1. **INTRODUCTION**
   1. **INTRODUCTION**

### ASP .NET MVC:

The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller. The ASP.NET MVC framework provides an alternative to the ASP.NET Web Forms pattern for creating Web applications. The ASP.NET MVC framework is a lightweight, highly testable presentation framework that (as with Web Forms-based applications) is integrated with existing ASP.NET features, such as master pages and membership-based authentication.

### MODEL:

 Model objects are the parts of the application that implement the logic for the application's data domain. Often, model objects store and retrieve model state in a database. For example, a Product object might retrieve information from a database, operate on it, and then write updated information back to a Products table in a SQL Server database.

In small applications, the model is often a conceptual separation instead of a physical one. For example, if the application only reads a dataset and sends it to the view, the application does not have a physical model layer and associated classes. In that case, the dataset takes on the role of a model object.

### VIEW:

  Views are the components that display the application's user interface (UI). Typically, this UI is created from the model data. A view can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants. An example would be an edit view of a Products table that displays text boxes, drop-down lists, and check boxes based on the current state of a Product object.

### CONTROLLER:

The controller accepts input and converts it to commands for the model or view.  These are the components that handle user interaction, work with the model, and ultimately select a view to render that displays UI. In an MVC application, the view only displays information; the controller handles and responds to user input and interaction. For example, the controller handles query-string values, and passes these values to the model, which in turn might use these values to query the database.

## Location Mapping/Restacking Tool:

## This tool provides a single window view to help real estate management. It helps to strategize the move globally, across facilities to make optimum and precise usage of available space. It helps senior managers to identify the location needs and plan accordingly. It also makes them to review existing resource or group or an organisation unit, current location. This tool will also be a great help during emergency relocation of teams. Region based administration roles can be created and region based location can be maintained separately. Major restacking can be done with minimal effort. Details/Extract of this application can be provided to other application as services. This tool deals with Digital Workplace and Mysource to explore where this tool best can be applied for enterprise usage.

**2. PROBLEM STATEMENT**

* 1. **PROBLEM STATEMENT**

The location mapping or restacking of employees/teams within the organization was currently maintained using Excel files.  As it is manual process, there was problem in effectively managing the available space and time consuming. In addition, there was no central control or reporting capabilities. In order to overcome these, we created service for mapping location and restacking using MVC .NET framework.

1. **SYSTEM CONFIGURATION**
   1. **HARDWARE REQUIREMENTS**

The minimum hardware configuration of the system on which the project was developed is as follows:

* + - Processor : Intel(R) Core(TM) 3.2 Ghz
    - CPU clock : 450 Mhz
    - RAM : 8 GB
    - HardDisk : 40 GB
    - Keyboard : Any type
    - Mouse : Any type
    - Monitor : Any type/Color Monitor preferable

### 3.2 SOFTWARE REQUIREMENTS

* Operating System : Windows 7
* Language Used : C sharp(Visual Studio 2013)

**4.2.1 Language:**

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

**4.2.2** **Working Environment:**

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

**4.2.3 Visualization Techniques:**

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

**4.2.4 Mathematical Function Library:**

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

**4.2.5 Application Program Interface (API):**

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

**5. METHODOLOGY**

**5.1. MODULES**

* Checking the constraints
* Calculation of Makespan and Total cost
* Calculation of Fitness value
* Optimization of scheduling using GA operators
  1. **MODULE DESCRIPTION**
     1. **Checking the constraints:**
* In this module checking of the CPU, memory constraints of incoming tasks with respect to the available resources.
* Initially all the VMs are in the ideal state once the scheduler receives the workflow then it will check the requirements of the user with the available resource, once all the constraints are satisfied then tasks are mapped to the resources.
* If the VM is allocated to the task then it will be changed to the busy state.
* Every VM is queued with n number of tasks and priority is given based on the waiting time of the tasks.
* Once all the tasks are mapped then next is to find the makespan (total execution time) and the total cost of consuming resources.
* Finally fitness value is evaluated using our own fitness function for the initial population then we employ Genetic Algorithm in order to generate new offspring to attain the best scheduled order.
  + 1. **Calculation of Makespan and Total cost:**
* In this module, calculation of the makespan and total cost of each chromosome has been done.
* Once the mapping of tasks to the available resources completed, we have to sort the tasks queued in the every VM pool based on its waiting time.
* After the sorting of tasks, execution starts by its order and find the summation of total execution time of each VM pool which is the makespan.
* Once the completion of makespan, next is to calculate the cost of allocated resources by making the product of cost of one unit and the makespan. Select the maximum execution time from the VM pool which consists of n number of tasks and assigned it to the makespan.
* This will be repeated for every VM.
* IP(initial population) indicates the set of chromosomes , E=Execution time of VMi where VM is the Virtual Machine.
* The formula to calculate the makespan and the total cost is as follows:
* ***Makespan=max(E) (6)***
* ***Totalcost=∑(E\* cost of VMi) (7)***
  + 1. **Calculation of Fitness value:**
* This module calculates the fitness value for set of chromosomes that is population set.
* Find the maximum and minimum makespan and also find the maximum and minimum total cost for the calculation of fitness value.
* Next is to apply those values in the below fitness calculation formula.
* The maximum fitness value denotes the best scheduled order that gives the minimum execution time and cost as compared to the other scheduled order in the given set of chromosomes reside in a population set.
* Repeat this module for n number of iterations once we attain the expected output.
* For better result find the best orders makespan and total cost from each iteration again find the fitness value.
* ***Max\_Makspn=max(Makespani) (8)***
* ***Min\_Makspn=min(Makespani) (9)***
* ***Max\_TotalCost=max(Totalcosti) (10)***
* ***Min\_TotalCost=min(Totalcosti) (11)***

