SYSTEMS ANALYSIS LECTURE 4 STRUCTURAL TASKS 1

Zuzana Bělinová

System structural tasks

- Several types of tasks
- Aiming to analyze
  - Capabilities of system‘s structure
  - Possibilities and impacts of structural changes

- Types of structural tasks
  - Path tasks
  - Finding antecedent and subsequent elements in system
  - Feedbacks and their identification
  - Finding elements or relations with specific parameters
  - Network‘s flow
  - System’s decomposition and integration
  - System‘s goals

Path tasks

- Typical tasks:
  - Finding all possible path between two elements
  - Path‘s length assessment
  - Finding path with predefined parameters (e.g. shortest, longest, of certain length, ...)
  - Tracing path of certain length
  - Finding system’s magnitude (set of all paths in the system)

- Basic graph theory algorithms
  - Dijkstra‘s algorithm for finding the shortest path,
  - Ford-Fulkerson algorithm for finding the maximum flow in a flow network

Basic theorem

- Non-zero coefficients in the i^{th} power of adjacency matrix determine the existence and amount of paths with length i between appropriate oriented doubles of elements.

- Adjacency matrix
  - Standard two dimensional square matrix
  - Dimension n equal to the number of elements in the system
  - Displays the existence of relation between particular elements (0 – no relation, 1 – existing relation)

Matrix „powers“

- In general there are no powers of matrixes
- For square matrix, multiplication by itself is possible
- Further on we call it exponentiation (doing left multiplication)
- Matrix multiplication for M(pxq) a N(qxr):

$$(M \cdot N)_{ij} = \sum_{k=1}^{q} m_{ik} n_{kj}$$

- S1 – adjacency matrix
- S2=S1xS1
- S3=S1xS2
- S4=S1xS3
- …..

Meaning of adjacency matric powers

- S1
  - Displays all paths of length 1 (direct relation between elements)
- S2
  - Displays paths of length 2
- …

Number of paths

- Number of paths of length n

$$S_n = \sum_{\forall i,j} (a_{ij})^n$$

- Total amount of paths in the system (system's magnitude)

$$C = \sum_{n=1}^{\infty} S_n$$

- $(E - S)^{-1} = E + C$,   (E is unit matrix)

Basic graph theory terms

- Directed graph – all the graph edges have a direction associated with them
- Acyclic graph – finite directed graph with no cycles
- For acyclic finite directed graphs the length of the longest path is limited

Work with adjacency matrix powers

- For the adjacency matrix of acyclic finite directed graphs there exist a matrix power from which all following powers are zero matrixes
- For cyclic directed graphs we search up to the r-th power of matrix, where $r=\min(p, q)$
  p – number of elements
  q – number of relations
  - Higher matrixes show the cyclic paths

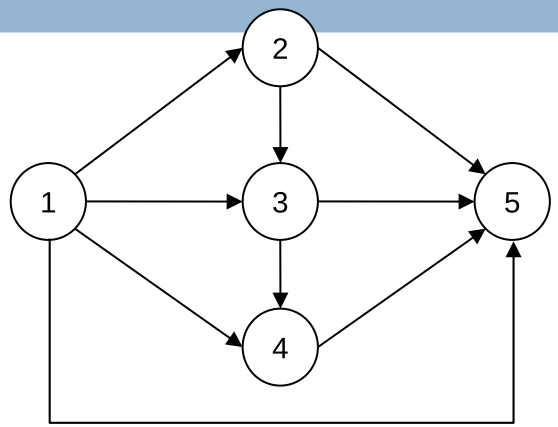
Antecedent and subsequent elements

- Antecedent elements = elements on paths leading **into** certain element.
- Subsequent elements = elements on paths leading **from** certain element.
- Generation = path length (number of following relations – edges) that connects certain element with its antecedents or subsequent
- Finding antecedent and subsequent elements:
  - Subsequent elements to element i, j^{th} generation can be found in the j^{th} power of adjacency matrix (SJ) in i^{th} row. Column marks of columns with non-zero value in the i^{th} row show the subsequents.
  - Antecedent elements to element i, j^{th} generation can be found in the j^{th} power of adjacency matrix (SJ) in i^{th} column. Row marks of rows with non-zero value in the i^{th} column show the antecedents.

