Distributed Genetic Algorithm with Multiple-Objectives to Improve and Develop of Evolutionary Algorithm

Khalil Ibrahim Mohammad Abuzanouneh
Qassim University, College of Computer, IT Department Al Qassim, Saudi Arabia

***Abstract*—In this paper, we argue that the timetabling problem reflects the problem of scheduling university courses, So you must specify the range of time periods and a group of instructors for a range of lectures to check a set of constraints and reduce the cost of other constraints ,this is the problem called NP-hard, it is a class of problems that are informally, it's mean that necessary operations to solve the problem will increases exponentially and directly proportional to the size of the problem, The construction of timetable is most complicated problem that was facing many universities, and increased by size of the university data and overlapping disciplines between colleges, and when a traditional algorithm (EA) is unable to provide satisfactory results, a distributed EA (dEA), which deploys the population on distributed systems ,it also offers an opportunity to solve extremely high dimensional problems through distributed coevolution using a divide-and-conquer mechanism, Further, the distributed environment allows a dEA to maintain population diversity, thereby avoiding local optima and also facilitating multi-objective search, by employing different distributed models to parallelize the processing of EAs, we designed a genetic algorithm suitable for Universities environment and the constraints facing it when building timetable for lectures.**

Keywords—Heterogeneous clusters; NP-hard; evolutionary multi-objective algorithm; parallel algorithms; Real-time scheduling

I. Introduction

Genetic Algorithm a heuristic used to find a vector x * of free parameters with associated values in an admissible region for which an arbitrary quality criterion is optimized as in given in figure 1.

$$f(\vec{x}) \rightarrow \max : \text{find } \vec{x}^* \text{ so that } \quad \forall \vec{x} \in M : f(\vec{x}) \leq f(\vec{x}^*) = f^*$$

Fig.1. A sequential Genetic Algorithm

The algorithm uses stochastic operator's selection, crossover and mutation on an initially random population in order to compute a whole generation of new strings.

Evolutionary Algorithm is part of the science of artificial intelligence so that it is similar to simulation solution including genetic evolution like the system through the representation of some genetic processes such as natural selection and the struggle for survival and mutations and live in groups [1].

Stages of the evolutionary algorithm begins by choosing a group of chromosomes so that each representing an individual solution to a problem, then you are causing a mutation on individuals process to produce new offspring , then a selection nearest the members of the ideal solution and the neglect of the solutions, we have to repeat this process for the production of successive generations, each with a larger number of qualities required prevailing until the arrival of the algorithm to termination as in given in figure 2.

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO
    1) SELECT parents;
    2) RECOMBINE pairs of parents;
    3) MUTATE the resulting offspring;
    4) EVALUATE new candidates;
    5) SELECT individuals based on their fitness for the next generation;
  DO
END
```

Fig.2.The General Scheme of an Evolutionary Algorithm

II. Background

A. *Parallel Genetic Algorithms (PGA):* is important for improvements using parallel models of Genetic Algorithms, Parallel Genetic Algorithms (PGA) are faster to find sub-optimal solutions, and able of cooperating with other search techniques in parallel [10], PGA is independent of the problem and can yield alternative solutions to the problem, parallel search from multiple points, easy parallelization, better search, higher efficiency and efficacy than sequential GAs [1].

B. *Parallel Multi-objective Evolutionary Algorithm*

Steps of the Parallel Multi-objective EA:

1) Create a community, the creation of the initial community, according to the problem to be solved and is usually within the terms of the initial problem and the available data [5].

2) Individual's evaluation, we use to assess individual fitness function for each by calculating the optimal solution function, this value is used to determine the closest solutions to solve optimization problems because they need to modify less than others to reach the optimal solution.

3) Mutation, which occurs randomly on one of the characteristics of the individual. So that the value of this status change to a random value within the terms of individuals [9].

4) The overall structure of the proposed methodology is shown in the pseudocode Multi Clustered Parallel GA [14].

5) Divide the individuals into n clusters based on the fitness value.

6) For each cluster perform the following:

a) Using Selection Mechanism, Select the individuals from each node.

b) Convert the individual to Gray code.

c) Mutate the Parent.

d) Convert the offspring to Binary Value, calculate the fitness Value.

7) Group the clusters together.

8) Allow the migration of individuals based on fitness.

9) Until Termination condition is reached repeat from point 5.

10)Select the best Individual.

Multi-core cluster: is a cluster where all the nodes in the cluster have multi-core processors and multi-cluster architecture is a multiple cluster system that is connected via the cluster interconnection networks [4], [11].

