FLOYD-WARSHALL AGAIN

Ph.D. Zoltán A.Vattai
Budapest University of Technology and Economics, Faculty of Architecture
zvattai@ekt.bme.hu

Abstract

It is about four and half decades since Warshall and (later) Floyd had published their much cited algorithms for to detect any loops (Warshall, 1959) or to calculete all-pairs Shortest Pathes (Floyd, 1962) in a (directed, weighted) graph structure. The basic idea being a triviality - that if both P[i,k] and P[k,j] pathes exist, than path P[i,j] also exists - gives us possibilities to perform all-pairs analyses to get manyfold information about the graph structure on the basis of tracking operative (adjacency) information systematically node-by-node (k) along the graph. The article shows that with some slight modification within the original „Floyd-Warshall Algorithm" and with some complementary terms a lot of information can be gained about the logical (or technical or social etc.) structure modelled by the graph that can be of much use for managerial decision support in fields of Construction Management too.

Keywords: operations research, graphs, computer applications

Introduction

Implying its Computing Science origin the so called Floyd-Warshall algorithm is frequently cited as tripartite loop routines organized around a single core of simple calculations derived from a proper representation („structure" or „adjacency" matrix) of the graph considered. See: *Figure 1* and *Figure 2*.

Figure 1. – Initial graph representation – written in Delphi / Turbo Pascal

```
var
   n : integer;                          { number of nodes of the Graph }
   i,j,k : integer;                      { running indices/identifyers of nodes (rows/columns) }
   D:  array [1..n,1..n] of integer;     { „distance" matrix of the Graph }
   P:   array [1..n,1..n] of integer;    { labels/pointers for tracking the shortest pathes }

Procedure Initialization;
   begin
      for i:=1 to n do
         for j:=1 to n do
            begin
               D[i,j]:=Wij;
               if D[j,j]=M then P[i,j]:=0 else P[i,j]:=i
            end
   end;

{ „M" refers to a properly chosen „marker" value indicating „no_connection". At Floyd M=„+∞" }
{ „Wij" refers to weights of Graph edges. Wij=M if no edge from i to j }
```

Figure 2. – Calculating all-pairs (shortest) distances – written in Delphi / Turbo Pascal

```
Procedure Calculating_Distances;
  begin
    for k:=1 to n do                          { outer loop – stepping „node by node" }
      for i:=1 to n do                        { inner loops – tabular calculations }
        for j:=1 to n do
          if D[i,k]+D[k,j]<D[i,j] then begin  { the core – testing „connections" }
            D[i,j]:=D[i,k]+D[k,j];            { – modification if needed }
            P[i,j]:=P[k,j]                    { – labelling if modified }
          end
  end;
```

Though it is rarely declared (or less evident) main idea of Floyd-Warshall routines is based on a simple triviality. Namely: considering a connected directed graph, if a path exists from node i to node k and also a path exists from node k to node j, then consequently a path do exists from node i to node j (at least the one via node k). In this context we do refer node k as a transfer node on a path from node i to node j.

Extending the above triviality it can be stated that in case of any graph there exist path from all the nodes from which path leads to a selected transfer node k to all the nodes to which path leads from the mentioned node k. It means that testing all the nodes of the graph as transfer nodes one by one we can gain certainty of existence of any and all the pathes throughout the entire graph. It is also easy to see that tracking nodes this way all existing pathes consisting of at least two edges on the graph get be considered once and only once. With information about the single edges in the other hand (read in the initial structure matrix) we can conclude that all-pairs analyses of the graph can be kept in hand this way. The question is what kinds of information can be gained about (better said by the aid of) the graph that can be calculated by separable functions, separated cycle by cycle at transfer nodes selected respectively.

Formulation

For to discuss the question above instead of using any computer syntax we do introduce the following denotions:

G[N,E]	: refers to a graph (G) having the set of nodes (N) and the set of edges (E)
G[N,E,*W*]	: refers to a graph as above but having weights (*W*) on edges respectively
ij	: refers to the edge from node i to node j on the graph
M	: marker value in tabular representation of graph, reads „no connection"
$\underline{\underline{W}}$	: initial tabular representation (structure/adjacency matrix) of the graph
w_{ij}	: element of $\underline{\underline{W}}$ referring to the „weight" of the edge from node i to node j
n	: number of nodes of the graph
k	: index of outer cycle, also refers to transfer node actually selected
$\underline{\underline{A}}^k$	: transformed matrix representation of the graph in cycle k
$a_{ij}{}^k$	: element of $\underline{\underline{A}}^k$ referring to ij pair of nodes (connection from i to j)
a_i^k	: row vector i of matrix $\underline{\underline{A}}^k$
$\acute{a}_i^k$	: column vector i of matrix $\underline{\underline{A}}^k$
i,j	: running indices of nodes, also referring to rows and columns of matrices

Using denotions above general routines of selecting nodes as transfer nodes one by one (outer cycle) and of testing connections (inner cycles) and of performing necessary modifications if any (core) can be formulated as shown in *Figure 3*.

