Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 8, No. 3, August 2017), Pages: 55-68
www.jacr.iausari.ac.ir

Scheduling of Real Time Processes Distribution on Multiprocessor Using Meta-Heuristic Ant Colony Algorithms, Genetic and PSO

Mostafa Soleymani[1], Hossein Nematzadeh[2]✉

1) *Department of Computer Engineering, University Collage of Rouzbahan, Sari, Iran*
2) Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran
soleymani@yahoo.com; nematzadeh@iausari.ac.ir

Received: 2016/08/01; Accepted: 2016/09/13

Abstract

Here we discuss the problem of distribution of Real time processes on multiprocessor with on-time maximum job accomplished. Scientists have been searching for producing optimized scheduling. This is an example of NP problems. This is not practical to approach this kind of problems with heuristic approach thus we must use meta-heuristic algorithms. These algorithms present many sets of answers in order to make options for scheduler, to choose the best process assignment to processor. Two examples are Branch and Bound, and Task Graph Algorithms. By studying the ant colony, Genetics and PSO Algorithms, we will design and consider several methods for our purpose and use them to produce Job assignment Scheduler, on processors. Each of these algorithms will provide us with a specific designing method and help us to make a scheduler engine of real time processes assignment on processors. We will compare each program to the first heuristic one, to assess the manufactured programs. In comparisons which are based on lost processes, Colony approach has 11.94 %, PSO approach 11.19 %, and Genetic approach has 7.52 % less process lost in compare to heuristic approach. It worth mention that 20 files each of which containing 50 Real time process have been used in these experiments.

Keywords: *Real Time; Scheduling; Branch and Bound; Task Graph*

1. Introduction

The vast development of alarming and predictive, aerospace research and medical systems and etc…, has created specific jobs that must been executed within a due time. Definition of problem is, presenting a method to implement process distribution scheduler on multi-processors by evolutionary algorithms. Maximum job done, minimum lost job and the highest efficiency is demanded by the problem.

A- The main purpose of this research is the study of :ultra-initiative approaches to design scheduler in order to distribute Real time processes on multi processors, requirements for execution of Real time processes and to provide execution of each process before expiry of due time.

B- Presenting scheduler engines for distribution of Real time processes on multi processors by evolutionary algorithms and development of scheduler software to

make this happen and finalizing the execution of Real time process is the milestones of our way.

C- Assessment of suggested methods and case study of ultra-initiative algorithms that used to design scheduler engines will be done by C++

The first meta-heuristic algorithm under research is Ant Colony. In this design, processors are sorted in the shortest path order. We use random pattern from genetic algorithm. We consider PSO algorithm as a systematic approach for designing. Specific calculation has been used in PSO for designing and due to this calculations, processors has been ranked in a descending order **[1], [2], [3], [4], [5], [6], [7], [8].**

2. Related Work

CAPS (Computing Analyzer and Process Simulator) Nelendra Badal and Hifzan Ahmad (2014) provided us with a powerful instrument to schedule processes. Choosing of processes from queue is done by FCFS rule. Comparative analysis shows that execution time of CAPS tool is less that all other existed tools. CAPS calculates waiting and working time. User could register simulation results in an excel file. **Bat Algorithm** Jein shen yang (2010) is a meta-heuristic algorithm which inspired categorization as a heuristic, from environment. It was inspired by pursuance of prey for hunting by a swarm of bats. In important problem like designing of gas turbines, optimal allocation of capacitor, and data clustering, K-means is used. BA, GA, PSO in compare to reasonbrux function (PSO=98%, GA=95%, BA=100%) and multiple peak (PSO=97%, GA=98%, BA=100%) has been yielded. Bat algorithm is a composition of swarm and path algorithms. **Proposed Fuzzy CPU Scheduling algorithm** (PFCS) Perina Ajmali Manoch cety (2013) is an algorithm for scheduling of real time processes in multi program operating systems on supercomputers which is designed on fuzzy logics. Fuzzy program first collects inputs then returns them in the fuzzy set which use fuzzy linguistic variables. In this algorithm, Mamdani inference system has been used. This algorithm has been mainly designed for optimization of turn around and waiting time. **Parallel particle swarm optimization** Gerhard Wenter, the other (2014), PPSO scheduler has been produced based on birds swarm by mass of particles optimization approach for processes scheduling. PPSO end of each stage update Parallel synchronous and updates parallel asynchronous when particles are available. PPSO algorithm use parallel particle mass optimization for maintaining a place as general convergence which is more costly than asynchronous model **[9], [10], [11], [12]**.