C. Multi-objective algorithm

Most of the problems that we face in life are not easy to define it as one goal, but must all parameters within the framework of the problem to get a better definition of the problem, and this in turn facilitates access appropriate solution for all parties. From here began the pluralistic definition of goals and how to reach a solution that suits conflicting objectives to define the problem.

In our problem we had two goals must be achieved and taken into consideration when finding a suitable solution for hard constraints and soft constraints, during the search for a solution to implement these objectives to get a satisfactory solution for both instructor and student [12],[13].

D. Parallel algorithm

One of the most important issues that must be considered when dealing with an evolutionary algorithm redistribution of tasks, so that the tasks are evenly distributed to all devices to accomplish all the tasks at the same time, the importance of the redistribution of loads in process Heterogeneous environments specifications [4], each computer has its own specifications which appears in varying speeds completion operations [1], [3],to implement parallel software engineering apply the four processes are: partitioning, communications, aggregation, and planning [2].

The division of tasks in two ways, either partial operation is divided into tasks or data division. In both cases, the system needs to contact operations between devices for coordination and exchange of data and information between processors, and communication processes that have several classifications of them (local or global, or structured or unstructured, or a fixed or non-fixed, or synchronous or asynchronous),to reduce processing and communication operations, and then assemble a single tasks in groups, so be quick communication and less costly, and to increase the usefulness of this process (assembly process) ,we develop processes to execute parallel operations on different processors and all operations that frequently communicate with each other on the same processor, it must be a balance between these two points to improve the performance of the algorithm as possible. This process is called planning and is divided into three levels of complexity according to change the number of fragmented or stability tasks, and the division is organized or unorganized, or the communication process during implementation it will change or not [2]. There is a lot of evidence of the high efficacy and efficiency of PGAs over traditional sequential GAs (for example [15].

The best topology for a parallel dGA [16]. , the ring and hypercube are two of the best topologies. The ring topology is easy to implement on very different hardware platforms (cluster of workstations, multi-computers), and its use is very extended.

These results only mean that the dGA is an efficient and robust search procedure. The actual performance of this algorithm is often better than the sequential GA in many problems; see for example [17]. This research maintains samples of very different zones of the search space in every population, thus showing a higher efficacy, and probability of obtaining a solution, for complex applications, the parallel dGA is important in order to have lower computational times and using larger populations than with sequential GAs, and the high number of non-standard and machine-dependent PGAs has led to efficient algorithms in many domains, these call distributed and cellular GAs (dGA and cGA),we refer to their computation/communication ratio, while the actual differences can be also found in the way in which they both structure their populations (Figure 3), when a distributed GA has sub-populations a cGA has typically one single string in every sub-algorithm. In a dGA, the sub-algorithms are loosely connected, while for a cGA they are tightly coupled, and in a dGA there exist only a few sub-algorithms while in a cGA there is a large number of them.

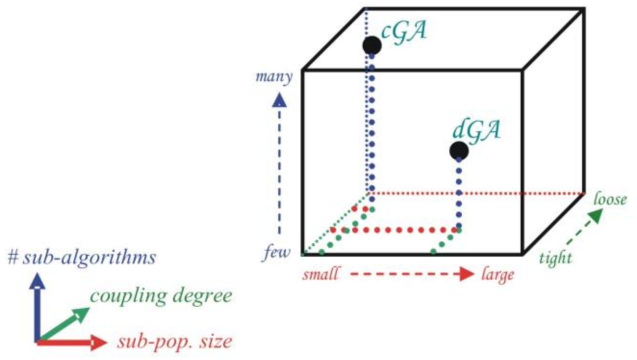

Fig.3. The Structured-Population Genetic Algorithm Cube

III. Related Work

The efficiency of schedule of courses is important and necessary for the satisfaction of instructors and students, as well as it is necessary for exploitation of human and material resources better. In the past, the solution to the scheduling problem was manually using trial and error and had access to resolve ideally devoid of errors is very difficult, even if any of this solution, it is not the best, for this is the use of scientific methods to solve the problem. The problem Study Began four decades ago, but not completed for a specific mechanism to resolve the problem until now. The study of the problem by (Gottlieb), the description of the problem that each lecture is one of a group of students and one teacher and a number of times that are randomly selected [6]. The problem of scheduling appointments in schools is easier than in the universities because the halls System (Class) is not changed ,but the students per group are variable and are chosen randomly for each course at the university. This increases in the complexity of the problem at the University greater than the complexity of halls system, so it is a traditional way to solve the problem is difficult, particularly if there are several objectives to be achieved at the same time [6]. Until the problem is defined as a Multi-objective, there must be more than one target fixes the problem and different goals from one university to another definition. It is noteworthy that most researchers had to solve the problem as a problem of a single goal , where the only goal is to reduce all constraints (Abramson and Abela 1992;Blum; 2002; Lima; 2001; Piola; 1994), but there are many options to define goals such as access to a timetable orderly and correctly, and a number of successive lectures to the instructor and others that make the problem of scheduling multiple objective problems, there are a few researchers who define the problem as a problem of multiple targets. Carrasc and Pat (2001) used the dual objective model to the problem of scheduling courses in school schedules to reduce soft constraints for the teacher and the class, Filho and Lorena (2001) they also define the problem as a dual goal, Dicev (2004) identified the problem as a dual goal, but the definition of different goals to reduce the gap between lecture time and the date of the completion of daily lectures, so that priority for early lectures given in one of the goals, and priority late for lectures given at the other goal [7].

