Shuffled Frog Leaping Algorithm in Distributed System

S. Sarathambekai
Assistant Professor (SG)
Department of IT
PSG College of Technology

K. Umamaheswari, Ph.D
Professor
Department of IT
PSG College of Technology

G. Tharanipriya
PG Scholar
Department of IT
PSG College of Technology

ABSTRACT
The general problem of multiprocessor scheduling is stated as scheduling tasks on a multiprocessor system so that a set of performance criteria can be optimized. Shuffled Frog Leaping (SFL) algorithm is a recently developed population based search algorithm, which is inspired by the interactive behavior and global exchange of information of frogs searching for food. It is combination of meme-based genetic algorithm or Memetic Algorithm (MA) and Particle Swarm Optimization (PSO). This algorithm is used in this paper to solve a task scheduling problem in distributed systems which aims at minimizing the tri-objectives such as makespan, flow time and reliability cost.

General Terms
Task Scheduling, Optimization algorithm

Keywords
Distributed system, PSO algorithm, SFL, Scheduling

1. INTRODUCTION
The multiprocessor system can be classified as homogeneous or heterogeneous systems [1, 2]. A homogeneous computing system is one in which all processing elements have the same execution speed. A heterogeneous computing system consists of different set of processors with different processing capacities. Multiprocessing is the use of two or more central processing units within a single computing system. The term also refers to the ability of a system to support more than one processor and the ability to allocate tasks between them. Multiprocessor Scheduling can be stated as scheduling a set of dependent or independent tasks in order to minimize certain objectives.

A job is called a task. The tasks may be dependent or independent [3]. A task that does not depend on other task is called as independent task. Independent tasks can be assigned to any available processor. If one task depends upon the completion of the other task then the tasks are known as dependent tasks.

Shuffled Frog Leaping Algorithm (SFLA) [4] is a meta-heuristic search algorithm. The main purpose of this algorithm is achieving a method to solve complicated optimization problems without any use of traditional mathematical optimization tools. This algorithm has been inspired from memetic evolution of a group of frogs when seeking for food. The initial population of frogs is partitioned into groups or subsets called "memeplexes" and the number of frogs in each subset is equal. The SFL algorithm is based on two search techniques: Local search and global information exchange techniques. Based on local search, the frogs in each subset improve their positions to have more foods (to reach the best solution). During global search, the obtained information between subsets is compared to each other (after each local search in subsets).

2. PROBLEM DEFINITION
This research work is aimed at scheduling of independent tasks in heterogeneous processors. The n represents the number of tasks and m represents the number of processors in the distributed systems. Expected Time to Compute (ETC) model can be characterized using three parameters: task heterogeneity, machine heterogeneity and consistency. Heterogeneities could be either high or low with respect to both tasks and machines. This matrix gives the exact time at which a processor completes executing a particular task. It is a size of n x m represents the computing capacity of the resources and the workload of the tasks, in some unit of time, where each position in ETC[n][m] indicates the expected time to compute task n in processor m. A set of n independent tasks are to be scheduled on a set of m heterogeneous processors. The environment is static, non-preemptive and the goal of the work is to minimize three objectives namely,

- Makespan (MS)
- Mean Flow Time (MFT)
- Reliability Cost (RC)

2.1 Objective Calculation
2.1.1Makespan
Makespan [5] is the maximum of completion time of all the tasks allocated to all processors. This is calculated using equation (1).

$$\text{Make span} = \max\{ \sum_{\forall \text{ task i allocated to processor}} C_{i,j} + W_j \}$$

$$j \in (1, 2, 3 \ldots, m) \qquad (1)$$

Where $C_{i,j}$(i ε{1,2,...,n}, j ε{1,2,...,m}) is the execution time for performing i^{th} task in j^{th} processor and W_j(j ε {1,2,...,m}) is the previous workload of P_j.

2.1.2 Mean Flow Time
The flow time refers to the response time of the processor. The flow time is calculated by summing up the completion time of each task where $F_{i,j}$is the finishing time of $task\ T_i$ on a processor P_j. The mean flow time [6] is used to evaluate flow time, which is calculated using equation (2).

$$MeanFlowtime = \frac{\sum_{i=1}^{m} M\_Flow_i}{m} \qquad (2)$$

$$M\_Flow_i = \frac{\sum_{j=1}^{k} F_{ji}}{k} \qquad (3)$$

2.1.3 Reliability Cost
Reliability cost [7] is defined to be the probability that the system will not fail during the time that it is executing the tasks. It is the indicator of how reliable a given system is when a group of tasks are assigned to it. Lower the reliability cost, higher the reliability. Reliability cost is the sum of

processor reliability and the link reliability. Link reliability is zero because the tasks are independent. Processor reliability is calculated based on the processor failure rate, which ranges from (0.95 to 1.05) x10^6. A task v_i on a processor p_j's failure rate is λ_j. The reliability cost of a task schedule is the summation over all task's reliability costs based on the given schedule. Given a heterogeneous processor P, the reliability cost is defined as

$$\text{Reliability Cost} = \sum_{j=1}^{m} \sum_{x(T_i)=j} \lambda_j C_{ij}(T_i) \quad (4)$$

2.2 Fitness Calculation

In SFL algorithm, all frogs at each iteration step are evaluated according to a measure of solution quality, called fitness. The fitness is calculated using the Adaptive Weighted Sum [8] Method. The objectives are summed up based on random weighting factors resulting in transformation of the multi-objective optimization problem into single objective optimization problem. The fitness is calculated using equation (5).

$$\text{Fitness} = \alpha_2 \text{MS} + (1-\alpha_1)\alpha_2 \text{MFT} + (1-\alpha_2)\text{RC} \quad (5)$$

Where $\alpha_i \in [0,1]$.

