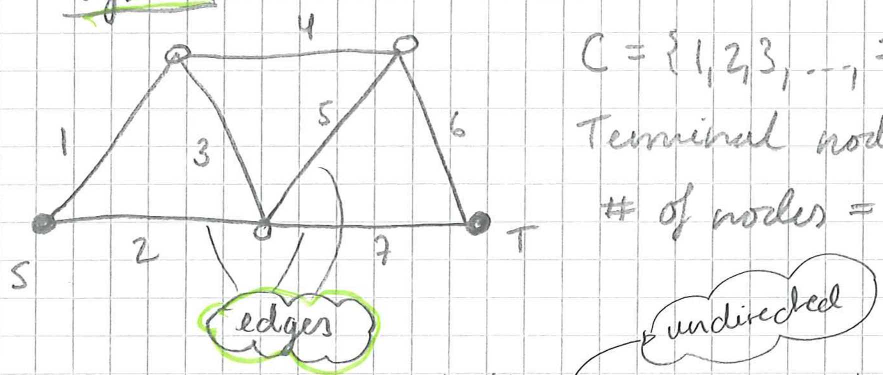
4.5: Computing the reliability of undirected network systems

Will present:

→ Algorithm for computing exact reliability of systems with indep. comp. state variables: The factoring algorithm.

→ Optimal performance: Assume undirected network system:



$C = \{1, 2, 3, \dots, 7\}$

Terminal nodes: S, T

\# of nodes $= 5$

Signals can be sent both ways along the edges.

(Need the following Thm for the algorithm:)

Thm 4.5.1: $i, j \in C$, $i \neq j$

(i) If i & j are in series, then $h(\vec{p})$ only depends on p_i and p_j through $p_i \cdot p_j$. Can replace i and j by a single comp. with reliability $p_i p_j$ without changing the system reliability. This is a series reduction.

(ii) If i & j are in parallel, then $h(\vec{p})$ only depends on p_i and p_j through $p_i \amalg p_j$. Can replace i and j by a single comp. with reliability $p_i \amalg p_j$ without changing the system reliability. This is a parallel reduction.

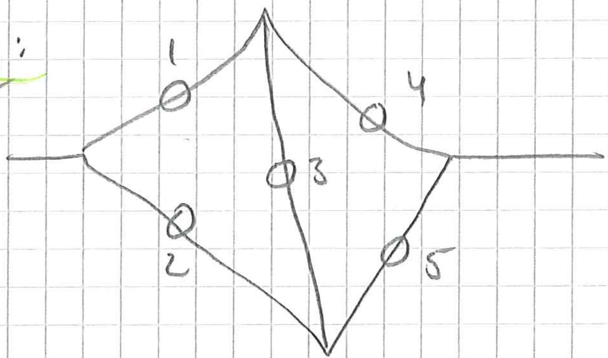
→ Common term for either series or parallel reductions:
<u>s-p reductions</u>

Def 4.5.2: A system is <u>s-p-reducable</u> if there are comp's in either series or parallel in the system. If not, the system is <u>complex</u>. A system which can be s-p-reduced to a single comp. is an <u>s-p-system</u>.

Ex:

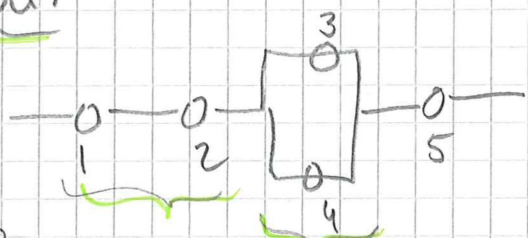
Complex system:

Bridge structure



Must pivot to find reliability

S-p-system:



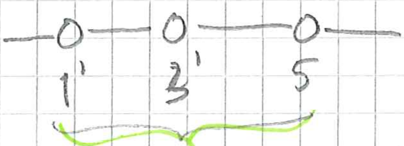
$1'$: $p_1 p_2$

series red.

$3'$: $p_3 \amalg p_2$

parallel red.

⇓



⇒ $h(\vec{p}) = \underline{p_{1'} \cdot p_{3'} \cdot p_5}$

$1''$: $p_{1'} p_{3'} p_5$

series red.

The factoring algorithm: (C, ϕ) bms. Assume at least one of its components is relevant (non-trivial bms). To compute the reliability $h(\vec{p})$:

Very important! Used over & over!

step 1: Perform all possible s-p-reductions. Call the reduced system (C^r, ϕ^r). Then, (C^r, ϕ^r) must also have at least one relevant component.

because s-p reductions do not remove relevance; can check this from the def. (do series & parallel red. separately)

Step 2: One of two cases can happen:

Case 1: (C^r, ϕ^r) contains precisely one relevant comp. with updated reliability p_e. Then $h(\vec{p}) = p_e$.

recall ex. prev. pg. with series & parallel red.

