PROJECT REPORT

TSP IN INTEGRATED CIRCUITS

BY

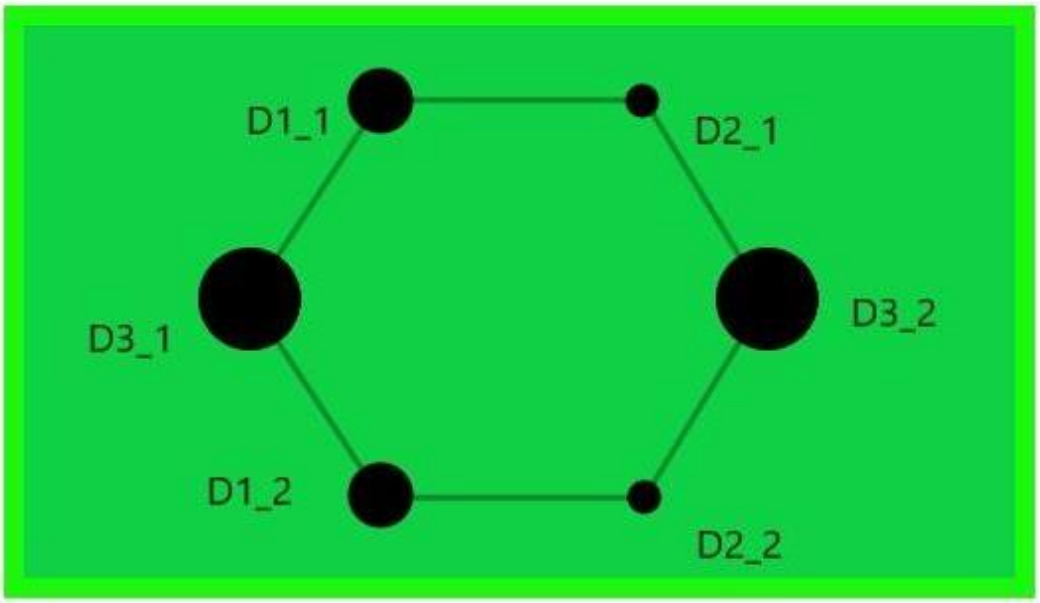
LAKSHMI PRIYA MOLAKALAPALLI

ABSTRACT

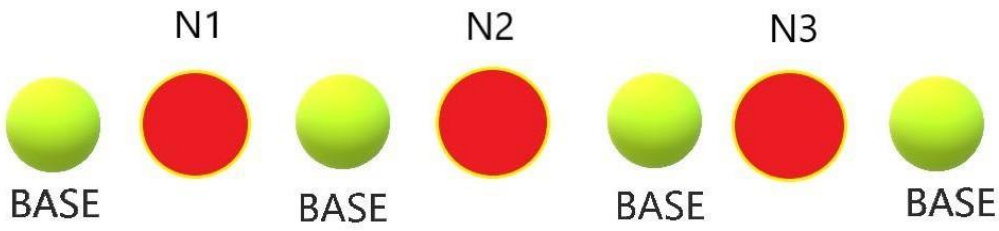
WE ALL KNOW THE DURING THE MANUFACTURE OF INTEGRATED CIRCUITS WE NEED TO DRILL HOLES ON THE CIRCUIT BOARD USING A ROBOTIC DRILL. SINCE DIFFERENT COMPONENTS ARE OF DIFFERENT SIZES IT IS NECESSARY FOR US TO DRILL DIFFERENT SIZE HOLES. EVERY TIME WE TRAVERSE FROM ONE NODE (HOLE) TO ANOTHER IT IS NECESSARY FOR US TO CHANGE THE SIZE OF THE PIN IN CASE IF BOTH THE NODES ARE OF DIFFERENT SIZES. HERE WE TRY TO MINIMIZE THE NO OF TIMES THE PIN BEING CHANGED WHILE TRAVERSING BETWEEN NODES.