(12)

* Fitness keeps a record of maximum *(Max\_TotalCost and Max\_Makspn*) and minimum *(Min\_TotalCost and Min\_Makspn)* values of *MakeSpan and Total cost* in order to provide a global evaluation to each solution.
* These values update on each iteration on the main loop of our algorithm.
* Additionally, the fitness equation enables the user to assign priority to a given optimization objective employing w1and w2as time and cost optimization weights respectively where w1+ w2= 1.
  + 1. **Optimization of scheduling using GA operator:**
* Two GA operators were used for the optimization process and those operators are crossover and mutation.
* Set of chromosomes are the inputs to Crossover.
* From the given set of input select two set of chromosomes, then perform crossover operation.
* Repeat the steps for N/2 times.
* Finally new set of chromosomes are generated that is known as offspring.
* Similarly mutation is also performs as same as crossover.
* In crossover, single point crossover operator is employed, where Single point crossover means only one intersection was set up in the individual code, at that point part of the pair of individual chromosomes are interchanged with one another. In mutation one bit is selected at random, and then the value in that point in one chromosome is interchanged with the value in the other chromosome.
* Repeat this module until the maximum fitness value is obtained that results the best scheduled order.
  1. **PROPOSED ALGORITHM:** 
     1. **ALGORITHM 1 – CALC\_FITNESS**

Input: N no of Chromosomes (different set of scheduled order)

Output: Selection of best pair of chromosome (the best schedule)

1: Read data from IP,Provider and Consumer dataset;

2: T= No of tasks order in N;

3: t=No of tasks in T

4: V= No of VMs provided;

5: Set N

6: **For** i=1: N

7: **For** j=1: T

8: **For** j=1: t

9: Check the CPU constraint and allocate the tasks to the VM

10: **End**

11: **For** k=1: V

12: Sort the tasks queued in VM based on its waiting time

13: E= ∑(Execution time of tasks in VM queue)

14: cost=E\*cost of VM per unit

15: **End**

16: makespan=max(E)

17: totalcost=∑cost

18: **End**

19: Max\_makespan=max(makspan);

20: Min\_makespan=min(makespan);

21: Max\_cost=max(totalcost);

22: Min\_cost=min(totalcost);

23:

24: **End**

25: **do**

25: Offspring=ALGORITHM 2(IP);

26: IP=Offspring;

27: **GoTo** Step 1;

28: **While** n;

29: Fittest Solution = max(Fitness);

* + 1. **ALGORITHM 2 – MF-GA (MULTI FOCUSED GENETIC ALGORITHM)**

Input: selected best pair of chromosomes (Two different scheduled order)

Output: offspring (new optimal scheduled order)

1: Evaluation

2: Crossover;

3: Mutation;

4: **Return** offspring (new schedule);

**ALGORITHM 5.3.1 - DESCRIPTION:**

Read the datasets of Provider and Consumer from the Excel then according to the number of user request we have randomly generate the Initial Population set.

Firstly Steps (8-10) performs the allocation of resources to the tasks based on their CPU constraints. Then Steps (11-15) sort the tasks according to their waiting time inside every VM. In steps (7-18) calculation of the makespan and totalcost will be done. Finally from the steps (6-24) evaluates the fitness value for IP that randomly generated. Steps (25-28) recursively performs the algorithm2 until it reaches the n number of iterations.

**ALGORITHM 5.3.2 - DESCRIPTION:**

This is purely based on GA where the input is obtained from the Algorithm1 and then it will be proceed with the GA operators say crossover and mutation.

Finally it returns the new population set.

1. **TESTING**

**6.1. TIME COMPLEXITY:**

We have illustrated the time complexity of proposed algorithm in the every module included in that in the following table.

|  |  |  |
| --- | --- | --- |
| STAGE | DESCRIPTION | COMPLEXITY |
| Generate initial population | Assign a random number [1:VM] to each gene (t) from the complete population (s) augmented | (t)(s)(IP) |
| Evaluation | Calculates F,fitness for each chromosome | (s) |
| Selection | Run roulette wheel (p) times to build a new population taking off number of chromosomes from elite operator | (s–l) |
| Crossover | The complete population has a maximum probability of 0.6 to go through crossover operator where each operation depends on a constant number of crossing points | (0.6)(s)(c) |
| Mutation | The complete population has a maximum probability of 0.3 to go through mutation operator where the number of interchangeable genes remains constant | (0.3)(s)(c) |

Table.2. Time complexity of the modules

1. **RESULTS AND DISCUSSION**

**7.1. RESULTS INTERPRETED:**

In general GA is used to attain the best optimal or the nearest best optimal solution from the set of population. Above statement is the reason why we did our implementation with GA even though PSO is better than GA. We have done implementation our proposed algorithm with the slight modification of general genetic algorithm, it will perform the existing genetic algorithms that are adopted for scheduling. The distinction of those GA’s with our algorithm is that the representation of the chromosome order.