In one of the other research, the researchers using the server and client architecture in the distributed evolutionary algorithm, so that there was a customer who apply evolutionary algorithm and named (O-clients) and others to resolve the problem by relying on the next transactions (O-clients) and those customers so-called (GEP-clients)[8]. In another research has the comparison between the two techniques for distribution, which is the first (Message Passing Programming) and the other (Shared Memory), so are easy to apply while the latter characterized the first efficient top except in special cases. This was after the application of the first three experiments on the sequential evolutionary algorithm, and the other on an evolutionary algorithm parallel technology (Message Passing Programming), and the third parallel evolutionary algorithm technology (Shared Memory) [9].

IV. Methodology

Description of the problem: we have implemented the algorithm on the data in the computer college at Qassim University. As an example of the experiments and development of the algorithm will be applied of all Qassim University Colleges. In the College of Computer, there are three disciplines: Computer Engineering, Computer Science and Information Technology. At the beginning of the semester, we have four student groups for each specialty, so it becomes a total 3 * 4 = 12 student groups. Where lectures system will apply on Sunday, Tuesday, and Thursday we have eight lectures of 60 minutes per lecture, but in the days Monday and Wednesday, there are five lectures each day by 90 minutes. To resolve the problem correctly, we define the problem to become composed of 6 different groups are as follows: groups of students, teachers, courses, halls, the time periods for lectures and restrictions.

The problem then being formulated to become as follows:

{St, Inst, Co, R, L, O}, where St = {st1, st2, st3 ..., sti} contain groups of students, each item which is stored the following values: (st_m, year , std_slot) where st_m: specialization, and year: the academic year, and st_slot: lectures time.

Ins= {inst1, inst2, inst3 ..., inst j} Instructors name, and every item in this collection contains the following information: (inst_no,inst_slot) where inst_no: instructors number, inst_slot: lectures times.

Co = {co1, co2, co3 ..., con} : reflect the, every element in this group stores the following values:

(sti, co_no, section, cap, inst_no, h_no, ph, ph_type, ph_inst), where the sti: student group number, and co_no: course number, section, cap : capacity for this course, inst_no: instructor number, h_th: the number of theoretical hours , h_lab: number of lab hours for this course, and h_type: the type of hours for this course, so the value of 0 if it is not for this course lab hours o and be 1 if the course has lab, 2 for multimedia lab, 3 if there electrical lab , 4 for electronics lab, either for h_inst lab they reflect Instructors hours number in the laboratory, shall be 0 if it is not necessary to the existence of the instructor in the laboratory, and is 1 if the instructor must stay a half lab time, and be 2 if the teacher must stay all time in the lab.

Ro = {ro1, ro2, ro3 ..., rok} expresses the halls and every item the stores by the following information: (cap, vcr, type_r) where that cap: reflect the capacity of the room, and VCR: expresses whether the room has a device for viewing or not shall be the value 0 if the room has a device for viewing and 1 if the room did not contain a device for viewing, but type_Ro: It reflects the room type is 0 if the lecture hall and 1 If a computer lab.

L = {lt1, lt2, lt3 ..., ltn} reflect the time periods in which every item contains the following data: (tn, dn) where tn: number The time period ranges from 1-8 days on Sunday, Tuesday, and Thursday , while its value ranging from 1 to 5 in the days Monday and Wednesday, and dn: thus ,expressed number of today ranges from 1-5.