3. Method

To solve the problem of choosing the best processor to control the process to acquire eligible. Kant diagram are measured each processor with mathematical calculations that in The basic concepts have been and continue to work on the schedule four engine with basic concepts and meta-heuristic offered .

3.1 Fundamentals

The problem we are facing is the kind of programs whose processes superficially accidental and occurring in different times but in a specific limit, not so widespread, and in a logical base like processor, memory and operating system. Problem definition: Real

time processes distribution scheduler on multi processors should has: a.minimum lost process b.maximum executed process and C-maximum profit.

Operational parameters of problems:

1- Real time processes can be loaded into processor by to parameters of execution time and due time A. predefined real time processes B. Real time processes in a time unit.

2-We consider five processors with identical speed

3-each processor has a horizontal kant diagram which in calculations two parameters must be consider(total time of execution on processor and the biggest due time).

Int cpu [5] [2]; defines cpu parameters **cpu [] [0]**; biggest due time

cpu [][1]; total time of process execution

As in Figure 1 amount of α is calculated from kant diagram and used in scheduler program operations.

α=total execution time remaning of processes on cpu × biggest due time of process

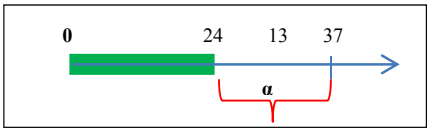


Figure 1: The Kant diagram of processor

This scheme is implemented based upon existed amounts of problem. First we show how to choose a process by considering kant diagram of the cpu in an example. Fara matrix keeps the processes. **Int Fara [50] [4];**

Ω=processing time of process − due time of process

Each incoming process must be compared to Ω parameter of kant diagram of each processor and if no violation of existed rules of kant diagram is seen, the permission of process assignment will be granted. Specification of several real time processes has been presented in diagram1 below and the way of assignment to processors by getting legal permission will be shown, see Table 1.

Table 1: many sample form real time process

ID proc	Rel. time	Proc. ime	Due time	Benefit
1	0	3	9	11
2	0	1	5	8
3	1	9	19	11
4	5	5	16	2

The first, second and third chosen process is seamlessly put on kant diagram, but with forth process coming according to this method there is no room for putting the process on cpu. The forth process is considered as lost which is shown in Figure 2.

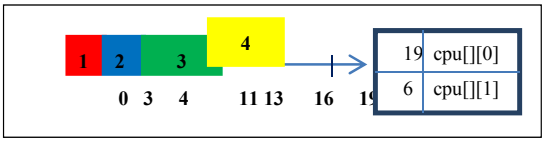


Figure 2: the cpu assignment of processes on cpu upon heuristic algorithm

The choosing method of Cpu's will be done as start to end of linear array of CPU numbers, at the beginning the first element of linear array of Cpu numbers is calculated by the required process and if approved, process is delivered to Cpu, otherwise we will proceed to the next element in the array, of course to the number of CPU's available to the program.

3.2 SAFO Heuristic Algorithm

First the main pattern has been described by showing the flowchart in figure 3.

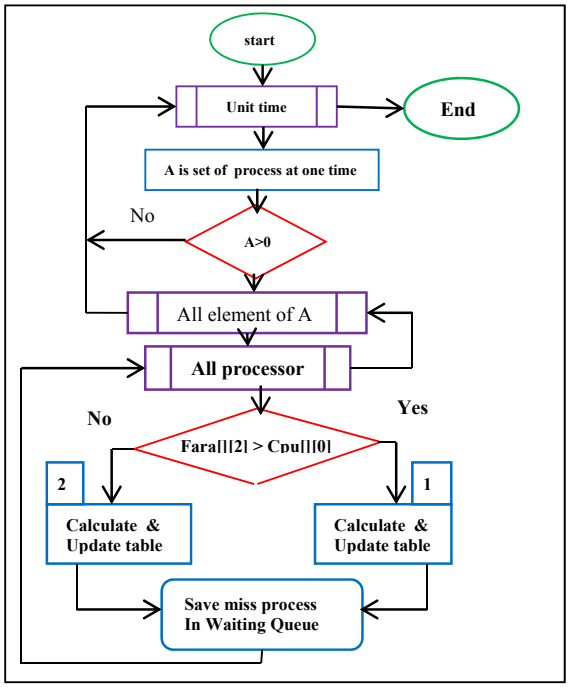


Figure 3: SAFO heuristic algorithm flowchart

And the method has been elaborated as below:

1) **WHILE internal time OR END**
2) **COLLECT whole inter process at one time in A set**
3) **IF not A>0 THEN goto 2**
4) **ELSE WHILE all A member and end all A goto 1**
5) **WHILE all processors and the end of time goto 4**
6) **IF Fara[][2] > cpu[][0] THEN goto part 1 ELSE goto part 2**
7) **however IF it doesn't accept by each processor put it on Waiting queue**
8) **And goto 5**

This scheduler engine will be used as a benchmark for assessing of other designing method and its designing structure as main core of distribution of processes on processors, in other designs.

3.3 Algorithm SAFO with Colony

By studying of designing method of Ant Colony which has been inspired by colporteur, the most important function that in related to the spirit of scheduler program can be used.

Description of colony method:

We define a two dimension array of integers containing the number of processors and processes existed in each processor:

Int Scup [N] [2]

Scup [N] [0] = number of processes

Scpu [N] [1] = number of processor

And for each process assigned to a processor, one unit is added to the variable Scpu of that processor and is sorted before each stage to sort the processors in ascending order.

First the main pattern will be describe by flowchart showed in Figure4:

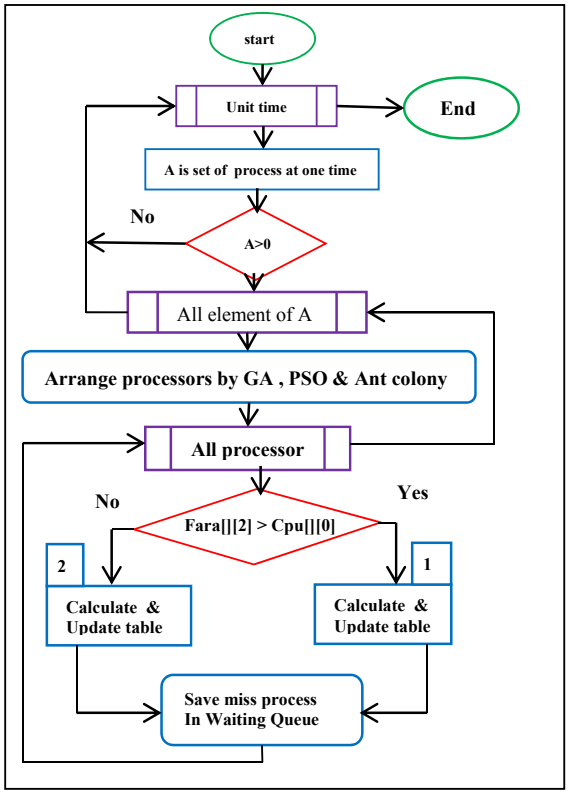


Figure 4: heuristic algorithm flowchart by Ant colony, Genetics and PSO methods

And the method has been elaborated as below:

1) **WHILE internal time OR END**
2) **COLLECT whole inter process at one time in A set**
3) **IF not A>0 THEN goto 2**
4) **ELSE WHILE all A member and end all A goto 1**
5) **SET processor by Ant Colony design**
6) **WHILE all processors and the end of time goto 4**
7) **IF Fara[][2] > cpu[][0] THEN goto part 1 ELSE goto part 2**
8) **however IF it doesn't accept by each processor put it on Waiting queue And goto 5**

By comparison of the obtained program with the probative program, we conclude that the results indicate the reduction of lost processes which is presented in results section and also we have achieved an excellent result on the load adjustment on the processors.