Case 2: (C^r, ϕ^r) contains several relevant comp's. In this case, choose a comp. $e \in C^r$ & do a pivotal decomposition, i.e., compute $h(\vec{p})$ by Thm. 3.1.1:

$$h(\vec{p}^{\,C^r}) = p_e\, h(1_e, \vec{p}^{\,C^r}) + (1-p_e)\, h(0_e, \vec{p}^{\,C^r}).$$

Then, compute $h(1_e, \vec{p}^{\,C^r})$ & $h(0_e, \vec{p}^{\,C^r})$ by repeated use of the algorithm.

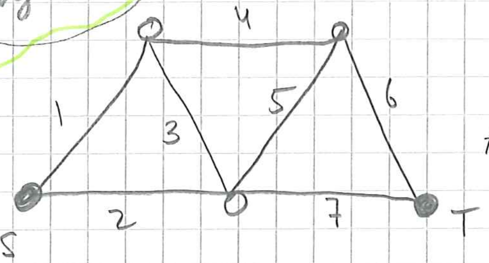
(+ Choice in Step 2: Efficiency of alg. depends on this choice.)

→ Can be shown: One should always pivot s.t. the resulting substructures are coherent. i.e., all comp's are relevant

→ The factoring alg. is of order $O(2^n)$ → in general
(same as state enumeration method).
In practise: Very efficient in some cases & for appropriately chosen pivoting comp's.

Roughly: State enumeration is always equally bad, but factoring can be very good

Example: (How to apply the factoring algorithm)



$C = \{1, \dots, 7\}$

Assume indep. comp. state var. & $P(X_i = 1) = p_i$, $i \in C$.

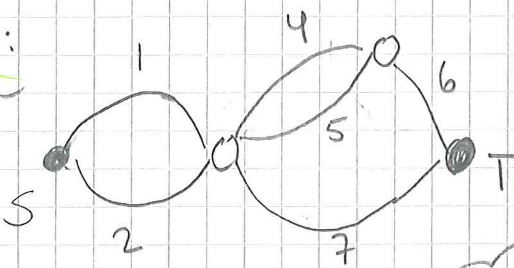
Not s-p-reducable (no series or parallel).

Pivotal decomposition:

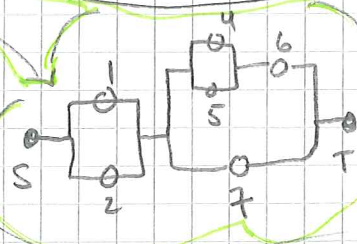
In the book, choose comp. 4: For illustration, we try comp. 3 instead: $h = p_3 h_{+3} + (1-p_3) h_{-3}$

$(C \setminus 3, \phi_{+3})$:

3 works



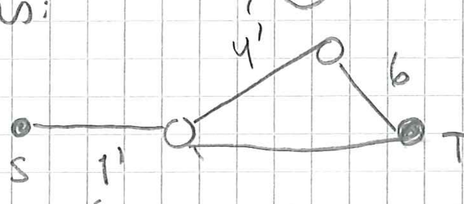
This is s-p-reducable!



Do parallel reductions:

$p_4 \amalg p_5$

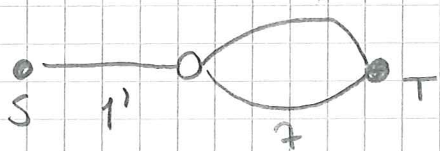
$((C \setminus 3)^r, \phi^r_{+3})$:



$p_1 \amalg p_2$

6' → $p_{4'} \cdot p_6$

Do series red:



Parallel red:

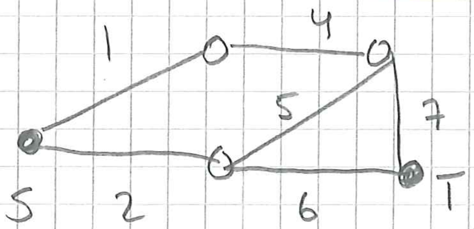
7' → $p_7 \amalg p_{6'}$

Finally, series red:

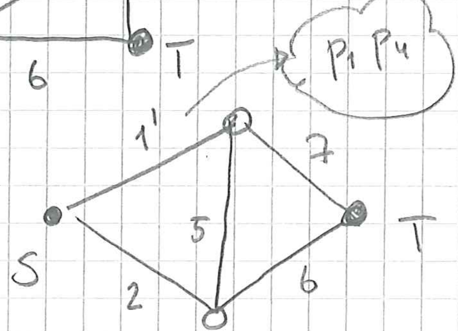
$p_{1'} \cdot p_{7'}$

So: $h_{+3} = p_{1'} \cdot p_{7'}$ (Can insert all of the expressions recursively, but this is not efficient. In practical alg; only the numbers are stored, not the symbolic expressions)

3 doesn't work; $(C \setminus 3, \phi_{-3})$



Series red:



A: bridge structure: Can pivot wrt. 5 if do not remember the formula:

$$h_{-3} = p_5 (p_{1'} \amalg p_2)(p_7 \amalg p_6) + (1-p_5)((p_{1'} p_7) \amalg (p_2 p_6))$$

(Could insert all of this in h, but in practise:)

$$h = p_3 h_{+3} + (1-p_3) h_{-3}$$

(Pretty much the same amount of work as when pivot wrt 4)