Task scheduling is the main problem in cloud computing scenario. Efficient task scheduling is essential for better utilization of resources. In this paper three scheduling algorithms are discussed such as Min-Min, Max-Min and Multi Focused Genetic Algorithm. The idea of genetic algorithm comes from natural selection which consists of population generation, selection, and mutation. This paper is our sincere attempt to reduce the makespan of algorithm by using enhanced Max Min for initializing the population in GA. The experimental results show that the MF-GA minimizes the makespan effectively than other algorithm. This idea can be further extended in which we can use execution cost of the resource as fitness criteria.

The following figure illustrates the graphical representation of the comparison of proposed algorithm with Min-Min and Min-Max algorithm. It depicts the total number of tasks allocated with the resources and the makespan acquired with the three of these algorithms. From that figure we can understand that MF-GA outperforms Max-Min algorithm and Min-Min algorithm will slightly increase the number of tasks allocated and decrease the execution time.

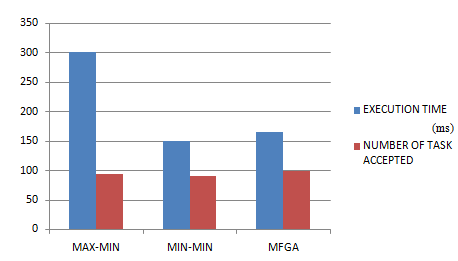


Fig.7.1. Comparison of the Algorithms

The following figures are the output that we obtained in our simulation. Fig.2. indicates the graph of the fitness value calculated for n number of iterations. From that visualization, we understood that maximum fitness value has the best scheduled order from the new population set. We know that range of the fitness between 0<=f<=1, we optimize the results using the MF-GA.

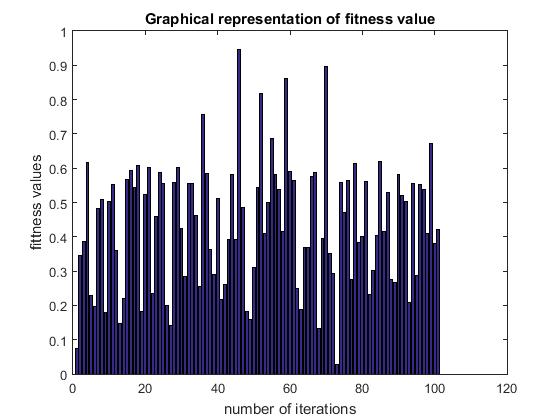


Fig 7.2.Illustration of fitness value

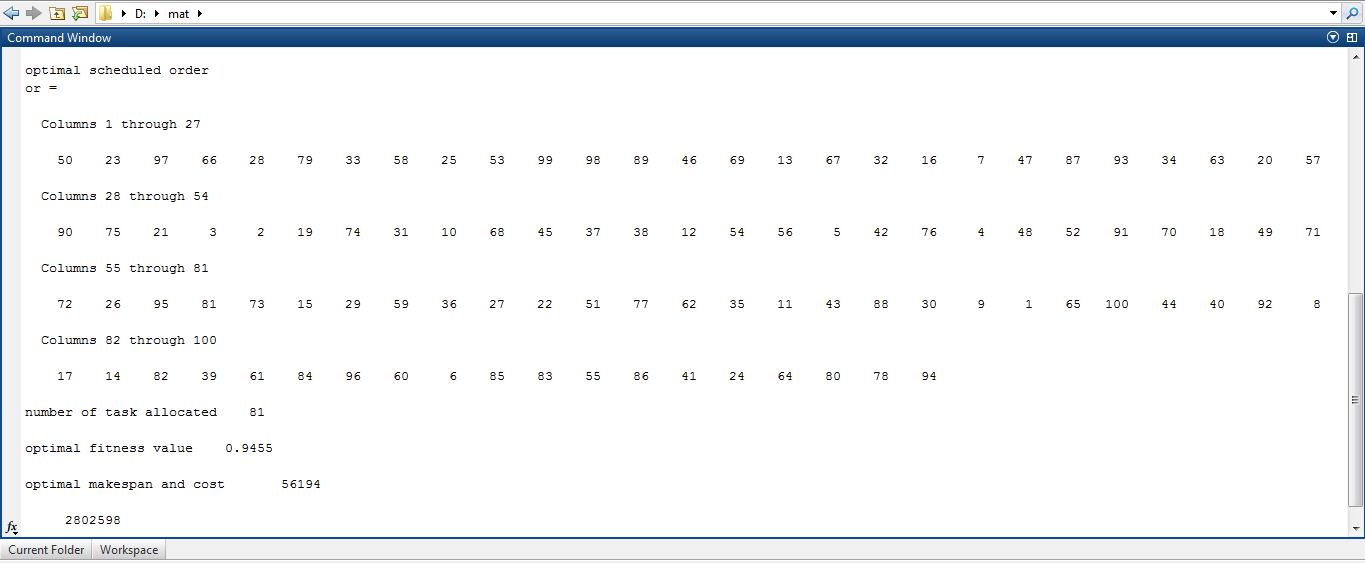


Fig.7.3. Results of the simulation

The above figure depicts the output obtained from the simulation environment that displays the best optimal scheduled order of the workflow, number allocated tasks to the available resources in that iteration, fitness value for that particular order, optimal makespan and the total cost for executing and allocating resources to that order.

