ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 132

2nd International Conference on Modelling, Simulation and Applied Mathematics (MSAM 2017)

Semi-Auto CPM Network Generation

Via Activities Predecessor & Duration Parsing

Chir-Ho Chang* and Ro-Yu Wu

300, Sec. 1, Wanshou, Rd, Guishan District, Taoyuan City 33306, Taiwan, R.O.C

*Corresponding author

***Abstract*— In order to notice a critical path method network working in a project, it is of crucial importance to visualize the network first. However, a project manager often needs to draw this kind of graph manually. In this paper, we develop a semi-auto network formation simulated system that reads the activities' relationships and durations and outputs a graph. The system generates the graphical version of the network from an project's activities table.**

Keywords-component; critial path method; semi-auto network generation

I. INTRODUCTION AND MOTIVATION

This paper deals with a fundamental issue: How to transform a relational table in CPM into a CPM network effectively and efficiently [1]. In contrast to the traditional approach, we almost always have to draw such tree graph by hand. Even, we use commercial software such as Microsoft Project, the barrage is still there. A manager has to decide the type of relation linkages. (i.e. FS, FF, SS, SF) under a friendly GUI of the software. It is quite inconvenient if you already have an activity, predecessor, and duration table. Why not transform the table directly into network table.

II. LITERATURE REVIEW

Critical Path Method is network-based method designed to assist in the planning, scheduling, and control of projects. It has been used widely [5]. According to H.A. TAHA [9], a project is defined as a collection of interrelated activities with each activity consuming time and resources. Right after defining a domain of a project, you collect a set of activities. But translate these activities into a network we need a relational description table that clearly states the sequences. We stick to H.A. TAHA and follow the mentioned three rules: Each activity is represented by one and only one arrow in the network. 2) Each activity must be identified by two distinct end nodes. And, 3) To maintain the correct precedence relationships.

III. AN ILLUSTRATIVE EXAMPLE

TABLE I. A STANDARD CPM RELATIONAL TABLE

ID	Activity	Predecessor	Duration
A	$XXXX_1$	-	K_1
B	$XXXX_2$	-	K_2
…	…	…	…
Z	$XXXX_4$	B	K_4

The table has four columns. The first index column indicates an ID of an activity. Table symbolizes the ID from Some of them might appear in the 3rd column later. The second column is the name of the activity. If an activity does not have a predecessor, a dash - symbol will be used in the 3rd column. The fourth column of the table is the time needed to finish an activity where the K with subscripts are all integers.

IV. DESIGN METHODOLOGY

I would like to divide the design process into four stages: 1) data structure, 2) file reading, 3) predecessor processing, and 4) Other operations.

A. Data Structure

The structure as describe in TABLE I. General representation of a single activity is depicted as follows,

Column No.1 #Column No.2#Column No.3#Column No.4

Column No.1: JOB ID
Column No.2: JOB's Name
Column No. 3: Previous JOB
Column No.4: Duration

They were connected by a pound sign. A useful MATLAB command to read such an activity is issued by the textread command:

[id,cntx,prev,dur]=textread('jobs.txt','%s%s%s%n','delimiter','#'). Some of the explanation for this command is stated in detail as follows.

1) The left four fields correspond to the output field after a file reading command is issued. Because there are four kind of outputs, the output must have four corresponding variables.
2) Jobs.txt is the name of the project's prime data file, remember to use single quotes to quote this name up.
3) '%s%s%s%n' Is the data type of the contents. For example，%s is a char data type or is called string and %n is a numeric data type.
4) 'Delimiter' is the key word for the separator.
5) '#' is used as a separator and is a constant

```
A#Product development and design#-#5
B#Market research and planning#-#12
C#Assured prototype#A#6
D#Marketing slides and manuals preparation#A#4
E#Cost estimation#C#5
F#Pilot test for product#D#9
G#Marketing investigation and report#B,E#17
H#Product pricing and sales forecast#H#8
I#Fianl report#F,G,I#6
```

FIGURE I. A SNAPSHOT ON TRANSFORMATION STAGE NO. 1

B. *Outputs after Reading a File*

The first output is the identification of the activity. And the second output variable, name of the activities, is listed in Figure II and is shown as follows,

```
'A'  'B'  'C'  'D'  'E'  'F'  'G'  'H'  'I'  'J'
```