Finally Cst = {cst1, cst2, cst3 ..., cst} a group which determinants that must be taken into account when searching for a solution, and Cst group: express the exact weight and value. Determinants are divided into two types difficult determinants (Hard), and determinants (Soft). The value should be easy or difficult determinants according to their importance.

G = {g1, g2, g3 ..., gj} represents a group of genes, which expresses the solution to the problem (chromosome), also shown in Figure data contained in the gene.

Each gene is a vector contains the following data (course, room_no, ph_tslot, lab_no, th_tslot),

Where course: The course number, but room_no: No. Hall, which will be given this course, and ph_tslot: expresses the number of time that will give the course, and lab_no: expresses the number laboratory or operator, and finally the th_tslot: expresses the time period in which will be given by the laboratory.

In this issue, we have a large and complex field of research, where it is to identify specific hall and laboratory hours if necessary process for each course suggested during the season, is also determined by the time period in which the course will be given and the laboratory, so that they are suitable for both the teacher and the student. To reduce the time and effort required to solve, our way to random selection for each of the halls and time periods to take into account the absence of a conflict with a teacher, and then later to improve the solution to meet the hard and soft student determinants (hard and soft constraints) easy for the teacher and determinants.

Constraints: the final solution takes into account the set of parameters that must be implemented so that the solution would be acceptable to both the student and the instructor, and these constraints are divided into hard constraints (infinity) soft constraints be valued according to their importance.

For hard Constraints are as follows:

1) You must not have an instructor lectures to the same time.

2) You must not be a student at the lectures in the same time.

3) You must not have an instructor lecture time at the time that does not exist in the university.

4) Lectures must not be in one hall at the same time and it accommodates the number of students.

For the soft constraints are as follows:

1) The students don't have lectures successive without intervals of rest.

2) Instructors or students do not have free time exceed three-hour.

3) Instructors don't have a lecture late every day.

4) Students don't have any consecutive lectures in places away.

V. Distributed Processing

In this section, we use distributed processing with the evolutionary algorithm for two main reasons:

1) To reduce the execution time by distributing operations on multiple processors

2) And the second by taking advantage of the division of society into several partial communities, which may help to give the best results [3], [4].

The researchers developed a set of definitions and methods relating to the process of distribution, and we'll show here what has been used in this project from these definitions and methods:

a) Acceleration ratio is the ratio of the time required to implement the algorithm on a single processor for the implementation of the algorithm on the processor.

b) The efficiency of the system: is the ratio of the acceleration for the number of processors that are divided by the algorithm.

c) Additional operations in the search process come from communication processes, and the waiting time for processing, for sharing data and processing.

There are several classifications of the process of distribution of any algorithm, including how to distribute the algorithm in terms of distribution operations or data which we used, called (Single Instruction, Multiple Data), which means that all processors will execute the same instructions, but on different data problem of this kind. The distribution of the process, he needs to redistribute loads after all the setups, especially if the application on a non-homogeneous system (heterogeneous environment) [4].

There are two methods that can be used in the distribution process : shared storage space and message passing computer, where the message passing computer used in this project, which means that each processor has its main memory, and can communicate with each other processors by sending.

The tasks divisions (partial division of society communities) at the start of the implementation of the algorithm in the central device, regardless of the needs of other devices, while not sent these partial societies only when a request for this data from their processors as I mentioned earlier, the redistribution of loads is important to us in this project to improve waiting time process, and this process is different from time to time because the evolutionary algorithm's dependence on random variables prevent the expected implementation the end of time, so the redistribution of loads is a dynamic process, and in the Central process device, we used algorithm centralized dynamic load balancing algorithm.

VI. Cluster Computing

This section provides the technique of linking set of computers in LAN in order to take advantage of the parallel processing power of those computers, clusters are designed to exploit the parallel processing power of multiple nodes and clusters operate by routing all work through one or more load-balancing front-end nodes, which then distribute the workload efficiently between the remaining active nodes and processing power can be optimized, the time required to solve the problem on N processors with the time required on a single processor. This is shown as:

$$S(n) = T(1) / T(n); \qquad (1)$$

where S (n) is the speedup achieved with n processors, T (1) is the time required on a single processor, and T (n) is the time required for N processors.