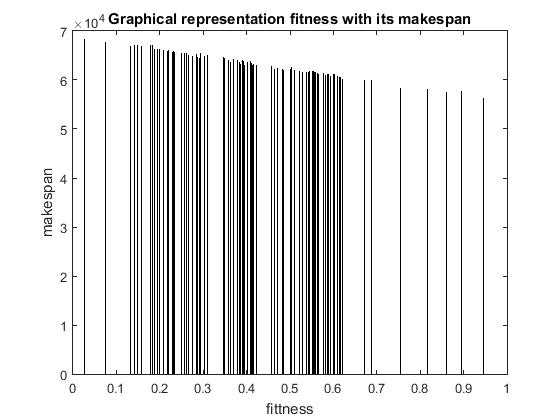


Fig.7.4. Illustration of the fitness with makespan

Fig.7.4. shows the graph plotted with the values of fitness values and its corresponding makespan values in each and every iterations. For example let us consider that the user gives 100 tasks for scheduling and in provider’s side has 10 VMs. And now the above graph depicts the plotted graph with makespan and fitness values.

Once the constraints are satisfied then the tasks were mapped on to the resources, there is a chance of rejection during the allocation due to unsatisfied constraints.

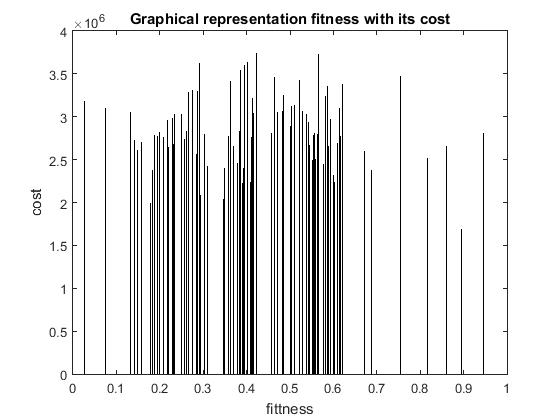


Fig.7.5 Illustration of the fitness with cost

Fig.7.5 shows that fitness value and its corresponding cost for that scheduling, we know that if the fitness value is increased then the cost and makespan decreases. This will depends on the weighted values such as w1 and w2.

