International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

Multiple Processor Scheduling with Optimum Execution Time and Processor Utilization Based on the SOSA

Sasmita Kumari Nayak, Chandra Sekhar Panda

Abstract: ***The multiple processor scheduling problem characterizes that different processor comprises of an arrangement of jobs or tasks designate proficient utilizing a limited number of processors. Herein development a multi-objective algorithm utilizing Symbiotic Organisms Search algorithm (SOSA) for scheduling an arrangement of reliant on tasks on obtainable resources in a multiple processor environment which minimizes the execution time and maximize the processor utilization. SOSA is a nature-inspired meta-heuristic algorithm utilized to compare with other meta-heuristic algorithms such as Water cycle algorithm (WCA), Genetic algorithm based Bacteria foraging optimization (GBF), Bacteria Foraging Optimization (BFO) and Genetic Algorithm (GA). SOSA reproduces the advantageous association methodologies received by life forms to survive and engender in the biological-community (ecosystem). Based on experimental results, we find the execution time as well as processor utilization using SOSA technique and then compare with the other mentioned algorithms. Acquired outcomes affirm the incredible execution of the SOSA in solving the multiple processor scheduling problems.***

Index Terms**: *Meta-heuristic Algorithm, Multiple processor scheduling, Optimization Problem, Symbiotic Organisms Search (SOSA).***

I. INTRODUCTION

In everyday life, the vast majority of the issues are elucidated with optimization. Normally, to achieve the best solution, optimization is a mathematical problem. Meta-heuristics are established continuously to upgrade either a population of solutions (for instance particle swarm optimization, genetic algorithms, Evolutionary algorithms, Swarm based algorithms) or a single solution (e.g. simulated annealing, Tabu Search) and for the most part utilize randomization and local search to elucidate an assumed optimization problem [1].

Multiple processors scheduling, is an NP-hard problem [2], a foundation of stimulating issues for specialists within the part of computer engineering. Different ways to deal with the multiple processor scheduling problem have been planned. In this article, a proficient technique is used derived from SOSA to elucidate the multiple processor scheduling problems with the collation of several traditional techniques.

SOSA has as of late presented with much consideration as metaheuristic algorithms for several optimization problems over the traditional metaheuristic methods. Traditional metaheuristic methods for instance Differential Evolution (DE) [3], Particle Swarm Optimization (PSO) [4], Bat Algorithm (BA) [5], Cuckoo Search (CS) Algorithm [6], Firefly Algorithm (FA) [7] - [10] gives the result local optima.

This article make known to another straightforward and effective metaheuristic algorithm named as Symbiotic Organisms Search (SOSA) which easily optimize the multiple processor scheduling. The organisms stay alive in ecosystem with this method which pretends the collaboration of symbiotic approaches [11].

The multiple processor scheduling problem using particle swarm optimization (PSO) is influenced in birds flocking action. The features of PSO algorithm are firsthand converting plan, an operation of finest velocity equation and community interrelationship among numerous variants, and an actual integration of local search [12]. However, in this article the multiple processor scheduling using SOSA algorithms is an active assimilation of global optimum search.

The multiple processor scheduling problem by using Ant colony system (ACS) algorithm, a heuristic is logical and operative algorithm well grows a near-optimal approach which will positively encourage practitioners to apply it to genuine problems [13]. The validation of SOSA has done in computational tryouts are directed to create a correlation through four prevailing algorithms i.e. Genetic algorithms (GAs), Bacteria Foraging Optimization (BFO), Genetic algorithm based Bacteria foraging optimization (GBF, a hybrid algorithm) [2] and a Water cycle algorithm (WCA) [15] in light of the same multiple processor scheduling problems. This article discerns the brunt of processor limit and task set on running time of calculation during simulation. Results prove the improvement over the above four mentioned methods. The remainder of this article goes on the following manner. Background of problem statement described in Sect - 2 and the concise presentation and working of SOSA illustrates in Sect-3 and Sect-4. Results and discussion i.e. simulation is depicted in Sect-5. Conclusion and future work is managed in Sect-6 and Sect-7.

Revised Manuscript Received on 30 July 2019.
***** Correspondence Author
Sasmita Kumari Nayak*, Computer Science and Engineering, Centurion University of Technology and management, Bhubaneswar, Odisha, India.
Chandra Sekhar Panda, Computer Science and Application department, Sambalpur University, Odisha, India.

II. MULTIPLE PROCESSOR SCHEDULING PROBLEM STATEMENTS

The goal of the multiple processor scheduling problem is minimization of total execution time. Here, total execution time can be called as total performance time also. This article ponders the distribution of task to the diverse processor with the related circumstances.