INTRODUCTION

WE ARE GOING TO USE TRAVELLING SALESMAN PROBLEM (MODEL) TO SOLVE THIS APPLICATION. IN A TRADITIONAL TSP THE CITIES ARE VISITED ONLY ONCE AND WE FIND THE SHORTEST PATH OR ELSE MINIMUM BUDGET PATH. BUT HERE THE CASE IS NOT VERY SIMILAR. HERE THE ROBOTIC DRILL STARTS FROM A BASE NODE WHERE IT USUALLY CHANGED THE SIZE OF ITS PIN. IT THEN TRAVERSES THROUGH VARIOUS NODES. IN ANY CASE THE STARTING NODE AND ENDING NODE OF THE DRILL IS THE BASE NODE. NOW WHEN WE LOOK DEEPER INTO THE PROBLEM AND UNDERSTAND IT, THERE ARE SEVERAL NODES IN THE I.C(INTEGRATED CIRCUIT). FOR EXAMPLE LET US CONSIDER THERE ARE 6 NODES IN AN I.C. FEW ARE OF SAME SIZE. NOW WE GROUP ALL THE NODES OF SAME SIZE INTO A CATEGORY.



WE FIGURE OUT THE TRAVERSAL SEQUENCE OF THESE CATEGORIES OF SIZES. THAT IS WHICH SIZED NODES ARE TO BE DRILLED FIRST AND FOLLOWED BY WHICH SIZE OF NODES IN A ORDER.THROUGH THIS WE BASICALLY REDUCE THE TIME OF DRILLING OVERALL AS THE PIN CHANGED IS ONLY ONCE FOR EACH SIZE OF NODE. THIS HELPS US TO COMPLETE DRILLING THE I.C IN A SHORT SPAN OF TIME IN AN EFFICIENT MANNER.



AS IN THE ABOVE FIGURE (TAKEN AS AN EXAMPLE) OUR STARTING NODE WILL BE BASE NODE AND THEN IT WILL FINISH ALL THE NODES OF SIZE CATEGORY N1 AND THEN IT GOES BACK TO BASE SINCE IT NEED TO CHANGE THE SIZE OF PIN FOR NEXT CATEGORY OF NODES. THE SAME PROCESS OCCURS UNTILL IT FINISHES ALL SET OF NODES AND AT LAST IT VISITS BACK TO THE BASE NODE. WHEN WE COMPARE OUR CASE WITH THE TRADITIONAL TSP IT IS ALMOST SIMILAR EXCEPT THAT THERE IN TRADITIONAL TSP ONLY STARTING NODE IS VISITED TWICE (SINCE IT IS THE ENDING NODE) WHEREAS HERE THE BASE NODE IS VISITED MULTIPLE TIMES. THERE WE FIND THE SHORTEST INTERMEDIATE PATH WHICH IS VERY SIMILAR TO THE OBJECTIVE OF THIS PROJECT BUT HERE WE CONSIDER THE TIME OF DRILLING ALSO ALONG WITH THE TRAVERSAL TIME. NOW AFTER WE SORT OUT THE PRIMARY SIZE TRAVERSAL SEQUENCE WE NEED TO SOLVE THE SUB-PROBLEM OF TRAVERSING BETWEEN THE NODES OF SAME SIZES IN CATEGORY OF SIZE. NOW INORDER TO DO THIS WE FIRST NEED TO KNOW THE INTER DISTANCES BETWEEN EVERYNODE. THIS DATA IS OBTAINED BY TAKING THE ENTIRE INTER DISTANCES IN AN ADJACENCY MATRIX FORMAT.

(NEXT PAGE CONTAINS AN EXAMPLE INPUT ADJACENCY MATRIX)

node 1 :

$$\begin{bmatrix} 0 & 4 & 1 & 5 \\ 4 & 0 & 2 & 3 \\ 1 & 2 & 0 & 2 \\ 5 & 3 & 2 & 0 \end{bmatrix}$$

node 2 :

$$\begin{bmatrix} 0 & 8 & 9 & 3 \\ 8 & 0 & 5 & 1 \\ 9 & 5 & 0 & 2 \\ 3 & 1 & 2 & 0 \end{bmatrix}$$

AS WE OBSERVE THIS IS A SYMMETRICAL MATRIX SINCE THE DISTANCE FROM ANY NODE TO ITSELF IS ZERO. ALSO, DISTANCE TRAVERSED FROM ONE NODE TO ANOTHER AND THE REVERSE IS SAME OR EXACTLY EQUAL. HERE ZERO REPRESENTS THE BASE NODE. TAKING THE MINIMUM DISTANCE AS A CONSTRAINT WE SORT OUT THE TRAVERSAL BETWEEN NODES. HENCE NOW WE HAVE OBTAINED TRAVERSAL SEQUANCE OF OUR NODES.THE SAME PROCESS IS APPLIE TO EVERY SIZE CATEGORY OF NODES. AT THE END WE ACHIEVE OUR OBJECTIVE OF TRAVERSAL OF NODES IN AN EFFICIENT MANNER.

