Available online at www.scholarsresearchlibrary.com

Scholars Research Library

Archives of Applied Science Research, 2014, 6 (5): 178-182
(http://scholarsresearchlibrary.com/archive.html)

ISSN 0975-508X
CODEN (USA) AASRC9

Multi-processor task scheduling by using eurygaster life

Fariborz Ahmadi, Hamid Salehi* and Khosro Karimi

Young research and elites club, Ghorveh branch, Islamic azad university, Ghorveh, Iran

---

ABSTRACT

Multiprocessor task scheduling is process of allocating some tasks to several processors so that the over-all time of tasks completion is to be minimized while precedence constraint are preserved. Multiprocessor task scheduleng is NP-complete problem that means it takes a long time to solve. In this paper, by simulating eurygaster life a novel approach has been proposed to solve this problem. These insects divide the search area of problem into several sections and investigate each section separately. Evaluation results show that proposed approach is more useful and faster that heuristics methods like genetic algorithm and PSO.

Keywords — evolutionary computation, genetic algorithm, eurygaster life, multiprocessor task scheduling, parallel systems.

INTRODUCTION

Multiprocessor task scheduling is a process of allowing task to be processed on several processors [1, 2]. This problem is belonging to NP-Hard category and has been focused to solve by researchers [1,2,3,4,5,7,9]. In this problem, a set of jobs with length J_i are distributed on the limited number of processor to be executed while the main purpose is finding appropriate scheduling so that the time to run all of the task without overlap be minimized[1,4]. In this paper, all processors are independent and also each task can be executed by each arbitrary processor. In the other hand, tasks may be having some dependence to each other and precedence constraints are appeared. Real parallel systems must handle these issues because parallelization level is reduced and in some condition parallel systems fails to exploit parallelism with correct execution of tasks [7,8]. Scheduling tasks on multi-processors is one of the important challenging in parallel and distributed systems, consequently many researches have been focused to solve this problem [6,7]. In this paper some assumption are preserved [1,5]. 1-All the tasks and processors are available at time Zero. 2- Processors never fail. 3- All processing time on the processors are known, deterministic, finite and dependent on sequence of the tasks to be processed. 4- Each processor is continuously available for assignment. 5- The first processor is assumed to be ready whichever and whatever task is to be processed on it first. 6- Processors may be idle 7- Splitting of task or task cancellation is not allowed.

In the other hand, eurygaster algorithm is an algorithm developed to solve NP-Hard problems. In this algorithm, each eurygaster shows one solution and the best solution is the answer of problem. In this algorithm a set of eurygasters distributed over wheat farms and ruin them. After ruining one farm, they migrate to adjacent farm to disturb it. This routing is continued until either all of the farms are ruined or the

best farm is reached.

We introduce eurygaster behavior in section2. Proposed approach and evaluation results are described in section 3 and section4, respectively. Finally, we draw some conclusion in section5.

Eurygaster behaviors

Eurygaster integricep is an insect pest that predominantly attacks grains, feeding on the leaves, stems and grains, reducing yield and injecting a toxin into the grains which adds a foul smell to the resulting flour, and substantially reduces the baking quality of the dough.

In winters eurygasters live under the plants and bushes in hillside, in several numbers and make a group. At the end of winter and at the beginning of spring when it gets warmer, these insects end their winter sleeps and get ready to move and fly to grain fields by moving over the high mountains and leaving the nests in groups. The first group by the use of its instinct finds the best and the nearest grain fields and stays there. Getting there, this group of insect sends signals to the air to show the other groups their being there. Based on the number of eurygasters in a place, the strength of signals will be different. If the number of eurygasters in a grain field is not great, the rate of diffused signals will be little and if the number of eurygasters in a grain fields is greater, the rate of diffused signals will be increased. They diffuse these signals to show the others that reside there. So that the other groups of eurygasters understand that they should not close to the grain field which contains the first group. Of course the other groups based on diffused signals by the first group and the strength of these signals they decide if they can land and stay there or not. If the power of diffused signals is low, it means that some of the other groups of eurygasters can land and stay by the other groups which are resident there and began to eat. While the strength of the signals in the sky is high, it means that the other groups cannot land on the field(s) containing eurygasters, and they must fly to other fields in which there are no eurygasters, to live and eat.

According to the passage mentioned above, the next group of eurygasters while flying from their nests to other fields to find the best grain fields searches the best and closest ones to land and eat based on the broadcasted signals by landed group(s). This process will continue until they will find a suitable and useful grain field to eat.