The concept of efficiency is defined as

$$E(n) = S(n) / n. \quad (2)$$

It measures how much speedup is brought per additional processor and the task is to be computed on a single processor, the time needed can be represented as:

$$T(1) = s' + np', \quad (3)$$

The scaled speedup can be written as:

$$s'(n) = \frac{T(1)}{T(n)} = \frac{(s' + np')}{(s' + p')} = n - (n-1).\frac{s'}{s' + p'}$$
$$= n - (n-1).s'' \quad (4)$$

where s'' is defined as $s'/(s'+p')$. s'' is the ratio of serial code, execution, while s is with reference to all code in the whole program for the problem, and also be noted that s is a constant that is only relevant to the computation problem, under the precondition that problem scale is fixed; where s'' is a constant under the precondition of problem scale changes as Gustafson described. Under Gustafson's Law, the speedup can be linearly increased with the number of processors hired in the computation [18].

VII. Parallel and Distributed EA

Parallel and distributed EA computing are a key technology in the present days of networked and high-performance systems. And for increased performance e by adding processors, memory and an interconnection network and putting them to work together for sharing the workload, it is hoped that an N-processor system will give rise to a speedup in the computation time, the MIND class of parallel architectures multiple processors work together through some form of interconnection, the different programs and data can be loaded into different processors which mean that each processor can execute different instructions at any given point in time, The processors will require some form of synchronization and communication in order to cooperate on a given application. 4 Figure 4 is the most commercially and useful of the parallel and distributed architectures that belong to it.

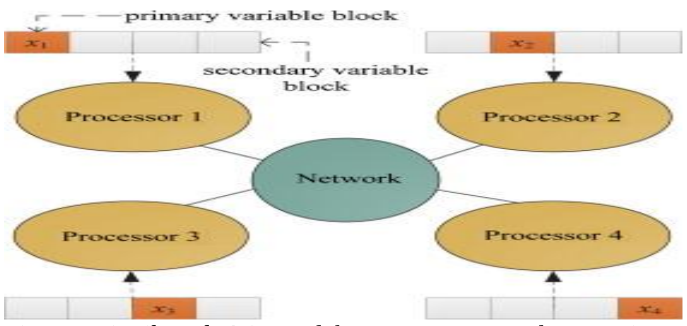

Fig. 4. Distributed GAs and homogeneous Implementations Used MIMD

The connectivity of the islands in a parallel distributed GA. We define the migration policy as a tuple of four values:

$$M = (m, \zeta, \omega_S, \omega_R) \quad (5)$$

Where m is the migration rate, ζ is the frequency of migration, $\zeta \in \{0, 1, ..., \infty\}$, being $0 \approx \infty$ (partitioned dGA),where ω_S is the *policy for selecting migrants*, we send a copy of the selected individual to the neighboring island; alternatively, the individual itself could be sent. It defines the connected nodes (topology) and the shared individuals, where ω_R is the *migration replacement policy*, used for integrating an incoming individual in the target island.

Some work is available at the convenience of using asynchronous communications [2], [12], [24]. This can be achieved by inserting an individual whenever it arrives, thus avoiding blocking every ζ steps, i.e., Emissions and receptions of migrants are managed in separate portions of the code.

The set of parameters Θ_M controlling the migration operator ω_M consists in the migration policy 0, plus a synchronization parameter (sync/async, labeled as **s** and **a** in the forthcoming graphs). The reproductive cycle of a parallel distributed GA is a composition of the island reproductive cycle and a migration operator:

$$\omega_d = \omega_M \circ \omega_{island} \quad (6)$$

To compute super-linear speedups we need to reduce both the number of necessary steps and the expected execution time Tnproc in relation with the sequential one T1, (Equation 7), [19].

$$S(n_{nproc}) = \frac{T(1)}{T(n_{nproc})} \quad (7)$$

There exists an exponential relationship between the speedup and the number of processors. In this relationship (Equation 8), s is the acceleration factor (super-linear speedup when s>1).

$$S(n_{nproc}) = n_{nproc.}\ s^{nproc-1} \quad (8)$$

VIII. Scheduling System

We have implemented in a cluster of 1 workstations and 5 processors, these machines have a technique employed to achieve parallelism has a number of processors that function asynchronously and independently, at any time, also different processors for executing different instructions on different data, implementations of homogeneous ,we used Java programming to solve the problem by using the principle of programming and the definition of the entities of the problem classes: teachers, students, and courses, and the halls, finally the genes which represents the solution.

A. The Results of Experiments

After many experiments and observations on the environment, Figure 5 shows the evolution of the average best fitness and the number of generations. Relationship between sequential execution and different variables Teachers=50, Sections=90, Classes=25.Generation=1000, Best Fitness=0.98.