**CONSUMER’S DATASET:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task id | CPU(MHz) | Exe Time | cache size(kb) | waiting time |
| 1 | 300 | 8627 | 512 | 18 |
| 2 | 200 | 7480 | 256 | 23 |
| 3 | 150 | 2034 | 64 | 25 |
| 4 | 200 | 9798 | 64 | 14 |
| 5 | 300 | 8191 | 64 | 22 |
| 6 | 300 | 4253 | 128 | 5 |
| 7 | 500 | 9343 | 512 | 13 |
| 8 | 250 | 4062 | 128 | 10 |
| 9 | 300 | 4451 | 128 | 24 |
| 10 | 200 | 9520 | 128 | 24 |
| 11 | 500 | 5833 | 128 | 11 |
| 12 | 150 | 2020 | 128 | 14 |
| 13 | 200 | 3037 | 256 | 11 |
| 14 | 500 | 2036 | 256 | 17 |
| 15 | 300 | 9765 | 512 | 21 |
| 16 | 300 | 7255 | 256 | 15 |
| 17 | 250 | 3738 | 256 | 11 |
| 18 | 300 | 2956 | 64 | 5 |
| 19 | 150 | 4251 | 128 | 19 |
| 20 | 150 | 7996 | 64 | 15 |
| 21 | 300 | 2356 | 512 | 18 |
| 22 | 150 | 2930 | 512 | 24 |
| 23 | 500 | 2278 | 128 | 12 |
| 24 | 250 | 5616 | 256 | 16 |
| 25 | 200 | 5079 | 512 | 24 |
| 26 | 500 | 5785 | 256 | 19 |
| 27 | 200 | 6852 | 64 | 8 |
| 28 | 300 | 9016 | 128 | 6 |
| 29 | 500 | 7357 | 256 | 8 |
| 30 | 300 | 2173 | 256 | 5 |
| 31 | 300 | 7624 | 512 | 24 |
| 32 | 300 | 5516 | 128 | 10 |
| 33 | 250 | 4927 | 64 | 9 |
| 34 | 300 | 7357 | 256 | 11 |
| 35 | 150 | 4105 | 128 | 11 |
| 36 | 150 | 3810 | 64 | 13 |
| 37 | 200 | 7000 | 512 | 24 |
| 38 | 150 | 6444 | 128 | 15 |
| 39 | 200 | 7580 | 64 | 25 |
| 40 | 150 | 9998 | 128 | 22 |
| 41 | 250 | 3216 | 64 | 14 |
| 42 | 250 | 7318 | 64 | 11 |
| 43 | 150 | 9001 | 128 | 23 |
| 44 | 200 | 9626 | 128 | 21 |
| 45 | 200 | 2673 | 64 | 20 |
| 46 | 150 | 3844 | 512 | 7 |
| 47 | 500 | 5470 | 256 | 10 |
| 48 | 150 | 5624 | 512 | 12 |
| 49 | 250 | 5676 | 128 | 19 |
| 50 | 250 | 3517 | 64 | 19 |
| 51 | 150 | 6783 | 256 | 14 |
| 52 | 300 | 8886 | 128 | 5 |
| 53 | 250 | 6258 | 64 | 13 |
| 54 | 150 | 3744 | 128 | 19 |
| 55 | 300 | 9933 | 64 | 15 |
| 56 | 250 | 9999 | 128 | 22 |
| 57 | 200 | 3017 | 512 | 20 |
| 58 | 300 | 9104 | 512 | 7 |
| 59 | 150 | 8708 | 512 | 20 |
| 60 | 250 | 9398 | 128 | 22 |
| 61 | 150 | 6664 | 512 | 20 |
| 62 | 200 | 8719 | 512 | 8 |
| 63 | 150 | 8488 | 64 | 6 |
| 64 | 200 | 2168 | 64 | 21 |
| 65 | 300 | 3077 | 256 | 17 |
| 66 | 300 | 2308 | 256 | 12 |
| 67 | 200 | 4011 | 64 | 15 |
| 68 | 200 | 2961 | 128 | 21 |
| 69 | 150 | 5532 | 512 | 15 |
| 70 | 300 | 6631 | 256 | 25 |
| 71 | 150 | 5608 | 256 | 25 |
| 72 | 500 | 2240 | 64 | 13 |
| 73 | 150 | 9942 | 512 | 14 |
| 74 | 300 | 4141 | 512 | 19 |
| 75 | 200 | 5239 | 64 | 11 |
| 76 | 300 | 4263 | 512 | 9 |
| 77 | 150 | 5523 | 512 | 9 |
| 78 | 500 | 7216 | 64 | 25 |
| 79 | 300 | 5481 | 128 | 16 |
| 80 | 200 | 5677 | 256 | 21 |
| 81 | 500 | 5107 | 128 | 6 |
| 82 | 300 | 4282 | 128 | 7 |
| 83 | 200 | 9128 | 512 | 5 |
| 84 | 250 | 2156 | 256 | 9 |
| 85 | 200 | 4593 | 256 | 22 |
| 86 | 250 | 7442 | 128 | 10 |
| 87 | 150 | 8099 | 256 | 24 |
| 88 | 250 | 3126 | 512 | 14 |
| 89 | 250 | 7264 | 256 | 7 |
| 90 | 150 | 8525 | 256 | 19 |
| 91 | 250 | 6186 | 64 | 20 |
| 92 | 200 | 7639 | 64 | 16 |
| 93 | 500 | 3282 | 128 | 24 |
| 94 | 150 | 8421 | 64 | 16 |
| 95 | 250 | 9198 | 512 | 20 |
| 96 | 200 | 3768 | 512 | 22 |
| 97 | 300 | 7789 | 256 | 21 |
| 98 | 250 | 4482 | 256 | 14 |
| 99 | 200 | 3988 | 512 | 14 |
| 100 | 250 | 5095 | 512 | 17 |

Table 7.1 Sample Dataset of Consumers

**CONSUMER’S DATASET:**

|  |  |  |  |
| --- | --- | --- | --- |
| VM ID | CPU(MHz) | L2 cache size(kb) | cost(per ms) |
| 1 | 200 | 128 | 8 |
| 2 | 500 | 64 | 8 |
| 3 | 200 | 512 | 5 |
| 4 | 250 | 64 | 9 |
| 5 | 150 | 256 | 6 |
| 6 | 150 | 256 | 5 |
| 7 | 200 | 512 | 4 |
| 8 | 500 | 512 | 2 |
| 9 | 200 | 512 | 8 |
| 10 | 500 | 10 | 7 |

Table 7.2 Sample Dataset of Providers

The above two different tables show that the sample datasets for both providers and consumers. From the table we can identify the different parameters required and provided by providers as well as consumers. By using these tables scheduler can easily allocate the incoming tasks to the available resources in the provider’s side. CPU capacity will indicates the speed of the VM in the terms of MIPS and Cache memory size in KB will depicts the required speed of the CPU/ processor.

In the consumer’s dataset waiting time shows the tolerance time that will be accepted by the requester. And the execution time is the estimation of performing that job requested y the user.

