A New Approximation Algorithm for Project Scheduling Problem

Bapita Roy
Department of Computer Application
Techno India College of Technology
Kolkata, India
bapitaroy14@gmail.com

Neha Modi
Department of Computer Application
Future Institute of Engineering & Management
Kolkata, India
neha.m77@gmail.com

Ramandeep Singh
Department of Computer Application
Techno India College of Technology
Kolkata, India
ramandeep.saini007@gmail.com

Suparna Konar
Department of Computer Application
Techno India College of Technology
Kolkata, India
suparna.konar7@gmail.com

***Abstract*— we have developed a project scheduling algorithm for better estimation of project in an incomplete information environment. It has been well approximated so that the generalised NP-hardness get relaxed. A case study example has been described for the claim of our scheme.**

***Keywords*— project scheduling, approximated, NP-hardness**

I. Introduction

This approximation algorithm is one of the tools in theory and practice of the project scheduling problem. We begin our exposition with the Max-Cut problem i.e. the problem of computing an edge cut with the maximum possible number of edges in a given graph.
This approximation algorithm is implemented using the idea of Max-Cut problem and gives impressive results in this area.

II. Problem Formulation

We have identified the following project scheduling problem as stated below:

Let us consider a network as a flow network in which a unit of flow enters at the origin node1 and exits at the terminal node n. The duration d_{ij} of an activity (i, j) in an AOA () network can then be interpreted as the time to traverse the arc from node i to node j. The intermediate node acts as transshipment centers. The task is to determine a path from source to sink which takes the longest time to traverse.

A. Semi Definite Problem Formulation & Relaxation

Let G = (N, E) be an undirected graph, which is the network, w_{ij} (i.e, d_{ij}) be weights of an edge (i,j) $\in$ E. $\forall$ k $\subset$ N. Let ∂(k) denote {{i,j}:i $\in$ k, j $\notin$ k} and w(∂(k))=

$$\sum_{(i,j)\in\partial(k)} w_{ij}\,.$$

So we define SDP relaxation as

$$C^{*} = \max\left[\frac{\left[\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(1-x_{ij})\right]}{4}\right]$$

$$s.t.\ x_{ii} = 1\ (i \in N), x \geq 0$$

B. Max Cut Problem Definition

Max Cut is the following computational problem.
We are given a graph G=(V,E) as the input , and we want to find a partition of the vertex set into two subsets , S and its compliment V\S, such that the number of edges going between S and V\S is maximized.We define a cut in a graph G= (V, E) as a pair (S, V\S), where S $\subseteq$ V. The edge set of the cut (S, V\S) is

$$E(S,V\setminus S)=\{e\in E: |e\cap S| = |e\cap(V\setminus S)| = 1\}\ ,$$

And the size of this cut is $|E(S,V\setminus S)|$ i.e., the number of edges. We also say that the cut is induced by S. Garey et al. [ ] has proved the following regarding the Max Cut problem:

Theorem 1: The decision version of the Max Cut problem is NP-complete.

The optimization version of the Max Cut problem is consequently NP-hard.

C. SDP MAX CUT Problem Formulation

Let G = (N, E) be an undirected graph i.e, (network) where N = {1 …n} Assume have edge-weights w = (w_{ij}) $\in R_{+}^{E}$. (Which is d_{ij}), for k $\subseteq$ N, ∂(k): = {i, j $\in$ E: i $\in$ k,j $\notin$ k}. So, the MAX-CUT problem

$$\max_{k\subseteq N} \sum_{i,j\in\partial(k)} w_{ij}$$

This can be formulated as

$$\max \ \frac{1}{4}\sum_{i,j=1}^{n} w_{ij}(1-x_i x_j)$$

$$\text{s.t.} \quad x_i \in \{-1,1\}, \ i \in N$$

Here putting $X = xx^T$ can be formulated as

$$\max \ \frac{1}{4}\sum_{i,j=1}^{n} w_{ij}(1-x_{ij})$$

$$s.t. \ x_{ii} \in \{-1,1\}, \ i \in N$$

D. GW Max Cut Problem Formulation

Let us consider a graph for approximating MAX CUT is denoted as $G=(V_i : i \in \{1,\dots,n\}, E)$. Let a set of almost optimal solution $u_1^*, u_2^*, \dots, u_n^*$ of the vector program

$$Maximize \sum_{(i,j)\in E} \frac{1-u_i^T u_j}{2}$$

$$s.t. \quad u_i \in S^{n-1}, i-1,2,\dots,n$$

That is a solution that satisfies

$$\sum_{(i,j)\in E} \frac{1-u_i^{*T} u_j^*}{2} \geq SDP(G) - 5*10^{-10} \geq Opt(G) - 5*10^{-4}$$

E. Project Network Scheduling Problem formulated as Geomanns-Willamson max Cut Problem

$Def^{\underline{n}}$: Almost Optimal Solution

In an approximation algorithm almost optimal solution is just as good as truly optimal solution. Under this convention, an optimal solution of a semi definite program or a vector program is the solution i.e accurate enough in the given context.