Figure 3. – Formulation of general routines

$\underline{\underline{A}}^{0} = \underline{\underline{W}}$ { initialization }

$\underline{\underline{A}}^{k} = \Phi(\underline{\underline{A}}^{k-1}), \quad k = 1,2,\ldots,n$ { outer cycle }

Inner calculations of matrix-transformation function Φ: { inner cycles and core }

$$a_{ij}^{k} = \begin{cases} \varphi(a_{ik}^{k-1}, a_{kj}^{k-1}, a_{ij}^{k-1}) \mid i \neq k \quad j \neq k \quad a_{ik}^{k-1} \neq M \quad a_{kj}^{k-1} \neq M \\ a_{ij}^{k-1} \quad \text{otherwise} \end{cases} \quad \forall\, ij$$

where $\varphi(a_{ik}^{k-1}, a_{kj}^{k-1}, a_{ij}^{k-1})$ refers to a properly selected trhee-variable function (core)

Calculations at problems discussed below are differing in selection of M (marker value), in initial representation of the garph (W) and in function $\varphi(a_{ik}^{k-1}, a_{kj}^{k-1}, a_{ij}^{k-1})$ (core). Special features are introduced at the individual problems below.

For to evaluate convergence of the algorithms it can be stated that initializations are done in n^2 steps, while overall matrix transformations (calculating $\underline{\underline{A}}^{n}$) need n^3 steps. However calculations are rarely over when getting any $\underline{\underline{A}}^{n}$ matrix of the graph. Further evaluations of results are usually necessary to gain any useful information just like for managerial purposes too.

Problems discussed below can be (are) referred in Operation Research as:

- Connectivity Analysis
- Finding Loops (Warshall, 1959)
- Finding Dominant and/or Dominated Sub-Graph
- All-Pairs Path Counting
- All-Pairs Shortest Path (Floyd, 1962), Gravity-Points, Center-Points, Diagonals
- All-Pairs Longest Path, Minimal Potentials, Scheduling (Network Techniques)

Connectivity Analysis

By definition a graph is considered to be connected if path exists between all pairs of nodes. The interpretation is evident in case of non-directed (symmetric) graphs, but it needs some re-interpretation for directed graphs, while weighted or non-weighted characteristics of the graph has no importance. As a general definition of connected graphs it can be stated that a graph is considered to be connected if **on its symmetric version** (all edges made non-directed or „two-directional") path exists between all pairs of nodes. This way connectivity is considered regardless of directions of edges that is regardless of the input or output „side" of the phenomenon modelled by the graph.

According to later definition using denotions introduced above initinal definition of structure matrix can be formulated as:

$$M = 0 \qquad w_{ij} = \left\{ \begin{array}{ll} 1 \mid ij \in \mathrm{E} \text{ or } ji \in \mathrm{E} & \\ M \quad \text{otherwise} & \end{array} \right\} \ \forall\, ij$$

Core funcion to be used is a simple constant function:

$$\varphi(a_{ik}^{k-1}, a_{kj}^{k-1}, a_{ij}^{k-1}) = 1$$

During calculations (in cycles of matrix transformations) practically we do complete path-existance information registered in $\underline{\underline{A}}^k$ matrices with newer discoveries revealing gradually during testing nodes one by one as transfer nodes. 1 values in cells of final $\underline{\underline{A}}^n$ matrix do indicate existence of any path between the pairs of nodes respectively.

In case of a connected graph all elements of $\underline{\underline{A}}^n$ matrix equal to 1. ($a_{ij}^n = 1 \quad \forall\, ij$)

In case of a not connected graph nodes of individual connected sub-graphs can be identified by considering row vectors of $\underline{\underline{A}}^n$. Node i and node j belong to the same connected sub-graph if $a_i^n = a_j^n$. The graph contains as many (connected) sub-graphs as many different a_i^n row vectors are in matrix $\underline{\underline{A}}^n$.