3.4 SAFO Algorithm with Genetics

In previous algorithms, selection of Processors was done in sequential order. Thus we looked into the outcomes accordingly. By genetic method we are to choose the processors randomly, and by Genetic method we offer several models of processors placement, then we will try to choose a processor. The proposed models implied in the calculation process, if the processor won't be attributed to a process in the calculation, we will go to the next process with the other proposed models. Otherwise the process is delivered to the waiting queue.

Genetic method has been defined and shaped by the following parameters:

Value: Coding is quantitative

Crossover: by single point method cut, with 0.5 probability.

CPU: the number of processors has been determined as 5.

Selection: Roulette method is used. In roulette we use Random (N) function to simulate falling of ball on a certain winner number. Then we sort out the vertical linear array so that the winner number locates at the middle of the array. To achieve the goal, the vertical linear array must be right shifted several times. For example if the random number is 2, by four times right shift of vertical linear array, the winner number 2 will be located at the middle of the vertical linear array, see Figure 5.

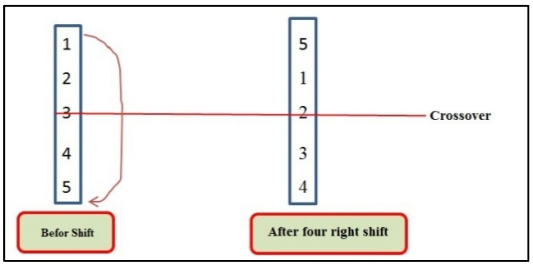


Figure 5: execution of roulette method in Genetic method in processor placement

Mutation: with 1/N transposition (N is the number of processors. We will work with 5 processors)

We have been able to obtain the model of CPU number placement according to the previous stages. Now it is Genetic Mutation turn. Due to the type of designing, cutting diagram is located on the middle element of linear array (because of odd number 5) Case one: substitution of the first and last element of array, see Figure 6.

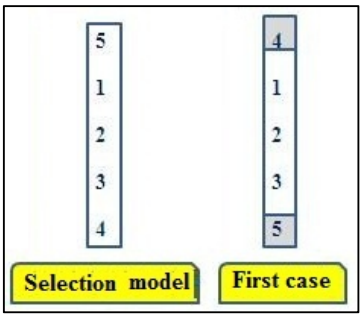


Figure 6: Using Genetic mutation in processor arrangement by single diagram method

The designing method considered for this scheduler, is as following:

Genetic method is more efficient in resolving the problems in which the sample space is very big. In this problem the number of processors is little and probability percent's in cut and direction are a little different from genetic method. This method is not so efficient for our problem after all.

First the main pattern will be described by flowchart shown in Figure4.

And method has been elaborated as follows:

1) **WHILE internal time OR END**
2) **COLLECT whole inter process at one time in A set**
3) **IF not A>0 THEN goto 2**
4) **ELSE WHILE all A member and end all A goto 1**
5) **SET processor by GA design**
6) **WHILE all processors and the end of time goto 4**
7) **IF Fara[][2] > cpu[][0] THEN goto part 1 ELSE goto part 2**
8) **however IF it doesn't accept by each processor put it on Waiting queue**
9) **And goto 5**

For assessing of the produced program, we compare the outputs of Genetic scheduler with according outcomes of Heuristic scheduler to find out how accurate they are. It's too hard to see any significant improvement in outputs. Because

Each time, with random selections we will see different presentation of processes distribution on processor, so this is hard to decide.

But the results show improvements in reducing of lost processes, but it's not significant.

3.5 SAFO Algorithm with PSO

This algorithm are based on more accurate calculation on each processor and comparison of them with each other. We follow the same sequential approach of heuristic scheduler, although by fundamental calculations with particle mass optimization method which we will describe in the following.

This time we will consider the nature of the problem which is of lawful one.

PSO method are defined and shaped upon the following parameters:

There are two parameter as the fundamentals of the PSO method. Generating particles' positions and velocities. X_k^i Position of i'th particle at time k and velocity V_k^i of particle i at time k which are produced. The numeral value of β obtained from process specification is used in PSO calculation.