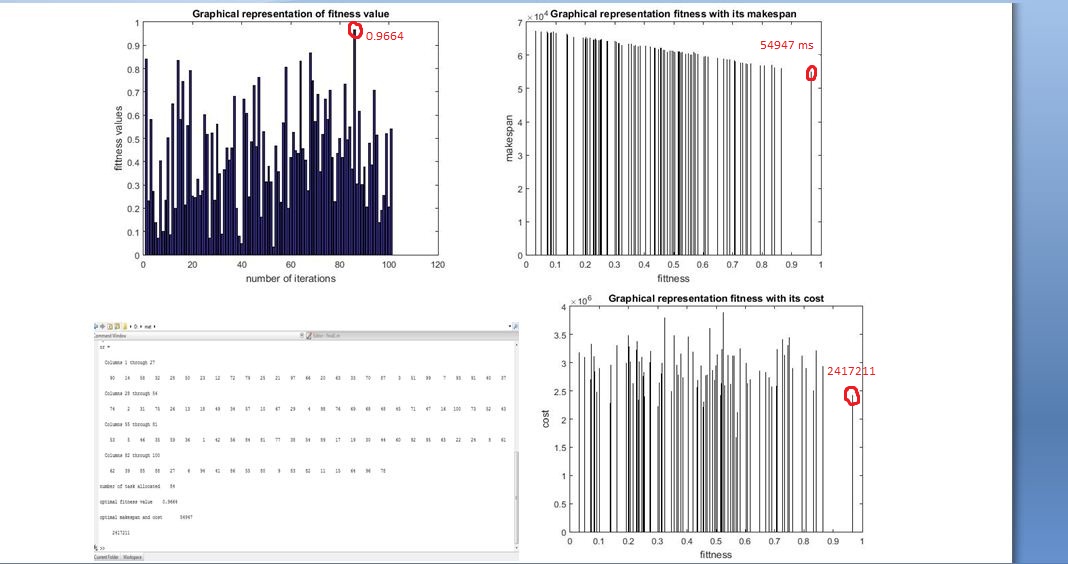


Fig.7.6 Illustration of Case (1)

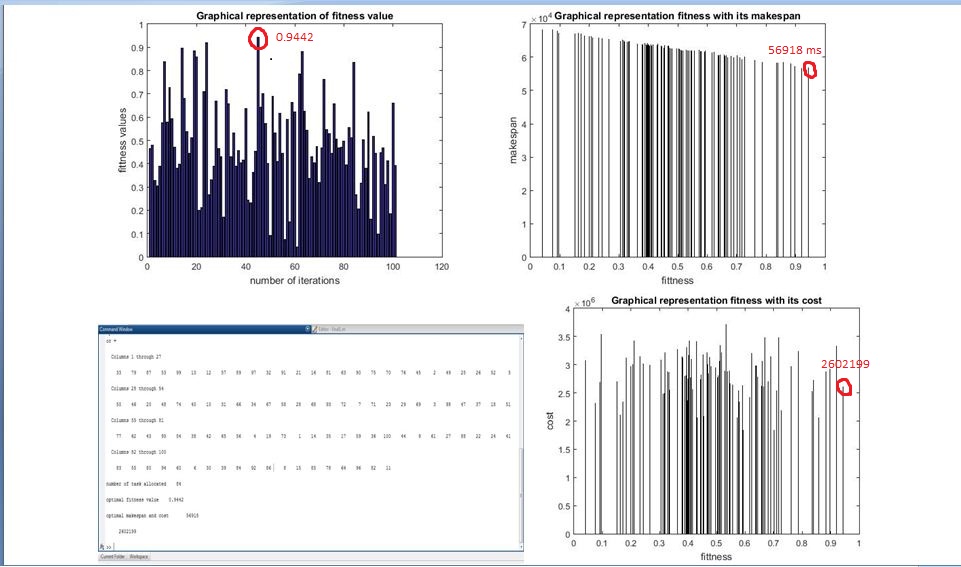


Fig.7.7 Illustration of Case (2)

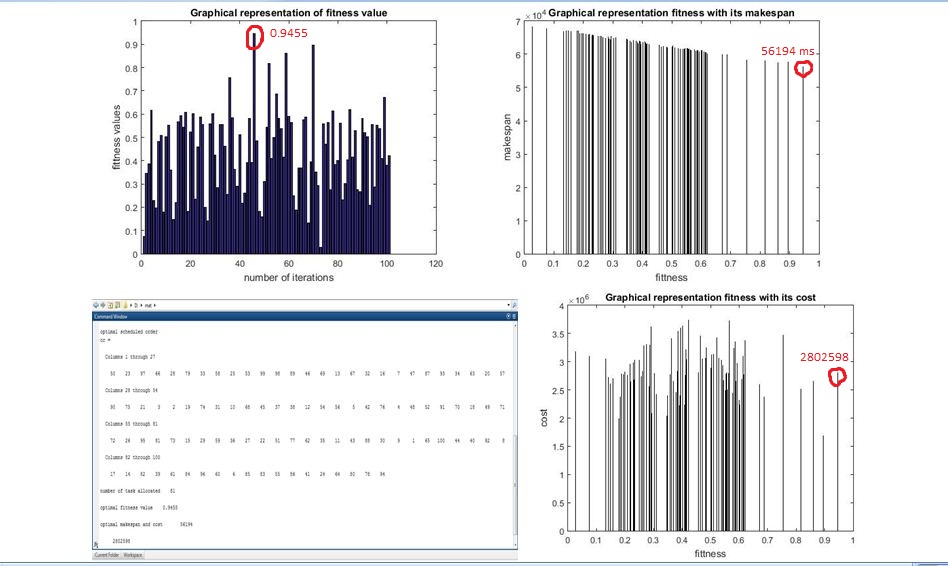


Fig.7.8 Illustration of Case (3)

