International Journal of Computer Networks and Applications
Volume 1, Issue 1, November - December (2014)

RESEARCH ARTICLE

An Empirical Model of Job Shop Scheduling With Related To Tiny Chemical Assembly Instructions Inside of Living Things and Gels Techniques

Ganesh Potta
Computer Science and Engineering
ganeshpotta392@gmail.com

Santosh Naidu P
Computer Science and Engineering
amsan2015@gmail.com

Abstract – Usage of (math-based/computer-based) useful things/valuable supplies is always an interesting research issue in the field of Grid figuring out/calculating. Job shop scheduling is a combinatorial optimization problem it finds possible number of solution for best solution. In this paper we are proposing a blended approach of (related to tiny chemical assembly instructions inside of living things) set of computer instructions and GELS set of computer instructions for identifying missed job or best solution from set of samples or (genetic information storage areas) which contains jobs, operation and time span.

Index Terms – Gels, Job-Shop, assembly instructions

1. INTRODUCTION

Shop scheduling problems belong to the class of multi-stage scheduling problems, where each job consists of a set of operations. For describing these problems, we use the standard3-parameter classification a j b j g. The limit/guideline points to/shows the machine (surrounding conditions), the limit/guideline b describes job characteristics, and the limit/guideline g gives the optimization judging requirement. In such a shop scheduling problem, a set of n jobs J1, J2,.....,Janh as to be processed one set of m machines M1,M2.....,Mm. The processing of a job Jib on a particular machine Mj is represented as an operation and shortened by (i,j). Each job Ji consists of a number ni of operations. For the pre-decided scheduling problems, the processing time pi j of each operation (I, j) is given in advance.

Among the shop scheduling problems, there are three basic types: a flow-shop, a job shop and an open-shop. In a flow shop problem (a = F), each job has exactly m operations, and the (related to computers and science) route (or machine order) in which the job passes through the machine sis the same for any job. Without loss of (statement that's not detailed), we assume that the (related to computers and science) route for any job is given by M1ï*f* M2ï*f*.....Mm. In a job shop problem (a = J), a specific (related to computers and science) route Mj1ï*f* Mj2ï*f* ....Mjni is given for each job Ji,1a‰¤ i a‰¤ n. Note that the number of operations per job in is equal to m for the classical job shop problems, but this number may be also smaller than m or larger than m (re circulation; in this case we may use the note/way of writing (I,j,k) for the k-th processing of job Ji on machine Mj). In an open shop problem (a = O), no (related to computers and science) routes are (forced (on people)/caused an inconvenient situation) on the jobs. Usually, it is assumed that each job has to be processed on any machine. [1]

(Related to tiny chemical assembly instructions inside of living things) sets of computer instructions have been originally developed by the work with a population composed of people. The (related to tiny chemical assembly instructions inside of living things) structure of a (related to the body function of living things) person is composed of several (genetic information storage areas). Each of these (genetic information storage areas) is composed of several (tiny chemical assembly instructions inside of living things) each of which consists of some (pairs of genes). In combinatorial optimization, a person is usually identified with a (genetic information storage area). A (genetic information storage area) is further split into number of (tiny chemical assembly instructions inside of living things) (in some papers a (tiny chemical assembly instruction inside of living things) is also identified with an (one of a pair of genes)). Before applying (related to tiny chemical assembly instructions inside of living things) set of computer instructions to scheduling problems, an appropriate (translating/putting into secret code) (or representation) of the solution must be introduced.

Representation of a solution In many (related to tiny chemical assembly instructions inside of living things) sets of computer instructions, often a binary (translating/putting into secret code) is used, (in other words), each (tiny chemical assembly instruction inside of living things) of an individual contains either the number 0 or the number 1. Of course, integers can be converted are presentation but for combination and

RESEARCH ARTICLE

arrangement problems, this is often not good/helpful. For flow shop problems with the combination [2] and arrangement condition a standard way of representing (able to be done) solution is the combination and arrangement code, (in other words), an individual consists of a string of lengthen, and the ith (tiny chemical assembly instruction inside of living things) contains the index of the job at position i, so an individual describes the job sequence chosen on all machines. In the case of a regular judging requirement, a combination and arrangement is (changed secret code into understandable language) into an(able to be done) solution by construction the resulting semi-active schedule.