Total time of processes executed in the processor × the Largest due time of processor = β

Position of particle V_k^i=is equal to the beta number of each processor, see fourmula 1.

$$\boldsymbol{X_k^i = X_{min} + \text{Rand } (X_{max} - X_{min})} \quad \textbf{(1)}$$

Velocity of particle X_k^i = we obtain the velocity of particle by position of particle, see fourmula 2.

$$\boldsymbol{V_k^i = \frac{\text{Rand}(X_{max} - X_{min})}{\Delta t}} \quad \textbf{(2)}$$

P^i =the best position of particle in current movement equal to the β number is obtained from incoming process with each processor and placed in **Scup [] [0].**

P_k^g= position of particle with the best global assessment in k current movements which after each stage is substituted with P^i and at first in variable **cpu [] [2]** with value 0.

In order to obtain of the next status of position and velocity of particles (updating) we will use the following formulas and define three parameters W, C1, C2:

$\mathbf{W} = \cup(\mathbf{0.4}, \mathbf{1.8})$ $\mathbf{C1} = \cup(\mathbf{1.5}, \mathbf{2.5})$ $\mathbf{C2} = \cup(\mathbf{1.5}, \mathbf{2.5})$

These parameters in run time of each **PSO** scheduler are obtained by reduction or increasing of the value of parameters. The formulas of position and velocity of particles is dexcribed as follows, see fourmula's 3 and 4.

(3)

$$\boldsymbol{V_{k+1}^i = (V_k^i \times W) + C1 \times Rand\,()\,(Scpu[\,][0] - X_k^i) + C2 \times Rand\,()\,(cpu[\,][2] - X_k^i)}$$

$$\boldsymbol{X_k^i = X_k^i + V_{k+1}^i} \quad \textbf{(4)}$$

The value of X_k^i is placed in array P^i, then array will be sort in descending order and this structure use for placement of the most efficient CPU. After finishing of each incoming process the value of P^i and P_k^g is substituted and used in the next execution.The designingmethod of this scheduler is based upon selection of processor with the ability of receiving next process(larger β number).position of particle(X_k^i) is calculated from β value with each of processors and obtain the velocity of particle V_k^i, then update the values of X_{k+1}^i and V_{k+1}^i.in this method we may be able to distribute on several processors, thus by sorting **Scpu** array in descending order, processor with larger particle velocity, wins the getting of process. Upon the existed data of the problem and applying **PSO** method, this design is implemented as follows, the concept of the method has been described below:

First the main pattern will be described by flowchart shown in Figure4.

And method has been elaborated as follows:

1) **WHILE internal time OR END**
2) **COLLECT whole inter process at one time in A set**
3) **IF not A>0 THEN goto 2**
4) **ELSE WHILE all A member and end all A goto 1**
5) **SET processor by PSO design**
6) **WHILE all processors and the end of time goto 4**

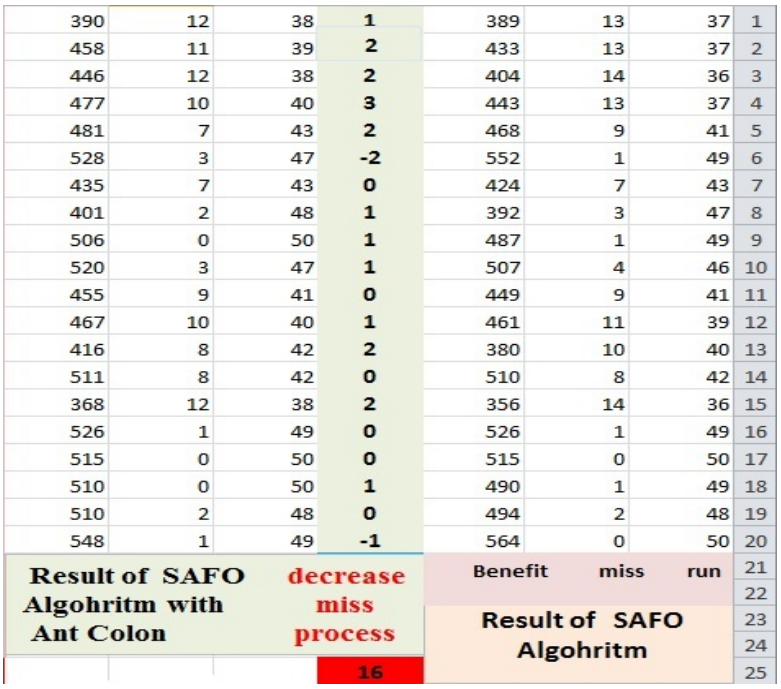
7) **IF Fara[][2] > cpu[][0] THEN goto part 1 ELSE goto part 2**
8) **however IF it doesn't accept by each processor put it on Waiting queue**
9) **And goto 5**