The system encompasses a schedule of tasks (D) and processors (A) having distinctive memory and resources accomplished on diverse processor experiences distinctive running time [2]. The connecting points are thought to be identical; however the cost of connecting points amongst two tasks will be accomplished once executed constantly with diverse processors. A task will make usage of resources from its execution processor. The purpose is attaining the minimum total performance time by assignment of tasks. This article discusses the projected dynamic multiple processor scheduling using SOSA. Table I exhibits an evocative example contains 7 tasks and 4 processors [2]. Every row and column represents the processors and the tasks correspondingly. From table, the cell [A2, D4] =1 proposes that task D4 is allocated to processor A2 specifically 1 and [A4, D3] =0 recommends task D3 is not consigned into processor A4 specifically 0.

The above problem is simulated by using the recent metaheuristic global optimal algorithm, so called SOSA. The detail of the SOSA is discussed in next section. Here, the objective function uses to compute total performance time. Average of the total performance time of set of tasks owed to the processors is computed by fitness function. $fitt\_fun_i$ is a fitness function of A_i processor, calculates the value of the allocated task by using (1) [2].

$$fitt\_fun_i = (1/makespan) \times \max(utilization) \quad (1)$$

The average utilization is computed found on particular execution of the processor. The utilization of the individual processor is given by (2) [2],

$$utilization(A_i) = Finish\_time(A_i)/makespan \quad (2)$$

The average processor utilization is assessed by the division of total processors utilization and total no. of processors namely n. Once the average processor utilization is enhanced, evade the processors being unused for quite a while. $Objctv\_fun$ may be computed with the help of (3) which finds average of total performance time of the tasks relegated with processors [2].

$$Objctv\_fun = \min\left\{\frac{\sum_{i=1}^{n} fitt\_fun_i}{n}\right\} \quad (3)$$

Table I: An Initial Ecosystem representation of task assignment using SOSA

	D1	D2	D3	D4	D5	D6	D7
A1	1	1	0	1	0	0	1
A2	0	0	0	1	1	0	1
A 3	0	0	0	0	1	1	0
A 4	1	0	0	1	1	1	0

The goal is to reduce the $Objctv\_fun$ referenced in (3). The outcome clearly indicates the optimum schedule along with the balance in the processor utilization.

III. THE SYMBIOSIS ORGANISMS SEARCH ALGORITHM (SOSA)

The natural phenomena are mimicked by all current metaheuristic algorithms. SOSA algorithm, is one among of metaheuristic algorithms, reproduces associations between any two organisms in the biological nature. Living beings seldom live in disconnection because of dependence on different species for sustenance and even subsistence [11]. In SOSA, the symbiosis is the close relationship between at least two living organisms, where no less than one gets a type of advantage from the relationship. For example, in the set of all animals incorporate certain types of flying creatures who eat parasites or bugs off the skin of elephants, and a honey bee which eats the nectar of blossoms and thus spreads the dust which enables the bloom to replicate. For the most part, in particular symbiotic relationships are 3 kinds, mutualism, commensalism, and parasitism. Symbiotic relationships of any group of organisms in ecosystem are shown in Fig. 1.

As a rule, organisms create symbiotic relationships as a procedure to adjust to deviations in their situation. Symbiotic relationships may likewise enable living beings to build wellness and survival advantage over the long haul. In this manner, it is sensible to reason that symbiosis has made and keeps on forming and support every single present-day biological nature inspired system. The more in detail of SOSA is discussed next.

IV. MULTIPLE PROCESSOR SCHEDULING USING SOSA

The character of the communication characterizes the primary rule of each stage. The outline of SOSA [11] in dynamic multiple processor scheduling is:

Step 1: Initialize the required factors.
Step 2: Repeat the procedure until termination criteria met.
Do the following three phases:
i) Mutualism phase
ii) Commensalism phase
iii) Parasitism phase
End