Antecedent and subsequent elements usage

- Search for possible spreading of error, disease, information, etc.
- Checking the results of regularization
- Searching of active sources
- Serching of active outputs and impact on the neighbourhood
- (tasks about the contamination, imunity, etc.)

Example



- Task: in the following system find out:
  a. Antecedent element to element 4, 2nd generation and all subsequent elements to element 3,
  b. Number of different paths between elements 1 and 5
  c. The longest path in the system
  d. The shortest path between element 1 and 4
  e. System's magnitude
  f. Trace the path from 1 to 5 with length 3 using the forward and backward algorithm

Adjacency matrix and its powers

S1	1	2	3	4	5
1	0	1	1	1	1
2	0	0	1	0	1
3	0	0	0	1	1
4	0	0	0	0	1
5	0	0	0	0	0

S3	1	2	3	4	5
1	0	0	0	1	2
2	0	0	0	0	1
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

S2	1	2	3	4	5
1	0	0	1	1	3
2	0	0	0	1	1
3	0	0	0	0	1
4	0	0	0	0	0
5	0	0	0	0	0

S4	1	2	3	4	5
1	0	0	0	0	1
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

S5=0

Example

a. Antecedent element to element 4, 2nd generation
   1, 2
   all subsequent elements to element 3
   1st generation – 4,5
   2nd generation – 5
   $\rightarrow$ 4,5
b. Number of different paths between elements 1 and 5
   Number of paths between elements i and j is a sum of all aij values in all matrixes
   1 paths length 1
   3 paths of length 2
   2 paths of length 3
   1 path of length 4
   total 7
c. The longest path in the system
   maximal path lenght is equal to the maximal non-zero adjacency matrix power
   longest path is of lenght 4

Example

d. The shortest path between element 1 and 4
searching for the first non-zero aij position in all the matrixes from the first one
direct path (length 1)

e. System's magnitude (number of all paths in the system)
Sum of all values in all the matrixes
9+8+4+1=22
$\mathbf{(E - S)^{-1} = E + C}$

S1	1	2	3	4	5
1	0	1	1	1	1
2	0	0	1	0	1
3	0	0	0	1	1
4	0	0	0	0	1
5	0	0	0	0	0

E-S1	1	2	3	4	5
1	1	-1	-1	-1	-1
2	0	1	-1	0	-1
3	0	0	1	-1	-1
4	0	0	0	1	-1
5	0	0	0	0	1

E	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

$(E\text{-}S1)^{-1}=E+C$

	1	2	3	4	5
1	1	1	2	3	7
2	0	1	1	1	3
3	0	0	1	1	2
4	0	0	0	1	1
5	0	0	0	0	1

C	**1**	**2**	**3**	**4**	**5**
1	**0**	**1**	**2**	**3**	**7**
2	**0**	**0**	**1**	**1**	**3**
3	**0**	**0**	**0**	**1**	**2**
4	**0**	**0**	**0**	**0**	**1**
5	**0**	**0**	**0**	**0**	**0**

$$\sum C=22$$

Note

- the task of finding shortest path can be often solved more effeciently using graph theory algorithms – e.g.
  - Dijkstra algorithm
  - Dantzig algorithm
  - Floyd algorithm
  - Etc.