3. ALGORITHM DESIGN

The SFL [4] is the memetic evolution of group of frogs for seeking the location that has the maximum amount of available food. It is based on evolution of memes carried by interactive individuals and a global exchange of information among the frog population. It combines the advantages of the MA and PSO algorithm. The characteristics of SFL: simple concept, great capability in global search and easy implementation. It involves a population of frogs with the same structure but different adapt abilities. The position vector of each individual frog represents a feasible solution of the optimization problems. The population is partitioned into a number of groups called as memeplex. Each memeplex represents a type of idea. The algorithm performs simultaneously an independent local search within each memeplex using a PSO. To ensure global exploration, after a default number of local iterations, the whole frogs are shuffled and reorganized into new memeplexes.

3.1 Pseudo Code for SFL

- Initialize the population based on the number of tasks and number of processors
- Evaluate the objectives- makespan, reliability cost and flow time using equation (1) - (4)
- Calculate the fitness using equation (5)
- Based on the fitness value, sort the frogs in descending order.
- Divide the population into memeplexes and find its local best $frog_{LB}$and local worst $frogt_{LW}$ and global best $frogt_{GB}$
- Update the local worst using equation (6) and (7)

$$D_i = rand * (frog_{LB} - frogt_{LW}) \quad (6)$$

$$frog_{new} = frog_{oldworst} + D_i \quad (7)$$

  Where rand is a random number between 0 and 1, $frog_{new}$ is new position of the worst frog.
- If this process produces a better solution replace the frog $frog_{oldworst}$ by $frog_{new}$
- Else replace $frog_{LB}$ in equation(6) by $frogt_{GB}$ and calculate $frog_{new}$
- If this process produces a better solution replace the frog $frog_{oldworst}$ by $frog_{new}$
- Else replace the frog $frog_{oldworst}$ by the randomly generated frog.
- The process will continue until the specified iterations.

4. IMPLEMENTATION

4.1. Frog Initialization

In the initial step, the independent tasks, number of processors, population size and the number of iterations are given as input. Each frog represents a feasible solution to the problem and a set of frogs are referred to as a population. The first generated frogs are termed as initial population. The frogs are represented in position vector form in which the elements are integer numbers between 1 and m, where m is the number of processors. An illustrative example is presented Table 1. It represents the number of tasks is 3, number of processor is 3 and population size is 3.

Table 1. Population Initialization

F \ T	T1	T2	T3
Frog 0	2	1	0
Frog 1	0	2	1
Frog2	1	2	2

In table 1, F represents frog. T represents task no. The values represent the processor number.

4.2 Objective Calculation

The three objectives namely makespan, reliability cost and flow time are calculated using the equation (1)-(4)

Table 2 represents the ETC matrix for the processors.

Table 2. Expected Time to Compute Matrix

P \ T	T1	T2	T3
P0	**3**	**2**	**5**
P1	**5**	**6**	**7**
P2	**8**	**9**	**4**

P represents the processor, T represents the task and the number represents the execution time of the task on a processor.

The Table 3 shows the objectives calculation for frog0. Similarly the objectives are calculated for remaining frogs in the population.

Table 3. Objective Calculation

Frog0=[2 1 0] **Processor Allocation :** Processor P2 is assigned to task1 Processor p1 is assigned to task2 Processor p0 is assigned to task 3 **Makespan, Reliabilitycost, flow time calculation for Frog 0:** **Makespan:** P0 ⟷ 0 T3 5 P1 ⟷ 0 T2 6 P2 ⟷ 0 T1 8 Makespan=8 **Frog 0:** Makespan=8 Reliability cost=0.00001821 Mean Flow time=6.3333	**Reliability cost:** **Processor Failure Rate:** Failure rate for P0=0.00000095 Failure rate for P1=0.00000096 Failure rate for P2=0.00000097 P0 =8*0.00000095 =0.00000760 P1 =6*0.00000096 =0.00000576 P2 =5*0.00000097 =0.00000485 Reliability cost = 0.00001821 **Mean Flow time:** Flow time of task in P0 =8/1 =8 Flow time of task in P1 =6/1 =6 Flow time of task in P2 =5/1 =5 Mean Flow time for frog0 =(8+6+5)/3 =6.333

4.3 Fitness Calculation

Fitness is calculated using the weighted sum method. The weights are randomly generated and the fitness is calculated using equation (5).

Fitness value for frog 0 = (0.818*0.057*8) + ((0.182)*0.057 * 6.333) + ((1-0.057)*1.82*10^{-4}) =1.1712000000000002

Similarly the fitness values are calculated for frogs in the population.