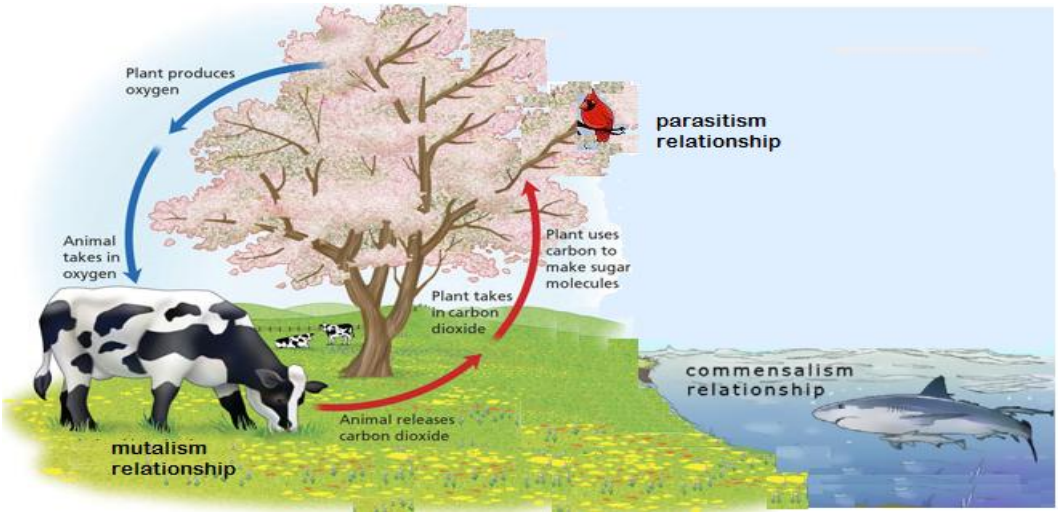

Fig. 1: Symbiotic organisms live respectively in an ecosystem

The SOSA suggested by [11], utilized to solve the multiple processor scheduling problem. In this method, population size is considered as the number of task allocated to the processor which is found from the given problem such as E represents no. of organisms in the ecosystem. After that, find the best organism Eco_{best} from the ecosystem. Then, an association process named as updation of organism which is applied to every organism. If a better organism is found then Eco_{best} is updated [14]. Fig. 2 describes complete SOSA procedures with three stages are examined here.

A. Mutualism Phase

Mutualism determines a relationship between two organisms where both benefit. For example, consider the cattle (domestic) animal and plant. This one is verifiable truth, those plants and cattle animal could not exist deprived of every one other. This association depends on the way that the cattle animal creature uses the oxygen that plants emit and breathe out carbon dioxide. Plants use the carbon dioxide to create oxygen for cattle animal.

In SOSA, Eco_i is an organism related with i^{th} fellow of ecosystem. Further, organism Eco_j is selected arbitrarily from the ecosystem to communicate with Eco_i. The two living beings' creatures partake in a mutualistic relationship with the goal of extending shared survival advantage in the biological system. Equation (4) and (5) are used to compute the novel solutions for Eco_i and Eco_j appeared as [11]:

$$EcoNew_i = Eco_i + R * Eco_{best} - MV * BF_1 \quad (4)$$

$$EcoNew_j = Eco_j + R * Eco_{best} - MV * BF_2 \quad (5)$$

$$MV = mean\left(\frac{Eco_i + Eco_j}{2}\right) \quad (6)$$

$$R = rand(1, E) \quad (7)$$

$$BF_1 = round(1 + rand) \quad (8)$$

Where MV, BF_1 and BF_2 represents as mutual vector, benefit factor of organism 1 and organism 2 respectively.

At this point the BF_1 and BF_2 can be act as a more prominent useful favorable position for only one organism than another organism.

The role of BF_1 and BF_2 is explained in this way. In any environment, certain mutualism associations possibly will provide a prominent useful preferred standpoint for only one organism than another organism. As such, organism A possibly will get an enormous advantage while collaborating with organism B. In the interim, organism B might only get sufficient or not all that huge advantage while cooperating with organism A. These elements speak to level of advantage to every organism, specifically, regardless of if an organism incompletely or completely profits by the collaboration [11].

B. Commensalism Phase

Commensalism determines a relationship between two organisms where one advantages and the other is unhurt. For instance, the association amid remora fish and sharks is commensalism. Remora affixes herself with shark and grubs sustenance remains, subsequently receiving an advantage. The shark is unpretentious through remora fish actions and acquires nominal, assuming any, gain as of the association [11].

Like the mutualism stage, an organism Eco_j is chosen arbitrarily from the ecological system which associated with Eco_i. Herein situation, organism Eco_i endeavors to return by the collaboration. Be that as it may, organism Eco_j itself neither advantages nor experiences the association. The novel candidate outcome of Eco_i is figured by the commensal beneficial symbiosis interaction amongst creature Eco_i and Eco_j, displayed now in (9).

$$EcoNew_i = Eco_i + rand(-1,1) * (Eco_{best} - Eco_j) \quad (9)$$

Following the principles, organism Eco_i is refreshed by its new fitness is superior to earlier fitness [11].