Connectivity Analysis can promote identifying problem fields that can be managed in no relation with others (individually) when modelling processes of complex production systems. It also can help to check reason-result mechanisms and to set boundaries and interfaces when developing controlling-monitoring scenarios. We may say in general Connectivity Analysis can be of use in case of looking for components more or less tightly related (or absolutely not related) in logical structures represented by graphs.

Finding Loops

The problem of finding loops (cycles) on graph structures is quite similar to the problem of connectivity analysis with the main difference that it can be characteristic problem of directed (not symmetric) graphs. The aim of calculations is to find nodes of the graph if any connected (via chain of edges through other nodes) to themselves.

We do keep directions of edges so definition of structure matrix can be formulated as:

$$M = 0 \qquad w_{ij} = \left\{ \begin{array}{ll} 1 \mid ij \in \mathrm{E} & \\ M \quad \text{otherwise} & \end{array} \right\} \ \forall\, ij$$

Core funcion can be either the same constant function used at Connectivity Analysis:

$$\varphi(a_{ik}^{k-1}, a_{kj}^{k-1}, a_{ij}^{k-1}) = 1 \qquad (\text{ Compare with Warshall ! })$$

Nodes connected to themselves if any are identified by $a_{ii}^{n} \neq M$ values in the diagonal of matrix $\underline{\underline{A}}^{n}$ indicating that „path" (better said „chain of edges") exist from given nodes to the same ones. We do refer these nodes as „loop-nodes" or „nodes-on-loop".

For to indentify edges of any loop (we refer them as „loop-edges" or „edges-on-loop") we have to consider the initial ($\underline{\underline{A}}^{0}$) matrix and the last one ($\underline{\underline{A}}^{n}$) together. Edges ($a_{ij}^{0}=1$) on loop can be identified by testing three necessary conditions: 1., originating point of the edge is a loop-node ($a_{ii}^{n} \neq M$); 2., ending point of the edge is a loop-node ($a_{jj}^{n} \neq M$); 3., path exists between the two nodes connected by the edge in reverse too ($a_{ji}^{n} \neq M$).

Though warning on existence of loop(s) on the graph even more identification of loop-nodes and loop-edges are of great help many times, knowing them is rarely enough. We may consider an accidentally false mistyping at entering precedence restrictions of a CPM network resulting in huge bulk of nodes and edges got be on loop. We would badly welcome some compass for finding data to be checked or rethought.

A slight help can be gained by assigning weights (e_{ij}) to loop-edges on the basis of estimated number of loops they are contributing in having the recognition that edges originating from the same node or ending in the same node are surely not components of the same loop. For to promote it we may isolate loop-edges in matrix $\underline{\underline{A}}^{n+1}$ using the algorithm below.

$$a_{ij}^{n+1} = \begin{cases} 1 \quad | \quad a_{ij}^{0}=1 \quad a_{ii}^{n} \neq M \quad a_{jj}^{n} \neq M \quad a_{ji}^{n} \neq M \\ 0 \quad \text{otherwise} \end{cases} \quad \forall\, ij$$

For weighting loop-edges ($a_{ij}^{n+1}=1$) we can state that number of loops (e_{ij}) an edge can contribute in is surely not less than number of loop-edges ending in its originating node ($\Sigma_j\, a_{ji}^{n+1}$) times number of loop-edges originating from its ending node ($\Sigma_i\, a_{ji}^{n+1}$).

$$e_{ij} \geq (\Sigma_j\, a_{ji}^{n+1}) \cdot (\Sigma_i\, a_{ji}^{n+1})$$

Having all loop-edges weighted we can focus our attention either to the environment of loop-edges of the higher weight-values.

Here we remark that information set in the initial $\underline{\underline{A}}^{0}$ matrix (edges of the graph) can be transmitted to matrix $\underline{\underline{A}}^{n}$ by changing the previously used constant function in the core to a kind of binary one.

$$\varphi(a_{ik}^{k-1}, a_{kj}^{k-1}, a_{ij}^{k-1}) = \max\{a_{ij}^{k-1}, 2\text{-}a_{ij}^{k-1}\}$$

Using this later function $a_{ij}^{n}=1$ values in matrix $\underline{\underline{A}}^{n}$ represent edges of the graph while $a_{ij}^{n}=2$ values are indicating that between the two nodes referred there exists connection via pathes consisting of at least two edges. This way we do not need to refer back to the initial $\underline{\underline{A}}^{0}$ matrix when isolating loop-edges (that is we do not have to keep it in the memory of the computer – in case of extensive graphs) hence $a_{ij}^{0}=1$ values can be read in matrix $\underline{\underline{A}}^{n}$ ($a_{ij}^{n}=1$) too.