TIME COMPLEXITY

SINCE WE ARE SOLVING THIS USING DYNAMIC PROGRAMING THERE ARE SUB PROBLEMS IN IT.

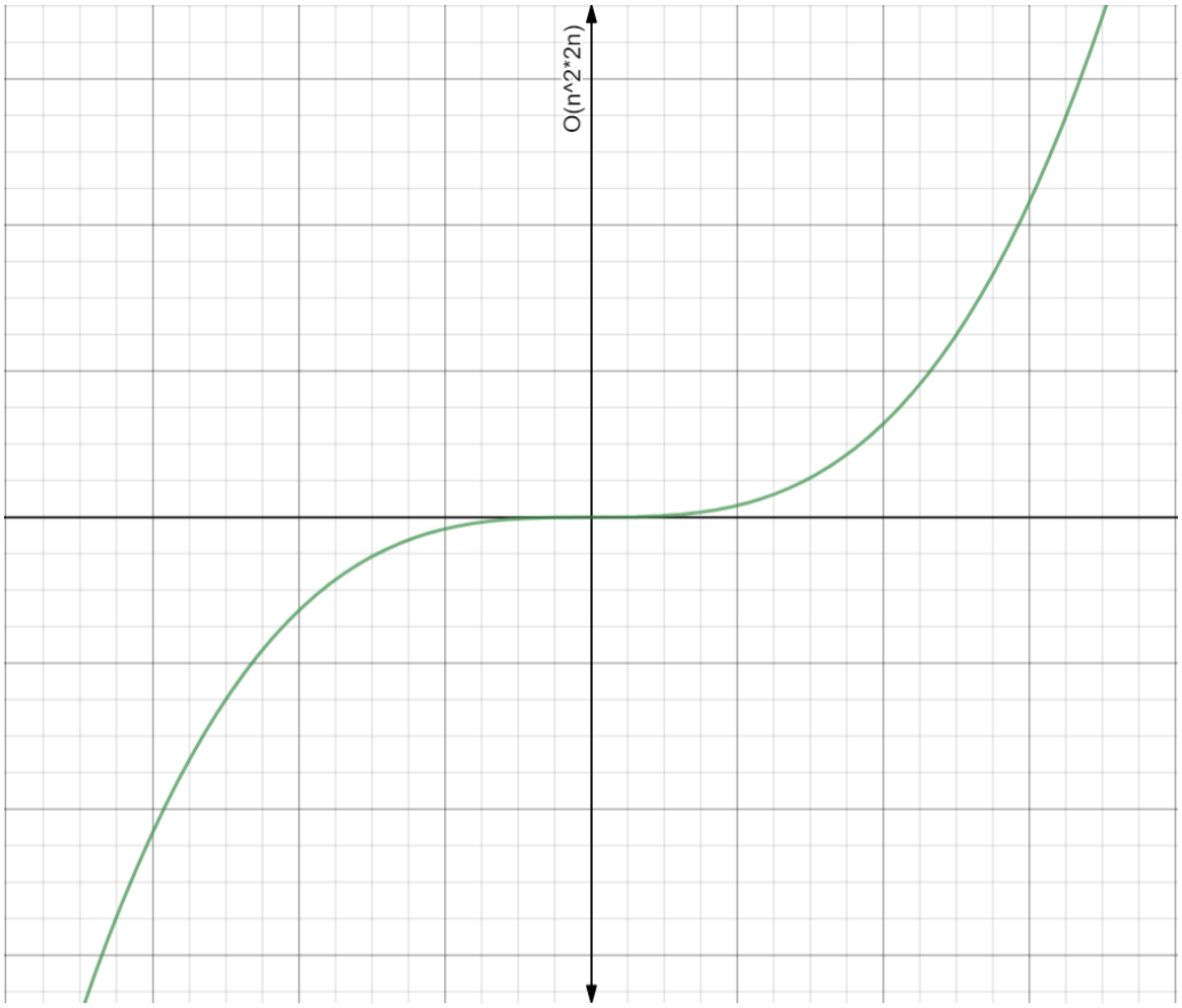
SUB PROBLEM- $O(n*2^n)$

TOTAL TIME COMPLEXITY: $O(n^2*2^n) * O(k)$

IT IS REPRESENTED GRAPHICALLY AS BELOW

Y AXIS REPRESENTS THE TIME COMPLEXITY

X AXIS REPRESENTS THE GROWTH OF n.