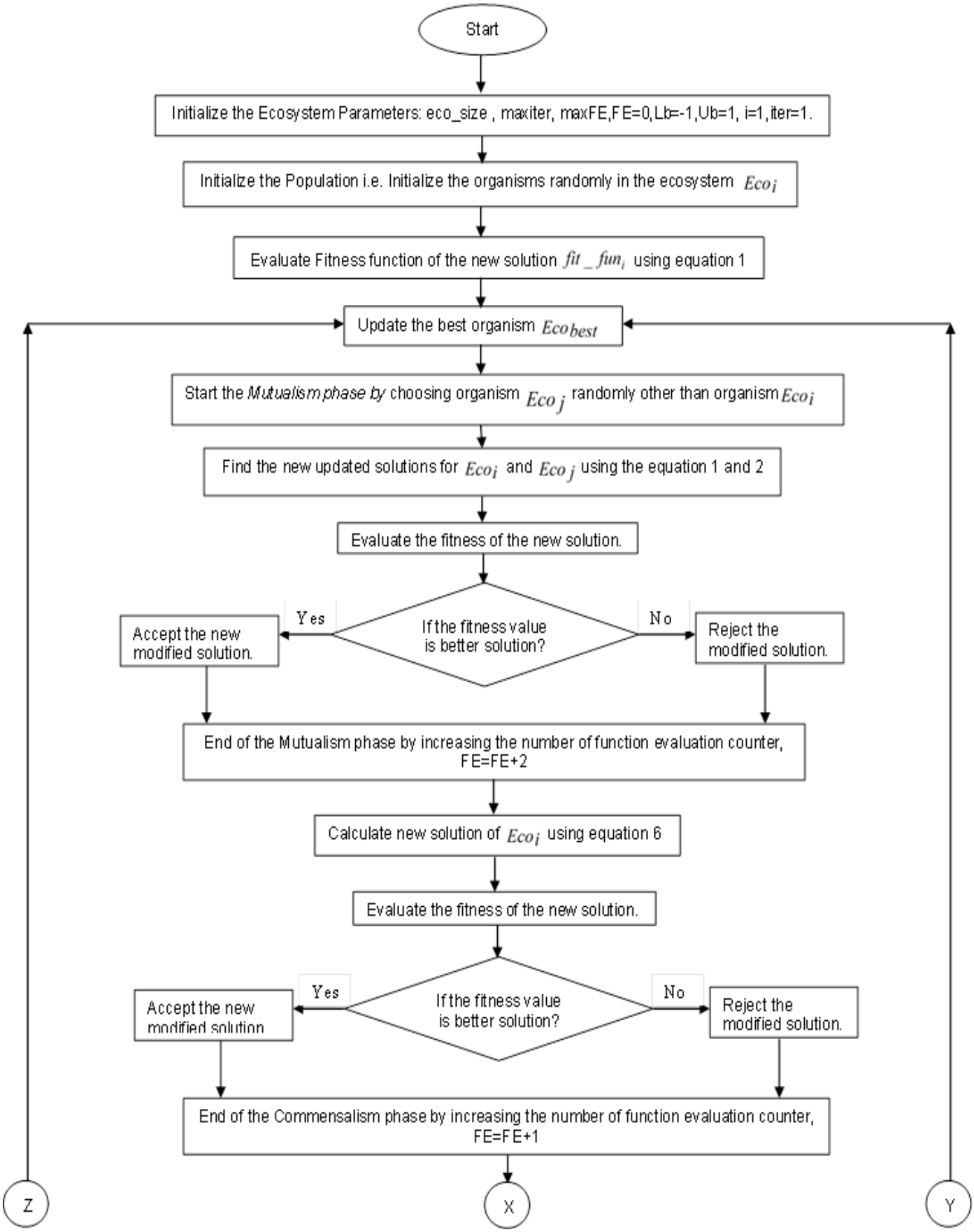

Fig. 2: SOSA Flowchart with multiple processor scheduling

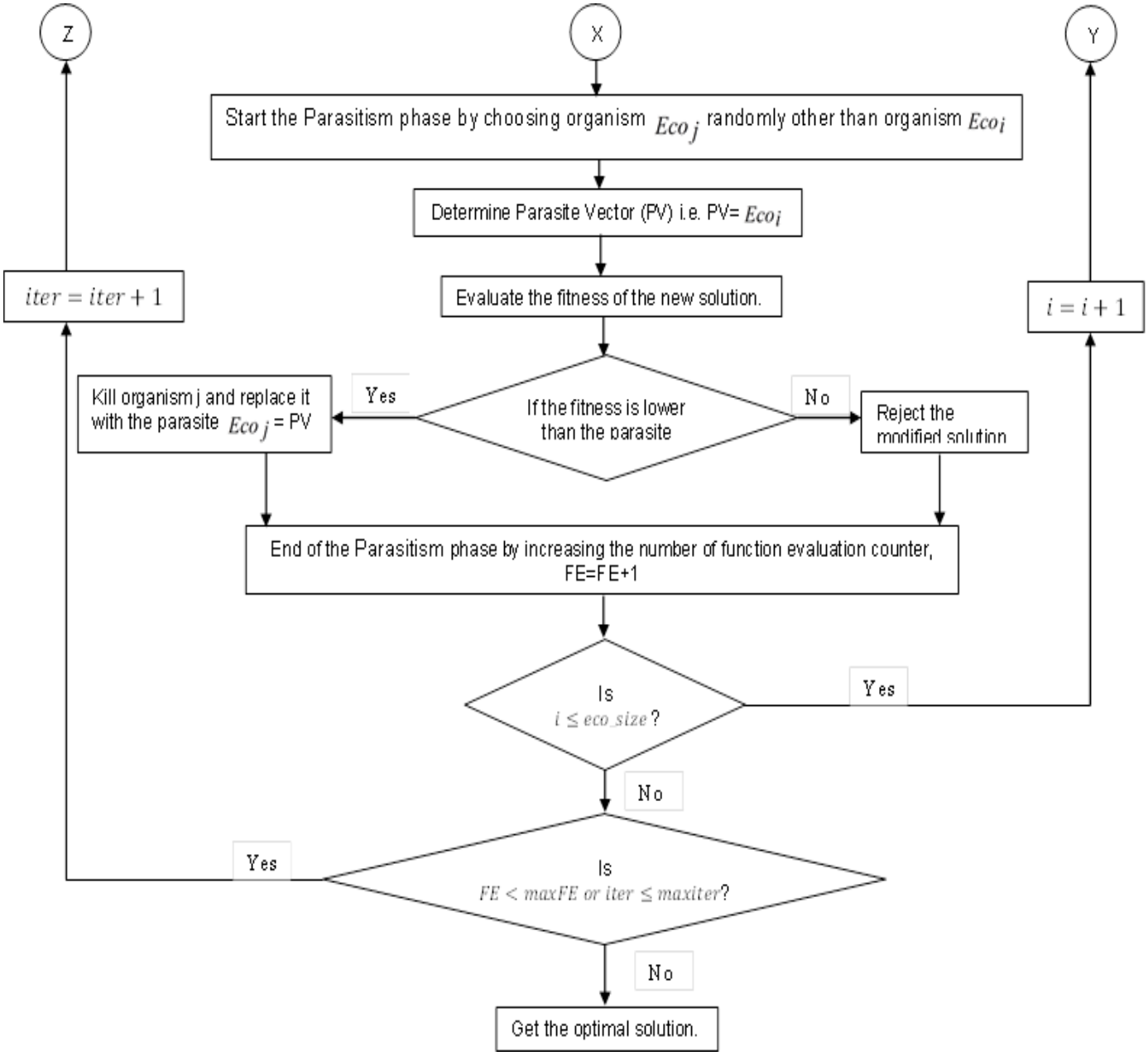

Fig. 2: SOSA Flowchart with multiple processor scheduling (Continued)

C. Parasitism Phase

Parasitism determines an association amid two organisms where one benediction and the other are hurt. For example, the relationship between woodpecker and tree is parasitism. The woodpecker eats insects. The tree makes its own wood. The tree is shelter for woodpecker. But the woodpecker puts holes in the tree.

In SOSA, organism Eco_i is given a part like the woodpecker over the formation of a non-natural parasite named as ‘‘Parasite_Vector (PV)’’. This is made within the search space by reproducing organism Eco_i using (10), at that point adjusting the arbitrarily chose dimensions utilizing an arbitrary value using (11). Organism Eco_j is chosen haphazardly from the ecological system and works for instance a person of the parasite vector using (12). PV attempts to supplant Eco_j in the ecosystem using (13). At that time mutually organisms are assessed to gauge their wellness.

$$PV = Eco_i \tag{10}$$

$$D = (1 : ceil(rand * E)) \tag{11}$$

$$PV\ (:, D) = rand(1, length(D)) * (Ub(D) - Lb(D)) + Lb(D) \tag{12}$$

$$Eco_j = PV \tag{13}$$

Where, PV is the parasite vector
E is the no. of organism or number of variables.
D is any random dimension
Ub and Lb is upper and lower bound value respectively.

V. SIMULATION

Here, the problem is simulated using MATLAB. The criteria are to minimize the execution time and increase the processor utilization. Because higher the percentage of the processor utilized by the less execution time of tasks.

For simulations the parameters set at: $eco\_size = E$, $\max iter = 100$, $\max FE = 5000$, $Lb = -1$, $Ub = 1$. Where, E signifies the no. of organism in ecosystem and set to the value as no. of task allocated to the processor. Arbitrarily pulled out no. of organisms and allocated within an array Eco , with 4 individual processors and 7 individual tasks. When any task is not allocated to processor, the value will be 0 otherwise 1.