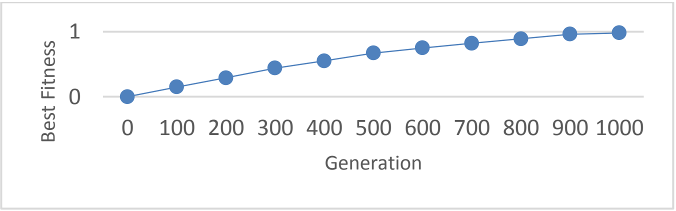

Fig.5.The Best Fitness Eevolution – Sequential GAs

Figure 6 describes the results when we run code to test the Superliner speedup in a shared memory system we used the data to access a specific element in the memory. The sequential execution will have a cache miss time for next element is read. Average running time = 4.52274 in a second.

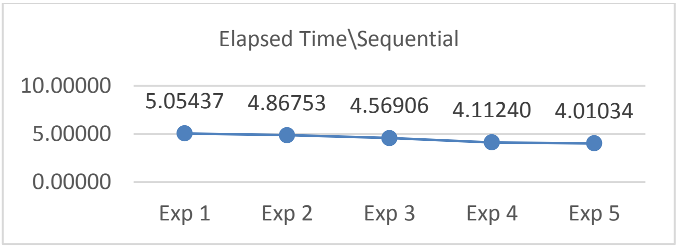

Fig.6.The Sequential System: S*ISD*

Figure 7 Shows the relationship between the number of different variables and the time required for the execution of the algorithm, note that the increased rate of time implementation and repetition stage with increasing variables of sections=100, Generation=1000 , Best Fitness=0.91.

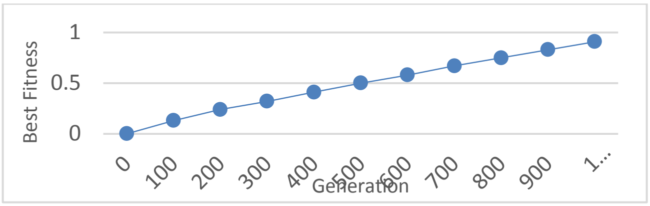

Fig.7.The best fitness evolution with increasing variables

Figure 8 shows the relationship between processor number, execution time, generation number and fitness in diagrams, and the analysis of the results. The execution time of the parallel multi-objective algorithm decreases efficiently as the number of processors increases. Processors=5 Fitness=0.91.

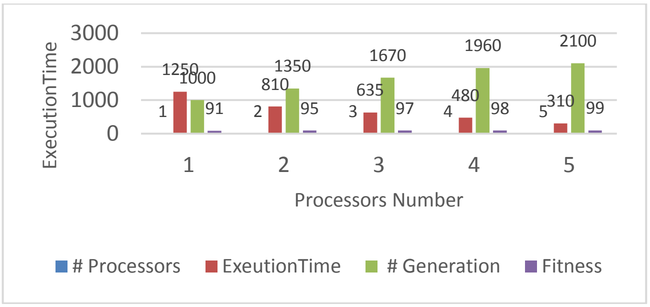

Fig.8.The Relationship between Execution Time and the Number of Processors- Parallel GAs

Figures (9, 10) Refer to the relationship between the number of generations, the hard cost, and soft cost respectively. We note through decreasing the cost of the hard and soft determinants with repeat stages of implementation, which means that it has been achieved two objectives at the same time.

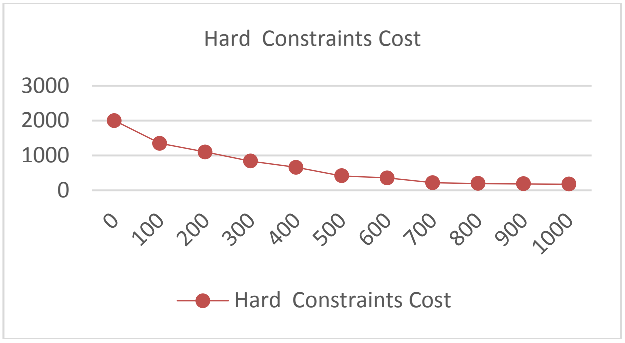

Fig.9. The Hard Cost

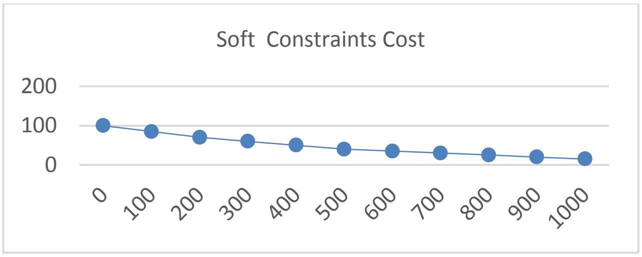

Fig.10.The Soft Cost

Figure 11 describes the result when we run code to test the Superliner speedup in a shared memory system in parallel execution, each element just fits in the level-1 cache memory to find the data it needs for its next operation, it will save time compared get it from random access memory, so there is no cache miss.