For job shop scheduling problems, the situation is a bit different. Here several (translating/putting into secret code) (success plans/ways of reaching goals) exist, and it is not clear in advance which is the best. Many authors tell the difference between a direct and an indirect representation. In the first case, a solution is directly (translated/put into secret code) into the genotype (i.e., the schedule itself is (translated/put into secret code)), but in the second case not (this means that an indirect representation (translates/puts into secret code) the instructions to a schedule builder).

2. LITERATURE SURVEY

Select Operation: Here, tournament [3] operator was used to choose the (genetic information storage areas). After accounting the fitness of each (genetic information storage area), a (genetic information storage area) which has more fitness than the others is selected for applying combination operators on it.

Crossover operation: The crossover operator used in this set of computer instructions is a basic two-point crossover which works as follows: Two random (genetic information storage areas) were selected by previous phase and exchanges a random part of those.

Combination Operator: Since in change operator, the points which the changes are applied on this are chosen randomly, so in proposed method instead of this operator, (genetic information storage areas) changes were done through GELS set of computer instructions for making the neighbour solutions, because the energetically/changing quickly as needed changing the ways in this set of computer instructions isn't randomly. When each (genetic information storage area) was selected by previous phase, was controlled by GELS set of computer instructions until the solution of its neighbour was gotten.

Each (tiny chemical assembly instruction inside of living things) on (genetic information storage area) is considered as a dimension of the problem. In fact the problem's dimensions are just the neighbouring solutions which are received/got by changing the current solution. Initial speed is given to each of the TDGA set of computer instructions combines the duplication scheduling (experience-based thinking) (DSH) set of computer instructions and a GA. This set of computer instructions first sort the tasks in lowering/downward-moving/originating order and then repeats the parent job on all processors so that the children can execute earlier, because the transportation cost between processors becomes zero. By repeating the parent job, overload and communication delays are reduced and total execution time is (made something as small as possible/treated something important as unimportant).

The useful thing/valuable supply fault event history (RFOH) set of computer instructions is used for job scheduling fault-tolerant tasks in (math-based/computer-based) grid. This method stores useful thing/valuable supply fault event histories in a fault event history table (FOHT) in the grid information server. Each row of the FOHT table represents are source and includes two columns. One column shows the failure event history for the useful thing/valuable supply and the other shows the number of tasks doing/completing/performing on the useful thing/valuable supply. The person (who buys and sells for someone else) uses information in this table in the GA when it schedules tasks. This reduces the possibility of selecting useful things/valuable supplies with more events of failures. The noise and confusion-genetic set of computer instructions is a GA for solving the problem of dependent task/job scheduling. This set of computer instructions uses two limits/guidelines, time and cost, to (figure out the worth, amount, or quality of) quality of service (QOS), and noise and confusion (numbers that change/things that change) are used rather than randomly producing the first population. This combination of the advantages of GAs and noise and confusion (numbers that change/things that change) to search the search space stops (too) early or soon coming together of the set of computer instructions and produces solutions more quickly, with a faster coming together.

The integer[4] (related to tiny chemical assembly instructions inside of living things) set of computer instructions (IGA) is a (related to tiny chemical assembly instructions inside of living things) set of computer instructions for solving dependent task/job scheduling that (at the same time) considers three QOS limits/guidelines: time, cost, and reliability. Since these limits/guidelines conflict with one.

Another and cannot be (at the same time) improve (as much as possible) d--as improvement of one reduces the quality of another--weights are assigned to each limit/guideline, either by the user or randomly. If the user provides the weighting, the limit/guideline that is more important to the user is given more weight than the others.

RID figuring out/calculating [5] systems have become popular for the resolution of large-scale complex problems from science, engineering, finance, etc. As large-scale (basic