FIGURE II. ID TRANSFORMATION STAGE

```
'Product development and design'
'Market research and planning'
'Assured prototype'
'Marketing slides and manuals preparation'
'Cost estimation'
'Pilot test for product'
'Marketing investigation and report'
'Product pricing and sales forecast'
'Fianl report'
```

FIGURE III. ACTIVITY NAME TRANSFORMATION

```
'-'  '-'  'A'  'A'  'A'  'C'  'D'  'B,E'  'H'  'F,G,I'
```

FIGURE IV. PREDECESSOR TRANSFORMATION

```
5  12  6  8  4  5  9  17  8  6
```

FIGURE V. DURATION TRANSFORMATION

C. *Decomposition of the Front-end Task*

1) *Step No.1: Seperation*

Considering the data structure of the original file, this study uses a comma to separate the ID symbol in the predecessor column. As a result, the first step in the end task is to do the separation.

2) *Step No.2: Looping process*

We use '#' as the first-layered separator and ',' as the second-layered separator. Since there might more than a comma in the predecessor temporary string, we need a loop command embedded in the decoding sub-system.

3) *Step No.3: Isolation*

Computer graphics, the computer does not know to perform several times

D. *Three Possible Ouputs*

Type I:'-' ➔n=0, s={}

Type II:Single character➔n=1, s={'A Symbolic Letter'}

Type III:Multiple Characters ➔n=the sum of predecessors' numbers, s={an array of all ID symbols that are predecessors}

To simplify the system design for the future, we use a cell array to record the corresponding ID symbols.

```
function ans=explode(prvs)
p=findstr(prvs,'-');
if(~isempty(p))
   ans=[];
else
   x=findstr(prvs,',');
   if(isempty(x))
      ans=prvs;
   else
      l=length(prvs);
      alx=1:l;
      p=setdiff(alx,x);
      ans=prvs(p);
   end
end
```

FIGURE VI. CODES THAT DISSECTS THE TEMPORARY STRING

E. *Node and an Arc Preparation*

The first and second arguments are entered at the beginning of the entry point for this drawing function, and the end point is determined by the length and the angle that a program determined. The third input argument is the length of the connection, and the fourth argument is the count of successors. The program optimizes the rendering of graphics by the number of inheritors. The last argument of the decision is processed in lines 21 to 24, with solid lines and dashed lines. The following function is described in two parts: Part I control the angles and part II deal with nodes and arcs.

```
%brachesinx:1-6
function [minx,maxx,miny,maxy]=genseg(startx,starty,
                                      len,branches,inx,scripttxt,b)
```

```
if(branches==1)
   theta=0;
elseif(branches==2)
   theta=[30,-30]*pi/180;
elseif(branches==3)
   theta=[45,0,-45]*pi/180;
elseif(branches==4)
   theta=[45,30,-30,-45]*pi/180;
elseif(branches==5)
   theta=[40,20,0,-20,40]*pi/180;
else
   theta=[50,30,15,-15,-30,-50]*pi/180;
end
```

FIGURE VII. CODES THAT DEALS ANGLES IN BRANCHES

```
stopx=startx+len*cos(theta(inx));
stopy=starty+len*sin(theta(inx));
drawnode(startx,starty);
drawnode(stopx,stopy);
xg=line([startx,stopx],[starty,stopy]);
set(xg,'LineWidth',3);
if(b==1)
   set(xg,'LineStyle','--');
end
xh=text(startx+(stopx-startx)/2-0.5,starty+(stopy-starty)/2,scripttxt);
set(xh,'FontSize',24);
minx=min([startx,stopx])-2;
maxx=max([startx,stopx])+2;
miny=min([starty,stopy])-2;
maxy=max([starty,stopy])+2;
axis([minx,maxx,miny,maxy]);
```