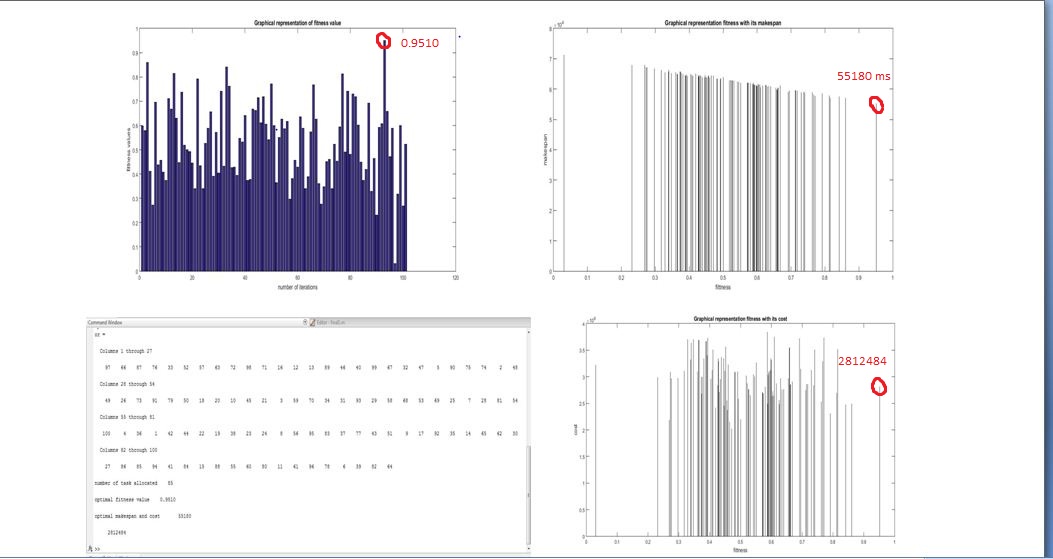


Fig.7.9 Illustration of Case (4)

The above figures depict the different test cases of our proposed work. Random distribution is employed in order to generate the data for both cloud providers and consumers. Firstly we run our simulation and get the results of best optimal scheduled order with the maximum fitness value. Likewise we did similar runs for the distinguish datasets.

And the fitness value is purely depends on the values of the dataset. We attained the following fitness values from the different runs such as 0.9510, 0.9455, 0.9442 and 0.9644. So the fitness value with 0.9664 shows that same number iterations with different dataset as per the values in dataset, it has the highest fitness value.

* 1. **CONCLUSION AND FUTURE WORK:**

From the results we have attained, shows that our proposed algorithm outperforms the existing algorithms such as Min-Min, Max-Min and general genetic algorithm. Scheduling algorithms have a lot to do with the efficiency of cloud computing environments through selection of suitable resources and assignment of workflows to them. Given the factors affecting their efficiency, these algorithms try to use resources optimally and increase the efficiency of this environment. This study attempted to present a new algorithm based on genetic algorithm, which can respond to most requirements of users and cloud environments and also find the best solution from the set of feasible solutions. Furthermore, it can be ideal for a static cloud environment, enhancing system efficiency. The parameters considered in this algorithm can fulfill the requirements of both users and cloud environments, such as meeting the deadlines, cost constraints, load balancing and enhancing fault tolerance.

The noteworthy point is that the newly proposed algorithm managed to complete all workflows based on estimating most requirements within a period shorter than that of min-min algorithm which focuses only on reducing the deadline. By taking into account the key parameters of load balancing, the new algorithm can greatly contribute to system performance, managing to utilize the resources optimally and complete the workflows within the shortest possible time.

Given the findings above and the results obtained by simulation of the newly proposed algorithm compared to its counterparts, it can be concluded that our algorithm yielded far better results through fulfilling most requirements of users and cloud environments, only a few of which had been previously met by other algorithms.

**PUBLICATIONS**:

In the initial stage of our work we have done literature review of several journals related to the resource allocation and task scheduling using Genetic Algorithm. Finally we published the survey paper on the title of “A SURVEY OF SCHEDULING USING GENETIC ALGORITHM IN CLOUD ENVIRONMENT” in the conference “National Conference on Big Data Analytics and Mobile Technologies NCBM 2017”, Thiagarajar College of Engineering, National Conference, Madurai, 23rd March 2017 .

1. **REFERENCES**

**REFERENCES:**

1. Dasgupta K, Mandal B, Dutta P, Mandal JK, Dam S. A Genetic Algorithm (GA) based load balancing strategy for cloud computing. Proc Technol 2013;10:340–7.