Initialize the population or organism in ecosystem randomly such as Eco_i . SOSA allowed repeating the procedure for either $\max iter$ or $\max FE$ times in simulation. Obtain the best fitness value in every iterations.

Here, best signifies the least value of fitness function. For every cycle, the SOSA has the capability to obtain the global minima in each and every situation with a high level of precision.

All functions i.e. fitness function, utilization and objective functions are continued with a 100 maximum number of iteration. In this article, we have taken 5 distinct cases, such as:

Case 1: Minimize the execution time using SOSA.
Case 2: Increasing the utilization of processor.
Case 3: The performance of execution time Vs. Processor Utilization using SOSA
Case 4: Comparing Processor Utilization of SOSA with other algorithms.
Case 5: Comparing execution time of SOSA with other algorithms.

For all the above cases are explained here with different values of the number of processor and number of task in increasing order.

A. Minimize the Execution Time Using SOSA

The execution time of multiple processor scheduling for 100 iterations using SOSA is illustrated in Table II and Fig. (3). The execution time is gradually decreasing in each iteration.

B. Increasing the Utilization of Processor

Table III illustrates the formation of data for 100 iteration outcome from SOSA for increasing the performance of processor utilization as shown in Fig. (4).

C. Performance of execution time vs. Processor Utilization using SOSA

Table IV illustrates the formation of data for 100 iteration outcome from SOSA for the relationship between execution time and processor utilization. Here, if the execution time is lower than the processor utilization will be more as shown in Fig. (5).