For assessment of produced program by PSO method, the outputs of PSO scheduler is compared to the similar outputs of heuristic scheduler to test the accuracy of them.in this scheduler we see the reduction of lost processes. because this problem is completely lawful and this kind of algorithms can provide us with proper solution for presenting an good scheduler.in this method, processors with the maximum space of performance on processor has been calculated, thus after each distribution of processes on each processor we have a processor with maximum space of performance.in comparison with other methods, this method is more efficient than genetic and heuristic algorithm.

4. Results

We introduce four algorithm of scheduler of processes distribution on multi processors each of which has been experimented with similar samples. Samples are files containing 50 Real time processes that each of them are different, regarding type of processes. Twenty files are divided into 4 category each of which shows different behavior from others. This affects the ability of produced schedulers and is a factor in outputs of them. A good scheduler engine

must has the ability of responding to any real time process in its life time, in other words must has the power for responding to 97 % of any Real time processes.one of the most successful schedulers in this respect is schedulers which are designed by branch and boundary algorithm. The reason for power of this kind of schedulers, is correct selection of processor among other processors from covering of all proper routs. We are confronting three kind of processes: matching processes, bad matching processes and miss matching. Matching processes are implied seamlessly to the processor. Bad matching processes that we try to somehow cover them by meta-heuristic algorithm and miss matching processes that we must do a clever designing for them.



390	12	38	1	389	13	37	1
458	11	39	2	433	13	37	2
446	12	38	2	404	14	36	3
477	10	40	3	443	13	37	4
481	7	43	2	468	9	41	5
528	3	47	-2	552	1	49	6
435	7	43	0	424	7	43	7
401	2	48	1	392	3	47	8
506	0	50	1	487	1	49	9
520	3	47	1	507	4	46	10
455	9	41	0	449	9	41	11
467	10	40	1	461	11	39	12
416	8	42	2	380	10	40	13
511	8	42	0	510	8	42	14
368	12	38	2	356	14	36	15
526	1	49	0	526	1	49	16
515	0	50	0	515	0	50	17
510	0	50	1	490	1	49	18
510	2	48	0	494	2	48	19
548	1	49	-1	564	0	50	20
Result of SAFO Algohritm with Ant Colon			decrease miss process	Benefit	miss	run	21
							22
				Result of SAFO Algohritm			23
							24
			16				25

Figure 7: comparison of lost processes of Ant colony algorithm with heuristic algorithm

Results of heuristic algorithm experiment with 20 samples of 50 real time processes in the right side of fig 7 are compared to results of similar experiments done by Ant colony algorithm in left side of fig 7.results of these experiments shows reduction in lost processes by ant colony algorithm that are approximately 11.94 % meaning 16 lost process. In this scheduler we use shortest-path which can make the heuristic algorithm more powerful. Except for optimization of reduction of lost processes, it successfully regulate loading which is an obvious parameter of power of this scheduler. The reason for ability of colony algorithm is overlapping of its designing with branch and bound algorithm in selection of processors.in branch and bound algorithm, every way of selecting a processor is looked into, so every other program that has the same behavior will be definitely powerful. heuristic algorithm by colony method has had the ability to operationally cover the match and bad matching processes but it is not so efficient in miss matching processes, see Figure 7.