Finding and eliminating loops (inconsistent restrictions) is typical problem at Network-techniques Aided Scheduling e.g. PERT, CPM, MPM, PDM, etc. where precedence restrictions are set in form of a directed graph.

Dominant/Dominated Sub-Graph

Problem of dominant and/or dominated sub-graph(s) has close relations with problems of loops. For initial representation of the graph ($\underline{\underline{W}}$) and for preparing matrix $\underline{\underline{A}}^n$ we may use the same algorithms (with constant function in the core) used at Finding Loops. The difference is in reading the results.

By definition: Node *i* of a graph is considered as „dominant node" if path exists from it to all other nodes of the graph. ($\Sigma_j a_{ij}^n - a_{ii}^n = n\text{-}1$)

Similarly: Node *j* is considered as „dominated node" if path exists to it from all other nodes of the graph. ($\Sigma_i a_{ij}^n - a_{jj}^n = n\text{-}1$)

Using terms of dominant/dominated (Do substitute the proper term!) nodes and sub-graphs some trivialities can be stated either as theorems:

- A directed graph may have at most one dominant/dominated sub-graph. The dominant/dominated sub-graph as a corporate unit acts like source/sink (origin/termin) in the direction of the rest of the graph.
- Nodes and edges of a dominant/dominated sub-graph consisting of more than one nodes are surely loop-nodes and loop-edges forming one single connected loop-system.
- There is no relation between existence of a dominant and that of a dominated sub-graph. A directed graph may have dominant sub-graph but no dominated one, and reverse.

Problem of dominant/dominated components may get high significance at evaluating working (technical and/or social) environments or dynamic systems. It can be explained briefly saying (e.g.): „If a dominant component goes wrong in a system then all other components will surely go wrong". Similarly: „If any component goes wrong in the system then all dominated components will surely go wrong". Similar way the most motivating group of a society (or of a company's staff) can be identified just like the least motivated or passive one of no any workable initiation („Staff-Mapping").

All-Pairs Path Counting

In case of directed garphs with no any loop we may earn information not only about existence of any path but about number of different pathes between any pair of nodes using a slightly modified algorithm described below. For to define initial representation of the graph ($\underline{\underline{W}}$) we may use the same routine described above at Finding Loops while for to prepare matrix $\underline{\underline{A}}^n$ we use the core function of:

$$\varphi(a_{ik}^{k-1}, a_{kj}^{k-1}, a_{ij}^{k-1}) = a_{ij}^{k-1} + a_{ik}^{k-1} \cdot a_{kj}^{k-1}$$

(Existence of any loop – that is any $a_{ii}^{n} \neq \mathrm{M}$ value in diagonal of matrix $\underline{\underline{\mathrm{A}}}^{n}$ – invalidates the results.)

Elements of resulting $\underline{\underline{\mathrm{A}}}^{n}$ matrix are the numbers of different pathes (differing in at least one node and in at least two edges) between respected pairs of nodes. The more interest it may earn anyhow is not the information collected in matrix $\underline{\underline{\mathrm{A}}}^{n}$ but is the information can be read by the aid of it. Based on a_{ij}^{n} values of matrix $\underline{\underline{\mathrm{A}}}^{n}$ a single weight (r_{st}^{ij}) or a whole matrix ($\underline{\underline{\mathrm{R}}}^{ij}$) reading the number of different pathes an edge (ij) is incorporated in between the nodes referred (s,t) can be calculated.

$$r_{st}^{ij} = \begin{cases} 1 & | \quad s = i \quad t = j \\ a_{jt}^{n} & | \quad s = i \quad t \neq j \\ a_{si}^{n} & | \quad s \neq i \quad t = j \\ a_{si}^{n} \cdot a_{jt}^{n} & \text{otherwise} \end{cases} \qquad \forall\, st \quad s \in \mathrm{N} \quad t \in \mathrm{N} \quad ij \in \mathrm{E}$$

Similar way, a single weight (r_{ij}^{st}) or whole matrix ($\underline{\underline{\mathrm{R}}}^{st}$) reading the number of different pathes the edges referred (ij) are inconporated in between a selected pair of nodes (s,t) can be calculated.