D. Comparing Processor Utilization of SOSA with Other Algorithms

In this case, the processor utilization of multiple processor scheduling using SOSA is equated with other methods for instance GA, BFO, GBF and WCA as shown in Fig. (6). Here we found that SOSA algorithm has more utilization of processor as compared with the other above mentioned algorithms as shown in Table V. The least value of processor utilization is 0.0823 which represents that the SOSA has more processor utilization after 100 iteration.

Table II: Performance of SOSA with performance time and no. of iterations as parameters

Iteration	Performance Time
10	14.8002
20	13.1701
30	11.9945
40	10.9851
50	9.5623
60	7.6693
70	5.7915
80	3.2371
90	1.7495
100	0.0244

Table III: Performance of SOSA with Processor Utilization (UP) and number of iterations as parameters

Iteration	Processor Utilization
10	5.6319
20	10.9190
30	14.3179
40	20.6449
50	27.1726
60	37.2086
70	48.1021
80	61.0137
90	94.3517
100	129.2776

Table IV: Performance of SOSA with Processor Utilization (UP), Performance Time and no. of iterations as parameters

Iteration	Performance Time	Processor Utilization
10	14.8002	5.6319
20	13.1701	10.9190
30	11.9945	14.3179
40	10.9851	20.6449
50	9.5623	27.1726
60	7.6693	37.2086
70	5.7915	48.1021
80	3.2371	61.0137
90	1.7495	94.3517
100	0.0244	129.2776

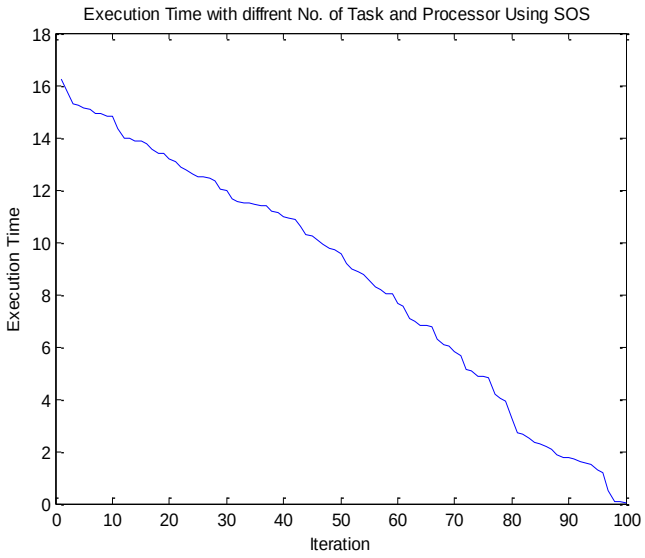

Fig. 3: Minimizing the Execution time

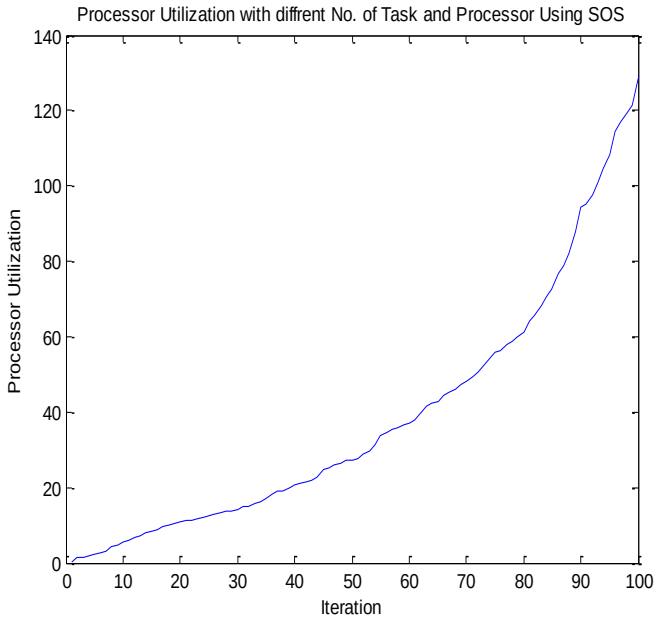

Fig. 4: Increasing the performance of processors

Table V: Performance of Processor Utilization with Iteration as parameter

Iteration	GA	BFO	GBF	WCA	SOSA
1	48.5000	1.3907	1.0126	108.2093	20.6449
40	25.9500	1.3762	1.0120	6.6745	14.3179
75	12.5263	1.2292	1.0117	2.2041	4.8571
100	8.9048	1.0565	1.0113	1.2312	0.0823