IMPLEMENTATION CODE

```
#include<iostream>

using namespace std;

int visited[100][100],n[100],nsize,min_distance=0;

int h=0;

void input(int mat[100][100][100])

{

   int i,j,k;

   cout<<"Enter The Adjacency Matrix(Give the first node distance from the base where pin changes)\n";

   for(k=0;k<nsize;k++)

   {

      for(i=0;i<n[k]+1;i++)

      {

         cout<<"Enter the Distance of each hole from the other considering first node as base of node size\t"<<k+1<<"\n";

         for(j=0;j<n[k]+1;j++)

         {

            cin>>mat[k][i][j];

            visited[k][i]=0;

         }

      }

   }

}

int minimum(int mat[100][100][100],int l,int x)

{

   int i,nc=999;

   int mini=999,kmini;

   for(i=0;i<n[h];i++)

   {
```

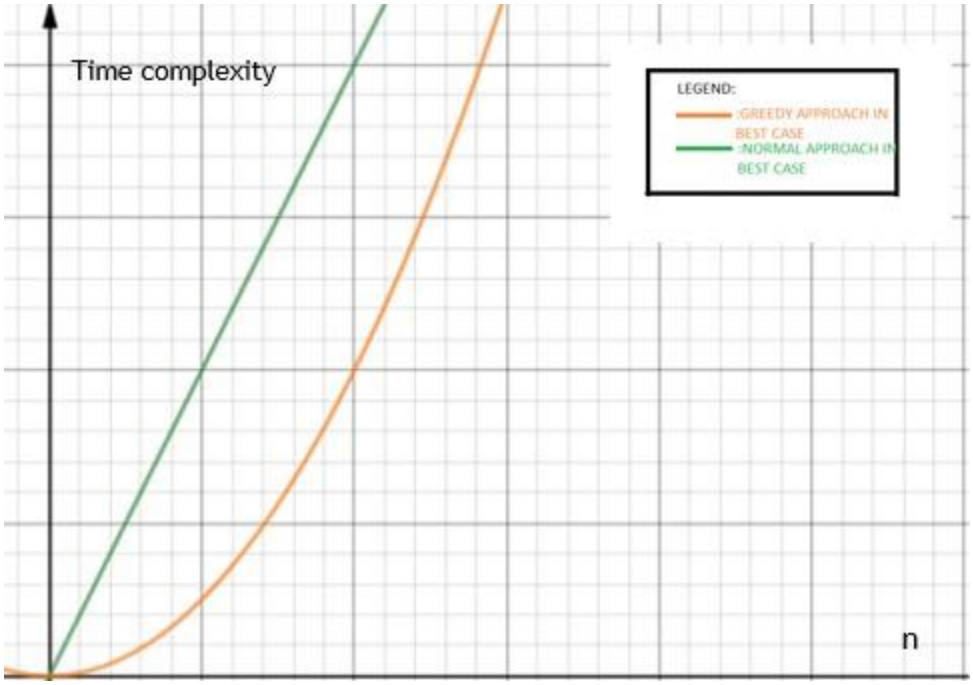
```
        if((mat[x][l][i]!=0) && (visited[x][i]==0))
          {
             if(mat[x][l][i]+mat[x][i][l]<mini)
              {
                 mini=mat[x][i][0]+mat[x][l][i];
                 kmini=mat[x][l][i];
                 nc=i;
              }
          }
    }
    if(mini!=999)
       min_distance+=kmini;
    return nc;

}
void path(int mat[100][100][100],int l,int x)
{
    int i,dist;
    visited[x][l]=1;
    cout<<x<<"."<<l<<"---->";
    dist=minimum(mat,l,x);
    if(dist==999)
    {
       dist=0;
       cout<<dist<<"\n";
       min_distance+=mat[x][l][dist];
       h++;
       return;
```

```
    }
    path(mat,dist,x);
}
int main()
{
   int matrix[100][100][100],i;
   cout<<"Enter the no. of sizes\n";
   cin>>nsize;
   for(i=0;i<nsize;i++)
   {
       cout<<"Enter the no. of nodes in size \t"<<i+1<<"\n";
       cin>>n[i];
   }
   input(matrix);
   for(i=0;i<nsize;i++)
   {
      cout<<"the path for node size\t "<<i+1<<"\n";
      path(matrix,0,i);
   }
   cout<<"The distance is\t"<<min_distance;
   return 0;
}
```

```
Enter the no. of sizes
2
Enter the no. of nodes in size(including the base node) 1
4
Enter the no. of nodes in size(including the base node) 2
4
Enter The Adjacency Matrix(Give the first node distance from the base where pin changes)
Enter the Distance of each hole from the other considering first node as base of node size      1
0
4
1
5
4
0
2
3
1
2
0
2
5
3
2
0
Enter the Distance of each hole from the other considering first node as base of node size      2
0
8
9
3
8
0
5
1
9
5
0
2
3
1
2
0
the path for node size   1
1.0---->1.2---->1.3---->1.1---->0
the path for node size   2
2.0---->2.3---->2.2---->2.1---->0
The distance is 28
Exit code: 0 (normal program termination)
```