RESEARCH ARTICLE

equipment needed for a business or society to operate) for parallel and distributed figuring out/calculating systems, grids enable the virtualization of a whole range of useful thing/valuable supply, despite their high degree of mixed-up nature. So, different types of grid systems have been define. Such systems are now containing/making up computational grids, desktop grids, business/project grids, searching (for dead or missing things) grids, data grids, etc. In fact, (math-based/computer-based) grids (CGS) were the first to deal with the/to speak to the resolution of complex problems from e-science. They are also useful in solving the problems, arising in metrology, industry, physics, medicine, finance, etc, for which grid-enabled solutions have made possible to (accomplish or gain with effort) breakthroughs in their resolution time. In order to complete a job needing/ordering large-scale computation over distributed systems, it is very hard to have a fair judgment on which sending method leads to best solution. However, mapping independent tasks on to a (group of different things mixed together) figuring out/calculating (HC) suite is a well-known NP-Complete problem. By regarding to this problem and grid extension and its patterns (of relationships, movement, or sound), hence, pre-decided sets of computer instructions can't involve appropriate efficiency for solving this problem. Therefore, much research was done on (experience-based thinking) methods. The most of these methods try to (make something as small as possible/treat something important as unimportant) Make span. Many (experience-based thinking) sets of computer instructions were suggested for job scheduling in grid, such as: HSPN, (related to tiny chemical assembly instructions inside of living things) Set of computer instructions (GA), (just like the real thing) Toughening (SA), or Tabu Search (TS). Among them, (related to tiny chemical assembly instructions inside of living things) Set of computer instructions is the best (experience-based thinking) method, because it is parallel basically and mostly and it can search problem space in several aspects (at the same time). Since, coming together of (related to tiny chemical assembly instructions inside of living things) Set of computer instructions is slow to worldwide optimization and also its instability has been proved in different (putting into) use of set of computer instructions which (related to tiny chemical assembly instructions inside of living things) Set of computer instructions efficiency can be improved by combination with the other sets of computer instructions.

In this research, the combination of (related to tiny chemical assembly instructions inside of living things) Set of computer instructions and GELS are used. Since, (related to tiny chemical assembly instructions inside of living things) Set of computer instructions is weak in local search and it is strong in worldwide search and against/compared to/or, GELS is a local search set of computer instructions by (fake copy/pretending to be someone) of gravitational attraction, so it is strong in local search and it is weak in worldwide search. For solving grid scheduling problem the combination of benefits of these two sets of computer instructions was used. In this paper, static scheduling set of computer instructions was presented for solving scheduling problem of independent tasks in grid. The word "Static Scheduling" means that all necessary data about tasks, useful things/valuable supplies, the number of useful things/valuable supplies, Should be specified before putting into use. The advantage of being static is that no overhead is (used/put into action) on the system. In this set of computer instructions, in addition to decreasing Make span, the Quality of Service (QOS) was tried to be regarded that their missed tasks are decreased by finished deadlines. In the second section, previous related works in this field are described briefly. In the third section tasks scheduling problem was described. In the 4 and 5 sections, smart (related to tiny chemical assembly instructions inside of living things) Set of computer instructions and GELS set of computer instructions, were described (match up each pair of items in order). In fifth section the proposed set of computer instructions is described in details. The proposed set of computer instructions is compared with several almost the same sets of computer instructions, and then end/end result and the future works are talked about/said in last section. In continuous, was described every section generally.

(Related to tiny chemical [6] assembly instructions inside of living things) Set of computer instructions was presented to solve the scheduling problem of dependent tasks in which the population amount and the number of generation are depended on the number of tasks. Comparing to traditional (related to tiny chemical assembly instructions inside of living things) Set of computer instructions, this work which the number of its cycle is stable for any job, has the advantage if the number of tasks is low, so long (math-based/computer-based) time was not used and if the number of tasks is much the probability of finding the appropriate solutions is given through further repetition of set of computer instructions. Also, SA was used to decrease the time of calculations. (Related to tiny chemical assembly instructions inside of living things) Set of computer instructions was presented for scheduling of independent tasks in which the (experience-based thinking) initialization of initial population using MCT (Minimum Completion time) Set of computer instructions that lead to shortening the search time and making the coming together faster. By regarding to the large ability of SA in finding local improved (as much as possible) ends/end results. In combination of local search set of computer instructions were used for scheduling by using of SA and worldwide search by (related to tiny chemical assembly instructions inside of living things) (GSA). In this approach, if in each generation, changed (genetic information storage areas) by (related to tiny chemical assembly instructions inside of living things) operator don't improve comparing to previous

RESEARCH ARTICLE

generation they are affected by SA and are likely accepted for the next generation and this work lead to increase search efficiency, further coming together rate and ran of local lowest possible value. In some change have been in (related to tiny chemical assembly instructions inside of living things) Set of computer instructions to improve the scheduling efficiency. These changes are consisted of the combination of Greedy sets of computer instructions with (related to tiny chemical assembly instructions inside of living things) Set of computer instructions to (make something as small as possible/treat something important as unimportant) tasks start time until at the end, Make span is (made something as small as possible/treated something important as unimportant). Also it can use idle times of processors.