2. Pop F, Dobre C, Cristea V. Genetic algorithm for DAG scheduling in grid environments. In: 5th IEEE int conf intell comput commun process; 2009. p. 299 305

3. Ge Y, Wei G. GA-based task scheduler for the cloud computing systems. In: Proc int conf web inf syst min, vol. 2; 2010. p. 181–6. http://dx.doi.org/10.1109/WISM.2010.87

4. Zheng Z, Wang R, Zhong H, Zhang X. An approach for cloud resource scheduling based on parallel genetic algorithm. In: 3rd IEEE int conf comput res dev. IEEE; 2011. p. 444–7. <http://dx>. doi.org/10.1109/ICCRD.2011.5764170

5. Zhao C, Zhang S, Liu Q, Xie J, Hu J. Independent tasks scheduling based on genetic algorithm in cloud computing. In:5th int conf wirel commun netw mob comput; 2009. p. 1–4.http://dx.doi.org/10.1109/WICOM.2009.5301850.

6. Ghorbannia Delavar A, Aryan Y. HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems. ClusterComput 2014;17:129–37. http://dx.doi.org/10.1007/s10586-013-0275-6.

7. Joseph CT, Chandrasekaran K, Cyriac R. A novel family genetic approach for virtual machine allocation. Proc Computer Science

8.Gu J, Hu J, Zhao T, Sun G. A new resource scheduling strategy based on genetic algorithm in cloud computing environment.J Comput 2012;7:42–52.

9. Kaur S, Verma A. An efficient approach to genetic algorithm for task scheduling in cloud computing environment. Int J InfTechnol Comput Science 2012;4:74–9.

10. Wang T, Liu Z, Chen Y, Xu Y, Dai X. Load balancing task scheduling based on genetic algorithm in cloud computing. In: IEEE 12th int conf dependable auton secur comput; 2014. p.146–52.

11. Zhu K, Song H, Liu L, Gao J, Cheng G. Hybrid genetic algorithm for cloud computing applications. In: IEEE Asia-Pacific serv comput conf; 2011. p. 182–7.

12. Tao F, Feng Y, Zhang L, Liao TW. CLPS-GA: a case library and Pareto solution-based hybrid genetic algorithm for energyaware cloud service scheduling.

13. Xiong A, Xu C. Energy efficient multiresource allocation of virtual machine based on PSO in cloud data center. Math Probl Eng 2014:2014.

14. Bahman Keshanchi a , Alireza Souri a , Nima Jafari Navimipour b , ∗ An

improved genetic algorithm for task scheduling in the cloud environments

using the priority queues: Formal verification, simulation, and statistical

testing, The Journal of Systems and Software, *124July 17 (2017) 1–21.*

15. Saima Gulzar Ahmada, Chee Sun Liewa, Ehsan Ullah Munirb, Ang Tan

Fonga,Samee U. Khanc,∗ , A hybrid genetic algorithm for optimization of

scheduling workflow applications in heterogeneous computing systems,Journal of Parallel Distribution Computation.

16. ZHANG Yan-huaa, Feng Leia, Yang Zhia, a\*,Optimization of Cloud

Database Route Scheduling Based on Combination of Genetic Algorithm and

Ant Colony Algorithm, Advanced in Control Engineering and Information

Science, *15 ( 2011 ) 3341 – 3345.*

17.Israel Casasa,b,∗, Javid Taheric, Rajiv Ranjanb,e, Lizhe Wangd, Albert

Y. Zomayaa, GA-ETI: An enhanced genetic algorithm for the scheduling of

scientificworkflows in cloud environments, Journal of Computational Science,

JOCS-540, 14, August 2016.

18. HAO-CHIN CHANG, YEH-PENG CHEN, TUNG-KUAN LIU, AND

JYH-HORNG CHOU, Solving the Flexible Job Shop Scheduling Problem

With Makespan Optimization by Using a Hybrid Taguchi-Genetic Algorithm,

Solving the FJS Scheduling Problem, VOLUME 3, 2015.

19. JASRAJ MEENA, MALAY KUMAR, AND MANU VARDHAN, Cost

Effective Genetic Algorithm for Workflow Scheduling in Cloud Under Deadline Constraint, CEGA for Workflow Scheduling in Cloud, VOLUME 4, (5065-5080) 2016.

20. Saima Gulzar Ahmada, Chee Sun Liewa, Ehsan Ullah Munirb, Tan Fong

Anga, Samee U. Khanc A Hybrid Genetic Algorithm for Optimization of

Scheduling Workflow Applications in Heterogeneous Computing Systems,

Journal Parallel Distribution Computation, S0743-7315(15)00186-0, 9 October 2015.

21. Saima Gulzar Ahmada, Chee Sun Liewa, Ehsan Ullah Munirb, Tan Fong

Anga, Samee U. Khanc A Hybrid Genetic Algorithm for Optimization of

Scheduling Workflow Applications in Heterogeneous Computing Systems,

Journal Parallel Distribution Computation, S0743-7315(15)00186-0, 9

October 2015.

22. HAO-CHIN CHANG, YEH-PENG CHEN, TUNG-KUAN LIU, AND

JYH-HORNG CHOU, Solving the Flexible Job Shop Scheduling Problem

With Makespan Optimization by Using a HybridTaguchi-Genetic Algorithm,

Solving the FJS Scheduling Problem, VOLUME 3, 2015.

23. JASRAJ MEENA, MALAY KUMAR, AND MANU VARDHAN, Cost

Effective Genetic Algorithm for Workflow Scheduling in Cloud Under Deadline Constraint, CEGA for Workflow Scheduling in Cloud, VOLUME 4,

(5065-5080) 2016.