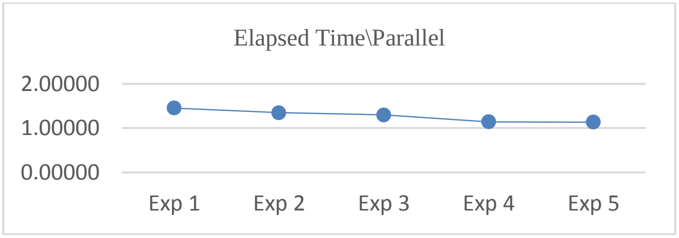

Fig.11. Parallel System: SIM*D*

TABLE I. SPEEDUP (AVG. SEQUENTIAL \ AVG. PARALLEL)

in second	Avg.Sequential	Avg.Parallel	Speedup
Elapsed Time	4.52274	1.27381	3.55055

Figure 12 describes the result when we run code with five slaves in one machine, we tested separately and the average running time of these five experiments. Elapsed time for each experiment is shown in Figure 12. Average running Time =2.80957s.

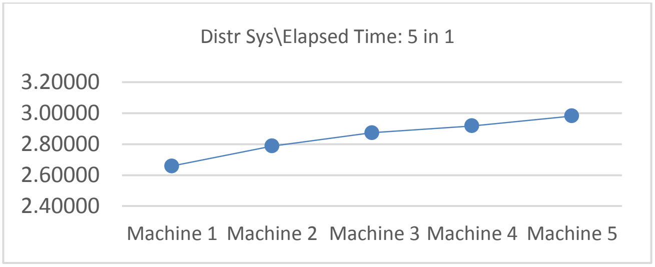

Fig.12. Distributed System: *MISD ((in second))*

Figure 13 describes the result with five slaves running on five different machines, we did five experiments with the same code and the same machines. Elapsed time for all experiments is shown in Figure 14. Average running time = 1.52401s

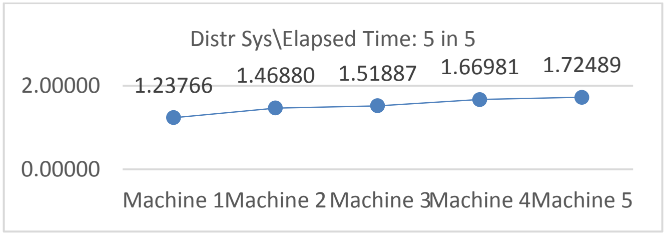

Fig.13. Distributed System: *MIMD (Five slaves in Fives machines)*

In table 2 describes the Ethernet latency slows down the application program. After comparing the results from table1 and table2, our conclusion we can gain Superliner speedup in distributed model to reduce access time, we need to use a fast Ethernet and store the data in a cache memory with each machine running with distributed system.

TABLE II. SPEEDUP IN DISTRIBUTED SYSTEM

Speedup in Distributed Model	
Five slaves in one machine	2.80957
Five slaves in Five machines	1.52401
Speedup	2.16679

IX. CONCLUSIONS AND RECOMMENDATIONS

A. Conclusions: In this paper, we have improved the performance of an evolutionary algorithm after research and tests, we concluded the following: 1. the application of multipurpose and techniques of the parallel and distributed algorithm gave the best performance in solving the problem. 2. After the application of the methodology presented with real data in all experiments of Computer College at Qassim University in all its complexity and constraints where the results were satisfactory. 3. The time required for the processes of communication between processors in parallel and distributed programming case, it takes less time for processing in the same conditions with one processor. 4. The process of redistribution of loads increased complexity as a result of the adoption of the original algorithm on the random variables, which makes it difficult to determine the time required for the implementation of the algorithm. 5. The main idea has been in parallel distributed GAs, so the impact of the research in this kind of algorithms is a larger than others of Parallel GAs.

B. *Recommendations*: After my research has been completed, I recommend the following: 1- Improve the performance Used a new architecture for multi-core multi-cluster [14].