FIGURE VIII. SCRIPTS THAT HANDLES GRAPHS

The computer does not have the ability to know whether he should stay at the position that is assigned by the first two argument of the planned source point. Therefore, the program does not move the source when a line has been drawn. On the other hand, it will search to find the point with the same origin by using a search program shown in Figure 9.

```
function ans=searchroot(char, prevx)
a='ABCDEFGHIJ';
cc=[];
len=length(prevx);
for i=1:len
    p=findstr(prevx{i}, char);
    if(~isempty(p))
       cc=[cc, a(i)];
    end
end
ans=cc;
```

FIGURE IX. SEARCH FOR THE SAME ORIGN

F. Codes of an Initial plot

```
function ans=initplot(prev, dur)
clf;
x=searchroot('-', prev);
lx=length(x);
minx=1000;
miny=1000;
maxx=-1000;
maxy=-1000;
for i=1:lx
    strtemp=[x(i),'-', num2str(dur(i))];
    [tmpminx, tmpmaxx, tmpminy, tmpmaxy]=genseg(0, 0, dur(i), lx, i, strtemp, 0);
    if(minx>tmpminx)
          minx=tmpminx;
    end
    if(maxx<tmpmaxx)
          maxx=tmpmaxx;
    end

    if(miny>tmpminy)
          miny=tmpminy;
    end
    if(maxy<tmpmaxy)
          maxy=tmpmaxy;
    end
    hold on;
end
axis([minx, maxx, miny, maxy]);
```

FIGURE X. THE CODE FOR THE INITIAL PLOT

G. Merage Process and a Whole Plot

At each round, we check whenever a new leaf is grown from a new node. If the new end node is a finished node, one has to merge the node. As an indication, we can find the duplication in the taboo array. For the nodes have been emerged then a dotted line is connected to the starting point.

```
function ans=wholeplot(prev, dur)
clf;
poolc=['0'];
poolxy000=[0, 0];
poolxyEND=[0, 0];
[gggx, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, '-', 0, 0, 1000, -1000, 1000, -1000, poolc, poolxy000, poolxyEND);

[ggga, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, 'A', gggx(1, 1), gggx(1, 2), minx, maxx, miny, maxy, poolc, poolxy000, poolxyEND);
[gggb, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, 'B', gggx(2, 1), gggx(2, 2), minx, maxx, miny, maxy, poolc, poolxy000, poolxyEND);

[gggc, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, 'C', ggga(1, 1), ggga(1, 2), minx, maxx, miny, maxy, poolc, poolxy000, poolxyEND);
[gggd, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, 'D', ggga(2, 1), ggga(2, 2), minx, maxx, miny, maxy, poolc, poolxy000, poolxyEND);
[ggge, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, 'E', ggga(3, 1), ggga(3, 2), minx, maxx, miny, maxy, poolc, poolxy000, poolxyEND);

[gggh, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, 'H', gggb(1, 1), gggb(1, 2), minx, maxx, miny, maxy, poolc, poolxy000, poolxyEND);

[gggg, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, 'G', gggd(1, 1), gggd(1, 2), minx, maxx, miny, maxy, poolc, poolxy000, poolxyEND);
[gggi, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, 'I', gggh(1, 1), gggh(1, 2), minx, maxx, miny, maxy, poolc, poolxy000, poolxyEND);
[gggf, poolc, poolxy000, poolxyEND, minx, maxx, miny, maxy]=
initplot(prev, dur, 'F', gggc(1, 1), gggc(1, 2), minx, maxx, miny, maxy, poolc, poolxy000, poolxyEND);
axis([minx, maxx, miny, maxy]);
```

FIGURE XI. THE CODE FOR THE INITIAL PLOT

V. EXAMPLES

A. Example No.1

http://www2.kimep.kz/bcb/omis/our_courses/is4201/Chap14.pdf

TABLE II. RELATIONAL TABLE

ACTIVITY	DESCRIPTION	IMMEDIATE PREDECESSORS
A	Select Office Site	—
B	Create Organizational and Financial Plan	—
C	Determine Personnel Requirements	B
D	Design Facility	A, C
E	Construct Interior	D
F	Select Personnel to Move	C
G	Hire New Employees	F
H	Move Records, Key Personnel, etc.	F
I	Make Financial Arrangements with Institutions in Des Moines	B
J	Train New Personnel	H, E, G