419	11	39	2	389	13	37	1
433	12	38	1	433	13	37	2
413	14	36	0	404	14	36	3
468	12	38	1	443	13	37	4
439	11	39	-2	468	9	41	5
541	3	47	-2	552	1	49	6
438	7	43	0	424	7	43	7
401	2	48	1	392	3	47	8
492	1	49	0	487	1	49	9
526	2	48	2	507	4	46	10
432	11	39	-2	449	9	41	11
456	11	39	0	461	11	39	12
379	10	40	0	380	10	40	13
524	8	42	0	510	8	42	14
354	13	37	1	356	14	36	15
540	0	50	1	526	1	49	16
497	2	48	-2	515	0	50	17
490	1	49	0	490	1	49	18
511	2	48	0	494	2	48	19
564	0	50	0	564	0	50	20
Result of SAFO Algohritm with GA			**Decrease Miss Process**	Benefit	miss	run	21
							22
				Result of SAFO Algohritm			23
							24
			1				25

Figure 8: comparison between lost processes of genetic algorithm with heuristic algorithm

Results of heuristic algorithm experiment with 20 samples of 50 real time processes in the right side of fig 8 are compared to results of similar experiments done by ant genetic algorithm in left side of Figure 8.results of these experiments doesn't show significant reduction in lost processes by Genetic algorithm that is only 7.52 % reduction, meaning 1 lost process can be seen. The reason for weakness of this designing is selecting of processor by a random method which conflicts with nature of this lawful schedulers.in each stage that a processor is selected, incoming process is assigned to it, but a better qualified processor might be available. For this reason, the scheduler engine with genetic method, couldn't meet our demands.in fully lawful designs there is no room for selecting of random methods.

423	10	40	3	389	13	37	1
449	11	39	2	433	13	37	2
414	12	38	2	404	14	36	3
472	11	39	2	443	13	37	4
441	9	41	0	468	9	41	5
506	4	46	-3	552	1	49	6
440	7	43	0	424	7	43	7
393	3	47	0	392	3	47	8
506	0	50	1	487	1	49	9
526	3	47	1	507	4	46	10
463	9	41	0	449	9	41	11
506	8	42	3	461	11	39	12
397	9	41	1	380	10	40	13
520	8	42	0	510	8	42	14
385	11	39	3	356	14	36	15
540	0	50	1	526	1	49	16
512	1	49	-1	515	0	50	17
510	0	50	1	490	1	49	18
486	3	47	-1	494	2	48	19
564	0	50	0	564	0	50	20
Result of SAFO Algohritm with PSO			Decrease Miss Process	Benefit	miss	run	21
							22
				Result of SAFO Algohritm			23
							24
			15				25

Figure 9: comparison of lost processes of PSO algorithm with heuristic algorithm

Results of heuristic algorithm experiment with 20 samples of 50 real time processes in the right side of figure 9 are compared to results of similar experiments done by PSO algorithm in left side of figure 9.results of these experiments shows reduction in lost processes by ant colony algorithm that are approximately 11.19 % meaning 15 lost process. In this scheduler, selection of processors is based on specific calculation of PSO algorithm. this calculations are estimated due to the amount of free space in each processor which is shown in fig1.Thus the selection of processors are based on which is the most qualified one.in this scheduler matching processes easily are assigned to processors and because selection of processors are done by accurate calculations, miss matching processes. This scheduler has the same efficiency as colony algorithm, except for bad matching processes that it isn't as much efficient. There is only one thing about PSO algorithm: w, c1 and c2 parameters which must be adjust for every file containing Real time processes to achieve better results. Based on accurate calculations of values, PSO algorithm is so successful in the experiments left side of figure 9.

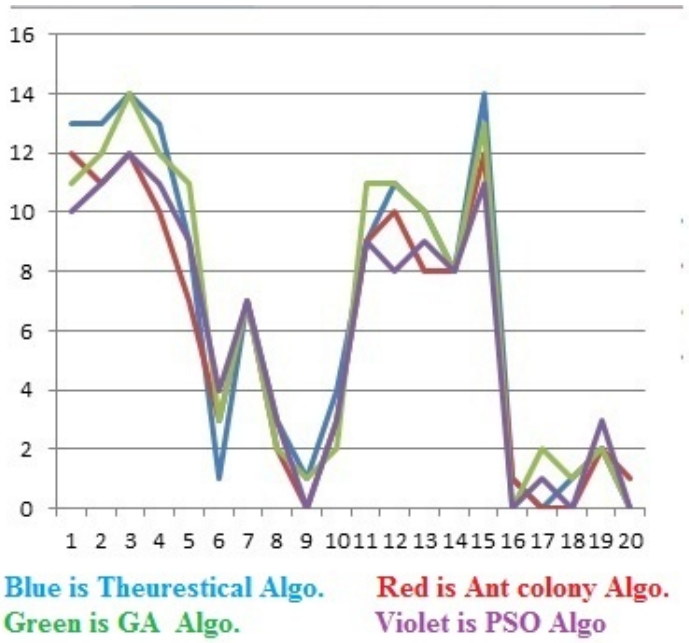


Figure 10: schematic comparison of four heuristic algorithm with ant colony, genetic and PSO Alg.