$$r_{ij}^{st} = \begin{cases} 1 & | \quad i = s \quad j = t \\ a_{jt}^{n} & | \quad i = s \quad j \neq t \\ a_{si}^{n} & | \quad i \neq s \quad j = t \\ a_{si}^{n} \cdot a_{jt}^{n} & \text{otherwise} \end{cases} \qquad \forall\, ij \quad ij \in \mathrm{E} \quad s \in \mathrm{N} \quad t \in \mathrm{N}$$

Number of pathes (r_i) a node (i) is incorporated in on the graph can be calculated via sums of values in column i ($\Sigma_j\, a_{ji}^{n}$) and in row i ($\Sigma_j\, a_{ij}^{n}$) of matrix $\underline{\underline{\mathrm{A}}}^{n}$.

$$r_i = \max\{\Sigma_j\, a_{ji}^{n}\,,\ \Sigma_j\, a_{ij}^{n}\,,\ (\Sigma_j\, a_{ji}^{n}) \cdot (\Sigma_j\, a_{ij}^{n})\} \qquad \forall\, i$$

Values of this kind related to components (to nodes and/or to edges) of the graph can quantify effect-range of them as numbers of pathes (better said: „of effect-chains") they are effected by and/or effecting on around the graph. Considering results of above calculations we may drive our attention to most effective (of highest weights) or to least effective (inessential/avoidable) components of the dynamic system modelled by the graph – depending on our actual interests.

All-Pairs Shortest Path

Having all pathes discovered by testing nodes as transfer nodes the possibility for to select one or more specific ones such as all-pairs shortest pathes (on a weighted graph, having no any negative loop) is also given. At initial representation of the graph we do keep the weights of edges while for marker value we do choose „positive infinity".

$$M = +\infty \qquad\qquad w_{ij} = \begin{cases} w_{ij} & | \quad ij \in \mathrm{E} \\ M & \text{otherwise} \end{cases} \qquad \forall\, ij$$

Core function to be used at preparing matrix $\underline{\underline{A}}^n$ reading the lengths of all-pairs shortest pathes („distance matrix") is:

$$\varphi(a_{ik}^{k-1}, a_{kj}^{k-1}, a_{ij}^{k-1}) = \min\{a_{ij}^{k-1}, a_{ik}^{k-1}+a_{kj}^{k-1}\} \qquad \text{(Compare with Floyd !)}$$

Elements of resulting $\underline{\underline{A}}^n$ matrix are lengths of the shortest pathes (distances) between respected pairs of nodes. (Existence of any negative loop – that is any $a_{ii}^n<0$ value in the diagonal of matrix $\underline{\underline{A}}^n$ – invalidates the results.)

Considering $\underline{\underline{A}}^0$ and $\underline{\underline{A}}^n$ matrices together path(es) of length read in respecting cell of matrix $\underline{\underline{A}}^n$ between any two nodes can be pursued. An edge (*sk*) is the first or the only edge of a path of given length (a_{st}^n) between two nodes (*s* and *t*) if the length (weight) of the edge (a_{sk}^0) and the length of the rest of the path (without that edge) (a_{kt}^n) adds the length of the path (a_{st}^n) considered.

$$a_{st}^n = a_{sk}^0 + a_{kt}^n \qquad \text{or} \qquad a_{st}^n = a_{st}^0$$

Starting from originating node of the path to direction of the terminal one edges of the path can be identified one-by-one on a „forward-pass". This way length of pathes and of edges are used as some kind of „labels". (Compare with labelling at Floyd!)

Distance information read in matrix $\underline{\underline{A}}^n$ may earn interest in logistics, in designing basic version (basic architecture) of „vario" products, in developing market policies etc..