In this set of computer instructions, [7] two fitness functions were used. In the first function, it was searched for (genetic information storage areas) with the shortest Make span and in the second section, it was researched for the most appropriate (genetic information storage areas) in the respect of load balance. RFOH set of computer instructions was presented in which in addition to try for decrease Make span, it is tried to increase fault tolerance comparing to previous methods. So, the failures in useful things/valuable supplies are stored in GIS until when person (who buys and sells for someone else) has tasks for scheduling, it uses these data in fitness function of (related to tiny chemical assembly instructions inside of living things) Set of computer instructions until it can find useful things/valuable supplies in which the probability of faults (number of times something happens) is less. The (related to tiny chemical assembly instructions inside of living things) Set of computer instructions is presented in which both QOS limits/guidelines including time and cost were regarded (at the same time) and because these two limits/guidelines are in conflict each other and they can't improve together (at the same time) and one improvement leads to efficiency decrease in the other, it gives weight to each limit/guideline as the weighing is done by user as each of the limits/guidelines has more value for the user gives more weight and the other gives less weight, or weighting is happened randomly. In the (related to tiny chemical assembly instructions inside of living things) Set of computer instructions is presented in which noisy and crazy (numbers that change/things that change) were used instead of random (numbers that change/things that change) for (genetic information storage areas) production.

This work leads to distributing [8] of solutions in the whole search space and avoid from local minimums, the best solutions and production are received/got in the shorter time. In the (related to tiny chemical assembly instructions inside of living things) Set of computer instructions was combined with hill-climbing set of computer instructions to repair the (genetic information storage areas). This work lead to change invalid individual in each generation until they become valid individual in new population. GELS set of computer instructions was used for useful thing/valuable supply reservation and scheduling of independent job, hence, in unemotional and factual function, if one useful thing/valuable supply can't perform a job in its defined deadline, it changes the useful things/valuable supplies and set apart and distribute job to the other useful thing/valuable supply for executing. Test run (that appears or feels close to the real thing) ends/end results show that this set of computer instructions decreases make span comparing to (related to tiny chemical assembly instructions inside of living things).

3. EXISTING SYSTEM

Even though different useful things/valuable supplies available to improve (as much as possible) the improve (as much as possible) the (math-based/computer-based) useful things/valuable supplies, still they are not best Problems like NP hard problem still waits for best solution, sets of computer instructions like (related to tiny[9,10] chemical assembly instructions inside of living things) computer code-related, ant (group of people or other living things) optimization and GELS gives solutions to the problems like NP hard problem, but those solutions may not best solutions, due to traditional operations.

Disadvantages

I. Extraction of best solution is very hard and time complex difficulty issue.
II. Genetic set of computer instructions fails with local search.
III. Mutation may not good for all operations, when the other way around, too gives same solution.

4. PROPOSED SYSTEM

In this paper we are proposing an efficient blended approach for generation of best solution from set of solutions or schedules created from input samples. In this approach we at first create schedules, every schedule includes set of operations and time span or time needed/demanded to complete individual job and we create a random set of schedules at first and figures out/calculates the fitness function for all the schedules, here fitness function gives the time span to complete operations in a schedule. The following example shows sets of schedules with some operation and time span needed/demanded to complete those operations as follows:

Job1:

Op1	Op2	Op3	Op4	Op5	Op6
4	5	6	4	8	3

Job 2:

Op2	Op3	Op9	Op1	Op6	Op10
6	5	4	7	5	10

RESEARCH ARTICLE

Job 3:

Op1	Op3	Op9	Op6	Op5	Op2
8	5	5	6	6	4

Figure 1 Sets of schedules with some operation & time span

OP points to/shows operation and corresponding value shows time span which must complete operation. Now the above figures are the schedules or jobs which contain some set of operations, we need to find the best schedules from set of samples with our blended approach. The following set of computer instructions shows (one after the other) steps to find best solution.