4.4 Leaping process

Based on the fitness, sort the frogs in descending order. Divide the frogs into memeplexes, each containing n1 frogs, in such a way that the first frog of sorted population goes to the first memeplex, the second frog goes to the second memeplex, n1 frog goes to n1 memeplex, and frog (n1)+1 goes to the first memeplex again.

4.5 Gbest , Lbest and Lworst Calculation

The smallest fitness value of the frog is the Gbest: $frogt_{GB}$. The smallest value in each memeplex is the Lbest: $frog_{LB}$. The largest value in each memeplex is defined as Lworst: $frogt_{LW}$. Figure1 shows the comparisons of Gbest and Lbest for different iterations. X axis represents the iterations and Y axis represents the Gbest and Lbest values for various iterations.

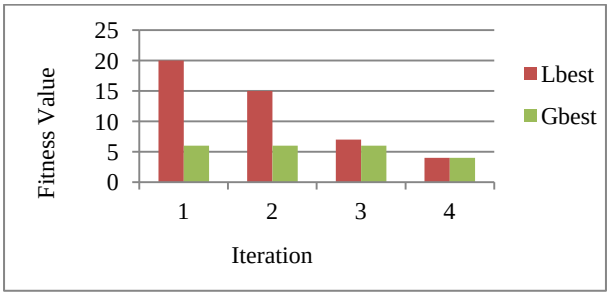

Fig 1. Comparison of Gbest and Lbest frogs

5. RESULT AND PARAMETER SETTINGS

5.1 Parameter Results

Table 4. Parameter settings

Parameter	value
No of processor	3
No of task	10
Population size	10
Memeplex size	2
No of iteration	20

5.2 Experimental Results

The algorithm is coded in java and executed in Net beans.

Table 5. Comparison of Fitness value of Gbest Frog

Iteration Number	Fitness Value
0	4235.678
5	4100.056
10	3876.234
15	3624.267
20	3401.187

The fitness value for iteration 1, 5, 10, 15 and 20 are calculated and presented in Table 5 and plotted in Figure 2.

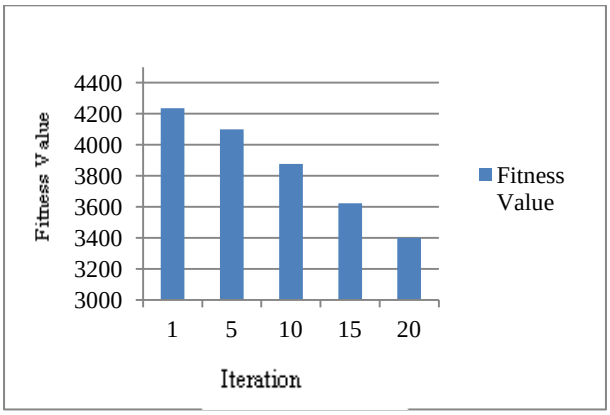

Fig 2. Comparison of Gbest value with respect to iterations

Figure 2 shows that increase in number of iterations will provide the better Gbest frog.

6. CONCLUSION

Task Scheduling is the challenging problem in heterogeneous computing systems. It is an extremely NP hard problem. This can be solved using heuristic/meta-heuristic algorithms efficiently because the traditional methods need more time for solving this NP hard problem. SFL is presented in this paper to solve this problem. SFL includes local search called leaping process which will avoid preventing the algorithm in local optima. The future work compares the proposed algorithm with the existing algorithm.

7. REFERENCES

[1] S. W. Choi, and Y. D. Kim, 2008 "Minimizing makespan on an m-machine re-entrant flow shop", Computers & Operations Research, Vol. 35, No. 5, pp. 1684–1696.

[2] DhodhiM K,Ahmad I,Yatama A,AhmadI, 2002 "An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems." Journal of Parallel and Distributed Computing, vol 62, pp 1338–61.

[3] Braun, T., Siegal, H., Beck, N, 2001," A comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing, vol. 61, pp 810-837.

[4] Muzaffar M, Eusuffand Kevin E, Lansey, 2003, "Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm", Journal of Water Resources Planning and Management, Vol. 129, No. 3, pp. 210-225.

[5] S.Sarathambekai, K.Umamaheswari, 2014, "Task Scheduling in Distributed Systems using Discrete Particle Swarm Optimization", International Journal of Advanced Research in Computer Science and Software Engineering, Vol 4, pp 510-522.

[6] R. Lindeke, 2005, "Scheduling of Jobs", IE 3265 – POM, Spring:www.d.umn.edu/~rlindek1/.../ Scheduling %20 of % 20Jobs_Sset11.ppt,.

[7] Xiao Qin and Hong Jiang, 2001, "Dynamic, Reliability-driven Scheduling of Parallel Real-time Jobs in Heterogeneous Systems", IEEE International conference on Parallel Processing, pp 113-122.

[8] I. Y. Kim and O. L. de Weck, 2006," Adaptive weighted sum method for multi-objective optimization: a new method for Pareto front generation", Springer-Structural and Multidisciplinary Optimization, Vol 31, pp 105-116.