Gravity-point of a weighted graph is the node where the sum of shortest pathes „to" (input side) or the sum of shortest pathes „from" (output side) other nodes of the graph is the minimum. Questions to be answered may sound like:

„Where to locate a Crop-Store to have the overall transportation cost (or time) be the minimum?" (input side)

$$j = ? \qquad \Sigma_i a_{ij}^n - a_{jj}^n \Rightarrow \min$$

„Where to locate the Dealers' Central Store to have the overall transportation cost (or time) be the minimum?" (output side)

$$i = ? \qquad \Sigma_j a_{ij}^n - a_{ii}^n \Rightarrow \min$$

Center-point of a weighted graph is the node where the longest of the shortest pathes „to" (input side) or „from" (output side) other nodes of the graph is the minimum. Question to be answered may sound like:

„Where to locate Emergency Center (Fire Department or Ambulance) to reach even the furthest lot the center is responsible for in the shortest time?" (input and/or output side)

$$j = ? \quad \max_i a_{ij}^n \Rightarrow \min \qquad \text{and/or} \qquad i = ? \quad \max_j a_{ij}^n \Rightarrow \min$$

Diagonal of the graph – by definition – is the longest of the (all-pairs) shortest pathes around the graph. Its length (*D*) can be read in matrix $\underline{\underline{A}}^n$ directly.

$$D = \max_{ij} a_{ij}^n$$

In its special way the Diagonal quantifies spread or integrity of the system modelled. It may function as a „measure" of transfer/response capability of dynamic systems like communication and/or alarm systems, „neural networks", etc..

All-Pairs Longest Path

Algorithms for to calculate $\underline{\underline{A}}^n$ matrix reading all-pairs longest pathes are differing from routines used for calculating all-pairs shortest pathes in „sign" only:

$$M = -\infty \qquad w_{ij} = \left\{ \begin{array}{ll} w_{ij} & | \quad ij \in E \\ & \\ M & \text{otherwise} \end{array} \right\} \quad \forall\, ij$$

$$\varphi(a_{ik}^{k-1}, a_{kj}^{k-1}, a_{ij}^{k-1}) = \max\{a_{ij}^{k-1}, a_{ik}^{k-1} + a_{kj}^{k-1}\}$$

This case positive loops are to be excluded from the graph. (Existence of any positive loop – that is any $a_{ii}^n > 0$ value in the diagonal of matrix $\underline{\underline{A}}^n$ – invalidates the results.)

Considering $\underline{\underline{A}}^0$ and $\underline{\underline{A}}^n$ matrices together path(es) of length read in respecting cell of matrix $\underline{\underline{A}}^n$ between any two nodes can be also pursued the same way it has been done at all-pairs shortest pathes.

Benefits of having lengths of all-pairs longest pathes known can be highly appreciated at scheduling problems where precedence restrictions of components are set in form of a directed weighted graph. „Overall execution time" (Π) of the project modelled and „deadlines" (time-potentials) such as „earliest times" (π_j) and „latest times" (π'_i) of „events" (represented by nodes of the graph) can be read in matrix $\underline{\underline{A}}^n$ almost directly.

$$\Pi = \max_{ij} a_{ij}^n$$

$$\pi_j = \max\{0, \max_i a_{ij}^n\} \qquad \forall\, j$$

$$\pi'_i = \min\{\Pi, \min_j(\Pi - a_{ij}^n)\} \qquad \forall\, i$$

Nodes of the overall longest path (better said: of „Dominant Sub-Graph") known as „Critical Path" can be recognized by checking condition if the earliest and latest times equal to each other ($\pi_i = \pi'_i$), while edges (between critical nodes) of it can be recognized by checking if difference of time potentials at ending (π_j) and at originating (π_i) nodes of the edge equals to the weight (w_{ij}) of it.

$$\pi_i = \pi'_i \qquad \pi_j = \pi'_j \qquad \pi_j - \pi_i = w_{ij} \qquad ij \in E$$

All-Pairs Longest Path algorithm may aid scheduling problems with no expressed starting (origin) and/or no ending (termin) point („open networks") and with upper and lower bounds either among precedence conditions or among durations. Approaching scheduling problems this way it is the dual of „Minimal Potentials" problem being

exposed while much of restrictions on structures of weighted graphs usually set at Scheduling Network Problems (PERT, CPM, MPM/PDM, etc.) can be released.

Research Background

The need for overviewing all-pairs connections in a logical structure had been generated by an R&D joint project within frames of cooperation of Hungarian Railways Company (MÁV) and of Budapest University of Technology and Management (BME), 1989-93, aiming developement of a Computer Aided Decision Support System for planning and managing reconstructional works and maintenance of hungarian railways' system.

Challenge of management problem was the „Permanent Scheduling" of works on a 3-years slipping time-span looking over thousands of jobs with accuracy of minutes. No expressed start, no expressed end, widely diversed responsibilities, dispersed locations across the country, but one complex „must-be-operating" (under traffic) railway system and a restricted common pool of some significant specialized resource series.