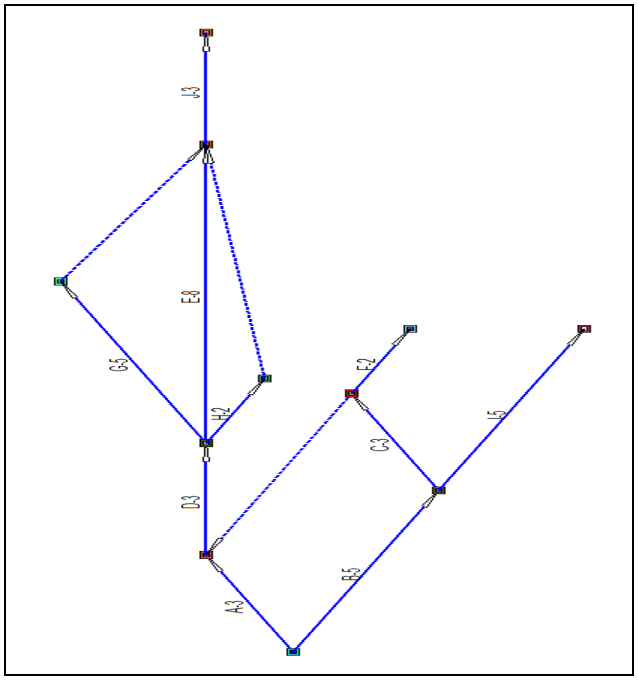

FIGURE XII. THE GENERATED CPM NETWORK FOR EXAMPLE NO.1

B. *Example No. 2*

TABLE III. RELATIONAL TABLE

ID	Activity	Predecessor(s)	Duration(days)
A	Clear site	-	1
B	Bring utilities to site	-	2
C	Excavate	A	1
D	Pour foundation	C	2
E	Outside plumbing	B, C	6
F	Frame house	D	10
G	Do electric wiring	F	3
H	Lay floor	G	1
I	Lay roof	F	1
J	Inside plumbing	E,H	5
K	Shingling	I	2
L	Outside sheathing insulation	F,J	1
M	Install windows and doors	F	2
N	Do brick work	L,M	4
O	Insulate walls and ceiling	G,J	2
P	Cover walls and ceiling	O	2
Q	Insulate roof	I,P	1
R	Finish interior	P	7
S	Finish exterior	I,N	7
T	Landscape	S	3