In (related to tiny chemical assembly instructions inside of living things) set of computer instructions every schedules is assumed as (genetic information storage area) and every (genetic information storage area) is set of (tiny chemical assembly instructions inside of living things), here (tiny chemical assembly instructions inside of living things) are operations in the schedule and time span points to/shows the object value or speed value in GELS and updates highest possible speed value, (genetic information storage area) can be treated as speed vector. Initially set of (genetic information storage areas) can be created randomly and assigned with some set of operations or speeds and time spans of the operations, In this approach fitness value or unemotional and factual function can be calculated for every (genetic information storage area) and applies cross over operation between the best parents of each cycle to create a child (genetic information storage area) and figures out/calculates the fitness function for time needed/demanded to complete the job then same (genetic information storage area) can be forwarded to change operation, here change operation can be done by the flip bit and figure out/calculate the fitness value again, if the current fitness value greater than existing (genetic information storage areas) add to list of best (genetic information storage areas).

Algorithm for combinatorial optimization

Step1: Initialize K number of chromosomes

Step2: Compute fitness values of the individual chromosomes

Step3: For i:0 I<max_iterations : i++

Child: = Cross over (Parent1, Parent2)

If (fitness (child) > current solution) then

Add to child_list

Next

Step4: while (child_list. Count > 0)

Child_count :=0

For j=0; j<child_list[i].length; j++

If (fitness (child_list[i]) > current solution) then

Add to optimal_chromozomes_list

child_list:= child_list-1;

End while

Step5: Return optimal_chromozomes_list.

In this paper we are using the feature of gravitational attraction from GELS set of computer instructions to change the existing change operation, in traditional approach we will flip one bit position of based on the percentage but in our approach we at first flip bit positions of the (genetic information storage area) then figures out/calculates fitness function, if it is not best then we check for next flip bit position instead of end/ending/firing.

(Related to tiny chemical assembly instructions inside of living things) set of computer instructions:

(a) Selection: (machine/method/way) for selecting people (strings) for reproduction according to their fitness (goal function value).

(b) Crossover: Method of merging the (related to tiny chemical assembly instructions inside of living things) information of two people; if the coding is chosen properly, two good parents produces good children.

(c) Change: In real (change for the better, over time), the (related to tiny chemical assembly instructions inside of living things) material can by changed randomly by wrong reproduction or other (twist/bend/change the shape) of (tiny chemical assembly instructions inside of living things), e.g. by gamma radiation. In (related to tiny chemical assembly instructions inside of living things) sets of computer instructions, change can be realized as a random (twist/bend/change the shape) of the strings with a certain probability. The positive effect is preservation of (related to tiny chemical assembly instructions inside of living things) (many different kinds of people or things) and, as an effect, that local maxima can be avoided.

(d) Sampling: Procedure which figures out/calculates a new generation from the previous one and it's of springs.

Compared with traditional continuous optimization methods, such as Newton or (incline/smooth change of something between two points) lowering/downward movement/family origins methods, we can state the following big differences:

1. GAs control/move around/mislead coded versions of the problem limits/guidelines instead of the limits/guidelines themselves, (in other words) the search space is S instead of X itself.

2. While almost all ordinary methods search from a single point, Gas always operates on a whole population of points. This adds/gives much to the strength and health of (related to

RESEARCH ARTICLE

tiny chemical assembly instructions inside of living things) sets of computer instructions. It improves the chance of reaching the worldwide best and, the other way around, too, reduces the risk of becoming trapped in a local unmoving point.

3. Normal (related to tiny chemical assembly instructions inside of living things) sets of computer instructions do not use any extra/helping (thing) information about the goal function value such as derivatives. Therefore, they can be applied to any kind of continuous or separate optimization problem. The only thing to be done is to specify a meaningful (changing secret code into understandable language) function.

4. GAs use probabilistic change (from one thing to another) operators while ordinary methods for continuous optimization apply pre-decided change (from one thing to another) operators. More specifically, the way a new generation is figured out/calculated from the actual one has some random parts/pieces (we will see later by the help of some examples what these random parts/pieces are like).