We conclude that all the grain fields in a wide area will be attacked by eurygasters, because they do not gather in a one place.so, when there is not enough food in a grain field in which the eurygasters have stayed for a time, they will fly to a new field with no eurygasters according to the process mentioned above.

Eurygaster Algorithm

In this section, eurygaster algorithm is described. Solving non-linear functions are so necessary in real life today and recently researchers interested in inventing methods to solve them. Thus, our approach contributes to solve NP-class problems. The great advantage of this algorithm is that it's so easy to implement and is also inexpensive in term of memory and speed. The second advantage of this algorithm is its convergence speed compared to other methods like GA and PSO.

The related semi-code of the proposed algorithm is as algorithm1. This algorithm is formed by combining of 3 sub-algorithms. In each phase, we described how to apply this algorithm to shortest path problem in a way it can be solved.

1 Initialization

In this phase the shortest path problem is divided into some partition and each partition is investigated separately. Also the structure of eurygaster is constituted. Suppose we have one graph like figure1.

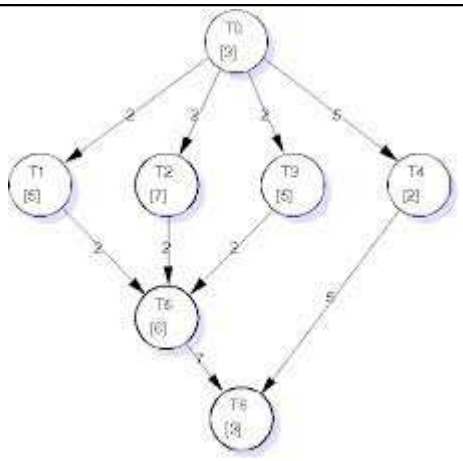

Figure1. Task graph

In this research, implementation has been executed on 4 processors. So a sample eurygaster for supposed task graph in figure 1 is according to figure2.

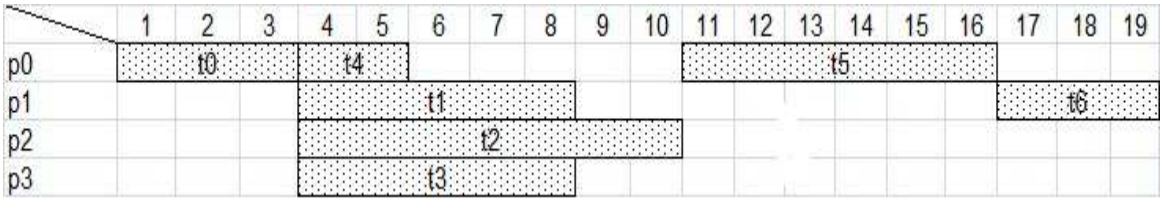

Figure2. Eurygaster Structure

I← the number of clusters

While I <> 0 do

1. Initialization: produce euragasters or particles according to characteristic of one partition
2. Distribution: distribute eurygasters on the regions of the partition
3. Evaluation: evaluate suitability of each eurygaster or particle depend on the problem
   - 3.1 If the suitable result of the partition is not obtained
     - 3.1.1. Change the position of Eurygasters in the partition
     - 3.1.2. goto 3
   - 3.2 If the result of the problem is not obtained
     - 3.2.1. I--
     - 3.2.2. goto 1

Else

- 3.2.3. Stop algorithm or break

End while

Algorithm1. Eurygaster Algorithm

4. Report the solution of the problem

In this figure, eurygaster is a matrix with 4 processors. Processes T0, T4, T5 are executed on processor p_0. Each process is executed on one of the processors if it is be free and dependant constriants are satisfied. Also, in this problem the area of problem is divided into three partitions as follows. The first partition involves all processes with least execution time, orderly. Also in this partition dependant constraint must be respected because this criterion is necessary to run program correctly. The logic behind this partition is that waiting time and turnaround time of processes are reduced because of running the short process as soon as possible. The second partition involves all processes with most execution time, orderly. Also in this partition dependant constraint must be satisfied like the first partition. The logic behind this partition is that after executing long processes, processor can execute several short processes. The last partition orders processes arbitrarily.

4-2 Distribution

In this phase, eurygasters according to their structures are produced and spread over the region of first partition. There are three probabilities in each partition. First, the answer is not found and the whole search space of partition is not investigated. In this case, some other eurygasters are produced and distributed to remaining section of partition. Second, the answer is not found and the whole search space of partition is investigated. In this case, some erygasters are created and spread over another partition. Finally, if the answer of problem found in partition, the algorithm is terminated.