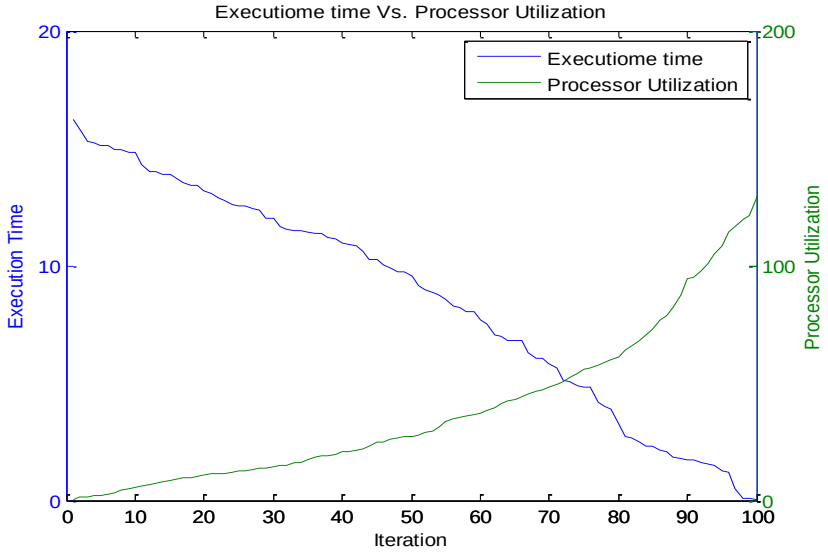

Fig. 5: Relationship between Execution time and processor utilization using SOSA.

E. Comparing execution time or performance time of SOSA with other algorithms

Here we found the main objective of multiple processor scheduling i.e. to minimize the execution time. Table VI shows that the SOSA algorithm gives minimum performance time as equated with former methods. The comparison of all algorithms for getting the minimum performance time using different no. of tasks and processors is displayed in Fig. (7). The bold values of Table VI represent the comparatively best values.

No. of Processor = 4, 17, 20, 24
No. of Task = 5, 10, 25, 27
No. of Iteration = 100

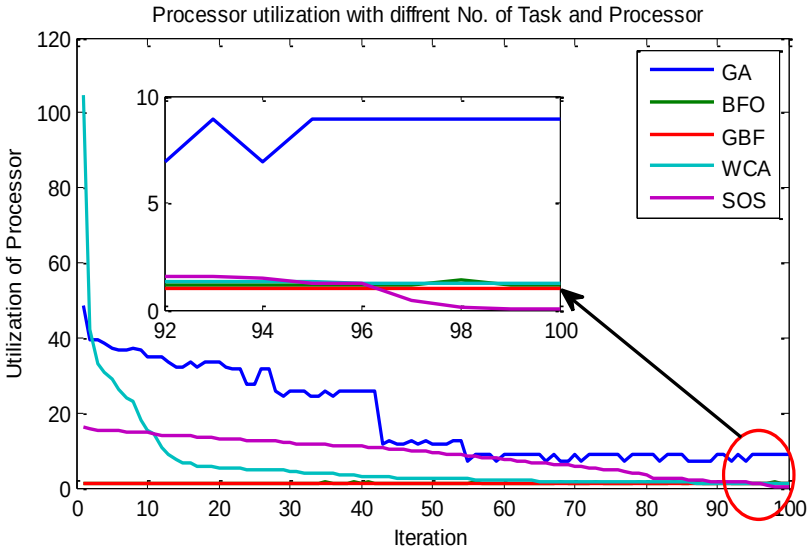

Fig. 6: Performance of utilization of processors with different methods.

Table VI: Work of Performance Time with variation in no. of processors and tasks

Task	Processor	GA	BFO	GBF	WCA	SOSA
5	4	7.0000	1.0000	1.0000	1.0000	**0.0614**
10	17	9.9375	1.0000	1.0036	0.7544	**0.0244**
25	20	46.0000	1.0000	1.0026	1.0000	**0.0067**
27	24	6.8966	1.0565	1.0030	1.0000	**0.0047**

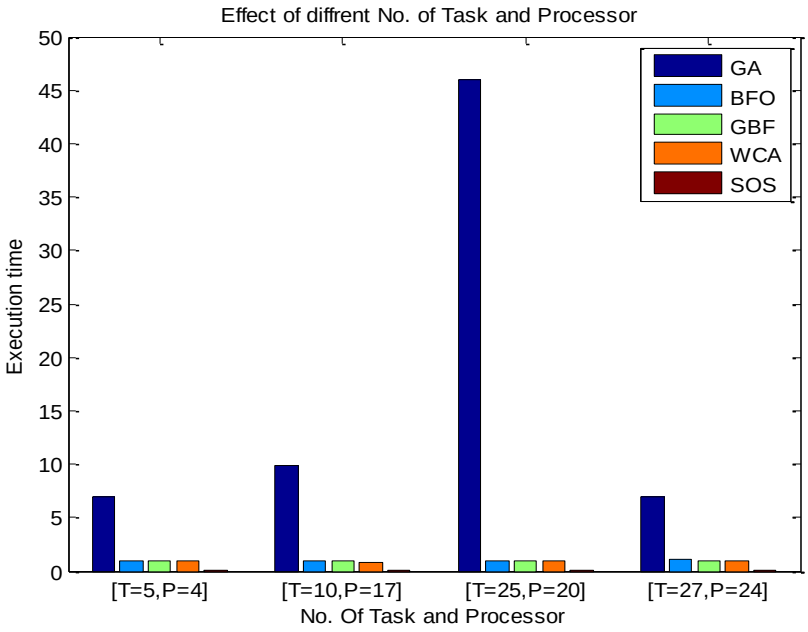

Fig. 7: Performance of Execution Time using different methods.