Forward algorithm

1. In the S1 matrix find list of elements with distance 1 from the initial element
2. Reduce the path length of 1
3. From the list of elements put the first element as the initial element
4. If the new reduced length is equal to zero and if in the current list of elements the final element exist, then END

Else

5. Return to step 3 and in the place of initial element put next element in the list
6. If all elements from the current list are used and condition 4 is not fulfilled, use as the current list the previous one
7. If the condition in 4. is not fulfilled for any list, there does not exist path of required length between chosen elements

- Disadvantage – looks through all the paths (solution by “force“)

Example

- Trace the path from 1 to 5 with length 3 using the forward algorithm
  - *Step 1. list of elements: 2,3,4,5*
  - *Step 2. length=2*
  - *Step 3. initial node 2*
  - *Steps 4. NO, 5. initial element 2*
  - *Step 1. list of elements 3,5*
  - *Step 2. length = 1*
  - *Initial node 3*
  - *4., 5.*
  - *Step 1. list of elements 4,5*
  - *length =0*
  - *Initial node 4*
  - *4. lenght 0, exists final element 5 – Found path 1-2-3-5*
  - *Previous list of elements, next element – 5, lenght 1*
  - *Previous list of elements, next element 3*
  - …
  - …
  - …
  - *Found path 1-3-4-5*

Backward algorithm

1. In the S1 matrix find all direct antecedent elements of final element and put them in list Q1. Reduce the path length of lenght 1. If the remaining length is zero, then END
2. In the power of adjacency matrix, which exponent is equal to the reduced path length find all subsequent elements of the initial element and put them in list Q2
3. Find common element in lists Q1 and Q2. It is the last but one element on the path (element ahead of the final element). Use this element as the final element and continue with step 1.

Example

- Trace the path from 1 to 5 with length 3 using the backward algorithm
  - *I find all the direct antecedent elements of node 5 in S_1 . Q1 contains nodes 1,2,3,4.*
  - *In the matrix power S_2 (3 minus1) I find end nodes from node 1. Q2 contains 3,4,5. Intersection of Q1 and Q2 are nodes 3 and 4.*

  - *Looking for paths from 1 to 4 length 2*
  - *I find all the direct antecedent elements of node 4 in S_1 . Q1 contains nodes 1,3.*
  - *In the matrix power S_1 (2 minus 1) I find end nodes from node 1. Q2 contains nodes 2,3,4,5. Intersection of Q1 and Q2 is node 3.*
  - *The remaining path length after subtracting 1 is zero.*
  - *Found path 1-3-4-5*

  - *Looking for paths from 1 to 3 length 2*
  - *I find all the direct antecedent elements of node 3 in S_1 . Q1 contains nodes 1,2.*
  - *In the matrix power S_1 (2 minus 1) I find end nodes from node 1. Q2 contains nodes 2,3,4,5.*
  - *Intersection of Q1 and Q2 is node 2.*
  - *The remaining path length after subtracting 1 is zero.*
  - *Found path 1-2-3-5.*

Feedbacks

- Important aspect
- Can be both
  - beneficially – control theory, living beings, robots, …
  - or undesirable –“deadlock“

Feedbacks

- In graph theory
  - Walk – sequence of nodes and edges
  - Closed walk – sequence of nodes and edges where the last node is also the first one
  - Loop – edge that connects the node to itself
  - Feedback – existence of a closed walk in a graph
  - Cycle – a closed walk evaluated by the number of repeating

Feedback identification

- Using power matrixes of the systems
- **In oriented graph there exists closed walk ⇔ there exist at least one non-zero element at the main diagonal of C matrix, where C = Σ S^i**

- Two certain indicators
  - Element on the main diagonal
  - (Element symetrically according the main diagonal) – finding it sooner
    - In any power of adjacency matrix

- Suspicious thing – elements below the main diagonal
  - There exists some back way (from element with higher number to element with lower number)
  - But is has to be also forward way between these elements to ensure feedback (otherwise it may be just unsuitable numbering of elements)

Taking care of feedbacks

- Unfoldment in time – detailed control of time relations
- Integration of some part of the system – feedback within one element

- Note: sometimes the feedback are necessary