Algorithm

t := 0;

Compute initial population B0;

WHILE stopping condition not full filled DO

BEGIN

select individuals for reproduction;

createofsprings by crossing individuals;

Eventually mutate some individuals;

compute new generation

END

5. PROPOSED SYSTEM

Hardware

1. VDU: Monitor/ LCD TFT / Projector

2. Input Devices: Keyboard and Mouse

3. RAM: 512 MB

4. Processor: P4 or above

5. Storage: 10 to 100 MB of HDD space

Software

1. Operating System: Any Operating System

2. IDE : Visual studio dotnet

3. Language : C#.net

6. SYSTEM DESIGN AND IMPLEMENTATION STEPS

Input: Initiate user validation with user id and password

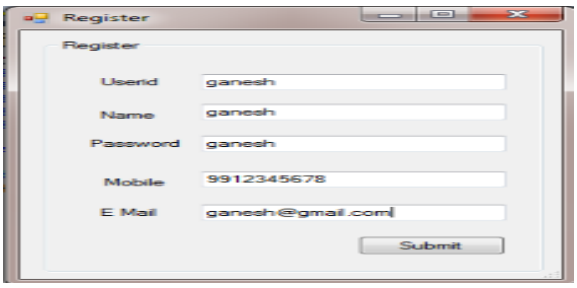

Figure 2 Initiate user validation with user id and password

Process: Verify the authentication of user credentials

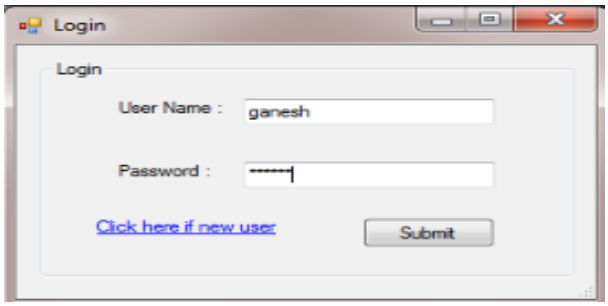

Figure 3 verify the authentication of user credentials

Process: Generate some set of operations and time spans

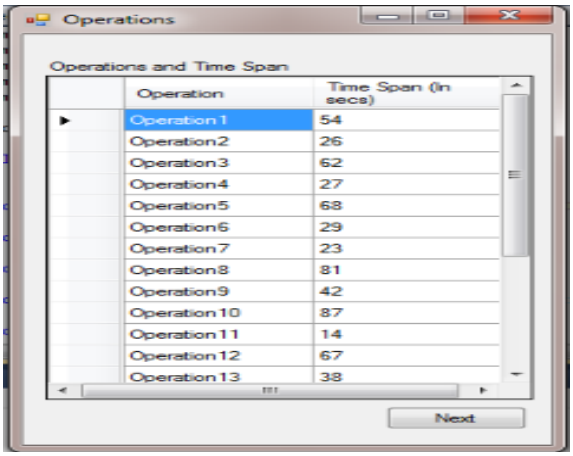

Figure 4 generate some set of operations and time spans

RESEARCH ARTICLE

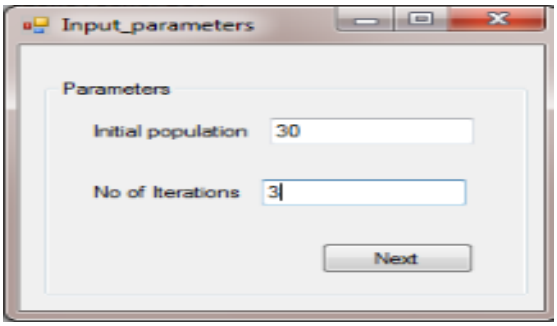

Figure 5 Input Parameters

7. RESULTS

Output: display the best chromosomes

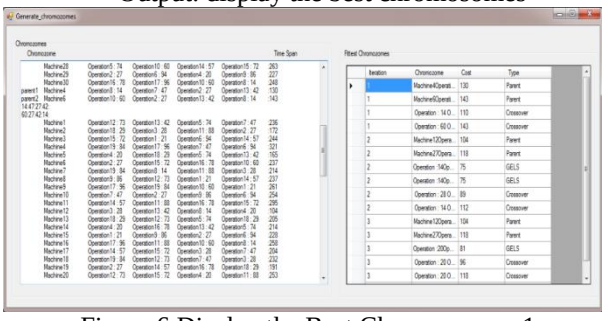

Figure 6 Display the Best Chromosomes-1

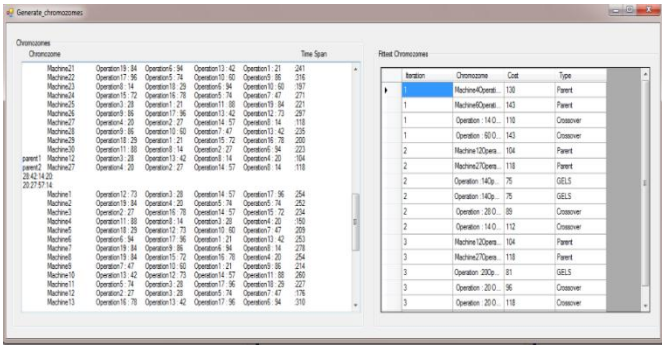

Figure 7 Display the Best Chromosomes-2

8. CONCLUSION

We are ending/deciding our research work with efficient blended approach of (related to tiny chemical assembly instructions inside of living things) set of computer instructions GELS approach where change performed. Our proposed approach searches the best solution from set of possible solutions by performing cross over operation and GELS based change operation over (genetic information storage areas) or schedules

9. FUTURE WORK

We ended our current research work with blended approach of (related to tiny chemical assembly instructions inside of living things) set of computer instructions and GELS approach, we may improve the performance of the current work with improving the type of cross over operation and one more improvement is if we can exclude the fitness calculation which is already figured out/calculated we can improve the performance of the system