F. Semi Definite Relaxation for Max Cut Problem

We have formulated the Max Cut problem as a constraint optimization problem. Let us consider the graph G= (V, E), where v=1… n. Let the variables $z_1, z_2, \dots, z_n \in \{-1,1\}$ corresponding to any values from {-1, 1} to this variables encodes a cut (S, V\S) , where $S=\{i \in V : z_i = 1\}$. Then the edge e_{ij} is transformed into the term $\left[\frac{1-z_i z_j}{2}\right]$.

If e_{ij} is not a cut edge, then $z_i z_j = 1$ and the value of the above term is zero. If e_{ij} is a cut edge, then $z_i z_j = -1$ and the value of the above term is one. Therefore the Max Cut problem can be formulated as

$$\left.\begin{aligned} &Maximize \sum_{(i,j)\in E} \frac{1-z_i z_j}{2} \\ &s.t. \quad z_i \in \{-1,1\}, i=1,\dots,n \end{aligned}\right\} \dots\dots\dots(1)$$

As the Max Cut is NP-complete problem, we cannot solve this optimization problem exactly in polynomial time.

Let us consider the optimal value is Opt (G). We replace each real variable $\mathbf{u_i} \in S^{n-1} = \{X \in R^n : \|X\| = 1\}$, the (n-1) dimensional unit sphere. So we get

$$\left.\begin{aligned} &Maximize \sum_{(i,j)\in E} \frac{1-u_i^T u_j}{2} \\ &s.t. \quad u_i \in S^{n-1}, i=1,2,\dots,n \end{aligned}\right\} \dots\dots\dots(2)$$

This is called a vector program since the unknowns are vectors.

The above consideration (2) gives a relaxation to formulation (1) with relaxed set of solutions and it therefore has value at least Opt (G).

To solve the formulation (1) , where the n variables $\mathbf{z_i}$ are elements of S° = {-1, 1} and thus determine a cut (S, V\S) via $S = \{i \in V : z_i = 1\}$, we have an almost optimal solution of relaxed version of formulation (2) where n vector variables are elements of S^{n-1}. Now we map S^{n-1} back to S° by choosing p$\in S^{n-1}$ by considering the mapping

$$u \rightarrow \begin{cases} 1, & is \ p^T u \geq 0 \\ -1, & otherwise \end{cases} \quad \dots\dots\dots(3)$$