VI. CONCLUSION

In this article suggested a SOSA is utilized to allocate a task to a processor in the multiple processor scheduling problems. More obvious after the graphical and experimental consequences, the recommended SOSA achieved outstandingly well than the other algorithms Viz. GA, BFO, GBF and WCA.

VII. FUTURE WORK

In future, our work will be extended by doing with novel propelled strategies for developing a better scheduling optimization algorithm.

REFERENCES

1. Binitha S, S Siva Sathya. A Survey of Bio inspired Optimization Algorithms. International Journal of Soft Computing and Engineering (IJSCE) 2012; ISSN: 2231-2307, Volume-2, Issue-2.
2. Sasmita Kumari Nayak, Sasmita Kumari Padhy, Siba Prasada Panigrahi, "A Novel algorithm for dynamic task scheduling", Future Generation Computer Systems, 2012, Volume 28, Issue 5, Pages 709-717.
3. RDEL: Restart Differential Evolution algorithm with Local Search Mutation for global numerical optimization, Ali WagdyMohamed, Egyptian Informatics Journal Volume 15, Issue 3, November 2014, Pages 175-188
4. Particle swarm optimization almost surely finds local optima, Manuel Schmitt, Rolf Wanka, Theoretical Computer Science Volume 561, Part A, 4 January 2015, Pages 57-72
5. An Effective Hybrid Of Bat Algorithm And Hill Climbing For Global Optimization Of High- Dimensional Functions, Ali Osman Topal, Oguz Altun, Yunus Emre Yildiz, Jnts, 2015/Vol. Xx (2).
6. Cuckoo Search Algorithm Based On Local Optimization In The PID Parameter Optimization, Lei Luo,Lixia Lv, Workshop on Advanced Research and Technology in Industry Applications (WARTIA 2016).
7. Yang X-S, Sadat Hosseini SS, Gandomi AH (2012) Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. Appl Soft Comput 12(3):1180–1186. doi:10.1016/j.asoc.2011.09.017
8. Gandomi AH, Yang X-S, Alavi AH (2011) Mixed variable structural optimization using firefly algorithm. Comput Struct 89(23–24):2325–2336. doi:10.1016/j.compstruc.2011.08.002
9. Talatahari S, Gandomi AH, Yun GJ (2012) Optimum design of tower structures using Firefly Algorithm. Struct Des Tall Spec
10. Wang G, Guo L, Duan H, Liu L, Wang H (2012) A modified firefly algorithm for UCAV path planning. Int J Hybrid Inf Technol 5(3):123–144
11. Min-Yuan Cheng , Doddy Prayogo, (2014), Symbiotic Organisms Search: A new metaheuristic optimization algorithm. Computers and Structures 139 (2014) 98–112.
12. Chao-Tang Tseng, Ching-Jong Liao, A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks. International Journal of Production Research. Volume 46, 2008 - Issue 17. Pages 4655-4670 (2008)
13. Kuo-Ching Ying, Shih-Wei Lin, Multiprocessor task scheduling in multistage hybrid flow-shops: an ant colony system approach. International Journal of Production Research. Volume 44, 2006 - Issue 16. Pages 3161-3177.
14. Ruskartina Eki, Vincent F. Yu, Santosa Budi, A. A. N. Perwira Redi. "Symbiotic Organism Search (SOSA) for Solving the Capacitated Vehicle Routing Problem", World Academy of Science, Engineering and Technology, International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering Vol:9, No:5, 2015.
15. Nayak S.K., Panda C.S., Padhy S.K. (2018) Efficient Multiprocessor Scheduling Using Water Cycle Algorithm. In: Ray K., Pant M., Bandyopadhyay A. (eds) Soft Computing Applications. Studies in Computational Intelligence, vol 761. Springer, Singapore.

AUTHORS PROFILE

Sasmita Kumari Nayak, received her B.Tech in IT from BPUT, Odisha, India in 2005 and M.Tech in IT from CET, Bhubaneswar, Odisha, India in 2010. Presently she is pursuing her Ph. D research under the guidance of Dr. Chandra Sekhar Panda and working as an Assistant Professor at Computer Science Engineering Department of Centurion University of Technology and management (CUTM), Bhubaneswar, Odisha, India. Her research interest includes Grid Computing, Multiprocessor scheduling, Soft Computing and Evolutionary Algorithms.

Dr. Chandra Sekhar Panda, had his M.C.A. from OUAT, Bhubaneswar, Odisha, India and Ph.D from Sambalpur University Odisha, India. Presently he is working with Sambalpur University, Odisha, India. His research interests include Programming Languages, Machine Vision, Image Processing, soft computing.