In Figure 10, 1- heuristic and colony algorithm,
2- Heuristic and pso algorithm,
3- Heuristic and genetic algorithm and,
4-heuristic algorithm,

Has been ranked consecutively based on reduction of lost processes. In the beginning we mentioned 20 files containing 50 Real time processes that has been sorted in four category. These four categories has been separated in Fig10.In files 1 to 5 and 11 to 15 ,number of bad matching and mismatching processes is high which seriously challenged

Our algorithm. But there are less bad matching processes, so there be seen lower statistics of lost processes especially in files 16 to 20 in which we face the minimum lost processes.

Experiments shows the power of designing by Meta heuristic algorithm, which is because of choosing of best processor from several processors.

5. Conclusion

During production period we were working with two main categories. The first one was processors which are lawful and second one was Real time processes which are being assessed in two class. The first class is predefined processes which can be easily work with, and the second is the Real time processes which are implied into the system without any rule and is the subject of this essay. During this operation we come to the general conclusion that we can overcome various Real time processes and we can achieve better response in less time (and of course with less power) in compare to branch and bound, and task graph algorithms. Our next goal is to overcome bad matching Real time processes by applying clever methods together with these produced schedulers.

References

[1] Marco Dorigo, Thomas Stu¨ tzle.(2004) " Ant colony optimization " p. cm. ‘‘A Bradford book.’’ Includes bibliographical references (p.). ISBN 0-262-04219- (alk. paper)

[2] Applegate, D., Cook, W., & Rohe, A. (2003). Chained Lin-Kernighan for large traveling salesman problems. INFORMS Journal on Computing, 15(1), 82–92.

[3] Mitchell, Melanie.(1999) "An introduction to genetic algorithms "p. cm. "A Bradford book." Includes bibliographical references and index.ISBN 0−262−13316−4 (HB), 0−262−63185−7 (PB)

[4] Ackley, D., and Littman, M. 1994. A case for Lamarckian evolution. In C. G. Langton, ed., *Artificial Life III,*Addison−Wesley.

[5] International Journal of Computer Applications (0975 – 8887) Volume 51– No.22, August 2012.

[6] Kennedy.J,Eberhart.R.C. "A discrete binary version of the particle swarm algorithm“, *Systems, Man, and Cybernetics, 1997.* Computational Cyberneties *and Simulation., 1997 IEEE International Conference on* ,vol.5, no., pp.4104,4108 vol.5, 12-15 Oct 1997.

[7] Wang. K,HUANG. L,ZHOU. C.G,"PARTICLE SWARM OPTIMIZATION FOR TRAVELING SALESMAN PROBLEM", Proceedings of the Second International Conference on Machine Learning and Cybernetics, 2-5 November 2003.

[8] Lin. T.L, Horng. Sh.J, Kao T.W, Chen. Y.H, Run. R.Sh, Chen R.J, Lai. J.L, "An efficient job-shop scheduling algorithm based on particle swarm ptimization",Expert Systems with Applications 37 (2010) 2629–2636.

[9] V. Singh, T. Gabba (2013), Comparative study of processes scheduling algorithms using simulator, *Int'l Journal of Computing and Business* algorithms *Research (IJCBR)*, vol. 4

[10] H.S. Kim, B. Malakooti, Bat intelligent hunting optimization application to ultiprocessor scheduling, Ph.D. Thesis,Department of Electrical Engineering and Computer Science, Case Western Reserve University, 2010.

[11] H. S. Behera, Ratikanta Pattanayak and Priyabrata Mallick, "An Improved Fuzzy- based CPU Scheduling(IFCS) Algorithm for Real Time System ",International Journal of Soft Computing and Engineering (IJSCE), Volume-2,Issue-1, March 2012.

[12] M. England, and Balac, "Flocking Behavior Rules," 2014.