IN CASE OF GREEDY APROACH THE ALGORITHM DIRECTLY GETS THE INPUT FROM THE USER. IT STARTS FROM THE NODE WHICH IS MOST NEAREST TO BASE NODE. AFTER REACHING THERE AGAIN IT REACHES TO ITS NEXT NEAREST NODE. LIKE WISE IT KEEPS ON MOVING FROM ONE NODE TO THE OTHER BY CONSIDERING THE NEAREST ONE. THIS LOOKS LIKE AN EFFICIENT ALGORITTHM BUT WHEN WE SEE THE RESULT IT IS DEFINITELY NOT BEST ONE SINCE THERE MIGHT HAVE SHORTER TRAVERSALS POSSIBLE AT AN OVERALL LEVEL EVEN THOUGH IT HAD NOT TRAVERSED TO ITS NEXT NEAREST NODE AT ANY POINT IN BETWEEN THE TRAVERSAL. SINCE WE ALWAYS COMPARE THE FINAL RESULT WE MAY ELIMANTE GREEDY APPROACH BUT, ANY WAY IN CASE OF BEST CASE SOME TIMES IT IS POSSIBLE THAT GREEDY APPROACH GIVES THE BEST RESULT. FOR EXAMPLE, CONSIDER THE NODES THAT ARE ALLIGNED IN A LINEAR LINE. IN THIS CASE GREEDY APPROACH OUTPERFORMS OUR NORMAL APPRAOCH.



HERE THE **ORANGE** GRAPH REPRESENTS THE TIME COMPLEXITY OF GREEDY APPROACH IN BEST CASE SCENARIO

GREEDY APPRAOCH: $O(n^2)$

THE GREEN GRAPH REPRESENTS THE TIME COMPLEXITY OF NORMAL APPROACH IN BEST CASE SCENARIO

NORMAL APPROACH: $O(2n)$

CONCLUSION:

STILL THE ALGORITHM CAN BE MUCH DEVOLOPED BY SIMPLYING THE PROGRAM. THIS CAN BE DONE BY MAKING IT MORE DYNAMIC. BY SPLITTING THE PROBLEM INTO MUCH SIMPLER SUB PROBLEMS WILL HELP US TO SOLVE IT MUCH MORE EFFICIENTLY. CONSIDERING THE BEST-CASE SCENARIO, WE HAVE TO WORK ON THE ALGORITHM TO MAKE IT MORE EFFICIENT IN COMPARISON WITH GREEDY APPROACH. WHEN WE GO DEEP INTO LEARNING OF TSP WE FIND THAT MST CAN BE USED TO DRAW AN EFFICIENT TRAVERSAL RESULT. IT MAY NOT BE A GOOD RESULT AS A TRADITIONAL ALGORITHM GIVES BUT ITS VALUE (TOTAL DISTANCE OR BUDGET) IS NOT MORE THAN TWICE THAT OF WHAT A TRADITIONAL ALGORITHM GIVES US. NOT ONLY MST, WE CAN DEVOLOP SEVERAL TECHNIQUES THAT COULD GIVE AN EFFICIENT RESULT AS CLOSE AS TO THE BEST RESULT IN ALL POSSIBLE CASES.

REFERENCES:

Solution of a Large-Scale Traveling-Salesman Problem BY George B. Dantzig, Delbert R. Fulkerson, and Selmer M. Johnson

Finding Tours In TSP By David Applagete, Robert Bixby, Vasek Chvatel

www.geeksforgeeks.com

www.stackoverflow.com

www.researchgate.com