Traditional Scheduling techniques (including traditional Network Techniques) prooved to be insufficient. „The project to be scheduled was not a project." A bunch of results of research cooperation was the development of a software system named „FITT" based on a network-like modular scheduling technique some elements of which have been discussed as „GTM" (General Time Model) in curricula of elective subjects of Department of Construction Technology and Management (ÉKT) of BME since nineties of last Century. …

Members of R&D group were:

- Tibor Vígh, MÁV KBPI
- Károly Bacher[+], BME ÉKT
- Dr. József Monori[+], BME ÉKT
- Dr. László Neszmélyi, BME ÉKT
- Zoltán A.Vattai, BME ÉKT

References

1. Frank S. Budnick, Dennis McLeavy, Richard Mojena, *Principles of Operations Research for Management*, IRWIN Homewood, Illinois, 1988
2. Robert W. Floyd, *Algorithm 97: Shortest Path*, Communications of the ACM 5(6): 345, June 1962
3. Frederick S. Hillier, Gerald J. Lieberman, *Introduction to Operation Research*, Holden Day Inc., 1986
4. James E. Kelley Jr., Morgan R. Walker, *Critical-path Planning and Scheduling*, Proceedings of Eastern Joint Computer Conference, p. 160-173, Boston MA, December 1959
5. Joseph J. Moder, Cecil R. Phillips, Edward W. Davis, *Project Management with CPM, PERT and Precedence Diagramming*, Van Nostrand Reinhold, New York, 1983

6. Albert D. Polimeni, H. Joseph Straight, *Foundations of finite mathematics*, Brooks/Cole Publishing Company, Monterey, California, 1985
7. B. Roy, *Les problémes d' ordonnancement*, Dunod, Paris, 1964
8. B. Roy and B. Sussmann, *Les problémes d' ordonnancement avec contraintes disjonctives*, Note DS no 9. bis, SEMA, Montrouge, 1964
9. Dr. Z. A. Vattai, *FITT - Vágányzárban végzett időkorlátos technológiák tervezése*, Építéstechnológia - építési menedzsment konferencia, Számítástechnika: tudományos és gyakorlati alkalmazások az építőiparban szekció, Szabadka, 1997
10. S. Warshall, *A Theorem on Boolean Matrices*, Journal of the ACM, Vol 9, 1, 11-12, 1962
11. Gary E. Whitehouse*, System Analysis and Design Using Network techniques*, Prentice-Hall Inc., Englewood Cliffs, New Yersey, 1973

APPENDIX I. – GLOSSARY

Adjacency Matrix - tabular representation of a graph having values in cells reading existences of edges. (See also: Structure Matrix)
CPM – Critical Path Method (J. Kelley 1957)
Diagonal of a squared matrix – set of cells with first and second index the same
Directed Edge – an edge having relation between the nodes related interpreted in one direction only. Usual graphical representation of a directed edge is an arrow from the „preceding" node (at tail) to the „succeding" one (at head). Remark: Any non-directed edge can be substituted by a pair of directed edges with opposing directions. In this context a non-directed edge works like a loop (See: Loop)
Directed Graph – a graph having all edges directed (See remark at Directed edge)
Edge – related pair of nodes
Graph – structured set (graphics) of nodes (vertices/circles) and edges (arcs/arrows)
Length of a path/loop – sum of weights of edges of the path/loop
Loop – self-closing flow (string/chain) of directed edges. (Inaccurately it is frequently explained as „self-closing path")
MPM – Metra Motentials' Method (B. Roy 1960)
Negative Loop – a loop having negative length
Path – connected flow (string/chain) of directed edges. Any path is identified by list of nodes it incorporates sequentially. In a path's „list of nodes" no repretition is allowed.
PDM – Precedence Diagramming Method (IBM 1972)
PERT – Program Evaluation and Review Technique (USA, Polaris, 1958)
Positive Loop – a loop having positive length
Squared Matrix – a matrix having the number of rows and of columns the same
Structure Matrix – tabular representation of a graph where cells are representing edges having the interpretation of „where from (row) - where to (column)". Cells of it are usually referred by indices (first index referring to row, second index referring to column). Values in cells are „weigths" of edges respectively and/or some marking value reading „no edge between the nodes referred in direction referred".
Symmetric Graph – a graph the structure matrix of which is symmetric („non-directed graph").
Weight – quantitative characteristic („parameter") assigned to edge
Weighted Graph – a graph having weights assigned to all edges