Reprinted from HAMDY A. TAHA OPERATIONS RESEARCH AN INTRODUCTION pp.266

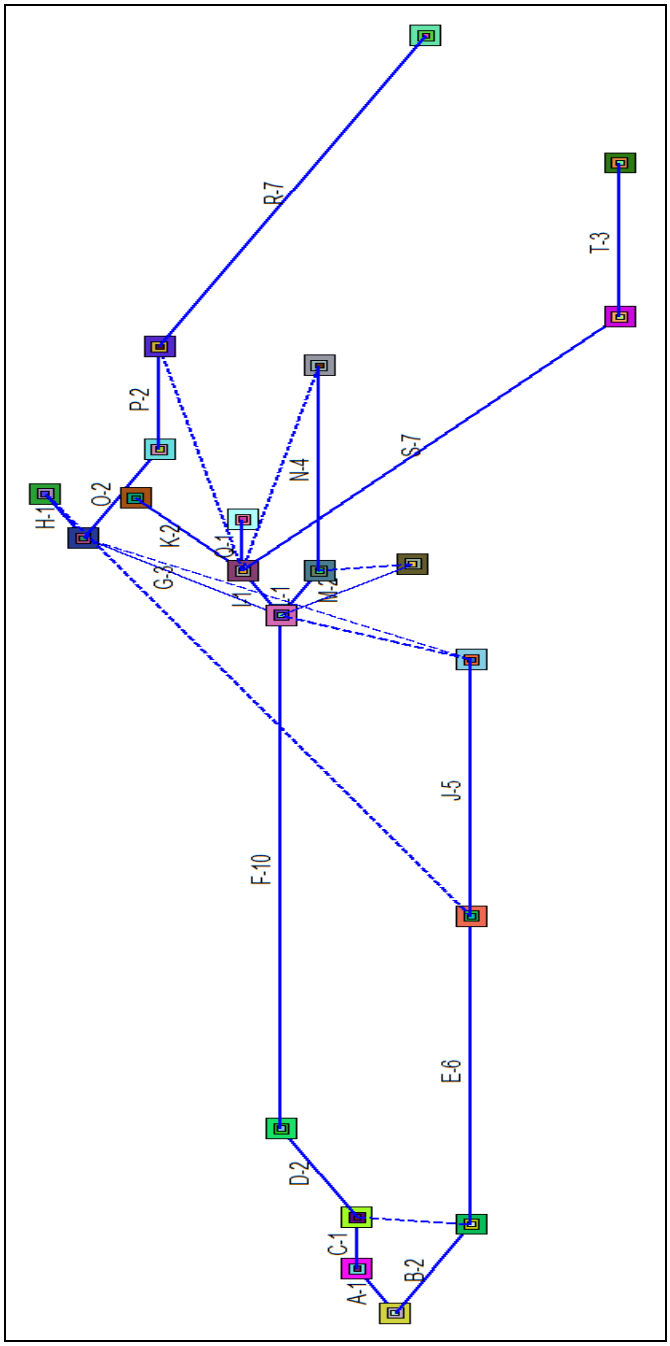

FIGURE XIII. THE GENERATED CPM NETWORK FOR EXAMPLE NO.2

VI. SUMMARY AND CONCLUSIONS

The critical path method can be of great importance to a project manager. But the formation of a project's network is traditionally difficult and can only be finished by hand. Though some commercial tools have been developed, one still have to provide the relations' information by dragging mouse. This work basically tried to develop a homemade version of network diagram for CPM. Since CPM network-generation is a labor-intensive task. This study uses a plain

list searching algorithm and smart branching techniques to increase the efficiency of a network-drawing. Authors developed several functions to help a manager to draw a CPM network: From a node to a line and from a line to a tree. Programs and functions are MATLAB scripts. The final plot was generated from these scripts on a personal computer. Two examples randomly picked from the internet and a popular textbook are used to demonstrate the effectiveness of our approach. The contribution of this study is significant in the sense that it activates the second stage of our research: To come up with a full-automatic version. It took around 20 minutes to complete a network-drawing in the past. For now, our system spent 5% to 50% of the time only construct of the same network.

There are two directions currently are under research. First of all, we found that when the number of activities increase, our system is more likely to come up with a twisted version of network [4]. Since complexity could fail our system eventually, some innovative programming scheme should be used [2, 3]. Future works also include the building of a full-automatic system. Which has the engine of writing the scripts shown in Figure 11 when provides a project precedence table to the future system.

ACKNOWLEDGMENT

The research grant of this research comes from “college students participate in the NSC program in Taiwan in 2015. Authors would like to express their thankfulness.

REFERENCES

[1] Borgatti, Stephen P., and Martin G. Everett. "Network analysis of 2-mode data." Social networks 19.3 (1997): 243-269.

[2] Batagelj, Vladimir, and Andrej Mrvar. "Pajek-program for large network analysis." Connections 21.2 (1998): 47-57.

[3] De, Prabuddha, et al. "Complexity of the discrete time-cost tradeoff problem for project networks." Operations research 45.2 (1997): 302-306.

[4] Grewal, Rajdeep, Gary L. Lilien, and Girish Mallapragada. "Location, location, location: How network embeddedness affects project success in open source systems." Management Science 52.7 (2006): 1043-1056.

[5] O'brien, James J. CPM in construction management. 1993.

[6] Ragsdale, C. "The current state of network simulation in project management theory and practice." Omega 17.1 (1989): 21-25.

[7] Shaffer, Louis Richard, J. B. Ritter, and W. Lyle Meyer. The critical-path method. McGraw-Hill, 1965.

[8] Wiest, Jerome D. "Precedence diagramming method: Some unusual characteristics and their implications for project managers." Journal of Operations management 1.3 (1981): 121-130.

[9] TAHA, Hamid A. Operations Research. 6TH Edition, 1997