4-3 Suitability

In order to find the solution of the problem, eurygasters should be evaluated so that the best answer is obtained. In order to reach the best solution, we must have a tool that justifies the solution. In this paper, the following function is used to examine the suitability of solutions.

Suitability (eurygaster)=1/(finish time of last process)

It is obvious that the best answer is one with highest suitability.

5. Evaluation Results

To evaluate results, we use standard task graph as a benchmark and all evaluation have been executed on four processors [6]. For better evaluation, proposed algorithm and genetic algorithm have been implemented and executed on each graph for 10 times. The best and the average time of both algorithms are shown in table1. For simplicity, the communication cost of tasks is set to zero. Table1 shows that proposed approach is faster in obtaining the solution of problem.

Table1. Evaluation results of proposed approach VS genetic algorithm

	Proposed approach		Genetic algorithm	
Benchmark	Best time	Average time	Best time	Average time
50tasks\Rand0100	0.00123	0.00776	0.00271	0.00977
50tasks\Rand0069	0.00287	0.00350	0.00276	0.00341
50tasks\Rand0019	0.00539	0.00856	0.00723	0.01391
50tasks\Rand0016	0.00465	0.00956	0.00481	0.01309
50tasks\Rand0002	0.018548	0.02310	0.01933	0.02684
100tasks\Rand0100	0.01936	0.02566	0.02045	0.02933
100tasks\Rand0069	0.01966	0.03270	0.02182	0.03859
100tasks\Rand0019	0.01342	0.01857	0.01833	0.02438
100tasks\Rand0016	0.01333	0.02342	0.01715	0.02742
100tasks\Rand0002	0.02088	0.02190	0.02328	0.02937

As you can see in table1, the execution time of our approach is less than genetic algorithm except in 50 tasks\rand0100 benchmark. The average time is the medium time of running algorithms on each benchmark 10 times and the best time is one of the tests that get solution faster than other tests in 10 times execution.

Proposed approach unlike genetic algorithm lacks the local optimum so the probability

Of getting the more accurate solution in this method is much more than the genetic algorithm. Moreover, in this method every space of the problem is searched for once while in the genetic algorithm every part of the problem space can be searched several times in different generations, so the rate of convergence in this algorithm is much more than the genetic algorithm. It is concluded that researchers work has accurate solution and also is faster in comparison to genetic algorithm.

CONCLUSION

In this article, one approach based on the behaviors of eurygasters has been presented to solve multiprocessor scheduling. This approach unlike genetic algorithm lacks the local optimum so the probability of getting the more accurate solution in this method is much more than the genetic algorithm. Moreover, in this method every space of the problem is searched for once while in the genetic algorithm every part of the problem space can be searched several times in different generations, so the rate of convergence in this algorithm is much more than the genetic algorithm. Also, this algorithm is easy to implement by computer. If takes a few lines to programming and doesn't need a huge memory or CPU speed. Both proposed algorithm and genetic algorithm were executed on standard task graphs and evaluation results show that convergence speed by proposed approach is faster and also more accurate.

REFERENCES

[1] R. Verma and S. Dhingra, *IJCSMS International Journal of Computer Science and Management Studies, Vol. 11*, Issue 02 ISSN (Online): 2231-5268
[2] M. R Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, pp 238. ISBN 0716710445, **1979**.
[3] F. Herrera, M. Lozano & J.L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artificial Intelligence Review 12: 265–319. Kluwer Academic Publishers. Printed in the Netherlands*, **1998**.
[4] A. Otman & A. Jaafar, "A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem," *International Journal of Computer Applications* (0975 – 8887) Volume 31– No.11, October 2011.
[5] A. Dhingra & P. Chandna, *International Journal of Engineering (IJE)*, Volume (3): Issue (5), **2009**.
[6] Standard task graph set is available online at: http://www.kasahara.elec.waseda.ac.jp/schedule
[7] Vahid Majid Nezhad1, Habib Motee Gader2 and Evgueni Efimov3, (**2011**), *IJCSI International Journal of Computer Science* Issues,Vol. 8.
[8] M.R. Gary and D.S. Johnson, Computers and Imractability: A Guide to the Theorv of NPComnleteness.W.H. Freeman and Company, **1979**.
[9] Haupt, R.L., Haupt, S.E., Parallel genetic algorithms, John Wiley & Sons, **2004**.