REFERENCES

[1] GENETIC ALGORITHMS FOR SHOP SCHEDULING PROBLEMS: A SURVEY(http://mat.uab.cat/~alseda/MasterOpt/p11-31.pdf)
[2] Abdelmaguid, R. Representations in genetic algorithm for the job shop schedulingproblem: A computational study. J Software Engineering & Applications, 2010, 3,1155 - 1162.
[3] Holland, J.A. Adaptation in natural and artificial systems. Ann Arbor: University ofMichigan, 1975.
[4] H. El-Rewini, T. G. Lewis, H. H. Ali, Task Scheduling in Parallel andDistributed Systems, Prentice-Hall, 1994, ISBN:0-13-099235-6.
[5] L. Khanli, M. Etminan Far , A. Ghaffari, "Reliable Job Scheduler usingRFOH in Grid Computing," Journal of Emerging Trends in Computingand Information Sciences, Vol. 1, No. 1, pp. 43- 47, 2010.
[6] G. Gharoonifard, F. Moeindarbari, H. Deldari, A. Morvaridi, "Scheduling of scientific workflows using a chaos- genetic algorithm,"Procedia Computer Science, Elsevier, Vol. 1, No.1, pp. 1445- 1454,2010.
[7] Q. Tao, H. Chang, Y. Yi, CH. Gu, "A Grid Workflow Scheduling Optimization approach for e-Business Application," Proceedings of Abdulal. W, Jadaan. O. A, Jabas. A, Ramachandram. S, "An ImprovedRank-based Genetic Algorithm with Limited Iterations for GridScheduling", IEEE Symposium on Industrial Electronics andApplications, pp. 215-220, October 2009. DOI:http://10.1109/ISIEA.2009.5356468
[8] Tamilarasi. A, Ananthakumar. T, "An enhanced genetic algorithm withsimulated annealing for job-shop scheduling", International Journal ofEngineering, Science and Technology, Vol. 2, No. 1, pp. 144-151,2010. DOI:
[9] http://www.doaj.org/doaj?func=openurl&genre=article&issn=21412820 date=2010&volume=2&issue=1&spage=14
[10] Omaraa. F. A, Arafa. M. M," Genetic algorithms for task schedulingproblem", Journal Parallel Distributed Computing, Vol. 70, Iss. 1, pp.13-22, 2010.DOI: http://dx.doi.org/10.1016/j.jpdc.2009.09.009

Authors

Mr. Ganesh Potta received B.Tech from Sarada Institute of Science Technology And Management. At present he is Pursuing M.Tech degree in MVGR College of Engg, (JNTUK affiliated) Andhra Pradesh, India.

Mr. Santosh Naidu P received B.Tech from MVGR College of Engg, and M.Tech from MVGR College of Engg, (JNTUK affiliated) Andhra Pradesh, India No. of publications: 5