**Assignment -1 Report**

**Roll no:23AT61R04**

Fiduccia-Mattheyses algorithm

Implemented in python

There are 5 sections in code:

1)Data input

2)Initial Solution

3)Gain Calculation

4)Constraint check

5)Solution update

**Data inputs:**

Takes the inputs of number of tasks, number of processers, Energy costs, Execution time

and Communication costs

t=number of tasks

p=number of processors

E[i­][j]=Energy cost of task(i) in processor(j) i={0,1,…t} j={0,1,2,….p}

T[i­][j]=Execution time of task(i) in processor(j) i={0,1,…t} j={0,1,2,….p}

C[i][j]=Communication cost between task(i) and task(j) i,j={0,1,…t}

**Initial Solution:**

Generates a random initial solution assign to Initial\_Array

Where,

Indices are tasks

Value at each index is which partition is the task located

Eg: t=3 p=2

Initial\_Array = [0 1 0]

Task 0 at partition 0

Task 1 at partition 1

Task 2 at partition 0

**Gain Calculation:**

now\_cost : represents the cost of each task in the present partition

maybe\_cost : represents the cost when a task is moved to a different partition

gain=now\_cost – maybe\_cost

choose the maximum gain and tentatively move that particular task to the partition which will have maximum gain

**Constraint check:**

Check the time constraint |***Aggr. Exec. Time on Pi - Aggr. Exec. Time on Pj | <=* D**

If satisfied move the task

If not choose the second max gain and check constraint once again continue until you satisfy

the time constraint

**Solution update :**

Do the same process until you move all the tasks and unlock all the tasks find the value of ‘k’ and its respective ‘G’

Update the solution to Final\_Array according to the value of k

Print the solution

**Generated Solution**









 













 













 



















 







 







 











 



 







 



 



 







 







 



 



 



 



 











 



 



 



 



 





 