2. Use algorithms to solve many problems in other ways and techniques 3. Applying algorithms in medical fields through the definition, diagnosis and analysis of complex diseases and finding the optimal treatment. 4. Use and application of these algorithms in industrial areas to get the best solutions and results after the definition of the project and identify targets. 5. The distance and locations between the halls we can use it as additional conditions and restrictions on the algorithm to enhance our solutions. 6. We can improve the application of the algorithm through the redistribution of tasks to get the final solutions. 8. We can improve the project used DBMS, which leads creating tables, relationships and reduce the execution time, and size of the data.

REFERENCES

[1] Entropic and Real-Time Analysis of the Search with Panmictic, Structured, and Parallel Distributed Genetic Algorithms. Enrique Alba, Carlos Cotta, Jose M Troya. LCC Technical Report ITI 99-7, 1999.

[2] A Parallel Implementation of Genetic Programming That Achieved Super-Linear Performance. David Andre, John R Koza, 1997.

[3] Designing Efficient and Accurate Parallel Genetic Algorithms. Erick Cantu-Paz. IlliGAL Report 99017, 1999.

[4] M. Aldasht, J. Ortega, and C. Puntonet, "Dynamic load Balancing in Heterogeneous Clusters Exploitation of the Processing Power", 2007.

[5] D. Datta, K. Deb, and CM Fonseca, "Solving class timetabling problem of IIT Kanpur use multi-objective evolutionary algorithm ", 2006.

[6] P. Pongcharoen, W. Promtetn, P. Yenradee, and C. Hicks, "Stochastic optimization timetabling tool for university course scheduling", 2007.

[7] H. Park, A. Grings, M. Santos, and A. Soares, "Parallel hybrid evolutionary computation: Automatic tuning of parameters for parallel gene expression programming", 2008.

[8] Hamid, Norhazlina, Walters, Robert John and Wills,Gary Brian (2014) Performance evaluation of multi-core multi- cluster architecture. In, Emerging Software as a Service and Analytics, Barcelona, ES,03-05 Apr2014. pp, 46-5.

[9] de Toro Negro, F., Ortega, J., Ros, E., Mota, S., Paechter, B., Martin, J.: PSFGA: Parallel processing ,and evolutionary computation for multi-objective optimization . Parallel Computing 30, 721–739 (2004).

[10] Miki, M., Hiroyasu, T., Watanabe, S.: The new model of the parallel genetic algorithm in multiobjective genetic algorithms. In: Congress on Evolutionary Computation CEC 2000, vol. 1, pp, 333–340 (2000).

[11] Alvaro Garcia-Piquer, Andreu Sancho-Asensio, Albert Fornells, Elisabet Golobardes, Guiomar Corral, Francesc Teixidó-Navarro. (2015) Toward high-performance solution retrieval in multiobjective clustering. Information Sciences 32012-25. 2015.

[12] Dr. P.M.G. Moreira and Dr. Paulo J. Tava, Kim C. Long, William S Duff, John W Labadie, Mitchell J Stansloski, Walajabad S Sampath, Edwin K.P. Chong. (2015) Multi-objective fatigue life optimization using Tabu Genetic Algorithms. International Journal of Structural Integrity 6677-688.7-Dec-2015.

[13] Lothar Thiele, Kaisa Miettinen, Pekka J. Korhonen Julian Molina, A Preference-Based Evolutionary Algorithm for Multi-Objective Optimization No Access Evolutionary Computation Fall 2009, Vol. 17, No. 3, Page 411-436.

[14] Hamid, Norhazlina, Walters, Robert John and Wills, Gary Brian, " An Architecture for Measuring Network Performance in Multi-Core Multi-Cluster Architecture (MCMCA)," International Journal of Computer Theory and Engineering vol. 7, no. 1, pp. 57-61, February 2015.

[15] J. L. Ribeiro Filho, C. Alippi, P. Treleaven. “Genetic Algorithm Programming Environments”. Parallel Genetic Algorithms: Theory & Applications, J. Stender (ed.), IOS Press. 1993.

[16] S. Lin, W. F. Punch and E. D. Goodman. “Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach”. Parallel & Distributed Processing. October 1994.

[17] S. Lin, W. F. Punch and E. D. Goodman. “Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach”. Parallel & Distributed Processing.Oct1994.

[18] Yuan Shi. Reevaluating Amdahl’s law and Gustafson’s law.Available:http://joda.cis.temple.edu/~shi/docs/Amdahl/amdahl.html.

[19] T. C. Belding, “The Distributed Genetic Algorithm Revisited”. In L. J. Eshelman. Proceedings of the Sixth ICGA. Morgan Kaufmann, CA, pp. 114-121, 1995.