III. Proposed Algorithm

A. Schematic flow diagram of proposed scheme

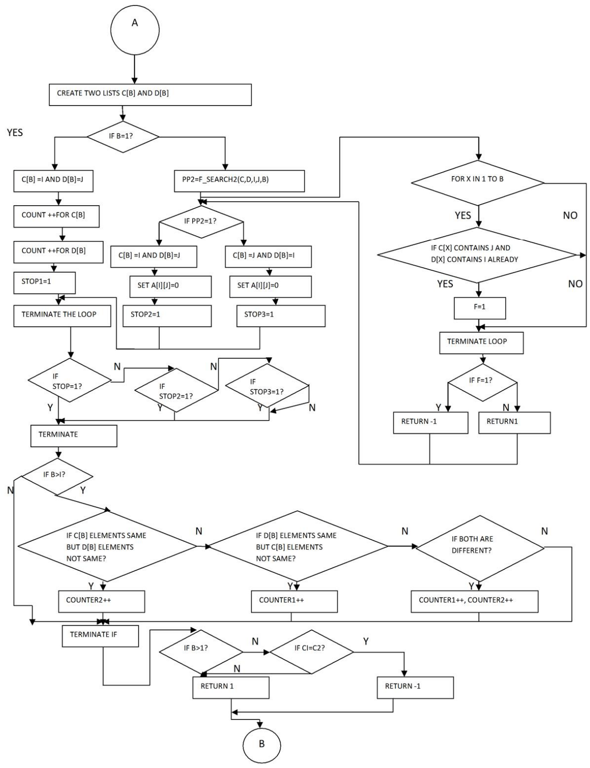


Fig. 1 F_{SEARCH} Function Flow Chart

B. Proposed Scheme

```
START
INPUT NODES
INPUT EDGES
INPUT WEIGHTS

FOR I = 1 TO NUMBER OF NODES
    FLAG=0
    FOR J=1 TO NUMBER OF NODES
        IF[ (WEIGHT OF J)<(WEIGHT OF J+1 )]
            THEN  SWAP WEIGHT OF J AND WEIGHT OF J+1
            FLAG=1
        END IF
    END FOR
        IF FLAG=0 THEN BREAK
END FOR

INITIALISE         [SUM       AS          ZERO]
FOR I =1 TO N
     INCREASE B
     EL= WEIGHT OF I
     P= FSEARCH(A,N.EL,B)
     IF P=1 THEN
          ADD WEIGHT OF I WITH EXISTING SUM
      BREAK
      ELSE
          ADD WEIGHT OF I WITH EXISTING SUM
    END IF

DEFINING FSEARCH(A,N,EL,B)
FOR I=1 TO N
    FOR J=1 TO N
          IF EL=A[I][J] THEN
                       C[B]=I
                       D[B]=J
          END IF
    END FOR
END FOR
IF B>1 THEN
    IF( C[B]=C[B-1] AND D[B] != D[B-1])
          THEN   INCREASE COUNTER1
    END IF
    IF (C[B]!=C[B-1] AND D[B]=D[B-1] )
          THEN   INCREASE COUNTER2
    END IF
    ELSE
        INCREASE COUNTER1 AND COUNTER2
   END IF
   IF B>1
          THEN  IF COUNTER1=COUNTER2 THEN
   RETURN -1
   ELSE RETURN 1
   END IF
   END IF
          SHOW THE SUM
END
```

Fig. 2 the Algorithm

IV. CASE STUDY

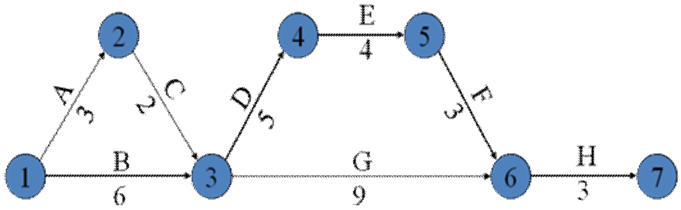
PROBLEM:

Let us consider the following network information related to a project. Estimate the project completion time.

Activity	Immediate Predecessor	Time (weeks)
A	-	3
B	-	6
C	A	2
D	B,C	5
E	D	4
F	E	3
G	B,C	9
H	F,G	3

Using Critical Path Method (CPM)

- **The PERT/CPM network:**



- **Calculate ES, LS, EF and LF for each activity:**

Activity	ES	LS	EF	LF	Slack	Critical?
A	0	1	3	4	1	
B	0	0	6	6	0	Yes
C	3	4	5	6	1	
D	6	6	11	11	0	Yes
E	11	11	15	15	0	Yes
F	15	15	18	18	0	Yes
G	6	9	15	18	3	
H	18	18	21	21	0	Yes

- **The activities that make up the critical path:**

Critical path activities: B -> D -> E -> F -> H = 21

Using proposed algorithm:

- **Arranging the weight according to descending order:**

Activity	Immediate Predecessor	Time (weeks)
G	B,C	9
B	-	6
D	B,C	5
E	D	4
A	-	3
F	E	3
H	F,G	3
C	A	2

- **Solving using proposed algorithm**

List 1	Weight	List 2
3	—— 9 ——▶	6
3	◀—— 6 ——	1
3	—— 5 ——▶	4
5	◀—— 4 ——	4
2	◀—— 3 ——	1
5	—— 3 ——▶	6
7	◀—— 3 ——	6

Total weight:- 33 [i.e. > CPM (21)]

Therefore, our proposed algorithm gives us greater completion time and with less time complexity.

V. CONCLUSIONS

We have worked on this problem of Project Management Scheduling and proposed the algorithm to reduce the complexity from O (n^2) to O (n). We really hope that it will replace the previous algorithms and will make its own stand.

This is an ongoing project and we're going to submit it as our final year project.

VI. REFERENCES

[1] Goemans. M.X. Williamson. D.p: improved approximation algorithms for max-cut and satisfiability problems using semidefinite programming. J . Assoc. Comput. Mach. 42(6).1115-1145(1995).

[2] A report on approximate graph coloring by semidefinite programming. By Pingke Li, Zhe Liu.

[3] Garey, Michael R.; Johnson, David S. (1979), Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, ISBN 0-7167-1045-5.
Maximum cut (decision version) ND16 in Appendix A2.2.
Maximum bipartite subgraph (decision version) is the problem GT25 in Appendix A1.2.